ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη iOS εφαρμογής για τουριστικό οδηγό»

Του φοιτητή
Αλέξανδρου Παλληκαρίδη
185407

Επιβλέπων
Δρ Τσιακμάκης Κυριάκος

Ιούνιος 2025

Τίτλος Π.Ε.: Ανάπτυξη iOS εφαρμογής για τουριστικό οδηγό

Κωδικός Π.Ε.: 24170

Ονοματεπώνυμο φοιτητή: Παλληκαρίδης Αλέξανδρος

Ονοματεπώνυμο εισηγητή: Τσιακμάκης Κυριάκος

Ημερομηνία ανάληψης Δ.Ε.: 27/3/2024

Ημερομηνία περάτωσης Δ.Ε.: 30/5/2025

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

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

«Στην οικογένειά μου»

Πρόλογος

Η ιδέα για την εκπόνηση της παρούσας πτυχιακής εργασίας με θέμα την ανάπτυξη μιας iOS εφαρμογής τουριστικού οδηγού γεννήθηκε μέσα από την εμπειρία που αποκόμισα κατά τη διάρκεια της πρακτικής μου άσκησης ως iOS Developer. Μέσα από την καθημερινή ενασχόλησή μου με τον σχεδιασμό και την υλοποίηση εφαρμογών για κινητές συσκευές, ήρθα σε επαφή με τις τεχνολογίες αιχμής του οικοσυστήματος της Apple, τις προκλήσεις που προκύπτουν από τη δημιουργία λειτουργικών και φιλικών προς τον χρήστη διεπαφών, καθώς και με τις πραγματικές ανάγκες των χρηστών και της αγοράς.

Η εμπειρία αυτή αποτέλεσε ένα σημαντικό εφαλτήριο για να συνδυάσω τις γνώσεις που απέκτησα με έναν τομέα ιδιαίτερης κοινωνικής και οικονομικής σημασίας, όπως είναι ο τουρισμός. Με αυτό το έργο, στόχος μου ήταν να δημιουργήσω ένα εργαλείο που να προσφέρει προστιθέμενη αξία τόσο στους επισκέπτες μιας περιοχής όσο και στις τοπικές επιχειρήσεις, ενισχύοντας την αλληλεπίδραση, την ενημέρωση και τη διάχυση της πληροφορίας.

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη της iOS εφαρμογής με τίτλο «River View Project», η οποία δημιουργήθηκε για λογαριασμό της ΚΟΙΝΣΕΠ RiverView. Η εφαρμογή αυτή σχεδιάστηκε ως ένα σύγχρονο ψηφιακό εργαλείο που αποσκοπεί στην ενίσχυση της τοπικής οικονομίας και την ανάδειξη του πολιτιστικού και φυσικού πλούτου της περιοχής. Μέσα από έναν φιλικό προς τον χρήστη τουριστικό οδηγό, ο επισκέπτης μπορεί να περιηγηθεί σε αξιοθέατα, να ενημερωθεί για σημεία ενδιαφέροντος και να έρθει σε επαφή με την τοπική κοινότητα.

Η εργασία αναλύει, πέρα από την τεχνική υλοποίηση της εφαρμογής, και το κοινωνικό υπόβαθρο στο οποίο βασίστηκε η ανάπτυξή της. Γίνεται αναφορά στην έννοια της κοινωνικής οικονομίας και στον ρόλο που διαδραματίζει στην υποστήριξη βιώσιμων μορφών επιχειρηματικότητας, ιδιαίτερα μέσα από την αξιοποίηση των ψηφιακών τεχνολογιών. Μέσα από τη σύνδεση της τεχνολογίας με την κοινωνική επιχειρηματικότητα, η εφαρμογή River View Project αναδεικνύει τη δυναμική των ψηφιακών λύσεων στην υποστήριξη της τοπικής ανάπτυξης, του βιώσιμου τουρισμού και της ενίσχυσης της κοινωνικής συνοχής.

Η συμβολή αυτής της εργασίας δεν περιορίζεται μόνο στην υλοποίηση μιας εφαρμογής, αλλά και στην ανάδειξη ενός μοντέλου συνεργασίας μεταξύ της τεχνολογίας και της κοινωνικής καινοτομίας, με απώτερο στόχο τη δημιουργία θετικού αντίκτυπου σε τοπικό επίπεδο.

«iOS application for a tourist guide»

«Alexandros Pallikaridis»

Abstract

This thesis focuses on the development of the iOS application titled "River View Project", which was created on behalf of the social cooperative enterprise KOINSEP RiverView. The application was designed as a modern digital tool aimed at strengthening the local economy and showcasing the cultural and natural heritage of the region. Through a user-friendly digital tourist guide, visitors can explore attractions, access useful information, and connect with the local community.

Beyond the technical implementation, this work also explores the social context that inspired the creation of the application. It introduces the concept of the social economy and highlights its role in supporting sustainable forms of entrepreneurship, particularly through the utilization of digital technologies. By bridging technology and social entrepreneurship, the River View Project demonstrates the potential of digital solutions in promoting local development, sustainable tourism, and social cohesion.

The contribution of this thesis lies not only in the development of a mobile application but also in the presentation of a collaborative model that integrates technology with social innovation—ultimately aiming to generate a positive impact at the local level.

Ευχαριστίες

Η ολοκλήρωση της παρούσας πτυχιακής εργασίας αποτελεί βασική προϋπόθεση για την περάτωση των προπτυχιακών σπουδών μου στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου Ελλάδος. Βασική προϋπόθεση για την ολοκλήρωση των σπουδών μου ήταν η υποστήριξη μερικών ατόμων τόσο από τον οικογενειακό μου κύκλο όσο και από τους ευρύτερους κοινωνικούς μου κύκλους.

Πρώτα απ' όλα, οφείλω να ευχαριστήσω τον επιβλέποντα της εργασίας, κ. Τσιακμάκη Κυριάκο για την εμπιστοσύνη του στην ανάθεση του θέματος της πτυχιακής εργασίας. Επιπλέον, θερμές ευχαριστίες οφείλω σε όλους τους καθηγητές του τμήματος για τις γνώσεις που μου μετέδωσαν κατά τη διάρκεια των προπτυχιακών μου σπουδών.

Τέλος, ένα μεγάλο ευχαριστώ οφείλω στους δικούς μου ανθρώπους για την κατανόηση και την υποστήριξη που μου έδειξαν απλόχερα, τόσο κατά το διάστημα της εκπόνησης της παρούσας πτυχιακής εργασίας όσο και γενικότερα, κατά την διάρκεια των σπουδών μου. Δε θα μπορούσα να παραλείψω να αναφερθώ στους υπέροχους γονείς μου αλλά και στην αδερφή μου, που συνεχίζουν να με πιστεύουν, να με ενθαρρύνουν και να με στηρίζουν απεριόριστα.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Αποσπασμάτων Κώδικα

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η παρούσα εργασία εξετάζει την ανάπτυξη της εφαρμογής <<River View Project>>, ενός ψηφιακού εργαλείου που στόχος του είναι η ανάδειξη της τοπικής πολιτιστικής και φυσικής κληρονομιάς. Κεντρικός σκοπός είναι η εισαγωγή του αναγνώστη σε έννοιες όπως η κοινωνική οικονομία, συμβάλλοντας έτσι στην κατανόηση των κινήτρων που οδήγησαν στον σχεδιασμό και την υλοποίηση της εφαρμογής. Το <<River View Project>> έχει ως σκοπό την ενίσχυση της τοπικής οικονομίας, προβάλλοντας σημεία ιδιαίτερου ενδιαφέροντος μέσω μιας φιλικής προς τον χρήστη και εύχρηστης ψηφιακής πλατφόρμας. Παράλληλα, η εργασία διερευνά τη διασύνδεση της τεχνολογίας με την κοινωνική επιχειρηματικότητα, αναδεικνύοντας τον ρόλο που μπορούν να διαδραματίσουν οι ψηφιακές λύσεις στην προώθηση της βιώσιμης τουριστικής ανάπτυξης και της τοπικής ενδυνάμωσης.

1.2 Κοινωνική Οικονομία

Κοινωνική Οικονομία είναι ο τομέας της οικονομίας που περιλαμβάνει οικονομικές, επιχειρηματικές και παραγωγικές δραστηριότητες οι οποίες οργανώνονται κυρίως με βάση κοινωνικούς σκοπούς και όχι την επίτευξη του μέγιστου κέρδους. Οι οργανισμοί της κοινωνικής οικονομίας – όπως είναι συνεταιρισμοί, κοινωνικές επιχειρήσεις, ενώσεις και μη κερδοσκοπικοί οργανισμοί – λειτουργούν με γνώμονα την αλληλεγγύη, τη δημοκρατική συμμετοχή, την κοινωνική συνοχή και την κάλυψη συλλογικών αναγκών. Προτεραιότητά τους αποτελεί η δημιουργία κοινωνικής αξίας, η ενίσχυση της απασχόλησης και η στήριξη ευάλωτων κοινωνικών ομάδων.

1.3 River View Project

Η River View Project Κοιν.Σ.Επ. είναι μία νέα κοινωνική συνεταιριστική επιχείρηση με έδρα το Δήμο Αγρινίου. Στοχεύει στην προστασία και ανάδειξη της φυσικής και πολιτιστικής κληρονομιάς της ευρύτερης περιοχής του Αγρινίου και της Αιτωλοακαρνανίας, εστιάζοντας στον ποταμό Αχελώο. Ακόμη στοχεύει στην ενίσχυση της κοινωνικής συνοχής, της επιχειρηματικότητας και της καινοτομίας, προωθώντας την τοπική ανάπτυξη. Η επιχείρηση επιδιώκει την ανάπτυξη εναλλακτικών μορφών τουρισμού, και στην ενίσχυση της τοπικής οικονομίας. Οι πρωτοβουλίες της επικεντρώνονται κυρίως σε μουσικές εκδηλώσεις, εκθέσεις τέχνης, προβολή τοπικών προϊόντων και παραγωγών, αθλητικές δραστηριότητες, επιμορφωτικές ημερίδες, εθελοντικές δράσεις και θεματικά workshops. Στα σχέδιά της περιλαμβάνονται επίσης, η δημιουργία νέων θέσεων εργασίας μέσω της ανάπτυξης εγκαταστάσεων εστίασης, διαμονής και αναψυχής.

1.4 Διάρθρωση Πτυχιακής Εργασίας

- Στο πρώτο κεφάλαιο γίνεται εισαγωγή και σύντομη περιγραφή του θέματος που πραγματεύεται η εργασία.
- Στο δεύτερο κεφάλαιο παρουσιάζονται οι τεχνολογίες οι οποίες χρησιμοποιήθηκαν προκειμένου να καταστεί δυνατή η υλοποίηση της εφαρμογής που περιγράφει το παρόν κείμενο.
- Στο τρίτο κεφάλαιο παρουσιάζονται τα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής.
- Στο τέταρτο κεφάλαιο παρουσιάζεται η ίδια η εφαρμογή, η λειτουργικότητά της και η εμπειρία χρήστη.
- Στο πέμπτο κεφάλαιο παρουσιάζονται μέρη του κώδικα που είναι υπεύθυνος για την υλοποίηση της εφαρμογής.
- Στο έκτο και τελευταίο κεφάλαιο του κειμένου αυτού, γίνεται μια σύνοψη της όλης προσπάθειας και παρουσιάζονται κάποια συμπεράσματα που προκύπτουν από τη λειτουργία της εφαρμογής.

Κεφάλαιο 2ο: Περιγραφή τεχνολογιών

2.1 Λειτουργικό σύστημα iOS

2.1.1 Εισαγωγή

Η εφαρμογή για την οποία συντάσεται το παρόν κείμενο απευθύνεται σε χρήστες Apple κινητών συσκευών, οι οποίες χρησιμοποιούν iOS λειτουργικό σύστημα.

Το Λειτουργικό σύστημα είναι το βασικό λογισμικό ενός υπολογιστικού συστήματος. Είναι υπεύθυνο για την διαχείρση του υλικού (hardware) και του λογισμικού (software) ενός υπολογιστή ή μίας κινητής συσκευής. Αποτελεί τον ενδιάμεσο κρίκο ανάμεσα στον χρήστη και το υλικό, επιτρέποντας την εκτέλεση εφαρμογών και την ομαλή λειτουργία του συστήματος. Βασικές λειτουργίες του είναι η διαχείριση της μνήμης, των αρχείων, των συσκευών εισόδου/εξόδου, των χρηστών και των διεργασιών.

Στους υπολογιστές, τα επικρατέστερα λειτουργικά συστήματα είναι τα εξής:

- Microsoft Windows
- macOS της Apple
- Linux (με πολλές διανομές, όπως Ubuntu και Debian)

Στις κινητές συσκευές (smartphones και tablets), κυριαρχούν δύο λειτουργικά συστήματα:

- Android (Google)
- iOS (Apple)

Το iOS, είναι λογισμικό κλειστού κώδικα το οποίο αναπτύσεται από την Apple, με σκοπό να χρησιμοποιείται αποκλειστικά από τις συσκευές που η ίδια κατασκευάζει και διανέμει στην αγορά.

2.1.2 Ιστορικό πλαίσιο

Το iOS είναι το λειτουργικό σύστημα που αναπτύσσει η Apple αποκλειστικά για τις φορητές συσκευές της. Παρουσιάστηκε πρώτη φορά τον Ιανουάριο του 2007, από τον Steve Jobs, κατά την διάρκεια της παρουσίασης του πρώτου iPhone με το όνομα iPhone OS. Από τότε, κάθε χρόνο η εταιρεία κυκλοφορεί νέες εκδόσεις, στις όποίες προσθέτει καινούργια χαρακτηριστικά και δυνατότητες στο λογισμικό. Οι εκδόσεις αυτές είναι οι εξής:

- iPhone OS 1 (2007)

Η πρώτη έκδοση του λειτουργικού συστήματος προκειμένου να υπάρχει υποστήριξη και να επιτευχθεί και η λειτουργία του και η κυκλοφορία του πρώτου iPhone. Υποστήριζε web browsing μέσω Safari ενώ παρείχε οπτικό τηλεφωνικό κατάλογο, email και άλλες βασικές λειτουργίες. Η πρώτη αυτή έκδοση υποστήριζε μόνο ενσωματωμένες εφαρμογές της εταιρείας.

- iPhone OS 2 (2008)

Συνοδεύτηκε με την κυκλοφορία του App Store. Αποτέλεσε μία τεράστια αλλαγή καθώς έδωσε την δυνατότητα σε developers να αναπτύξουν εφαρμογές για την πλατφόρμα και να τις διαθέσουν μέσω του App Store στους χρήστες. Συντέλεσε στην μεγάλη επιτυχία του οικοσυστήματος.

- iPhone OS 3 (2009)

Με την τρίτη έκδοση του λογισμικού, γίνονται διαθέσιμες και άλλες υπηρεσίες για τα iPhone όπως η αποστολή και λήψη MMS, η αναζήτηση τύπου Spotlight, η λειτουργία εντοπισμού συσκευής Find My Phone και άλλες.

- iOS 4 (2010)

Το 2010 η Apple παρουσίασε το iPad, βασισμένο στο ίδιο λειτουργικό με το iPhone. Το λειτουργικό σύστημα μετονομάστηκε σε iOS καθώς πλέον έπαψε να χρησιμοποιείται από μία μονάχα συσκευή. Με το νέο λογισμικό προστέθηκε η δυνατότητα για προσαρμογή της οθόνης και έγινε δυνατή η ταυτόχρονη χρήση πάνω από μίας εφαρμογής.

- iOS 5 (2011)

Προστίθεται η δυνατότητα λήψης ειδοποιήσεων από τις εφαρμογές ενώ προστίθενται και τα iCloud, iMessage, Siri.

- iOS 6 (2012)

Με την έκδοση αυτή η Apple προσθέτει ακόμη περισσότερες native εφαρμογές. Παρουσιάζεται για πρώτη φορά το Apple Maps, το οποίο συνεργάζεται με την Siri και στην ουσία αποτελεί την απάντηση της εταιρείας στο Google Maps.

- iOS 7 (2013)

Γίνεται μία ριζική αλλαγή στην εμφάνιση του λογισμικού, αφού γίνεται ένας πλήρης επανασχεδιασμός του UI ενώ προστίθεται και το AirDrop, το οποίο είναι ένα εργαλείο επικοινωνίας μεταξύ των Apple συσκευών, που χρησιμεύει στον άμεσο διαμοιρασμό αρχείων.

- iOS 8 (2014)

Προστίθεται η δυνατότητα ενεργοποίησης της Siri μέσω φωνητικών εντολών. Προστίθενται ακόμη νέες δυνατότητες στις εφαρμογές ενώ ενισχύονται οι έξυπνες δυνατότητες των συσκευών της εταιρείας.

- iOS 9 (2015)

Προστίθεται η δυνατότητα διαχωρισμού της οθόνης, με την οποία εξασφαλίζεται η παράλληλη εκτέλεση εφαρμογών στο οπτικό πεδίο του χρήστη.

- iOS 10 (2016)

Προστίθεται ξεχωριστή οθόνη για τα widget, υλοποιείται ολικός επανασχεδιασμός της οθόνης κλειδώματος, παρέχοντας στον χρήστη την δυνατότητα να αλληλεπιδρά με τις ειδοποιήσεις πιό εύκολα αλλά και πιό γρήγορα.

- iOS 11 (2017)

Αποτελεί την έκδοση στην οποία δίνεται έμφαση στην βελτίωση της εμπειρίας για του χρήστες iPad. Προστίθεται το Dock, με το οποίο εναλλάσονται οι εφαρμογές που τρέχουν στην ίδια χρονική στιγμή, ο διαχειριστής αρχείων και το ARKit.

- iOS 12 (2018)

Αποτελεί μία έκδοση στην οποία η Apple προχωρά κυρίως σε διορθώσεις και βελτιώσεις στις native εφαρμογές της, με σκοπό την υποστήριξη και την ομαλή λειτουργία των παλαιότερων συσκευών της.

- iOS 13 (2019)

Στην έκδοση αυτή η εταιρεία προχωρά στον διαχωρισμό των εκδόσεων για τα iPhone και τα iPad. Πλέον το λογισμικό των iPad θα ονομάζεται iPadOS, ενώ το λογισμικό των iPhone παραμένει το iOS. Αναπτύσονται λειτουργίες ασφαλείας, όπως το Face ID και παρουσιάζεται το Dark Mode.

- iOS 14 (2020)

Προστίθεται η δυνατότητα για εμφάνιση Widgets στην αρχική οθόνη. Στην παρούσα έκδοση βελτιώνεται η πολιτική απορρήτου και η χρήση δεδομένων των χρηστών από τις εφαρμογές.

- iOS 15 (2021)

Στην έκδοση αυτή γίνεται βελτίωση στις εφαρμογές της εταιρείας όπως το FaceTime, το Maps, το iMessage και το Live Text, στο οποίο προστίθεται η δυνατότητα αναγνώρισης κειμένου από φωτογραφίες κ.α.

- iOS 16 (2022)

Στην έκδοση αυτή του λογισμικού δίνεται έμφαση στην βελτίωση εμπειρίας του χρήστη ενώ παρέχονται και περισσότερες δυνατότητες εξατομίκευσης της συσκευής. Ενδεικτικά εντοπίζονται αλλαγές στην οθόνη κλειδώματος και στο iMessage.

- iOS 17 (2023)

Η έκδοση αυτή έρχεται με αναβαθμίσεις σε βασικές εφαρμογές του λογισμικού. Περιλαμβάνει ακόμη βελτιώσεις στην λειτουργία του AirDrop και αναβαθμίσεις στις προτάσεις ορθογραφίας του πληκτρολογίου μέσω AI.

- iOS 18 (2024)

Με την νέα αυτή έκδοση παρέχεται η δυνατότητα μεγαλύτερης μορφοποίησης τόσο της αρχικής οθόνης όσο και της οθόνης κλειδώματος. Ακόμη προστίθενται νέες λειτουργίες στο Apple Pay και στο Apple Wallet. Μία ακόμη μεγάλη αλλαγή είναι ο ολικός επανασχεδιασμός και η δυνατότητα τροποποίησης του κέντρου ελέγχου.

2.1.3 Βασικά χαρακτηριστικά

Το iOS είναι το λειτουργικό σύστημα, που σχεδιάζεται και υλοποιείται από την Apple και προορίζεται για τις κινητές συσκευές της. Η Apple εκμεταλλεύεται πλήρως αυτή την αποκλειστικότητα στην χρήση και την διανομή του iOS για να προσφέρει στον χρήστη μία ολοκληρωμένη εμπειρία μέσα από το οικοσύστημα της επιτυγχάνοντας την απρόσκοπτη επικοινωνία μεταξύ των προϊόντων που σχεδιάζει και προσφέρει στην αγορά. Με γνώμονα αυτό κάθε αναβάθμιση και προσθήκη στις λειτουργίες του iOS αποσκοπεί σε αναβάθμιση της εμπειρίας χρήσης του οικοσυστήματος της εταιρείας.

Το iOS είναι ένα λογισμικό κλειστού κώδικα. Αυτό σημαίνει πως ο πηγαίος κώδικας που έχει γραφτεί για το λογισμικό δεν είναι προσβάσιμο από τρίτους, παρά μόνο από τους προγραμματιστές της Apple. Παράλληλα διασφαλίζεται πώς το λογισμικό θα χρησιμοποιείται και θα διανέμεται μόνο στις συσκευές

που η εταιρεία επιθυμεί. Έτσι, η εταιρεία εξασφαλίζει πως μόνο μέσω αυτής και των προϊόντων της θα μπορεί κάποιος να χρησιμοποιήσει το λογισμικό αυτό στοχεύοντας σε ένα συγκεκριμένο κοινό που είναι διατεθημένο να γίνει μέρος του οικοσυστήματος που η εταιρεία έχει δημιουργήσει τα τελευταία χρόνια.

Ο συνδυασμός ενός λογισμικού κλειστού κώδικα και μίας συγκεκριμένης και περιορισμένης γκάμας συσκευών, παρέχει την δυνατότητα στην Apple εκμεταλλευόμενη τις δυνατότητες του λογισμικού και των συσκευών της, να προσφέρει στον χρήστη την βέλτιστη εμπειρία χρήσης. Μεγαλύτερος και αμέσως ανταγωνιστής για το λογισμικό της Apple είναι το Android της Google. Το Android είναι ένα λογισμικό ανοιχτού κώδικα. Μία βασική διαφορά μεταξύ των δύο λογισμικών είναι πώς στο iOS ο σχεδιασμός και η διεπαφή του χρήστη εμφανίζονται πιο απλά, αυτό από μερίδα του κοινού κρίνεται ως αρνητικό στον αντίποδα όμως υπάρχουν αυτοί που υποστηρίζουν πως το iOS είναι σε θέση να απευθυνθεί σε ένα μεγαλύτερο φάσμα κοινού εξαιτίας της ευχρηστίας του.

Αδιαμφισβήτητα ένα μεγάλο πλεονέκτημα του iOS εντοπίζεται στο κομμάτι των εφαρμογών που εκτελούνται. Αυτό επιτυγχάνεται καθώς η ίδια η εταιρεία σχεδιάζει και υλοποιεί το λειτουργικό της για μία συγκεκριμένη και περιορισμένη γκάμα συσκευών της. Έτσι παρέχεται η δυνατότητα για στοχευμένο σχεδιασμό των εφαρμογών, προσφέροντας πολύ καλές επιδόσεις και περιορίζοντας σε πολύ μεγάλο βαθμό τα λάθη. Όσον αφορά την ανάπτυξη ιδιόκτητων εφαρμογών κλειστού κώδικα η Apple έχει καταφέρει να δημιουργήσει εφαρμογές υψηλού επιπέδου, οι οποίες δεν υστερούν σε κάποιο γνώρισμα από τις αντίστοιχες, μαζικώς χρησιμοποιούμενες εφαρμογές της Google για το λογισμικό Android.

Στο κομμάτι της ασφάλειας η Apple δημιουργώντας ένα λογισμικό κλειστού πηγαίου κώδικα, υπερτερεί ενάντια των ανταγωνιστών της, αφού έχει την δυνατότητα να προστατευθεί σε μεγαλύτερο βαθμό από κακόβουλες παρεμβάσεις, σε σχέση με ένα λογισμικό ανοιχτού πηγαίου κώδικα, όπως είναι το Android.

2.1.4 Ανάπτυξη εφαρμογών για iOS

Καθώς οι πωλήσεις συσκευών της Apple έχουν μία διαρκώς αυξανόμενη πορεία, αυξάνεται και η ζήτηση για ανάπτυξη εφαρμογών συμβατών με το iOS. Η διάθεση των εφαρμογών αυτών στο κοινό επιτυγχάνεται μέσω του App Store. Μία εφαρμογή που διατείθεται μέσω του App Store, πρέπει να τηρεί κάποια αυστηρά κριτήρια, τόσο τεχνικά όσο και σχεδιασμού. Τα κριτήρια αυτά θέτονται από την Apple με σκοπό την εξασφάλιση της αξιοπιστίας, της ασφάλειας και μίας γενικής ομοιομορφίας, προσφέροντας μία ξεχωριστή εμπειρία χρήσης.

Η ανάπτυξη εφαρμογών για το iOS επιτυγχάνεται μέσω περιβάλλοντος που έχει δημιουργήσει η Apple για τον σκοπό αυτό και με την χρήση πολυάριθμων βιβλιοθηκών και σύγχρονων εργαλείων που κυκλοφορούν. Στη συνέχεια κάποια από αυτά θα αναλυθούν, καθώς χρησιμοποιούνται για την ανάπτυξη της εφαρμογής, για την οποία συντάσσεται το παρόν κείμενο. Η γλώσσα προγραμματισμού

που χρησιμοποιείται για την ανάπτυξη iOS εφαρμογών είναι η Swift, μία νέα σχετικά γλώσσα υψηλού επιπέδου.

2.2 Frameworks για iOS

2.2.1 Εισαγωγή

Με τον όρο framework αναφερόμαστε σε ένα σύνολο εργαλείων, βιβλιοθηκών, δομών και κανόνων που προσφέρουν στους προγραμματιστές υποδομή, πάνω στην οποία μπορούν να αναπτύξουν λογισμικό.

Συγκεκριμένα ένα framework μπορεί να προσφέρει στον προγραμματιστή έτοιμο κώδικα και λειτουργίες(όπως διαχείριση δικτύου, κ.τ.λ.), ώστε να μην ξεκινά η δουλεία του προγραμματιστή από το μηδέν. Μπορεί να προσφέρει ένα συγκεκριμένο τρόπο οργάνωσης του κώδικα, προσφέροντας δομή και συνέπεια στην ανάπτυξη εφαρμογών. Τέλος, βοηθά στην εξοικονόμηση χρόνου και στον περιορισμό των λαθών.

2.2.2 Κατηγορίες

Υπάρχουν πολλά frameworks, έτοιμα προς χρήση. Κάθε framework βασίζεται σε κάποια γλώσσα προγραμματισμού και εξυπηρετεί τις ανάλογες λειτουργικότητες, με βάση τον τύπο της εφαρμογής που θα δημιουργήσουμε χρησιμοποιώντας το. Με γνώμονα τον τύπο της εφαρμογής διακρίνονται στις εξής κατηγορίες:

1. Web Frameworks

Frameworks τα οποία χρησιμοποιούνται για ανάπτυξη web εφαρμογών.

2. Front-end Frameworks

Frameworks τα οποία προσφέρουν τον σκελετό για την ανάπτυξη διεπαφών για ιστοσελίδες.

3. Back-end Frameworks

Frameworks τα οποία προσφέρουν server-side λειτουργίες που μπορούν να αποτελέσουν θεμέλια για περαιτέρω ανάπτυξη. Η επιλογή του κατάλληλου framework εξαρτάται από την γλώσσα προγραμματισμού που θα χρησιμοποιηθεί.

4. Mobile App Development Frameworks

Frameworks τα οποία υποστηρίζουν την ανάπτυξη εφαρμογών για κινητές συσκευές για συγκεκριμένο περιβάλλον είτε αυτό ανήκει στα native, hybrid ή cross-platform. Τα τρία περιβάλλοντα που προαναφέρθηκαν θα αναλυθούν περαιτέρω παρακάτω.

5. Content Management Frameworks

Frameworks τα οποία παρέχουν επαναχρησιμοποιήσιμα κομμάτια με σκοπό την διαχείριση και την κοινή χρήση των χαρακτηριστικών κάποιου web framework ή CMS.

6. Data Science Frameworks

Frameworks τα οποία χρησιμοποιούνται για εξόρυξη και ανάλυση δεδομένων αλλά και για σχεδιασμό αλγορίθμων.

Ένας ακόμη μέσο κατηγοριοποίησης των frameworks είναι με βάση τον τύπο της εφαρμογής που θα αναπτυχθεί με την χρήση τους. Συγκεκριμένα χωρίζονται σε δύο κατηγορίες:

• Native Frameworks

Frameworks τα οποία χρησιμοποιούνται για ανάπτυξη εφαρμογών που προορίζονται για ένα συγκεκριμένο λειτουργικό σύστημα. Τα native frameworks μπορεί να περιπλέξουν την διαδικασία ανάπτυξης μίας εφαρμογής ή ακόμη και να ανεβάσουν το κόστος ανάπτυξης της, αφού απαιτούν μεγάλη εξατομίκευση, περισσότερο χρόνο και μεγαλύτερο ανθρώπικο δυναμικό προγραμματιστών.

• Hybrid Frameworks

Frameworks τα οποία χρησιμοποιούνται για ανάπτυξη εφαρμογών που μπορούν να εγκατασταθούν και να χρησιμοποιηθούν σε πάνω από ένα λειτουργικά συστήματα αλλά και πλατφόρμες. Αυτό πρακτικά σημαίνει ότι η διαδιακασία ανάπτυξης μπορεί να γενικευθεί αφού θα έχουμε μία πιο ευρύτερη χρήση από ένα μεγαλύτερο μέρους του κοινού. Ακόμη απλοποιείται η εργασία των προγραμματιστών.

2.2.3. MapKit

Το MapKit είναι ένα framework που παρέχεται από την Apple και επιτρέπει στους developers να ενσωματώσουν χάρτες, πλοήγηση, τοποθεσίες, pins (annotations), διαδρομές, γεωκωδικοποίηση και άλλες συναφείς λειτουργίες σε εφαρμογές iOS, iPadOS, macOS και watchOS. Χρησιμοποιείται για την εμφάνιση χαρτών μέσα σε views (MKMapView), για την εισαγωγή pins (annotations), για τον εντοπισμό της τρέχουσας τοποθεσίας του χρήστη, για να τον υπολογισμό διαδρομών και την πλοήγηση,

για γεωκωδικοποίηση αλλά και αντίστροφη γεωκωδικοποίηση (μετατροπή διεύθυνσης σε

```
import MapKit
let mapView = MKMapView(frame: view.bounds)
view.addSubview(mapView)
```

συντεταγμένες και αντίστροφα). Βασική οπτική συνιστώσα του MapKit είναι το MapView.

Κώδικας 1: Χρήση του MapKit [26]

Το MapKit παρέχει την δυνατότητα να ορίσουμε τον τύπου του χάρτη που θα εμφανιστεί (.standard, .satellite, .hybrid), να εμφανίσουμε την τρέχουσα τοποθεσία και να κάνουμε zoom, scroll, rotate, tilt στα πλαίσια του MapView.

Το MapKit δεν έχει ενσωματωμένη διαχείριση GPS γι' αυτό και χρειάζεται να χρησιμοποιήσουμε το "CoreLocation", όπως φαίνεται παραπάνω. Για να ανακτήσουμε την τρέχουσα θέση του χρήστη θα εκτός από το "CoreLocation" να προσθέσουμε και την παρακάτω γραμμή κώδικα:

```
let request = MKDirections.Request()
request.source = MKMapItem(placemark: MKPlacemark(coordinate: sourceCoord))
request.destination = MKMapItem(placemark: MKPlacemark(coordinate: destCoord))
request.transportType = .automobile
let directions = MKDirections(request: request)
directions.calculate { response, error in
    guard let route = response?.routes.first else { return }
    mapView.addOverlay(route.polyline)
    mapView.setVisibleMapRect(route.polyline.boundingMapRect, animated: true)
```

"mapView.showsUserLocation = true"

Κώδικας 2: Υπολογισμός Διαδρομής [26]

```
import CoreLocation
let geocoder = CLGeocoder()
geocoder.geocodeAddressString("Thessaloniki, Greece") { placemarks, error in
    if let location = placemarks?.first?.location {
        print(location.coordinate)
    }
}
```

Κώδικας 3: Γεωκωδικοποίηση [26]

```
let location = CLLocation(latitude: 37.9838, longitude: 23.7275)
geocoder.reverseGeocodeLocation(location) { placemarks, error in
    print(placemarks?.first?.name)
```

Κώδικας 4: Αντίστροφη Γεωκωδικοποίηση [26]

Το MapKit είναι ενσωματωμένο framework μέσα στο Xcode. Είναι ένα ισχυρό και σχετικά εύκολο στη χρήση εργαλείο για χαρτογραφικές δυνατότητες στις iOS εφαρμογές και ενσωματώνεται άψογα τόσο με UIKit όσο και με SwiftUI.

2.2.4 UIKit

Το UIKit είναι το βασικό framework διεπαφής χρήστη (UI framework) που παρέχει η Apple για την ανάπτυξη εφαρμογών σε iOS, iPadOS, watchOS και tvOS. Περιλαμβάνει ένα σύνολο κλάσεων, πρωτοκόλλων και APIs που επιτρέπουν τη δημιουργία και διαχείριση της διεπαφής χρήστη, της αλληλεπίδρασης, της πλοήγησης, του animation και της διαχείρισης touch gestures. Το UIKit προσφέρει έτοιμα UI components όπως κουμπιά, ετικέτες, εικόνες, πίνακες, συλλογές κ.λπ.. Μπορεί να διαχειρίζεται τη ζωή της εφαρμογής (life cycle), γεγονότα αφής, συμβάντα πληκτρολογίου, multitasking και ειδοποιήσεις. Υποστηρίζει animation και μεταβάσεις, καθώς και προσαρμοσένες διαστάσεις (custom layout) σε διαφορετικά μεγέθη οθόνης. Παρέχει την δυνατότητα τόσο για προγραμματισμό όσο και για σχεδίαση του UI, μέσω κώδικα ή Interface Builder (Storyboard/XIB αρχεία). Βασικά στοιχεία του UIKit είναι το UIView και το UIViewController. Το UIView είναι η βασική μονάδα διεπαφής στο UIKit. Όλα τα οπτικά στοιχεία είναι υποκλάσεις του UIView (π.χ. UILabel, UIButton, UIImageView, UITextView). Το UIViewController είναι υπεύθυνο για την παρουσίαση του UI, τη διαχείριση γεγονότων και την επιχειρησιακή λογική. Στην ουσία διαχειρίζεται την εμφάνιση μίας οθόνης μέσα στην εφαρμογή. Τα κύρια στοιχεία του UIKit είναι τα εξής:

- UILabel

Χρησιμοποιείται για προβολή κειμένου

- UIButton

Ένα διαδραστικό κουμπί, με το οποίο επιτελούνται συγκεκριμένες λειτουργίες

- UITextField, UITextView

Χρησιμοποιείται για εισαγωγή κειμένου

- UIImageView

Χρησιμοποιείται για προβολή εικόνων

- UITableView

Πίνακας με λίστες δεδομένων

- UICollectionView

Πλέγμα δεδομένων με δυνατότητα layout customization (vertical & horizontal)

- UIScrollView

Χρησιμοποιείται για περιεχόμενο το οποίο θα είναι scrollable

- UIStackView

Είναι μία διάταξη με Views, που είναι stack-based. Κι εδώ υπάρχει η δυνατότητα για layout customization.

Το UIKit παρέχει αναγνώριση κινήσεων (gestures) όπως:

- Tap
- Swipe
- Pinch
- Long Press
- Pan

Προσφέρει συστήματα πλοήγησης όπως:

- UINavigationController:

Push-based πλοήγηση σε στοίβα (stack)

- UITabBarController:

Πλοήγηση μέσω tabs

- UISplitViewController:

Ιδανικό για iPad (master-detail interface)

- Modal presentation:

Παρουσίαση οθονών με fade/slide χωρίς push

Το UIKit προσφέρει δυναμική προσαρμογή διάταξης για όλες τις συσκευές με τη χρήση του Auto Layout και των Size Classes.

2.2.4.1 Auto Layout

Το Auto Layout είναι ένα σύστημα που υπολογίζει δυναμικά το μέγεθος αλλά και την θέση των στοιχείων σε ένα User Interface, με βάση κάποια ορισμένα constraints. Αυτό επιτρέπει στα interfaces να είναι responsive είτε η αλλαγή συμβαίνει εσωτερικά (π.χ. μεταβολή στο μέγεθος και στην εμφάνιση ενός κειμένου από κάποιο request που ίσως απέτυχε) είτε εξωτερικά (π.χ. μεταβολή στην εμφάνιση μίας εικόνας με βάση την αλληλεπίδραση με τον χρήστη). Βασικές έννοιες του Auto Layout είναι τα constraints και η δυναμική αλληλεπίδραση. Αρχικά τα constraints είναι κανόνες που ορίζουν την σχέση μεταξύ των στοιχείων μίας οθόνης (π.χ. ένα UILabel θα έχει πάντα απόσταση ίση με 8 από ένα UIImageView). Η δυναμική αλληλεπίδραση εξασφαλίζει ότι το Auto Layout αυτόματα θα μεταβάλει τόσο την θέση όσο και το μέγεθος μεταξύ των στοιχείων μίας οθόνης όταν κάποιες συνθήκες μεταβάλλονται (π.χ. χρήστης με προβλήματα όρασης μεταβάλει το μέγεθος της γραμματοσειράς της εφαρμογής). Το Auto Layout μπορεί να χειρίζεται τόσο εσωτερικές όσο και εξωτερικές μεταβολές. Οι εξωτερικές μεταβολές συμβαίνουν όταν το μέγεθος της οθόνης μεταβάλλεται, π.χ. ο χρήστης μεταβάλει το μέγεθος ενός παραθύρου (επεκτηνόμενο UIView), ο χρήστης κάνει rotate την οθόνη του, ή χρειάζεται να εμφανίζονται και να κρύβονται στοιχεία, ανάλογα με το μέγεθος της οθόνης της συσκευής. Οι εσωτερικές μεταβολές συμβαίνουν όταν το μέγεθος ορισμένων στοιχείων της οθόνης μεταβάλλεται, π.χ. μεταβολή στο περιεχόμενο της οθόνης (μεταβολή στο μέγεθος ενός κειμένου ή μία εικόνας). Η χρήση του Auto Layout αντικατέστηκε το Programmatic Layout. Σε αυτό το είδος layout ο προγραμματιστής καθορίζει προγραμματίστηκα για κάθε στοιχείο της οθόνης το ακριβές μέγεθος και την ακριβή θέση του. Το Auto Layout προτιμάται, καθώς είναι πιο έυκολο στην κατανόηση και την συντήρηση, ιδιαίτερα για responsive interfaces.

2.2.5. SwiftUI

Το SwiftUI είναι ένα declarative UI framework που παρουσιάστηκε από την Apple στο WWDC 2019. Έχει σχεδιαστεί ώστε να προσφέρει έναν πιο απλό, καθαρό και επαναχρησιμοποιήσιμο τρόπο κατασκευής της διεπαφής χρήστη (UI) σε εφαρμογές για iOS, iPadOS, macOS, watchOS και tvOS, χρησιμοποιώντας τη γλώσσα Swift. Σε αντίθεση με το κλασικό UIKit (το οποίο είναι imperative), το

SwiftUI ακολουθεί δηλωτική προσέγγιση (declarative syntax). Αντί να λέμε "πώς" να φτιάξουμε το UI βήμα-βήμα, δηλώνουμε τι θέλουμε να φαίνεται και αφήνουμε το framework να το διαχειριστεί. Βασικά χαρακτηριστικά του SwiftUI είναι τα εξής:

• Declarative Syntax

Δήλωση του UI και της συμπεριφοράς του, με σαφή και σύντομο τρόπο

• Cross-platform

Συγγραφή ίδιου κώδικα για όλες τις πλατφόρμες του οικοσυστήματος της Apple

• State-driven UI

Κάθε μεταβολή στην κατάσταση (state) προκαλεί ανανέωση του UI

• Built-in animation

Απλή δημιουργία animation χρησιμοποιώντας μία μόνο εντολή

• Preview σε πραγματικό χρόνο

Δυνατότητα προβολής του UI στο Xcode, παράλληλα με την συγγραφή του κώδικα

Βασικές έννοιες του SwiftUI είναι οι εξής:

• Views

Κάθε στοιχείο της διεπαφής στο SwiftUI είναι ένα View. Όλα τα Views είναι structs που υιοθετούν το

```
struct ContentView: View {
   var body: some View {
      Text(“Hello World!")
         .font(.title)
         .foregroundColor(.blue)
   }
```

View πρωτόκολλο και επιστρέφουν ένα άλλο View στο body.

Κώδικας 7: Παράδειγμα View, SwiftUI [24]

• State Management

Η SwiftUI είναι state-driven: Όταν αλλάζει μια μεταβλητή κατάστασης, το View ανανεώνεται αυτόματα. Ακολουθούν κάποια από τα βασικά states:

- @State: Κατάσταση που ανήκει στο ίδιο View.
- @Binding: Δέσμευση για να μεταδίδεται κατάσταση μεταξύ Views.
- @ObservedObject & @StateObject: Παρακολουθούν ObservableObjects για μεταβολές.

```
struct CounterView: View {
    @State private var counter = 0
    var body: some View {
        VStack {
            Text("Μέτρηση: \(counter)")
            Button("Αύξησε") {
                counter += 1
            }
        }
    }
```

- @Environment: Πρόσβαση σε shared global πληροφορίες.

Κώδικας 8: Παράδειγμα State, SwiftUI [24]

- Layout Systems

Τα Views τοποθετούνται με χρήση στοιχειοθετημένων containers όπως:

- VStack, HStack, ZStack

Στοίχιση κατακόρυφα, οριζόντια, και επικάλυψη

- Spacer

Προσθέτει "ευέλικτο" κενό χώρο

- GeometryReader

Απόκτηση διαστάσεων/θέσης View στο layout.

- Modifiers

```
Text("Hello")
    .font(.largeTitle)
    .foregroundColor(.red)
```

Κάθε View μπορεί να τροποποιηθεί με modifiers, τα οποία επιστρέφουν νέο View.

Κώδικας 9: Παράδειγμα Modifier, SwiftUI [24]

- Navigation

Η SwiftUI προσφέρει NavigationStack, από το iOS 16 και μετά. Ακόμη προσφέρει NavigationView (σε

```
NavigationStack {
   NavigationLink("Μετάβαση", destination: DetailView())
```

παλαιότερες εκδόσεις iOS) για πλοήγηση μεταξύ των Views.

Κώδικας 10: Παράδειγμα Navigation, SwiftUI [24]

- Forms & Controls

Η SwiftUI προσφέρει πλήρη υποστήριξη από controls:

- TextField, SecureField
- Toggle
- Slider
- Picker
- DatePicker
- Form: Εμφάνιση σε iOS-style form layout.
- Animation

Τα animations στο SwiftUI είναι απλά και περιεκτικά

Το SwiftUI προσφέρει σημαντικά πλεονεκτήματα για την ανάπτυξη εφαρμογών στο οικοσύστημα της Apple, όπως η απλότητα στη σύνταξη, η ταχύτερη ανάπτυξη UI μέσω της δηλωτικής προσέγγισης, καθώς και η δυνατότητα προβολής σε πραγματικό χρόνο (live previews) μέσα από το Xcode. Επιπλέον, ενσωματώνεται πλήρως με τη γλώσσα Swift, ενισχύοντας την ασφάλεια τύπων (type-safety) και επιτρέποντας την υιοθέτηση σύγχρονων αρχιτεκτονικών όπως το MVVM. Ωστόσο, το SwiftUI παρουσιάζει και ορισμένους περιορισμούς, όπως η περιορισμένη υποστήριξη σε παλαιότερες εκδόσεις του iOS (διαθέσιμο μόνο από iOS 13 και άνω), οι λειτουργικές ελλείψεις σε σύγκριση με το UIKit (ειδικά σε πιο σύνθετες διεπαφές ή custom συμπεριφορές), καθώς και η αστάθεια ή ασυμβατότητα μεταξύ διαφορετικών εκδόσεων του SwiftUI, η οποία μπορεί να δυσκολέψει τη συντήρηση μακροχρόνιων έργων. Παρ' όλα αυτά, το SwiftUI αποτελεί τη στρατηγική κατεύθυνση της Apple και

ένα ιδιαίτερα ισχυρό εργαλείο για μελλοντικές εφαρμογές. Το SwiftUI φέρνει μια νέα εποχή στην ανάπτυξη εφαρμογών για το οικοσύστημα της Apple. Η δηλωτική προσέγγιση, η αλληλεπίδραση με το state, η δυνατότητα δημιουργίας επαναχρησιμοποιούμενων components και η υποστήριξη όλων των Apple platforms, το καθιστούν ένα από τα πιο ισχυρά εργαλεία UI ανάπτυξης. Παρόλο που σε πιο σύνθετα σενάρια ή εφαρμογές legacy μπορεί να απαιτείται UIKit, το SwiftUI είναι το μέλλον για νέες εφαρμογές και υιοθετείται όλο και περισσότερο από τους επαγγελματίες developers.

2.3 Web APIs

2.3.1 Εισαγωγή

Το API (Application Program Interface) αποτελεί ένα σύνολο από κανόνες, πρωτόκολλα και εργασίες τα οποία παρέχουν την δυνατότητα σε διαφορετικά προγράμματα λογισμικά να επικοινωνούν μεταξύ τους. Πιό απλά ένα API αποτελεί στην ουσία τον ενδιάμεσο, αφού επιτρέπει σε μία εφαρμογή να ανακτά δεδομένα ή και υπηρεσίες από κάποια άλλη. Με τον όρο Web API αναφερόμαστε στην ανάκτηση δεδομένων μία διαδιακτυακής εφαρμογής μέσω HTTP αιτημάτων. Βασικά πλεονεκτήματα των APIs είναι τα εξής:

- Καθιστούν απλή την επαναχρησιμοποίηση λειτουργιών
- Επιτρέπουν την σύνδεση μεταξύ συστημάτων
- Καθιστούν τις εφαρμογές ευέλικτες

2.3.2 Στάδια ανάπτυξης ενός API

1. Καθορισμός σκοπού και απαιτήσεων

Στο στάδιο αυτό πρέπει να καθοριστεί το πρόβλημα το οποίο θα επιλύσει το API, αλλά και ποιοί θα είναι οι χρήστες του(developers, εφαρμογές, υπηρεσίες).

2. Σχεδιασμός

Στο στάδιο αυτό γίνονται πλάνα που αφορούν τις λειτουργίες του API. Καθορίζονται οι κανόνες επικοινωνίας, εξατάζονται οι πόροι αλλά και η ασφάλεια του. Καθορίζονται οι τεχνολογίες που θα χρησιμοποιηθούν και οι οποίες θα καθορίσουν τελικά και την αρχιτεκτονική του.

3. Υλοποίηση

Στο στάδιο αυτό, εφόσον πληρούνται οι προϋποθέσεις που έχουν προηγουμένως οριστεί, θα γίνει η υλοποίηση του API. Είναι το στάδιο κατά το οποίο θα συνταχθεί ο backend κώδικας που θα εξυπηρετεί τα αιτήματα του API.

4. Τεκμηρίωση

Στο στάδιο αυτό η ομάδα των προγραμματιστών που αναπτύσσει το API θα συντάξει έναν πλήρη οδηγό χρήσης που θα περιλαμβάνει πλήρη περιγραφή των endpoints, των παραμέτρων και των απαντήσεων. Εργαλεία όπως είναι το Swagger θα βοηθήσουν στην κατανόηση.

5. Δοκιμές

Στο στάδιο αυτό η ομάδα των προγραμματιστών θα προχωρήσει σε ενδελεχή έλεγχο του API για πιθανά λάθη που θα οδηγήσουν σε ανεπιθύμητη συμπεριφορά. Ακόμη θα ελεχγθεί η ασφάλεια και η απόδοση του API.

6. Δημοσίευση και Έκθεση

Στο στάδιο αυτό το API θα γίνει διαθέσιμο σε διακομιστές ή σε cloud και θα δοθεί πρόσβαση σε εξουσιοδοτημένους χρήστες.

7. Συντήρηση και Βελτιώσεις

Στο στάδιο αυτό θα γίνει παρακολούθηση της χρήσης και θα καταγραφούν οι αποδόσεις του API. Ακόμη θα υπάρξουν οι κατάλληλες αναβαθμίσεις και θα υποστηριχθούν νέες τεχνολογίες.

8. Απόσυρση

Στο στάδιο αυτό είναι πολύ σημαντικό να καταρτιστεί ένα σχέδιο αντικατάστασης του API. Είναι πολύ σημαντικό να μην εγκαταλειφθούν οι χρήστες που ήδη το χρησιμοποιούν, αφού ο απότομος τερματισμός ενός API θα είχε ως αποτέλεσμα την δημιουργία προβλημάτων σε κάθε εφαρμογή που το χρησιμοποιεί.

2.4 Architecture Patterns

Τα architecture patterns είναι προκαθορισμένα δομικά σχέδια που χρησιμοποιούνται για την οργάνωση του κώδικα με τρόπο που να είναι επανεκμεταλλεύσιμος, ευανάγνωστος και ευέλικτος. Αποτελούν κατευθυντήριες γραμμές για το πώς να δομηθεί μια εφαρμογή, τόσο σε υψηλό επίπεδο (π.χ. πώς επικοινωνούν τα modules), όσο και σε πιο χαμηλό επίπεδο (π.χ. πώς οργανώνονται οι κλάσεις). Δημοφιλή architecture patterns είναι τα εξής: MVC, MVVM, VIPER, Clean Architecture. Ο προγραμματιστής αποφασίζει για την χρήση του κατάλληλου architecture pattern, ανάλογα με την

πολυπλοκότητα της εφαρμογής που θα υλοποιήσει κάθε φορά. Για παράδειγμα για απλές και μικρές εφαρμογές μπορεί να χρησιμοποιηθεί το MVC, που είναι και το πιο απλό architecture pattern. Ένα από τα κύρια προβλήματα του MVC είναι πως υπάρχει η δυνατότητα να καταλήξει ο Controller, υπερβολικά "φορτωμένος", με αποτέλεσμα ο κώδικας να γίνεται πιο δύσκολος στην ανάγνωση και στην κατανόηση του. Στις περισσότερες εφαρμογές το architecture pattern είναι είτε το MVVM είτε ένας συνδυασμός MVVM και Clean Architecture. Για modular εφαρμογές με πολλές λειτουργίες, μπορεί να χρησιμοποιηθεί και το VIPER. HClean Architecture, σε συνδυασμό με το MVVM (Model View ViewModel), αποτελούν ένα από τα πιο δημοφιλοί παραδείγματα αρχιτεκτονικής που παρέχει οργάνωση και δομή στον κώδικα. Το MVVM προσφέρει ένα πρότυπο για τη διαχείριση του UI (User Interface) και για τη διασύνδεση με τα δεδομένα, με αυτό τον τρόπο αποφεύγεται η σύγχυση μεταξύ της λογικής της εφαρμογής και της παρουσίασης των δεδομένων της εφαρμογής. Η Clean Architecture διαχωρίζει τον κώδικα σε ξεχωριστά επίπεδα με διαφορετικές ευθύνες και εξασφαλίζει την επαναχρησιμοποίηση, τη διαχείριση των εξαρτήσεων και τη δομή του κώδικα. Ο συνδυασμός αυτών των δύο architecture patterns οδηγεί στην ανάπτυξη ευανάγνωστου κώδικα και καθιστά πιο εύκολη τόσο την συντήρηση του όσο και την αναβάθμιση του.

2.4.1 Clean Architecture

Η Clean Architecture είναι ένα αρχιτεκτονικό πρότυπο λογισμικού που προτάθηκε από τον Robert C. Martin (γνωστός και ως "Uncle Bob"). Στόχος της είναι να δημιουργεί καλά οργανωμένα, ανεξάρτητα, ευέλικτα και εύκολα στη συντήρηση συστήματα, με σαφή διαχωρισμό των ευθυνών και με χαλαρή σύζευξη (loose coupling) μεταξύ των επιπέδων της εφαρμογής. Η Clean Architecture βασίζεται στην επίπεδη (layered) αρχιτεκτονική, όπου τα layers χωρίζονται με σαφείς ρόλους και οι εξαρτήσεις κατευθύνονται προς τον πυρήνα της εφαρμογής (Dependency Rule).

Η Clean Architecture οργανώνεται σε ομόκεντρους κύκλους (rings), όπου το κάθε εσωτερικό layer δεν γνωρίζει τίποτα για τα εξωτερικά:

- Frameworks & UI (UIKit, SwiftUI)
- Interface Adapters (Presenters, ViewModels, Controllers)
- Application Business Rules (Use Cases / Interactors)
- Enterprise Business Rules (Entities, core models, domain logic)

Τα κύρια components της Clean Architecture είναι τα εξής:

- Entities

Core Business rules. Είναι ανεξάρτητα απο API, Databases και UI.

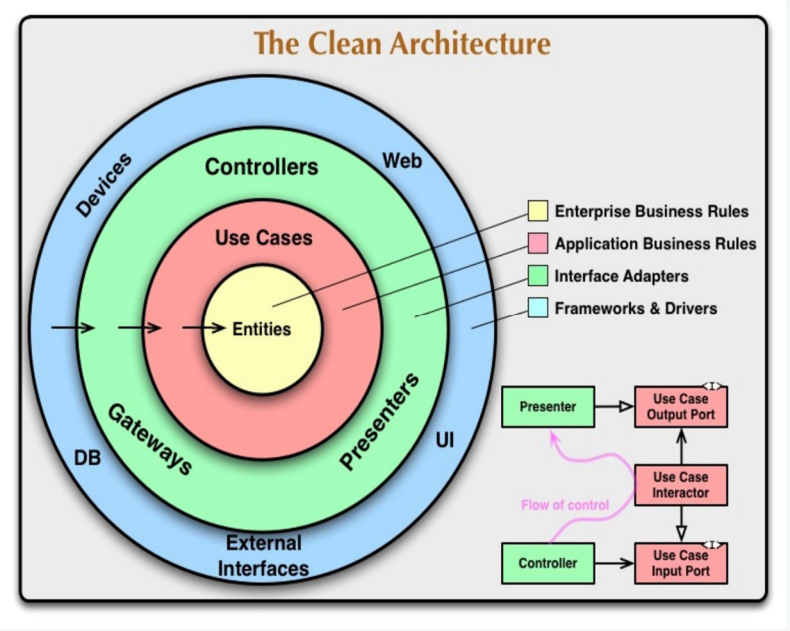
• Use Cases

Τα Use Cases εφαρμόζουν τους κανόνες τηε εφαρμογής (application logic).

• Interface Adapters

Μετατρέπουν τα δεδομένα από και προς μορφές που χρησιμοποιούν τα Use Cases (π.χ. ViewModels, Controllers).

• Frameworks & Drivers
Εξωτερικά συστήματα όπως UI, API clients, Databases, iOS SDK.



Εικόνα 2.1: Clean Architecture [15]

2.4.1.1 Frameworks & Drivers

Στο επίπεδο αυτό περιλαμβάνονται οι εξωτερικές εξαρτήσεις που χρησιμοποιεί η εφαρμογή. Μπορεί να είναι λογισμικά ή εργαλεία όπως κάποια βάση δεδομένων ή το UI της εφαρμογής. Αυτά τα στοιχεία είναι αποκομένα από τον πυρήνα της εφαρμογής, για να εξασφαλίσουμε ότι σε περίπτωση αλλαγή τους δεν θα προκύψει κάποιο ανεπιθύμητο πρόβλημα. Έτσι προσδίδει σταθερότητα στην εφαρμογή.

2.4.1.2 Interface Adapters

Οι Interface Adapters αποτελούν ένα ενδιάμεσο επίπεδο στην Clean Architecture που μετατρέπει τα δεδομένα μεταξύ των εσωτερικών επιπέδων (Use Cases, Entities) και των εξωτερικών συστημάτων (όπως Βάσεις Δεδομένων, Web APIs, UI frameworks κ.λπ.). Ο σκοπός τους είναι να προσαρμόζουν ή να μεταφράζουν τα δεδομένα από και προς μορφές που είναι καταλληλότερες για:

- Την επιχειρηματική λογική της εφαρμογής (Entities, Use Cases)
- Τα εξωτερικά συστήματα (UI, Database, API)

Στο επίπεδο αυτό περιλαμβάνονται τα εξής:

- Presenters
- Views
- Controllers
- ViewModels

Αυτά μεταφέρουν τα δεδομένα μεταξύ UI και Use Cases.
Δηλαδή: Ο Controller/ViewModel λαμβάνει input από τον χρήστη και το προωθεί στον UseCase. Μετά, ο Presenter/ViewModel λαμβάνει τα δεδομένα που επιστρέφονται και τα εμφανίζει στη View.

2.4.1.3. Use Cases

Τα Use Cases αποτελούν το κεντρικό επίπεδο της εφαρμογής, και εκφράζουν τους κανόνες λειτουργίας της, δηλαδή τι μπορεί να κάνει ο χρήστης με το σύστημα. Είναι ένα μεσαίο layer ανάμεσα στα Entities (επιχειρηματική λογική) και στα Interface Adapters (UI, DB, API, κ.λπ.). Στο επίπεδο αυτό περιέχονται οι κανόνες λειτουργίας της εφαρμογής, που είναι συγκεκριμένοι για το εκάστοτε project. Ορίζει:

- Τι κάνει το σύστημα
- Πότε το κάνει
- Με ποιον τρόπο συνεργάζονται οι οντότητες (Entities) για να το κάνουν.

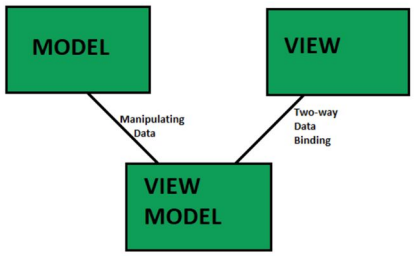
2.4.1.4. Entities

Τα Entities είναι το πιο εσωτερικό και σταθερό κομμάτι της Clean Architecture. Αντιπροσωπεύουν τη βασική επιχειρηματική λογική (enterprise-wide business rules). Είναι τα αντικείμενα που μοντελοποιούν την ουσία του προβλήματος που λύνει η εφαρμογή. Τα Entities, μπορούν να είναι δομές (structs) ή data structures + συναρτήσεις. Τα Entities δεν εξαρτώνται από frameworks, βάσεις

δεδομένων ή UI ενώ αντιπροσωπεύουν κανόνες και λογική που παραμένουν σταθεροί ακόμα και αν αλλάξει τελείως η τεχνολογική στοίβα.

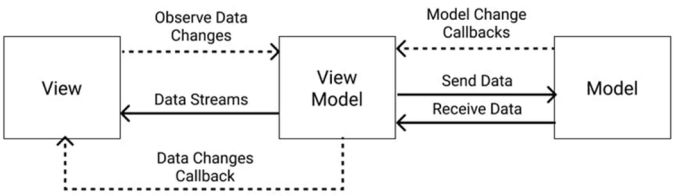
2.4.2 MVVM

Το MVVM ως architecture pattern, αποτελεί μία από τις πιο δημοφιλείς και αναγνωρίσιμες αρχιτεκτονικές. Στο μοντέλο αυτό ο κώδικας οργανώνεται, διαχωρίζοντας την εφαρμογή σε τρία κομμάτια. Πρώτα έχουμε το Model, μετά το View και τέλος του ViewModel.



Εικόνα 2.2: Σχήμα MVVM [16]

Το Model αντιπροσωπεύει τα δεδομένα και τη λογική του domain (π.χ. ένα User). Το View είναι πρακτικά το UI που βλέπει ο χρήστης (π.χ. UIView, UIViewController, ή SwiftUI View). Το ViewModel αποτελεί πρακτικά την "γέφυρα" ανάμεσα στο Model και στο View. Περιέχει την λογική παρουσίασης και τη διαχείριση κατάστασης. Στόχος του MVVM είναι η αποσύνδεση του UI από τη λογική παρουσίασης και τη διαχείριση δεδομένων. Το View δεν ξέρει τίποτα για το Model. Όλη η επικοινωνία περνάει μέσα από το ViewModel. Το MVVM αποτελεί ένα από τα πιό δημοφιλή architecture patterns στο iOS, αφού προσφέρει διαχωρισμό UI και λογικής, ευκολία στο testing, καθαρό, επεκτάσημο και ευανάγνωστο κώδικα ενώ συνδυάζεται και με SwiftUI.



Διάγραμμα 2.1: MVVM Implementation [16]

2.4.3 Clean Architecture & MVVM

Η χρήση του MVVM μαζί με την Clean Architecture είναι ιδιαίτερα δημοφιλής στη Swift και σε iOS development γενικότερα, επειδή τα δύο αυτά αρχιτεκτονικά πρότυπα συμπληρώνουν ιδανικά το ένα το άλλο. Αυτό συμβαίνει γιατί ο συνδυασμος τους καλύπτει ορισμένα ζητήματα που απορρέουν από το MVVM. Ο συνδυασμός MVVM και Clean Architecture προσφέρει:

- ισχυρή modular αρχιτεκτονική
- καθαρότητα στον κώδικα
- εύκολο testing
- επεκτασιμότητα και συντηρησιμότητα

Είναι μία από τις καλύτερες πρακτικές στο iOS development σήμερα, ιδιαίτερα σε μεσαίες και μεγάλες εφαρμογές.

2.5. Firebase

Το Firebase είναι μία πλατφόρμα της Google, που προσφέρει ένα ευρύ φάσμα εργαλείων για την ανάπτυξη και διαχείριση εφαρμογών για πολλές υποστηριζόμενες πλατφόρμες όπως το iOS. Στην εφαρμογή, για την οποία συντάσεται και το παρόν κείμενο, γίνεται χρήση της πλατφόρμας Firebase, για το backend κομμάτι της. Η πλατφόρμα έχει πολλές λειτουργίες. Ενδεικτικά κάποιες από αυτές είναι η αυθεντικοποίηση των χρηστών (authentication), η αποθήκευση των δεδομένων σε cloud, αναλύσεις σχετικά με την εφαρμογή (Analytics) και αναλύσεις σφαλμάτων σχετικά με την εφαρμογή (Crashlytics). Το Firebase δίνει την δυνατότητα στους προγραμματιστές να εκμεταλλευτούν τα πλεονεκτήματα του cloud με σκοπό την ανάπυτξη γρήγορων και αξιόπιστων εφαρμογών.

2.5.1 Firebase Analytics

Το Firebase Analytics (πλήρες όνομα: Google Analytics for Firebase) είναι μια ισχυρή και δωρεάν πλατφόρμα ανάλυσης χρήστη που παρέχεται από τη Google και ενσωματώνεται εύκολα σε εφαρμογές iOS, Android και web. Χρησιμοποιείται για την παρακολούθηση συμπεριφοράς χρηστών, ανάλυση αλληλεπιδράσεων και βελτίωση της εμπειρίας χρήστη σε εφαρμογές. Το Firebase Analytics επιτρέπει στους developers να:

- Καταγράφουν γεγονότα (events) όπως screen views, clicks, purchases, logins κλπ.
- Ομαδοποιούν τους χρήστες με βάση χαρακτηριστικά (user properties).

- Δημιουργούν audiences για marketing ή A/B testing.
- Παρακολουθούν funnels, retention, conversion rates κ.ά.

2.5.2 Firebase Crashlytics

Το Firebase Crashlytics είναι ένα real-time crash reporting tool που παρέχεται από τη Google μέσω του Firebase, και είναι ένα από τα πιο χρήσιμα εργαλεία για εντοπισμό, ανάλυση και επίλυση σφαλμάτων (crashes) σε mobile εφαρμογές (iOS και Android). Το Crashlytics καταγράφει αυτόματα:

- Σφάλματα (crashes)
- Non-fatal errors (π.χ. exceptions)
- Stack traces
- Πληροφορίες για τη συσκευή, την έκδοση της εφαρμογής, και τις συνθήκες σφάλματος.

Με λίγα λόγια: σου λέει πού, πότε, και γιατί έσκασε η εφαρμογή σου.

2.6 Git

Το Git είναι το πιο διαδεδομένο σύστημα ελέγχου έκδοσης λογισμικού στον κόσμο σήμερα. Δημιουργήθηκε το 2005 από τον Linus Torvalds, δημιουργό του Linux. Πρόκειται για ένα ώριμο, ενεργά συντηρούμενο λογισμικό ανοικτού κώδικα, που χρησιμοποιείται τόσο σε εμπορικά όσο και σε ανοικτά έργα λογισμικού. Σε αντίθεση με παλαιότερα συστήματα όπως το CVS ή το Subversion (SVN), το Git είναι κατανεμημένο (DVCS): κάθε προγραμματιστής έχει ένα πλήρες αντίγραφο του ιστορικού του έργου στο δικό του υπολογιστή. Τα βασικά πλεονεκτήματα του είναι τα εξής:

- Απόδοση

Το Git είναι γρήγορο σε λειτουργίες όπως commit, branching, merging και σύγκριση εκδόσεων. Δεν βασίζεται στα ονόματα αρχείων, αλλά στο πραγματικό τους περιεχόμενο, χρησιμοποιώντας τεχνικές όπως delta encoding και συμπίεση.

- Ασφάλεια:

Χρησιμοποιεί τον αλγόριθμο SHA-1 για να διασφαλίσει την ακεραιότητα του ιστορικού και των αλλαγών του κώδικα, προστατεύοντας από τυχαίες ή κακόβουλες παρεμβάσεις.

- Ευελιξία:

Υποστηρίζει διάφορες ροές εργασίας ανάπτυξης, λειτουργεί καλά σε μικρά και μεγάλα έργα και είναι συμβατό με υπάρχοντα πρωτόκολλα και εργαλεία. Η διαχείριση παρακλαδιών και ετικετών (tags) είναι πλήρως ενσωματωμένη.

Το Git θεωρείται δύσκολο στην εκμάθηση, ειδικά λόγω της ορολογίας του. Ωστόσο, προσφέρει τεράστια δύναμη και ευελιξία σε ομάδες ανάπτυξης. Παρόλο που είναι κατανεμημένο, μπορεί να χρησιμοποιηθεί και με κεντρικό αποθετήριο, διατηρώντας ταυτόχρονα την ανεξαρτησία κάθε προγραμματιστή. Συνολικά, το Git είναι το κορυφαίο εργαλείο ελέγχου έκδοσης για σύγχρονες ομάδες λογισμικού, συνδυάζοντας απόδοση, ασφάλεια, ευελιξία και ισχυρή κοινότητα υποστήριξης.

2.7 Ανακεφαλαίωση Περιγραφής Τεχνολογιών

Στο παρόν κεφάλαιο παρουσιάστηκαν αναλυτικά οι βασικές τεχνολογίες που αξιοποιήθηκαν για την ανάπτυξη της εφαρμογής. Αρχικά, εξετάστηκε το λειτουργικό σύστημα iOS, το οποίο αποτελεί τη βάση για την υλοποίηση εγγενών εφαρμογών στις φορητές συσκευές της Apple. Έγινε αναφορά στα βασικά χαρακτηριστικά του λειτουργικού, όπως η ασφάλεια, η σταθερότητα, η υψηλή απόδοση και η στενή ενοποίηση με το υλικό των συσκευών.

Στη συνέχεια, παρουσιάστηκαν τα κυριότερα frameworks για την ανάπτυξη εφαρμογών σε iOS, με ιδιαίτερη έμφαση στα UIKit και SwiftUI. Το UIKit αποτελεί το παραδοσιακό framework της Apple για την κατασκευή γραφικών διεπαφών, προσφέροντας πληθώρα εργαλείων για την υλοποίηση πολύπλοκων και δυναμικών περιβαλλόντων χρήστη. Το SwiftUI, από την άλλη πλευρά, εισάγει μία πιο σύγχρονη, δηλωτική προσέγγιση στον σχεδιασμό διεπαφών, μειώνοντας τον απαιτούμενο κώδικα και ενισχύοντας την ευχρηστία, ιδίως σε εφαρμογές που προορίζονται για πολλαπλές πλατφόρμες του Apple οικοσυστήματος. Παράλληλα, έγινε αναφορά και στο MapKit, το framework της Apple για την ενσωμάτωση χαρτογραφικών λειτουργιών και γεωγραφικής πληροφορίας στις εφαρμογές.

Ακολούθως, αναλύθηκε η έννοια των WebAPIs, τα οποία αποτέλεσαν κρίσιμο μηχανισμό για την επικοινωνία της εφαρμογής με απομακρυσμένους διακομιστές. Μέσω των WebAPIs, η εφαρμογή δύναται να ανταλλάσσει δεδομένα σε πραγματικό χρόνο, να καταναλώνει εξωτερικές υπηρεσίες και να εμπλουτίζει τη λειτουργικότητά της δυναμικά.

Η εργασία περιλαμβάνει επίσης την παρουσίαση των αρχιτεκτονικών προτύπων (Architecture Patterns) που εφαρμόστηκαν, με βασικότερο το MVVM (Model–View–ViewModel). Η χρήση του συγκεκριμένου προτύπου επιτρέπει την καθαρή διάκριση ανάμεσα στα δεδομένα, τη λογική παρουσίασης και τη διεπαφή χρήστη, συμβάλλοντας σε πιο ευέλικτη, επεκτάσιμη και ευκολότερα διαχειρίσιμη ανάπτυξη κώδικα.

Σημαντικό ρόλο στην ανάπτυξη έπαιξε και η πλατφόρμα Firebase, η οποία παρείχε βασικές λειτουργίες backend χωρίς την ανάγκη υλοποίησης και συντήρησης υποδομών. Πιο συγκεκριμένα, αξιοποιήθηκαν τα FirebaseAuth για τη διαχείριση αυθεντικοποίησης χρηστών, FirebaseFirestore για την αποθήκευση και ανάκτηση δεδομένων σε πραγματικό χρόνο, καθώς και το FirebaseCore για την αρχικοποίηση και παραμετροποίηση της σύνδεσης με τις υπηρεσίες της πλατφόρμας.

Τέλος, παρουσιάστηκε το Git, το οποίο χρησιμοποιήθηκε ως σύστημα ελέγχου εκδόσεων καθ' όλη τη διάρκεια ανάπτυξης της εφαρμογής. Μέσω του Git, διασφαλίστηκε η ιστορικότητα των αλλαγών στον πηγαίο κώδικα, η ευχέρεια συνεργασίας μεταξύ διαφορετικών προγραμματιστών και η σταδιακή υλοποίηση λειτουργιών με δυνατότητα εύκολης επαναφοράς σε προηγούμενες καταστάσεις.

Η κατανόηση και η σωστή αξιοποίηση των παραπάνω τεχνολογιών αποτέλεσαν θεμέλιο λίθο για τον επιτυχημένο σχεδιασμό και την υλοποίηση της εφαρμογής, ενισχύοντας τόσο τη λειτουργικότητά της όσο και την τεχνική της αρτιότητα.

Κεφάλαιο 3ο: Εργαλεία για την υλοποίηση της εφαρμογής

3.1 Xcode

Το Xcode είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών(IDE), που δημιουργήθηκε και διατήθεται από την Apple, για την δημιουργία εφαρμογών για πλατφόρμες όπως:

- iOS
- macOS
- watchOS
- tvOS

Διαθέτει τα απαραίτητα εργαλεία και προσφέρει τις κατάλληλες δυνατότητες σε έναν προγραμματιστή για την ανάπτυξη κώδικα. Ενδεικτικά τα εργαλεία που προσφέρει το Xcode, για την ανάπτυξη εφαρμογών για το οικοσύστημα της Apple είναι τα εξής:

- Interface Builder

Γραφικό εργαλείο που βοηθά στην σχεδίαση του UI, χωρίς να απαιτεί την συγγραφή κώδικα.

- Simulator

Εργαλείο που βοηθά στην προσομοίωση λειτουργίας των εφαρμογών σε εικονικές συσκευές, χωρίς να απαιτείται από τον developer να έχει στην κατοχή του κάποια φυσική συσκευή.

- Debugger & Profiler

Εργαλείο που βοηθά στην εύρεση σφαλμάτων και την βελτιστοποίηση της απόδοσης των εφαρμογών.

- Build System & Test Tools

Εργαλεία που βοηθάνε στην δημιουργία και την διανομή των εφαρμογών καθώς και στο testing.

3.2 Swift

3.2.1. Τι είναι και ποιά τα χαρακτηριστικά της

Η Swift είναι μία σύγχρονη γλώσσα προγραμματισμού υψηλού επιπέδου. Σχεδιάστηκε με σκοπό να αντικαταστήσει την Objective-C για να κάνει την ανάπτυξη εφαρμογών ευκολότερη, ταχύτερη αλλά και πιο αξιόπιστη. Χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών που απευθύνονται στο οικοσύστημα της Apple. Τα χαρακτηριστικά της Swift είναι τα εξής:

• Γρήγορη και Αποδοτική

Η Swift είναι κατασκευασμένη με τέτοιον τρόπο ώστε ο κώδικας της να εκτελείται πολύ γρήγορα, προσφέρωντας καλύτερες αποδόσεις σε σχέση με τον προκάτοχο της, την Objective-C.

• Ασφαλής

Η Swift περιλαμβάνει λειτουργίες που βοηθούν στην μείωση των λαθών, όπως: Type safety και Optionals.

• Καθαρή και απλή σύνταξη

Ο κώδικας ης Swift, είναι καθαρός, εύκολος τόσο στην ανάγνωση όσο και στην συντήρηση, γεγονός που την καθιστά ιδανική για νέους προγραμματιστές.

• Διαλειτουργικότητα με Objective-C

Η Swift έχει την δυνατότητα να συνεργάζεται με υπάρχοντα Objective-C frameworks επιτρέποντας την εύκολη μετάβαση παλαιών projects.

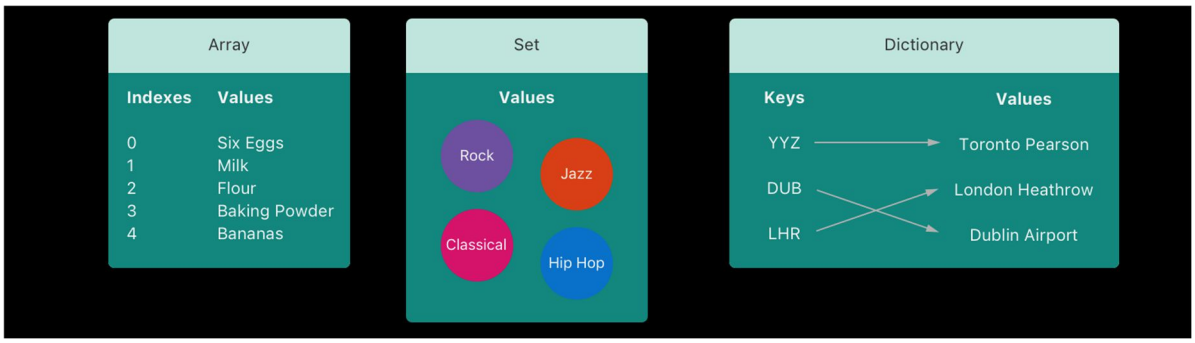
• Ανοιχτού κώδικα γλώσσα

Η Swift είναι μια ανοιχτού κώδικα γλώσσα προγραμματισμού η οποία βελτιώνεται συνεχώς τόσο από την ίδια την Apple όσο και από τους developers.

3.2.2 Τύποι δεδομένων

Η Swift προσφέρει πολλούς από τους ήδη γνωστούς τύπους δεδομένων, από άλλες προγενέστερες γλώσσες προγραμματισμού, συμπεριλαμβανομένων των Int, Double, Bool, String. Επίσης η Swift συμπεριλαμβάνει και έναν επιπλέον τύπο δεδομένων που είναι τα tuples. Χρησιμοποιούνται για την

αποθήκευση δεδομένων διαφορετικών τύπων, με την χρήση μίας δομής. Η Swift συμπεριλαμβάνει



Εικόνα 3.1: Collection Types [6]

ακόμη και 3 βασικούς τύπους collection types. Οι τύποι αυτοί είναι το Set, το Dictionary και το Array.

Η Swift παρέχει την δυνατότητα δήλωσης μίας μεταβλητής ως Optional. Πρακτικά αυτό σημαίνει ότι παρέχεται η δυνατότητα μη αρχικοποίησης της μεταβλητής με κάποια τιμή. Μία μεταβλητή που είναι

```
1. let shortForm: Int? = Int("42")
2. let longForm: Optional<Int> = Int("42")
```

τύπου Optional μπορεί να αποκτήσει μία τιμή ή όχι.

Κώδικας 12: Τρόποι δήλωσης Optional μεταβλητών [7]

Μία Optional μεταβλητή είναι πρακτικά ένα enum με δύο cases, την Optional.none και την Optional.some. Πρακτικά το πρώτο case ισοδυναμεί με nil και το δεύτερο case αποθηκεύει μία τιμή. Ακριβώς αυτό φαίνεται και στον παρακάτω κώδικα.

```
1. let number: Int? = Optional.some(42)
2. let noNumber: Int? = Optional.none
3. print(noNumber == nil)
4. //prints “true”
```

Κώδικας 13: Εξήγηση του enum για τις Optional μεταβλητές [7]

Ένα χαρακτηριστικό γνώρισμα της Swift σε σχέση με τις υπόλοιπες γλώσσες προγραμματισμού είναι πως επιτρέπει την χρήση Unicode κωδικών για την ονοματοδοσία μίας μεταβλητής. Επιτρέπεται ακόμη και η χρήση δεσμευμένων λέξεων, αν και δεν συνιστάται. Βασικοί κανόνες, όπως είναι η απαγόρευση χρήσης αριθμών στην αρχή του ονόματος, η χρήση κενών χαρακτήρων αλλά και μαθηματικών συμβόλων διατηρούνται.

```
let π = 3.14159
let 你好 = “你好世界”
let 🐶🐮 = "dogcow"
```

Κώδικας 14: Ασυνήθιστοι τρόποι δήλωσης μεταβλητών [7]

3.2.3 Χαρακτηριστικά της Swift

Τα κύρια χαρακτηριστικά της γλώσσας Swift είναι τα παρακάτω.

3.2.3.1 Multi-paradigm

Η Swift παρέχει την δυνατότητα υποστήριξης διάφορων προγραμματιστικών προσεγγίσεων, όπως:

1. Object Oriented

Η Swift επιτρέπει τον διαχωρισμό των οντοτήτων ενός προγράμματος σε αντικείμενα, τα οποία αποτελούν στιγμιότυπα κλάσεων. Οι κλάσεις ομαδοποιούν κοινά χαρακτηριστικά (μεταβλητές) και συμπεριφορές (μεθόδους) που μπορούν να επαναχρησιμοποιηθούν. Αυτή η προσέγγιση καθιστά τον κώδικα πιο οργανωμένο και εύκολα επεκτάσιμο. Ένα βασικό γνώρισμα της αντικειμενοστραφούς φιλοσοφίας είναι η ενθυλάκωση (encapsulation), δηλαδή η απόκρυψη των εσωτερικών λειτουργιών μιας κλάσης από τον εξωτερικό κόσμο. Ακόμη, μέσω της κληρονομικότητας (inheritance), μια υποκλάση μπορεί να αποκτήσει τις ιδιότητες μιας υπερκλάσης και να τις επεκτείνει. Τέλος, ο πολυμορφισμός (polymorphism) επιτρέπει σε αντικείμενα διαφορετικών τύπων να ανταποκρίνονται με διαφορετικό τρόπο στο ίδιο μήνυμα ή συνάρτηση, προσφέροντας ευελιξία στον σχεδιασμό.

2. Functional

Στον συναρτησιακό(functional) προγραμματισμό ο έλεγχος της ροής γίνεται μέσα από την σύνθεση συναρτήσεων και επικεντρώνεται στην επίλυση προβλημάτων μέσα από αλγοριθμικές προσεγγίσεις. Οι μεταβλητές σε αυτή την προσέγγιση είναι σταθερές(let) και δεν συμβαίνουν μεταβολές στην κατάσταση. Η προσέγγιση αυτή παρέχει μία άμεση αλγοριθμική επίλυση προβλημάτων στην οποία η συνάρτηση αποτελεί την βασική οντότητα εκτέλεσης κώδικα.

3. Imperative

Στον προστακτικό τρόπο προγραμματισμού δίνεται βάση στην εκτέλεση μιας αλληλουχίας εντολών που τροποποιούν την κατάσταση του προγράμματος. Σε αντίθεση με τη συναρτησιακή προσέγγιση, εδώ χρησιμοποιούνται μεταβλητά δεδομένα και διαδοχικές μεταβολές. Παρότι είναι πιο κατανοητός σε απλά σενάρια, μπορεί να παρουσιάσει δυσκολίες στην κλιμάκωση, ειδικά σε περιβάλλοντα που απαιτούν παράλληλη εκτέλεση.

4. Block-structed

Η Swift οργανώνει τις εντολές της μέσω καθορισμένων μπλοκ, τα οποία περικλείονται με αγκύλες({}). Κάθε μπλοκ εκτελείται εφόσον πληρείται μία συνθήκη (π.χ. if-else, switch), διατηρώντας έτσι τη

λογική ακεραιότητα και τη σειριακή ροή του προγράμματος. Αυτή η μορφή επιτρέπει την ομαδοποίηση εντολών και καθιστά τον έλεγχο ροής σαφέστερο.

3.2.3.2 Control Flow

Με τον όρο Control Flow αναφερόμαστε στους μηχανισμούς που παρέχουν την δυνατότητα στον προγραμματιστή να καθορίσει ποιό κομμάτι του κώδικα θα εκτελεστεί και πότε. Η Swift διαθέτει ένα πλούσιο σύνολο εργαλείων για ροή ελέγχου.

3.2.3.2.1 Συνθήκη ελέγχου if

Η συνθήκη ελέγχου "if", αποτελεί έναν από τα πιό συνήθη στοιχεία στον προγραμματισμό. Στην πιό απλή μορφή της η συνθήκη ελέγχου "if" έχει μόνο μία συνθήκη "if", την οποία ελέγχει. Αν η συνθήκη ισχύει τότε το block κώδικα που βρίσκεται ανάμεσα στις αγκύλες ({}), εκτελείται. Αν, παρόλα αυτά, η συνθήκη δεν ισχύει, τότε το block κώδικα δεν εκτελείται.

Υπάρχουν περιπτώσεις στις οποίες ο έλεγχος πρέπει να επεκταθεί σε περισσότερους ελέγχους. Αυτό επιτυγχάνεται με την χρήση του "else if", που παρέχει την δυνατότητα να εξεταστούν πολλαπλές πιθανές περιπτώσεις. Στην περίπτωση που κάνένας έλεγχος δεν είναι αληθής τότε θα χρησιμοποιηθεί μία τελική δήλωση "else", που θα εκτελέσει τις εντολές που περικλύονται στις αγκύλες({}) της.

```
var temperatureInFahrenheit = 30
if temperatureInFahrenheit <= 32 {
    print("It's very cold. Consider wearing a scarf.")
}
// Prints "It's very cold. Consider wearing a scarf."
```

Κώδικας 15: Συνθήκη ελέγχου if [9]

```
temperatureInFahrenheit = 40
if temperatureInFahrenheit <= 32 {
   print("It's very cold. Consider wearing a scarf.")
} else {
   print("It's not that cold. Wear a T-shirt.")
}
// Prints "It's not that cold. Wear a T-shirt."
```

```
let someCharacter: Character = "z"
switch someCharacter {
case "a":
   print("The first letter of the Latin alphabet")
case "z":
   print("The last letter of the Latin alphabet")
default:
   print("Some other character")
}
// Prints "The last letter of the Latin alphabet"
```

Κώδικας 16: Συνθήκη ελέγχου if-else, με else αποτέλεσμα συνθήκης [9]

```
temperatureInFahrenheit = 90
if temperatureInFahrenheit <= 32 {
   print("It's very cold. Consider wearing a scarf.")
} else if temperatureInFahrenheit >= 86 {
   print("It's really warm. Don't forget to wear sunscreen.")
} else {
   print("It's not that cold. Wear a T-shirt.")
}
```

Κώδικας 17: Συνθήκη ελέγχου if με πρόσθετους ελέγχους else if [9]

3.2.3.2.2 Συνθήκη ελέγχου Switch

Η συνθήκη ελέγχου switch μοιάζει με την συνθήκη “if… else if”. Βασικά διαφορά μεταξύ των δύο είναι ότι αντί να χρησιμοποιοιείται επαναλαμβανόμενα το “else if”, χρησιμοποιούνται οι περιπτώσεις “case”. Σε κάθε περίπτωση, αν η συνθήκη της είναι αληθής, τότε εκτελούνται οι εντολές της συνθήκης. Σε περίπτωση που καμία από τις περιπτώσεις δεν είναι αληθής θα εκτελείται η περίπτωση “default”.

Κώδικας 18: Συνθήκη ελέγχου switch [9]

3.2.3.3. Δομή επανάληψης for

Η δομή επανάληψης “for”, χρησιμοποιείται έτσι ώστε, όταν μία καθορισμένη συνθήκη είναι αληθής, να εκτελούνται οι κάποιες εντολές. Η επανάληψη των εντολών αυτών θα συνεχίζεται μέχρις ότου η συνθήκη που ήταν αληθής, πάψει να είναι.

```
let names = ["Anna", "Alex", "Brian", "Jack"]
for name in names {
    print("Hello, \(name)!")
}
// Hello, Anna!
// Hello, Alex!
// Hello, Brian!
// Hello, Jack!

let numberOfLegs = ["spider": 8, "ant": 6, "cat": 4]
for (animalName, legCount) in numberOfLegs {
    print("\(animalName)s have \(legCount) legs")
}
// cats have 4 legs
// ants have 6 legs
// spiders have 8 legs

for index in 1...5 {
    print("\(index) times 5 is \(index * 5)")
}
// 1 times 5 is 5
// 2 times 5 is 10
// 3 times 5 is 15
// 4 times 5 is 20
```

Κώδικας 19: Δομή επανάληψης for [9]

3.2.3.4 Functions

Η συνάρτηση είναι ένα ονοματισμένο μπλοκ κώδικα το οποίο εκτελεί ένα συγκεκριμένο έργο. Μπορεί να δεχθεί παραμέτρους εισόδου, να εκτελέσει λογικές πράξεις και να επιστρέψει μια τιμή εξόδου. Στην Swift οι συναρτήσεις είναι δομικά στοιχεία του κώδικα που παρέχουν την δυνατότητα ομαδοποίησης κάποιων εντολών, με σκοπό την επαναχρησιμοποίηση τους και την καλύτερη οργάνωση του κώδικα. Οι συναρτήσεις στην Swift έχουν τα εξής χαρακτηριστικά:

1. Πολλαπλές παράμετροι

2. Δεν επιστρέφουν πάντοτε τιμή (Void)

3. Μπορούν να έχουν προεπιλεγμένες τιμές παραμέτρων

4. Μπορούν να επιστρέφουν πάνω από έναν τύπους με την χρήση Tuples

5. Συναρτήσεις μέσα σε συναρτήσεις (Nested Functions)

```
func greet(person: String) -> String {
    let greeting = "Hello, " + person + "!"
    return greeting
}

func greetAgain(person: String) -> String {
    return "Hello again, " + person + "!"
}
print(greetAgain(person: "Anna"))
// Prints "Hello again, Anna!"

func sayHelloWorld() -> String {
    return "hello, world"
}
print(sayHelloWorld())
// Prints "hello, world"

func greet(person: String, alreadyGreeted: Bool) -> String {
    if alreadyGreeted {
        return greetAgain(person: person)
    } else {
        return greet(person: person)
    }
}
print(greet(person: "Tim", alreadyGreeted: true))
```

Κώδικας 20: Συναρτήσεις [10]

3.2.3.5 Structures and Classes

Τόσο οι δομές όσο και οι κλάσεις είναι δύο βασικοί τύποι για τη δημιουργία σύνθετων μονάδων δεδομένων. Χρησιμοποιούνται για την οργάνωση των δεδομένων και της λειτουργικότητας σε ένα ενιαίο, επαναχρησιμοποιήσιμο μπλοκ κώδικα. Παρόλο που έχουν πολλές ομοιότητες, υπάρχουν σημαντικές διαφορές στη συμπεριφορά και στη χρήση τους.

Η δομή είναι ένας τύπος αξίας δεδομένων (value type) που αντιγράφεται. Συνιστάται για μοντέλα δεδομένων που είναι απλά και δεν απαιτούν κληρονομικότητα.

Η κλάση είναι τύπος αναφοράς (reference type) και μοιράζεται την ίδια αναφορά στη μνήμη όταν αντιγράφεται. Είναι κατάλληλη για πιο σύνθετα αντικείμενα, όπου απαιτείται κληρονομικότητα και διαμοιρασμένη κατάσταση.

```
struct SomeStructure {
    // structure definition goes here
}
```

```
class SomeClass {
    // class definition goes here
}
```

Κώδικας 21: Παράδειγμα Δομής [8]

Κώδικας 22: Παράδειγμα Κλάσης [8]

Κοινά χαρακτηριστικά των κλάσεων και των δομών είναι τα εξής:

- Μπορούν να έχουν ιδιότητες (properties) και μεθόδους (methods)
- Μπορούν να επεκταθούν μέσω extensions
- Μπορούν να συμμορφώνονται με protocols
- Μπορούν να ορίσουν initializer(s)

Βασική διαφορά μεταξύ των κλάσεων και τον δομών είναι πως οι δομές αντιγράφονται κατά την ανάθεση τους σε μία νέα μεταβλητή, ενώ οι κλάσεις δημιουργούν μία αναφορά(reference) προς το αρχικό αντικείμενο. Πρακτικά οποιαδήποτε αλλαγή πραγματοποιηθεί σε μία κλάση επηρεάζει τόσο την κλάση την ίδια όσο και την αρχική και το αντίθετο.

```
struct Resolution {
    var width = 0
    var height = 0
}
class VideoMode {
    var resolution = Resolution()
    var interlaced = false
    var frameRate = 0.0
    var name: String?
}
```

Κώδικας 23: Σύκγριση μεταξύ Class και Struct [8]

3.2.3.6 Memory management

Η διαχείριση μνήμης (memory management) στη Swift είναι ένας βασικός μηχανισμός που εξασφαλίζει ότι η μνήμη που χρησιμοποιείται από τα αντικείμενα ενός προγράμματος δεσμεύεται και απελευθερώνεται σωστά. Στην Swift υπάρχουν κανόνες για την προώθηση ασφαλών πρακτικών memory management. Ακόμη, η Swift αναλαμβάνει την αυτόματη διαχείριση της μνήμης. Πρακτικά αυτό σημαίνει πως μειώνονται σε μεγάλο βαθμό λάθη που σχετίζονται με την μνήμη. Αυτό επιτυγχάνεται κυρίως μέσω του συστήματος Automatic Reference Counting (ARC).

Το ARC είναι ένας μηχανισμός διαχείρισης μνήμης που χρησιμοποιείται από τη Swift, ο οποίος παρακολουθεί πόσες ισχυρές αναφορές (strong references) υπάρχουν προς κάθε instance(αντικείμενο) κλάσης. Όταν ο αριθμός των αναφορών πέσει στο μηδέν, το αντικείμενο απελευθερώνεται από τη μνήμη.

3.3 Βιβλιοθήκες

Στον κόσμο του προγραμματισμού, είτε μιλάμε για προγραμματισμό web εφαρμογών, είτε για προγραμματισμό mobile εφαρμογών τόσο για iOS όσο και για Android, οι βιβλιοθήκες (libraries) είναι επαναχρησιμοποιήσιμα κομμάτια κώδικα τα οποία έχουν φτιαχτεί για να προσφέρουν έτοιμες λειτουργίες και εργαλεία, ώστε να επιταχύνουν την ανάπτυξη, να μειώσουν τα σφάλματα και να διευκολύνουν τον προγραμματιστή. Οι βιβλιοθήκες χωρίζονται σε δύο κατηγορίες:

- Static libraries (συνδέονται κατά τη μεταγλώττιση)
- Dynamic libraries / frameworks (φορτώνονται δυναμικά κατά την εκτέλεση)

Οι βιβλιοθήκες στην ανάπτυξη iOS εφαρμογών προσφέρουν:

- Επαναχρησιμοποίηση κώδικα: Χρησιμοποιούμε λειτουργίες που έχουν ήδη υλοποιηθεί.
- Λιγότερος χρόνος ανάπτυξης: Δεν χρειάζεται να "ανακαλύψουμε τον τροχό" κάθε φορά.
- Ευκολότερη συντήρηση: Λιγότερος custom κώδικας σημαίνει λιγότερα bugs.
- Καλύτερη απόδοση: Οι βιβλιοθήκες είναι συνήθως βελτιστοποιημένες.
- Ενσωμάτωση τρίτων υπηρεσιών: π.χ. Firebase, Stripe, Google Maps κ.ά.

Κάποιες από τις πιο δημοφιλείς βιβλιοθήκες που χρησιμοποιούνται στην ανάπτυξη iOS εφαρμογών είναι οι εξής:

- Kingfisher

Το Kingfisher χρησιμοποιείται για φόρτωση και caching εικόνων από URLs

- Alamofire

Το Alamofire είναι ένα network library, που προσφέρει την δυνατότητα στους προγραμματιστές να πραγματοποιούν HTTP requests με ευκολία.

- SDWebImage

Το SDWebImage είναι μία βιβλιοθήκη παρόμοια με το Kingfisher, συνεπώς επίσης χρησιμοποιείται για φόρτωση και caching εικόνων από URLs.

• RxSwift

Η RxSwift χρησιμοποιείται για διαχείριση ασύγχρονων ροών δεδομένων.

3.3.1 Βιβλιοθήκες της εφαρμογής

3.3.1.1 FirebaseFirestore

Το Firestore είναι μία ευέλικτη βάση δεδομένων που χρησιμοποείται για ανάπτυξη λογισμικού σε mobile πλατφόρμες, web πλατφόρμες και servers. Προσφέρεται από την Firebase και το Google Cloud. Η συγκεκριμένη βιβλιοθήκη παρέχει στους προγραμματιστές την δυνατότητα πρόσβασης σε NoSQL cloud database σε πραγματικό χρόνο. Είναι μία ιδανική επιλογή για δυναμικά δεδομένα όπως chat messages, λίστες προϊόντων κ.λ.π. Λειτουργεί με παρόμοιο τρόπο με την Firebase Realtime Database, καθώς έχει την δυνατότητα να διατηρεί τα δεδομένα που αποθηκεύονται σε αυτή συγχρονισμένα σε όλες τις εφαρμογές-πελάτες μέσω προγραμμάτων ακρόασης σε πραγματικό χρόνο, ενώ προσφέρει και υποστήριξη εκτός σύνδεσης για κινητά και web. Με αυτό τον τρόπο προσφέρει στους προγραμματιστές την δυνατότητα να δημιουργήσουν εφαρμογές που θα καλύπτουν τις απαιτούμενες ανάγκες ενώ παράλληλα θα μπορούν να λειτουργούν ανεξάρτητα από ενδεχόμενα προβλήματα σύνδεσης στο διαδίκτυο ή ακόμη και αποτυχία σύνδεσης σε αυτό. Μέσα από το Firestore ο προγραμματιστής έχει την δυνατότητα, με την χρήση κατάλληλων queries, να έχει πρόσβαση σε συγκεκριμένα αρχεία σε ένα collection ή σε όλα τα αρχεία ενός collection, με την προϋπόθεση ότι αυτά θα ταιριάζουν με τις παραμέτρους στο query του. Τα queries αυτά μπορούν να περιλαμβάνουν διάφορα φίλτρα ή ακόμα και να συνδυάζουν χρήση φίλτρων και ταξινόμησης. Όπως και στην Realtime Database, το Firestore χρησιμοποιεί συγχρονισμό των δεδομένων για να ενημερώνει τα δεδομένα του σε κάθε συσκευή με την οποία είναι συνδεδεμένο. Παράλληλα είναι κατασκευασμένο με τέτοιο τρόπο, ώστε να προσφέρει την δυνατότητα για αποτελεσματικά εφάπαξ queries. Μία πολύ σημαντική δυνατότητα του Firestore είναι ότι μπορεί να παρέχει υποστήριξη εκτός σύνδεσης, αφού μπορεί να αποθηκεύσει δεδομένα που χρησιμοποιεί η εκάστοτε εφαρμογή, προσφέροντας την δυνατότητα τόσο της παρουσίασης των δεδομένων όσο και της μεταβολής αυτών μέχρις ότου η εφαρμογή αποκαταστήσει την σύνδεση της στο διαδίκτυο. Σε περίπτωση που έχει πραγματοποιηθεί η οποιαδήποτε μεταβολή δεδομένων, όσο η εφαρμογή δεν ήταν συνδεδεμένη στο διαδίκτυο, τότε το Firestore έχει την δυνατότητα να συγχρονίσει αυτές τις τοπικές αλλαγές με το Firestore. Το Firestore είναι κατασκευασμένο με τέτοιο τρόπο ώστε να μπορεί να υποστηρίξει τόσο μία μικρή εφαρμογή όσο και κάποιες από τις μεγαλύτερες εφαρμογές του πλανήτη.

3.3.1.2 FirebaseAuth

Το FirebaseAuth χρησιμοποιείται για την αυθεντικοποίηση των χρηστών. Προσφέρει την δυνατότητα αυθεντικοποίησης μέσω προσωπικών διαπιστευτηρίων (email, password), μέσω χρήσης των λογαριασμών σε Google, Apple, Facebook, GitHub και μέσω τηλεφώνου. Στην τελευταία περίπτωση η αυθεντικοποίηση, επιτυγχάνεται μέσω αποστολής SMS. Η χρήση του κρίνεται ιδιαίτερη σημαντική αφού οι περισσότερες εφαρμογές χρειάζεται να γνωρίζουν ποιος τις χρησιμοποιεί. Με αυτό τον τρόπο οι εφαρμογές αποθηκεύουν τα δεδομένα των χρηστών στο cloud και μπορούν στην συνέχεια να προσφέρουν την ίδια εμπειρία χρήσης σε όλες τις συσκευές, στις οποίες είναι ο κάθε χρήστης συνδεδεμένος.

3.3.1.3 FirebaseCore

Το FirebaseCore αποτελεί την βασική βιβλιοθήκη για αρχικοποίη και παρακολούθηση της σύνδεσης με την Firebase (π.χ. μέσω FirebaseApp.configure()).

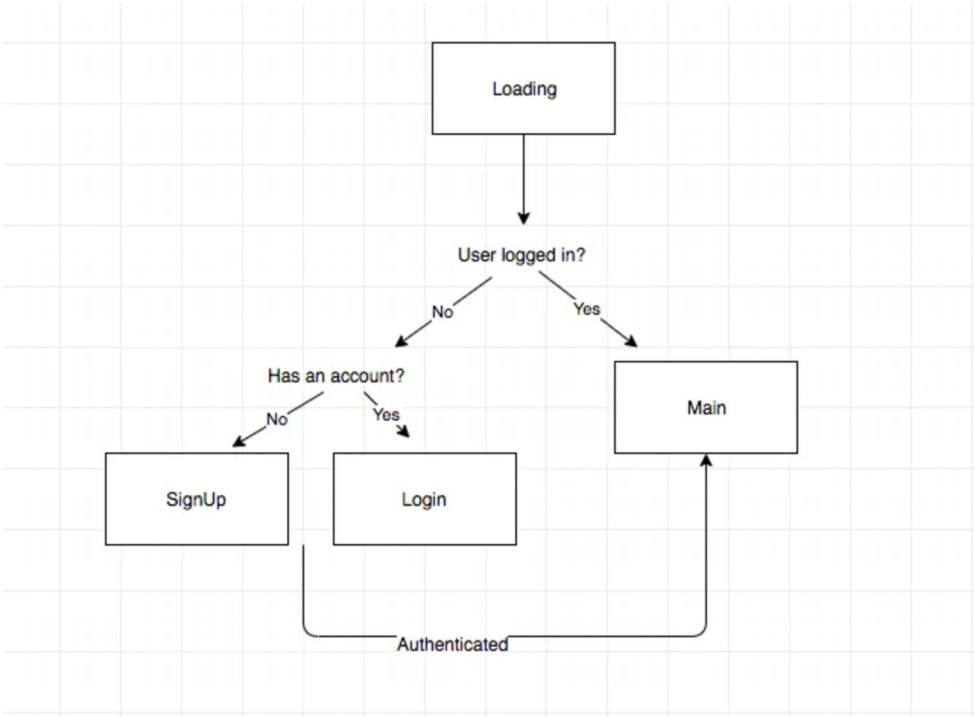
3.4 Ανακεφαλαίωση εργαλείων για την υλοποίηση της εφαρμογής

Στο παρόν κεφάλαιο παρουσιάστηκαν τα βασικά εργαλεία που αξιοποιήθηκαν για την ανάπτυξη της εφαρμογής. Ειδικότερα, έγινε αναφορά στο Xcode, το επίσημο περιβάλλον ανάπτυξης της Apple για iOS και macOS εφαρμογές. Το Xcode παρέχει μια ολοκληρωμένη εργαλειοθήκη που περιλαμβάνει επεξεργαστή κώδικα, σχεδιαστικό περιβάλλον διεπαφής χρήστη (Interface Builder), προσομοιωτές συσκευών (Simulators), δυνατότητες debugging και διαχείρισης εξαρτήσεων, διευκολύνοντας έτσι την οργάνωση και την ανάπτυξη της εφαρμογής σε όλες τις φάσεις της υλοποίησης.

Στη συνέχεια, περιγράφηκε η γλώσσα προγραμματισμού Swift, η οποία χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής. Η Swift αποτελεί μία σύγχρονη, ισχυρή και ασφαλή γλώσσα προγραμματισμού που έχει αναπτυχθεί από την Apple και ενδείκνυται για την υλοποίηση iOS εφαρμογών. Συνδυάζει την απλότητα στη σύνταξη με υψηλή απόδοση και δυνατότητες που ευνοούν τον καθαρό και επεκτάσιμο προγραμματισμό.

Τέλος, έγινε αναφορά στις βιβλιοθήκες της Swift που χρησιμοποιήθηκαν κατά την ανάπτυξη, τόσο τις εγγενείς (όπως Foundation, RxSwift) όσο και εξωτερικές(Kingfisher, Alamofire), οι οποίες συνέβαλαν στην επιτάχυνση της ανάπτυξης, την επαναχρησιμοποίηση κώδικα και την προσθήκη εξειδικευμένων λειτουργιών. Οι βιβλιοθήκες αυτές προσέφεραν λειτουργικότητες που σχετίζονται με την επικοινωνία με απομακρυσμένες υπηρεσίες, τον χειρισμό δεδομένων, καθώς και τον σχεδιασμό της διεπαφής χρήστη.

Η ορθή επιλογή και αξιοποίηση των ανωτέρω εργαλείων αποτέλεσε καθοριστικό παράγοντα για την επιτυχημένη υλοποίηση της εφαρμογής, τόσο από πλευράς απόδοσης όσο και από πλευράς ευχρηστίας και συντηρησιμότητας του κώδικα.



Διάγραμμα 3.1: Application Logic Using Firebase

Κεφάλαιο 4ο: Ανάπτυξη Εφαρμογής - Παρουσίαση Λειτουργίας Εφαρμογής

4.1 RiverView Project

Στο κεφάλαιο αυτό γίνεται μία αναλυτική περιγραφή της ανάπτυξης της εφαρμογής, που συνοδεύει αυτή την εργασία. Ακόμη θα παρουσιαστεί η δομή που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής και οι σχέσεις μεταξύ των αρχείων της εφαρμογής. Στο αρχικό φάκελο της εφαρμογής(root file), βρίσκονται χωρισμένα σε Groups (στο Xcode οι φάκελοι στους οποίους θα προσθέσουμε κάποια αρχεία ονομάζονται Groups) τα αρχεία από τα οποία αποτελείται η εφαρμογή και μέσω των οποίων υλοποιείται η λειτουργικότητα της.

Η αρχιτεκτονική που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι το MVVM (Model, View, ViewModel). Η χρήση της συγκεκριμένης αρχιτεκτονικής εξασφαλίζει ότι η επιχειρησιακή λογική (business logic), το Model δηλαδή, διαχωρίζεται από την εμφάνιση (View), μέσω του ViewModel. Το ViewModel διαχειρίζεται την επικοινωνία και τα δεδομένα μεταξύ των δύο, το οποίο έχει ως αποτελέσμα καθαρότερο και ευανάγνωστο κώδικα, ο οποίος θα είναι πιο εύκολος στην συντήρηση και κατανόηση του.

Νωρίτερα έγινε αναφορά στην δημιουργία Groups, τα οποία ομαδοποιούν και διαχωρίζουν τα αρχεία της εφαρμογής, ενώ παράλληλα εξυπηρετούν και την αρχιτεκτονική που επίσης αναφέρθηκε παραπάνω. Η δημιουργία των Groups αυτών, βοηθά στην πλοήγηση μέσα στο project, ενώ η ονοματοδοσία τους είναι ενδικτεική της λειτουργίας που επιτελούν μέσα στην εφαρμογή. Ενδεικτικά στο Group Scenes, περιέχονται τα αρχεία που θα χρησιμοποιηθούν από πολλές οθόνες.

4.1.1 Scene Delegate

Το αρχείο SceneDelegate αποτελεί βασικό συστατικό της αρχιτεκτονικής μιας iOS εφαρμογής από την έκδοση iOS 13 και έπειτα, στο πλαίσιο της υποστήριξης πολλαπλών σκηνών (multi-window support). Η κύρια ευθύνη του είναι η διαχείριση της σκηνής (UIScene), δηλαδή του κύκλου ζωής ενός παραθύρου ή διεπαφής χρήστη της εφαρμογής. Στον παραπάνω κώδικα, το SceneDelegate αξιοποιείται για την αρχικοποίηση και διαμόρφωση του κύριου παραθύρου της εφαρμογής. Συγκεκριμένα, στην scene(_:willConnectTo:options:) μέθοδο, το windowScene χρησιμοποιείται για να δημιουργηθεί ένα νέο παράθυρο (UIWindow), το οποίο στη συνέχεια συσχετίζεται με το κατάλληλο rootViewController, δηλαδή τον αρχικό View Controller της εφαρμογής – στην προκειμένη περίπτωση, έναν login controller μέσω του LogInModule. Με αυτόν τον τρόπο, η εφαρμογή παρακάμπτει τη χρήση του storyboard και φορτώνει το αρχικό περιβάλλον χρήστη με προγραμματιστικό τρόπο (programmatic UI). Οι υπόλοιπες

μέθοδοι του SceneDelegate παρέχουν σημεία εισόδου για να διαχειριστεί κανείς μεταβάσεις κατά τη διάρκεια ζωής της σκηνής (π.χ. όταν η εφαρμογή γίνεται ενεργή, όταν μεταβαίνει στο υπόβαθρο, κ.λπ.), προσφέροντας έτσι μεγαλύτερο έλεγχο και ευελιξία στη ροή της εφαρμογής.

4.1.2 App Delegate

Το αρχείο AppDelegate.swift αποτελεί θεμελιώδες κομμάτι της αρχιτεκτονικής μιας iOS εφαρμογής και έχει ως κύρια ευθύνη τη διαχείριση του κύκλου ζωής της εφαρμογής σε επίπεδο συστήματος. Η κλάση AppDelegate υιοθετεί το πρωτόκολλο UIApplicationDelegate και είναι το πρώτο σημείο εισόδου της εφαρμογής κατά την εκκίνηση. Στο συγκεκριμένο έργο, στην μέθοδο application(_:didFinishLaunchingWithOptions:), πραγματοποιείται η αρχικοποίηση του Firebase μέσω της εντολής FirebaseApp.configure(). Αυτό είναι απαραίτητο ώστε οι επιμέρους υπηρεσίες της πλατφόρμας Firebase (όπως FirebaseAuth και FirebaseFirestore) να είναι διαθέσιμες καθ' όλη τη διάρκεια λειτουργίας της εφαρμογής. Επιπλέον, η μέθοδος application(_:configurationForConnecting:options:) καθορίζει τη διαμόρφωση (UISceneConfiguration) για νέες σκηνές που δημιουργούνται, στοιχείο που ενισχύει τη διαχείριση πολλαπλών παραθύρων (scenes) στην εφαρμογή. Η AppDelegate συντονίζει επομένως τις ενέργειες που αφορούν σε παγκόσμιες ρυθμίσεις και υπηρεσίες, και λειτουργεί σε άμεση συνεργασία με το αρχείο SceneDelegate, το οποίο αναλαμβάνει την εμφάνιση και διαχείριση των επιμέρους παραθύρων της διεπαφής χρήστη.

4.2 Αρχεία ViewController

4.2.1 LoginViewController

Ο LoginViewController είναι στην ουσία η αρχική οθόνη της εφαρμογής. Είναι η πρώτη οθόνη που θα συναντήσει ένας χρήστης κατά την εκκίνηση της εφαρμογής. Μέσω αυτής θα μπορέσει να πλοηγηθεί στις υπόλοιπες οθόνες της εφαρμογής. Συγκεκριμένα μέσω αυτής της οθόνης μπορεί με το πάτημα του κουμπιού “Σύνδεση” να μεταφερθεί στην main οθόνη της εφαρμογής (MainViewController), όπου περιλαμβάνονται τα βασικά στοιχεία της εφαρμογής. Ακόμα μέσα από την οθόνη LoginViewController μπορεί να μεταβεί στην οθόνη του SignUp (SignUpViewController), όπου θα γίνει η εγγραφή του χρήστη.

4.2.2 SignupViewController

Ο SignupViewController είναι ο view controller, μέσω του οποίου ο χρήστης έχει την δυνατότητα να δημιουργήσει λογαριασμό. Η οθόνη διαθέτει κουμπί όπου μετά την συμπλήρωση των απαραίτητων στοιχείων ο χρήστης θα δημιουργήσει λογαριασμό και θα μεταβεί στην επόμενη οθόνη

(LoginViewController). Ο χρήστης μπορεί να μεταβεί σε αυτή την οθόνη και με την χρήση gesture, π.χ. σε περίπτωση που πλοηγήθηκε σε αυτή την οθόνη ενώ έχει ήδη δημιουργήσει λογαριασμό.

4.2.3 MainViewController

Στην οθόνη αυτή εμφανίζονται βασικές πληροφορίες για την εφαρμογή. Εμφανίζεται ένα carousel με κάποιες φωτογραφίες από διάφορα τοπία της περιοχής, ένα section που περιέχει δράσεις της ΚΟΙΝΣΕΠ κι άλλο ένα που περιέχει ανακοινώσεις της ΚΟΙΝΣΕΠ. Η οθόνη διαθέτει επίσης κι ένα κουμπί με το οποίο ο χρήστης αποσυνδέεται από την εφαρμογή και μεταφέρεται στην αρχική οθόνη της εφαρμογής (LoginViewController).

4.2.3.1 AnnouncementPopupViewController

Είναι η οθόνη που εμφανίζεται όταν ο χρήστης επιλέξει μία συγκεκριμένη ανακοίνωση. Η οθόνη αυτή καταλαμβάνει ορισμένο μέγεθος της οθόνης, χρησιμοποιώντας έναν custom UIPresentationController. Στην οθόνη αυτή εμφανίζονται όλες οι πληροφορίες σχετικά με μία ανακοίνωση της ΚΟΙΝΣΕΠ. Ο χρήστης μπορεί να μεταβεί στην οθόνη αυτή με το κατάλληλο gesture (tap) σε μία ανακοίνωση, ενώ μπορεί να μεταβεί πάλι πίσω στην οθόνη από την οποία πλοηγήθηκε σε αυτή (MainViewController), με ένα gesture (tap),πατώντας οπουδήποτε εκτός της οθόνης αυτής (AnnouncementPopupViewController).

4.2.4 LocationsViewController

Η συγκεκριμένη οθόνη παρουσιάζει όλα τα τοπία της περιοχής, στην οποία δραστηριοποιείται η ΚΟΙΝΣΕΠ. Χωρίζει το κάθε τοπίο σε υποκατηγορίες (Φράγματα, Τριχωνίδα, Αχελώος, Αρχαία Στράτος). Σε κάθε υποκατηγορία εμφανίζονται όλα τα τοπία που ανήκουν σε αυτήν.

4.2.4.1 LocationsPopupViewController

Είναι η οθόνη που εμφανίζεται όταν ο χρήστης κάποιο από τα τοπία. Η οθόνη αυτή καταλαμβάνει όλο το διαθέσιμο μέγεθος της οθόνης. Ο χρήστης μπορεί σε αυτή την οθόνη να δει όλες τις πληροφορίες σχετικά με το τοπίο που επέλεξε. Η μετάβαση του χρήστη στην οθόνη αυτή επιτυγχάνεται με την χρήση ενός συγκεκριμένου gesture (tap). Η μετάβαση του χρήστη, στην οθόνη που βρισκόταν πριν επιλέξει κάποιο από τα τοπία (LocationsViewController) επιτυγχάνεται με την χρήση gesture (swipe down).

4.2.5 MarketViewController

Στο συγκεκριμένο view controller, προβάλεται το eShop που ενσωματώνει η εφαρμογή. Στην οθόνη αυτή βρίσκεται ένα κουμπί, που είναι στην ουσία το καλάθι του eShop, και παρουσιάζονται όλα τα προϊόντα που μπορεί ο χρήστης να προσθέσει στο καλάθι του. Για κάθε προϊόν εμφανίζεται η φωτογραφία του, ο τίτλος του κι ένα κουμπί με το οποίο γίνεται η προσθήκη του στο καλάθι του eShop.

4.2.5.1 CartViewController

Είναι η οθόνη που εμφανίζεται όταν ο χρήστης επιλέξει το κουμπί του καλαθιού του eShop. Εδώ εμφανίζεται μία οθόνη που καταλαμβάνει ορισμένο μέρος της οθόνης. Αυτό το επιτυγχάνει χρησιμοποιώντας έναν custom UIPresentationController. Η οθόνη αυτή έχει ένα κουμπί, με το πάτημα του οποίου ο χρήστης μπορεί να πλοηγηθεί στην προηγούμενη οθόνη (MarketViewController). Αυτός είναι ο ένας τρόπος πλοήγησης από το CartViewController στο MarketViewController. Ο δεύτερος τρόπος είναι με ένα gesture (tap) εκτός των ορίων της οθόνης του CartViewController. Στην οθόνη αυτή θα βρούμε επίσης ένα view (UICollectionView), το οποίο όταν έχουν προστεθεί αντικείμενα στο καλάθι, θα περιέχει τα αντικείμενα αυτά. Σε αντίθετη περίπτωση το view αυτό θα είναι κενό. Η οθόνη περιλαμβάνει ένα κουμπί “Αποστολή Παραγγελίας”. Η κατάσταση (state) του κουμπιού μεταβάλλεται, αναλόγως με το αν έχει αντικείμενα ή όχι το καλάθι.

4.2.6 InfoViewController

Στην οθόνη αυτή εμφανίζονται όλα τα στοιχεία επικοινωνίας της ΚΟΙΝΣΕΠ.

4.3 Models

Στο πλαίσιο της αρχιτεκτονικής MVVM (Model-View-ViewModel), τα Models διαδραματίζουν έναν θεμελιώδη ρόλο, καθώς εκπροσωπούν τη δομή και τα δεδομένα της εφαρμογής. Η κύρια λειτουργία τους είναι η αποτύπωση των πραγματικών οντοτήτων της εφαρμογής με τρόπο αφηρημένο και ανεξάρτητο από την παρουσίαση ή τη λογική χειρισμού τους. Συνολικά, τα Models στη Swift

συμβάλλουν στην καθαρή διαχωρισμένη ευθύνη και στην ευκολία συντήρησης και επεκτασιμότητας

```
struct MarketItem {
    let title: String
    let subtitle: String
    let imageName: String

    init?(dictionary: [String: Any]) {
        guard let title = dictionary["title"] as? String,
                let subtitle = dictionary["price"] as? String,
                let imageName = dictionary["imageUrl"] as? String else { return nil }
        self.title = title
        self.subtitle = subtitle
        self.imageName = imageName
    }
}

struct CartItem {
    let title: String
    let subtitle: String
    let imageName: String
```

της εφαρμογής.

```
struct User {
    var email: String
     var password: String
```

Κώδικας 25: Μοντέλα MarketItem & CartItem

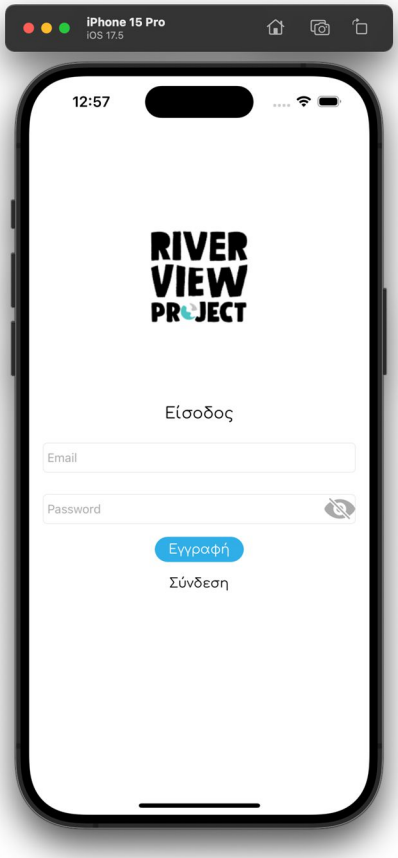
Κώδικας 26: Μοντέλο User

```
struct Announcement {
    let title: String
    let subtitle: String
    let imageName: String
    let text: String

    init?(dictionary: [String: Any]) {
        guard let title = dictionary["title"] as? String,
              let subtitle = dictionary["date"] as? String,
              let imageName = dictionary["imageUrl"] as? String,
              let text = dictionary["description"] as? String else { return nil }
        self.title = title
        self.subtitle = subtitle
        self.imageName = imageName
        self.text = text
    }
}

struct ImageBanner {
    let imageName: String

     init?(dictionary: [String: Any]) {
        guard let imageName = dictionary["imageUrl"] as? String else { return nil }
        self.imageName = imageName
    }
}

struct Event {
    let title: String
    let subtitle: String
    let imageName: String

    init?(dictionary: [String: Any]) {
        guard let title = dictionary["title"] as? String,
              let subtitle = dictionary["date"] as? String,
              let imageName = dictionary["imageUrl"] as? String else { return nil }
        self.title = title
        self.subtitle = subtitle
        self.imageName = imageName
    }
```

Κώδικας 27: Μοντέλα Announcement, ImageBanner, Event

4.4 Παρουσίαση Λειτουργία Εφαρμογής

4.4.1 Αρχική οθόνη Εφαρμογής



Εικόνα 4.1: Αρχική οθόνη (μη συνδεδεμένος χρήστης)

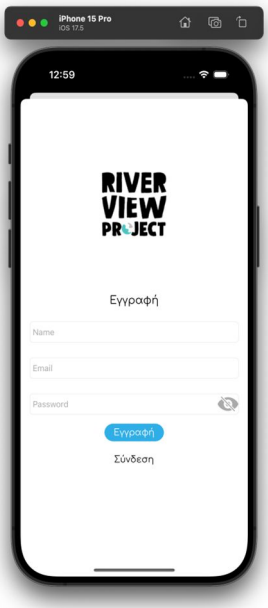
Η παραπάνω εικόνα δείχνει την πρώτη οθόνη που βλέπει ο χρήστης όταν ανοίξει την εφαρμογή. Το πρώτο που συναντά κανείς είναι το λογότυπο της εφαρμογής, ακολουθούμενο από τη λέξη “Είσοδος” και δύο πεδία κειμένου, το πεδίο email και το πεδίο password. Τα παραπάνω αποτελούν τυπικά στοιχεία μιας οθόνης σύνδεσης, που χρησιμοποιούνται για τον έλεγχο ταυτότητας του χρήστη. Το πεδίο κειμένου email επιτρέπει στον χρήστη να εισάγει την καταχωρισμένη διεύθυνση email του, η οποία λειτουργεί και ως το μοναδικό αναγνωριστικό για τον λογαριασμό. Το πεδίο κειμένου password χρησιμοποιείται για την εισαγωγή του κωδικού πρόσβασης του λογαριασμού. Είναι σύνηθες οι χαρακτήρες που εισάγονται σε αυτό να καλύπτονται για τη διατήρηση του απορρήτου του χρήστη.

Επόμενο είναι το κουμπί ‘Εγγραφή’. Το κουμπί αυτό είναι ένα βασικό μέρος της διεπαφής σύνδεσης, παρέχοντας στους χρήστες την δυνατότητα να δημιουργήσουν έναν νέο λογαριασμό, σε περίπτωση που δεν έχουν ήδη. Όταν πατηθεί, το κουμπί καθοδηγεί τον χρήστη σε μια φόρμα εγγραφής όπου μπορεί να εισάγει τα απαραίτητα στοιχεία όπως όνομα, email και κωδικό πρόσβασης. Με αυτόν τον τρόπο οι νέοι χρήστες μπορούν να δημιουργήσουν έναν λογαριασμό στο σύστημα ούτως ώστε να έχουν

πρόσβαση στις δυνατότητές του. Η ύπαρξη ενός κουμπιού εγγραφής διευκολύνει την ενσωμάτωση νέων χρηστών στην εφαρμογή, προσφέροντας έναν άμεσο και ευδιάκριτο τρόπο δημιουργίας λογαριασμού.

Ακριβώς κάτω από αυτό, το κουμπί με την ένδειξη "Σύνδεση" λειτουργεί ως σημείο εκκίνησης για τη διαδικασία αυθεντικοποίησης. Κατά την επιλογή του, ο χρήστης μεταφέρεται σε φόρμα εισαγωγής διαπιστευτηρίων, όπου καλείται να συμπληρώσει το email και τον κωδικό πρόσβασής του. Με την ενεργοποίηση του κουμπιού σύνδεσης, ενεργοποιείται μηχανισμός επαλήθευσης των δεδομένων που καταχωρίστηκαν, μέσω διασταύρωσής τους με τα εγγεγραμμένα στοιχεία στη βάση χρηστών (Firebase). Η επιτυχής επαλήθευση επιτρέπει την ασφαλή πρόσβαση του χρήστη στο περιβάλλον της εφαρμογής. Η βασική λειτουργία της διεπαφής σύνδεσης είναι ο έλεγχος ταυτότητας του χρήστη, επιτρέποντας μόνο σε εξουσιοδοτημένα άτομα την πρόσβαση στην εφαρμογή. Τα δύο βασικά πεδία εισαγωγής —email και password— παρέχουν το απαιτούμενο μέσο για την εισαγωγή των διαπιστευτηρίων του χρήστη. Όταν ο χρήστης συμπληρώσει τα στοιχεία του και πατήσει το κουμπί "Σύνδεση", η εφαρμογή ενεργοποιεί τη διαδικασία επαλήθευσης, συγκρίνοντας τα παρεχόμενα στοιχεία με εκείνα που είναι αποθηκευμένα στη βάση δεδομένων, όπως έχει υλοποιηθεί μέσω του Firebase Authentication. Εφόσον τα στοιχεία ταιριάζουν με κάποιον καταχωρημένο λογαριασμό, ο χρήστης αποκτά πρόσβαση στην εφαρμογή και μεταφέρεται στο κύριο περιβάλλον λειτουργίας της. Αντίθετα, σε περίπτωση ασυμφωνίας, εμφανίζεται κατάλληλο μήνυμα σφάλματος, ενημερώνοντας τον χρήστη για την αποτυχία σύνδεσης. Συνολικά, η διεπαφή σύνδεσης έχει σχεδιαστεί ώστε να είναι απλή, κατανοητή και ασφαλής, εξασφαλίζοντας μια θετική εμπειρία χρήσης και διευκολύνοντας την άμεση και προστατευμένη είσοδο στην εφαρμογή.

4.4.2 Οθόνη Εγγραφής Χρήστη



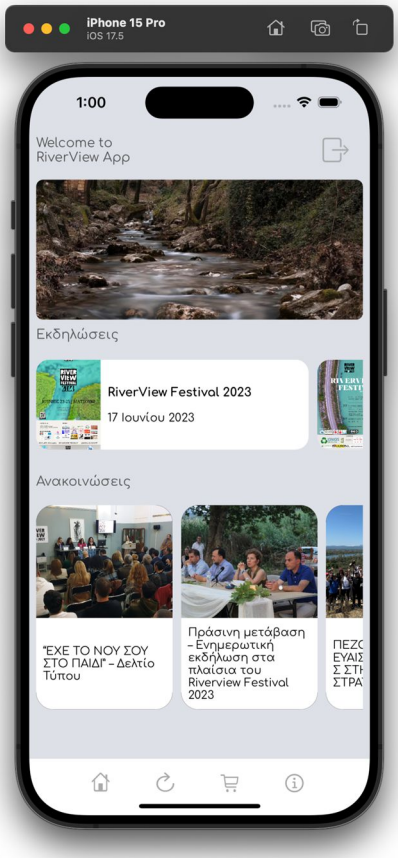
Εικόνα 4.2: Οθόνη Εγγραφής Χρήστη

Στην παραπάνω εικόνα παρουσιάζεται η διαδικασία δημιουργίας νέου λογαριασμού. Όπως και στη διεπαφή σύνδεσης, έτσι και η οθόνη εγγραφής περιλαμβάνει τα βασικά πεδία προς εισαγωγή, email και password. Επιπλέον περιλαμβάνει το πεδίο Name. Τα στοιχεία αυτά καλούνται να συμπληρώσουν οι νέοι χρήστες, ώστε να δημιουργήσουν το προσωπικό τους προφίλ στην εφαρμογή. Το email λειτουργεί ως μοναδικός αναγνωριστικός παράγοντας για τον λογαριασμό, ενώ ο κωδικός πρόσβασης καθορίζεται από τον χρήστη για λόγους ασφάλειας.

Το βασικό κουμπί ενέργειας της διεπαφής φέρει την ένδειξη "Εγγραφή" και ενεργοποιεί τη διαδικασία δημιουργίας λογαριασμού. Όταν ο χρήστης εισάγει τα απαιτούμενα στοιχεία και πατήσει το κουμπί, η εφαρμογή ελέγχει την εγκυρότητα των παρεχόμενων πληροφοριών, όπως αν το email είναι ήδη καταχωρημένο ή αν ο κωδικός πληροί τα προκαθορισμένα πρότυπα ασφαλείας. Εφόσον οι έλεγχοι ολοκληρωθούν επιτυχώς, δημιουργείται νέος λογαριασμός.

Η διεπαφή εγγραφής αποτελεί κρίσιμο κομμάτι της διαδικασίας απόκτησης νέων χρηστών, προσφέροντας μια απλή, κατανοητή και προσβάσιμη εμπειρία. Με τον καθαρό σχεδιασμό και την ευκολία χρήσης της, η διεπαφή αυτή ενισχύει τη διάθεση των επισκεπτών να εγγραφούν και να αλληλεπιδράσουν με την πλατφόρμα, συμβάλλοντας στην ενίσχυση της χρήσης και της απήχησης της εφαρμογής.

4.4.3 Main Οθόνη



Εικόνα 4.3: Κύρια Οθόνη Εφαρμογής

Στην εικόνα που παρουσιάζεται παραπάνω βλέπουμε την αρχική οθόνη που εμφανίζεται στον χρήστη αμέσως μετά την επιτυχή σύνδεσή του στο σύστημα. Ξεκινώντας από το επάνω μέρος της διεπαφής, στα δεξιά εντοπίζεται ένα κουμπί αποσύνδεσης. Με το πάτημά του, ο χρήστης εξέρχεται από τον λογαριασμό του και μεταφέρεται πίσω στην οθόνη εισόδου της εφαρμογής.

Ακριβώς πιο κάτω, βρίσκεται ένα carousel με εικόνες σχετικές με το έργο. Οι φωτογραφίες απεικονίζουν είτε δράσεις που έχουν πραγματοποιηθεί, είτε στιγμιότυπα από τον οργανισμό, είτε πρόσωπα που συμμετέχουν σε διάφορες εκδηλώσεις. Η εναλλαγή των εικόνων γίνεται αυτόματα με τη χρήση χρονοδιακόπτη που προγραμματίζει την προβολή κάθε φωτογραφίας για συγκεκριμένο χρονικό διάστημα. Παράλληλα, ο χρήστης μπορεί να αλλάξει εικόνα χειροκίνητα, με κίνηση σάρωσης (swipe) δεξιά ή αριστερά.

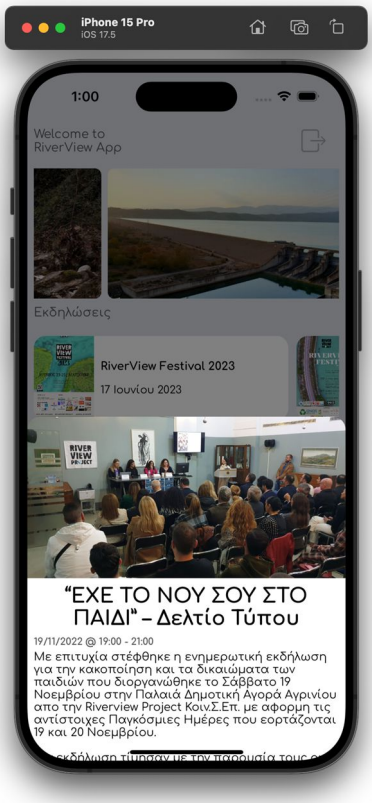
Πιο κάτω ακολουθεί ένα UICollectionView που παρουσιάζει συγκεντρωμένα τα events του οργανισμού. Τα γεγονότα αυτά εμφανίζονται με χρονολογική σειρά και περιλαμβάνουν τόσο περασμένες εκδηλώσεις όσο και μελλοντικές. Με αυτόν τον τρόπο, η ενότητα λειτουργεί ως ένα χρονολόγιο ή ημερολόγιο δράσεων για τον χρήστη. Στη συνέχεια της ίδιας οθόνης εντοπίζουμε ένα δεύτερο UICollectionView, το οποίο εμφανίζει τις επίσημες ανακοινώσεις της ΚΟΙΝΣΕΠ. Οι ανακοινώσεις ταξινομούνται επίσης με βάση την ημερομηνία δημοσίευσης. Ο χρήστης μπορεί να πατήσει σε οποιαδήποτε ανακοίνωση για να εμφανιστεί ένα αναδυόμενο παράθυρο (pop-up) με περισσότερες λεπτομέρειες – το οποίο θα εξετάσουμε στη συνέχεια. Υπάρχει επίσης η δυνατότητα οριζόντιας κύλισης για την προβολή επιπλέον ανακοινώσεων.

Στο κάτω μέρος της οθόνης παρουσιάζεται μια επιπλέον ενότητα, η οποία αφορά τις δράσεις που οργανώνει η ΚΟΙΝΣΕΠ. Όπως και με τις ανακοινώσεις, κάθε επιλεγμένη δράση εμφανίζεται σε νέο παράθυρο που υπερκαλύπτει τη βασική διεπαφή, παρουσιάζοντας σχετική φωτογραφία και περιγραφή της δράσης.

Τέλος, στο κάτω μέρος της οθόνης υπάρχει μια σταθερή γραμμή πλοήγησης (navigation bar), η οποία επιτρέπει τη γρήγορη μετάβαση σε βασικές ενότητες της εφαρμογής. Από δεξιά προς τα αριστερά, περιλαμβάνει:

- την αρχική οθόνη,
- την ενότητα του τουριστικού οδηγού,
- την οθόνη με το κατάστημα της εφαρμογής
- και μία τελευταία ενότητα με πληροφορίες για το έργο Riverview.

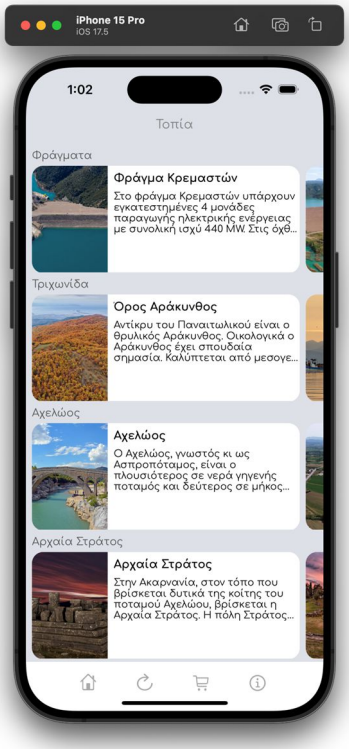
Ανάλογα με την οθόνη στην οποία βρίσκεται ο χρήστης, η αντίστοιχη επιλογή στη μπάρα τονίζεται χρωματικά, ώστε να διευκολύνεται ο προσανατολισμός και η εμπειρία πλοήγησης.



Εικόνα 4.4: Popup ανακοινώσεων

Στην παραπάνω εικόνα, βλέπουμε το παράθυρο που ανοίγει πατώντας σε μία ανακοίνωση. Με πάτημα εκτός του παραθύρου μπορούμε να το κλείσουμε, ενώ κάνοντας scroll μπορούμε να δούμε όλα τα περιεχόμενα του.

4.4.4 Οθόνη τοπίων

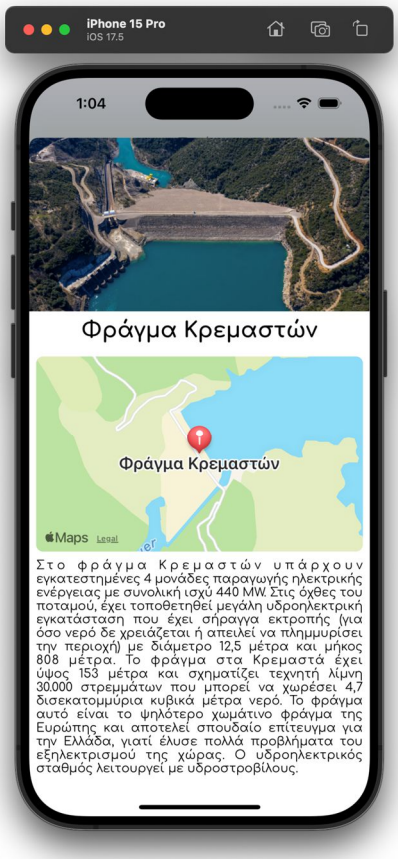


Εικόνα 4.5: Οθόνη τοπίων

Στην οθόνη αυτή επιδιώκεται η ανάδειξη της φυσικής ομορφιάς του τόπου στον οποίο δραστηριοποιείται η ΚΟΙΝΣΕΠ. Η σύνδεση με το φυσικό περιβάλλον, η προβολή του φυσικού κάλλους της περιοχής και η ενίσχυση της περιβαλλοντικής συνείδησης των πολιτών αποτελούν βασικές προτεραιότητες της συγκεκριμένης εταιρείας. Η εικόνα που προβάλλεται παραπάνω αντιστοιχεί στη σχετική οθόνη της εφαρμογής και απεικονίζει πώς παρουσιάζεται αυτή στον τελικό χρήστη.

Κατά την υλοποίηση αυτής της οθόνης, αρχικά έγινε διαχωρισμός των διαθέσιμων τοπίων σε επιμέρους κατηγορίες, ανάλογα με τα χαρακτηριστικά που περιλαμβάνει κάθε τοπίο. Έτσι, δημιουργήθηκαν συνολικά τέσσερις βασικές κατηγορίες, μέσα στις οποίες κατανεμήθηκαν τα αντίστοιχα δεδομένα. Για την απεικόνιση των κατηγοριών χρησιμοποιήθηκαν τέσσερα διαφορετικά UICollectionViews, επιτρέποντας στον χρήστη να περιηγηθεί σε καθένα απ' αυτά κάνοντας swipe προς τα δεξιά ή αριστερά, ώστε να μπορέσει να δει το σύνολο των διαθέσιμων επιλογών.

Όταν ο χρήστης επιλέξει ένα από τα τοπία αυτά, μεταφέρεται σε μια νέα οθόνη, η οποία έχει σχεδιαστεί για να παρέχει περισσότερες πληροφορίες και λεπτομέρειες σχετικά με το συγκεκριμένο τοπίο.



Εικόνα 4.6: Popup τοπίων

Ως επέκταση της λειτουργικότητας της προηγούμενης οθόνης, η παραπάνω εικόνα απεικονίζει τη σελίδα που εμφανίζεται στον χρήστη μόλις επιλέξει ένα συγκεκριμένο τοπίο. Όπως φαίνεται, στο πάνω μέρος της οθόνης προβάλλεται μια φωτογραφία του τοπίου, η οποία προέρχεται από ένα image view. Ο χρήστης μπορεί να επιστρέψει στην προηγούμενη οθόνη με τη λίστα των τοπίων, κάνοντας κύλιση προς τα κάτω (swipe down).

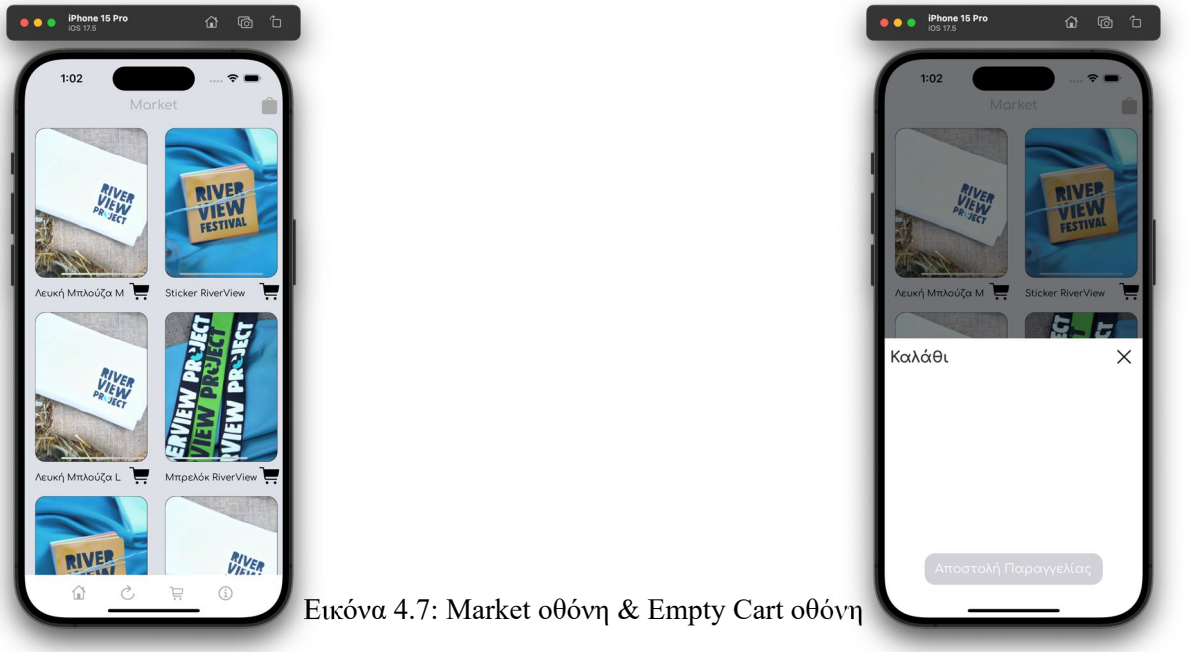
Το πιο ουσιαστικό στοιχείο της οθόνης βρίσκεται λίγο χαμηλότερα: αξιοποιώντας τους χάρτες της Apple, έχει ενσωματωθεί ένας διαδραστικός χάρτης στον οποίο έχει τοποθετηθεί ένα pin που υποδεικνύει την ακριβή γεωγραφική θέση του επιλεγμένου τοπίου.

Τέλος, στο κάτω μέρος της οθόνης υπάρχει ένα κείμενο με περιγραφή του σημείου, προσφέροντας στον χρήστη πληροφορίες και στοιχεία σχετικά με το τι πρόκειται να συναντήσει κατά την επίσκεψή του.

4.4.5 Market Οθόνη

Το επόμενο τμήμα της εφαρμογής αφορά το κατάστημα, το οποίο παρουσιάζεται στην παρακάτω εικόνα. Η οθόνη αυτή έχει ιδιαίτερη σημασία, καθώς συμβάλλει τόσο στην προβολή των τοπικών προϊόντων όσο και στην οικονομική ενίσχυση του project. Μέσω αυτής της ενότητας, επιδιώκεται η ανάπτυξη της πρωτοβουλίας μέσα από διάφορες δράσεις, εκδηλώσεις και άλλες ενέργειες που ενδέχεται να προκύψουν στο μέλλον.

Σε αυτό το περιβάλλον, κάθε παραγωγός έχει τη δυνατότητα να προβάλλει και να προωθήσει τα προϊόντα του. Επιπλέον, το ίδιο το project μπορεί να προωθήσει είδη που το αντιπροσωπεύουν — όπως ρούχα ή άλλα διαφημιστικά αντικείμενα που σχετίζονται με την ταυτότητά του. Δίπλα σε κάθε προϊόν υπάρχει ένα εικονίδιο με καλαθάκι· πατώντας το, ο χρήστης προσθέτει το αντίστοιχο προϊόν στο καλάθι αγορών του. Το καλάθι του χρήστη είναι ορατό στο επάνω δεξί μέρος της οθόνης. Με ένα απλό πάτημα, ανοίγει ένα αναδυόμενο παράθυρο που εμφανίζεται πάνω από την εφαρμογή και στο οποίο παρουσιάζονται τα προϊόντα που έχουν προστεθεί, έτοιμα για αγορά.



Εικόνα 4.7: Market οθόνη & Empty Cart οθόνη

Ανοίγοντας το καλάθι αγορών, παρατηρούμε ότι, εάν ο χρήστης δεν έχει προσθέσει κάποιο προϊόν, δεν μπορεί να προχωρήσει στην ολοκλήρωση της παραγγελίας. Η ύπαρξη τουλάχιστον ενός αντικειμένου στο καλάθι κρίνεται απαραίτητη για την υποβολή οποιασδήποτε παραγγελίας. Επιπλέον, δίπλα σε κάθε προϊόν υπάρχει η δυνατότητα διαγραφής, ώστε ο χρήστης να αφαιρεί όποιο αντικείμενο δεν επιθυμεί πλέον.

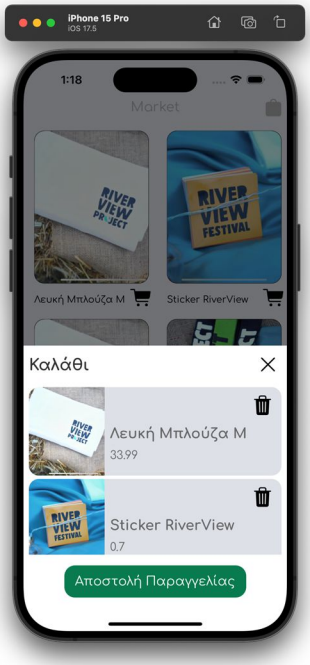
Στο κάτω μέρος της οθόνης εμφανίζεται το κουμπί «Αποστολή Παραγγελίας». Με το πάτημα αυτού του κουμπιού, η παραγγελία καταχωρείται στη βάση δεδομένων με την οποία επικοινωνεί η εφαρμογή. Ο διαχειριστής της εφαρμογής διαθέτει ξεχωριστή, απλή εφαρμογή μέσω της οποίας έχει πρόσβαση σε

όλες τις παραγγελίες, καθώς και στα στοιχεία των χρηστών που τις υπέβαλαν. Με αυτόν τον τρόπο μπορεί να επικοινωνήσει μαζί τους για επιβεβαίωση ή αποστολή της παραγγελίας.

Αξίζει να σημειωθεί ότι η μόνη διαθέσιμη μέθοδος πληρωμής είναι η αντικαταβολή· η εφαρμογή δεν υποστηρίζει εξ αποστάσεως πληρωμές. Μόλις ολοκληρωθεί η παραγγελία, τα περιεχόμενα του καλαθιού διαγράφονται και αυτό επανέρχεται σε κενή κατάσταση.

Ο χρήστης έχει επίσης τη δυνατότητα να εξέλθει από την οθόνη του καλαθιού μέσω ενός κουμπιού στο πάνω δεξιά μέρος, χωρίς όμως να χάσει τα προϊόντα που έχει ήδη προσθέσει. Έτσι, μπορεί να επιστρέψει και να προσθέσει επιπλέον προϊόντα που ενδέχεται να είχε ξεχάσει.

Σε αυτό το σημείο καθίσταται απαραίτητη η εγγραφή και η είσοδος του χρήστη στην εφαρμογή. Χωρίς την ταυτοποίηση, ο διαχειριστής δεν θα είχε τη δυνατότητα να γνωρίζει ποιος έκανε την παραγγελία, ούτε θα μπορούσε να συλλέξει τα απαραίτητα στοιχεία επικοινωνίας. Αντί να απαιτείται μια πρόσθετη φόρμα συμπλήρωσης στοιχείων, όλα γίνονται αυτόματα με μια απλή εγγραφή.



Εικόνα 4.8: Cart οθόνη

4.4.6 Contact οθόνη

Τέλος όσον αφορά την επικοινωνία μεταξύ της εφαρμογής και της ΚΟΙΝΣΕΠ, το οποίο είναι απαραίτητο ώστε ο χρήστης να γνωρίζει πώς και πού μπορεί να έρθει σε επαφή με τον οργανισμό.

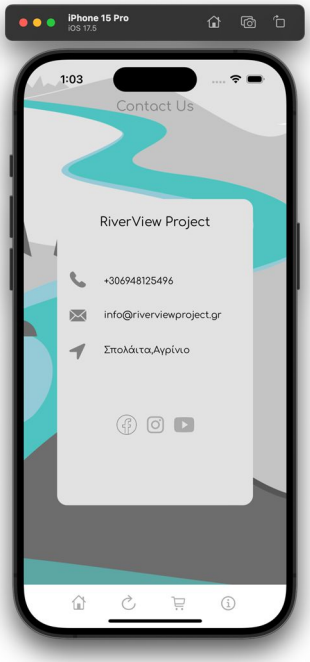
Αρχικά, εμφανίζεται αριθμός κινητού τηλεφώνου που ανήκει στην εταιρεία. Πατώντας επάνω στον αριθμό, ανοίγει αυτόματα η εφαρμογή κλήσεων της συσκευής, με τον αριθμό ήδη προσυμπληρωμένο, ώστε ο χρήστης να μπορεί να πραγματοποιήσει άμεσα την κλήση.

Στη συνέχεια, παρέχεται η διεύθυνση ηλεκτρονικού ταχυδρομείου της εταιρείας. Ο χρήστης μπορεί να πατήσει πάνω σε αυτή και να μεταφερθεί σε εφαρμογή αποστολής e-mail, με το πεδίο του παραλήπτη ήδη συμπληρωμένο. Έτσι, διευκολύνεται σημαντικά η αποστολή μηνύματος στην ομάδα του project.

Αμέσως μετά, εμφανίζεται γεωγραφική τοποθεσία των γραφείων της εταιρείας. Πατώντας σε αυτό το σημείο, ο χρήστης μεταφέρεται στους χάρτες της Apple, όπου εμφανίζεται το στίγμα της εταιρείας. Μέσα από τους χάρτες, μπορεί να λάβει οδηγίες πλοήγησης και να επισκεφθεί τα γραφεία αυτοπροσώπως, εφόσον το επιθυμεί.

Τέλος, ιδιαίτερη έμφαση δίνεται στα social media, καθώς αποτελούν βασικό εργαλείο προβολής και επικοινωνίας. Στο κάτω μέρος της οθόνης, εμφανίζονται τρία εικονίδια, που αντιστοιχούν σε:

- Facebook
- Instagram
- YouTube



Εικόνα 4.9: Info οθόνη

Πατώντας στο πρώτο εικονίδιο, ο χρήστης μεταφέρεται στη σελίδα του οργανισμού στο Facebook. Με το δεύτερο, κατευθύνεται στο προφίλ Instagram του οργανισμού. Τέλος, το τρίτο εικονίδιο οδηγεί στη σελίδα της εταιρείας στο YouTube.

Με αυτόν τον τρόπο, διασφαλίζεται συνεχής επαφή μεταξύ του χρήστη και της ΚΟΙΝΣΕΠ Riverview, ενισχύοντας τη διαδραστικότητα και τη διαρκή ενημέρωση του κοινού για όλες τις εξελίξεις που αφορούν την ΚΟΙΝΣΕΠ.

Κεφάλαιο 5ο: Επίλογος - Συμπεράσματα

Η παρούσα πτυχιακή εργασία είχε ως στόχο την ανάπτυξη μιας ολοκληρωμένης iOS εφαρμογής για λογαριασμό της ΚΟΙΝΣΕΠ RiverView, καθώς και την παρουσίαση των βασικών λειτουργιών του λειτουργικού συστήματος iOS, σε συνδυασμό με τις τεχνολογίες που αξιοποιούνται για την ανάπτυξη mobile εφαρμογών.

Μέσα από την εργασία αυτή, αναλύθηκαν τόσο τα θεωρητικά θεμέλια του iOS λογισμικού όσο και οι πρακτικές εφαρμογές που υλοποιήθηκαν στο πλαίσιο της εφαρμογής. Η εφαρμογή που δημιουργήθηκε καλύπτει τις ανάγκες της οργάνωσης RiverView για ολοκληρωμένη επικοινωνία με τους χρήστες, προβολή των προϊόντων και δράσεων, διαχείριση παραγγελιών, καθώς και ενημέρωση μέσα από κοινωνικά δίκτυα. Η αρχιτεκτονική της εφαρμογής σχεδιάστηκε με γνώμονα τη λειτουργικότητα, τη χρηστικότητα και την αποδοτικότητα, χρησιμοποιώντας τεχνολογίες όπως η Swift και το UIKit.

Η διαδικασία σχεδίασης και ανάπτυξης της εφαρμογής κατέδειξε τη σημασία της σαφούς κατανόησης των αναγκών του τελικού χρήστη, αλλά και της συνεργασίας με τον φορέα υλοποίησης. Παράλληλα, προσέφερε την ευκαιρία για εξοικείωση με πρακτικές ανάπτυξης που ακολουθούνται στην επαγγελματική αγορά.

Συμπεράσματα:

- Η ανάπτυξη εφαρμογών για το οικοσύστημα της Apple απαιτεί συνδυασμό τεχνικής γνώσης και εμπειρίας χρήσης του iOS.
- Η δημιουργία μιας εφαρμογής για έναν πραγματικό οργανισμό, όπως η ΚΟΙΝΣΕΠ RiverView, ενίσχυσε την πρακτική διάσταση της γνώσης που αποκτήθηκε κατά τη διάρκεια των σπουδών.
- Η σύνδεση θεωρίας και πράξης βοήθησε στην ανάπτυξη δεξιοτήτων ανάλυσης απαιτήσεων, σχεδιασμού UX/UI και debugging.
- Η εργασία ανέδειξε τη δυνατότητα των τεχνολογιών mobile development να προωθήσουν κοινωνικούς σκοπούς, ειδικά όταν εντάσσονται στο πλαίσιο φορέων όπως οι ΚΟΙΝΣΕΠ.
- Η επιλογή των εργαλείων και των τεχνολογιών (Swift, Xcode, UIKit, κ.λπ.) κρίθηκε κατάλληλη για την επίτευξη ενός σταθερού και ευέλικτου τελικού προϊόντος.
- Η διαδικασία ενίσχυσε την κριτική σκέψη, την αυτονομία στη λήψη αποφάσεων και τη συστηματική επίλυση προβλημάτων.

Συνοψίζοντας, η εκπόνηση αυτής της πτυχιακής εργασίας υπήρξε μια πολύτιμη εμπειρία που συνέβαλε όχι μόνο στην επιστημονική και τεχνική κατάρτιση, αλλά και στην προσωπική ανάπτυξη του συγγραφέα, αποτελώντας ένα γόνιμο βήμα προς την επαγγελματική του πορεία στον χώρο της ανάπτυξης λογισμικού και ειδικότερα στον τομέα των mobile εφαρμογών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] «The History of iOS and its Evolution» [Ηλεκτρονικό]. Available: https://www.mobileappdaily.com/history-of-ios (last accessed: 20/05/2025).
[2] «iOS» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/IOS (last accessed: 20/05/2025).
[3] «What is a Framework?» [Ηλεκτρονικό]. Available: https://www.codecademy.com/resources/blog/what-is-a-framework/ (last accessed: 22/05/2025).
[4] «Application Framework» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Application_framework (last accessed: 22/05/2025).
[5] «API» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/API (last accessed: 22/05/2025).
[6] «Collection Types» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/collectiontypes/ (last accessed: 22/05/2025).
[7] «Optional» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/swift/optional (last accessed: 22/05/2025).
[8] «Structures and Classes» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/classesandstructures/ (last accessed: 22/05/2025).
[9] «Control Flow» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/controlflow/ (last accessed: 22/05/2025).
[10] «Functions» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/functions/ (last accessed: 22/05/2025).
[11] «The Basics» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/thebasics/ (last accessed: 22/05/2025).
[12] «Swift Data Types» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/thebasics/ (last accessed: 22/05/2025).
[13] «Memory Management» [Ηλεκτρονικό]. Available: https://www.sngular.com/insights/281/memory-management-in-swift (last accessed: 23/05/2025).
[14] «Architecture Patterns» [Ηλεκτρονικό]. Available: https://www.geeksforgeeks.org/types-of-software-architecture-patterns/ (last accessed: 23/05/2025).
[15] «Clean Architecture» [Ηλεκτρονικό]. Available: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (last accessed: 23/05/2025).
[16] «MVVM» [Ηλεκτρονικό]. Available: https://www.geeksforgeeks.org/introduction-to-model-view-view-model-mvvm/ (last accessed: 23/05/2025)
[17] «Git» [Ηλεκτρονικό]. Available: https://www.atlassian.com/git/tutorials/what-is-git (last accessed: 23/05/2025)
[18] «Firebase» [Ηλεκτρονικό]. Available: https://swiftpackageindex.com/firebase/firebase-ios-sdk (last accessed: 24/05/2025)
[19] «Firestore» [Ηλεκτρονικό]. Available: https://firebase.google.com/docs/firestore. (last accessed: 24/05/2025)
[20] «FirebaseAuth» [Ηλεκτρονικό]. Available: https://firebase.google.com/docs/auth. (last accessed: 24/05/2025)
[21] «FirebaseCore» [Ηλεκτρονικό]. Available: https://firebase.google.com/docs/reference/ios/firebasecore/api/reference/Classes (last accessed: 24/05/2025)
[22] «UIKit» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/uikit (last accessed: 25/05/2025)
[23] «Auto Layout» [Ηλεκτρονικό]. Available: https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html (last accessed: 25/05/2025)
[24] «SwiftUI» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/SwiftUI (last accessed: 25/05/2025)
[25] «Libraries» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Library_(computing) (last accessed: 25/05/2025)

[26] «MapKit» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/mapkit/ (last accessed: 25/05/2025)