ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«iOS Client για ανοιχτό σύστημα ανακοινώσεων / ενημέρωσης με RESTful API»

Της φοιτήτριας
Μούσα Αναστασίας
Αρ. Μητρώου: 175088

Επιβλέπων
Αντώνης Σιδηρόπουλος
Αναπληρωτής Καθηγητής

Ημερομηνία: 10/9/2024

Τίτλος Π.Ε.: iOS Client για ανοιχτό σύστημα ανακοινώσεων / ενημέρωσης με RESTful API
Κωδικός Π.Ε.: 22259
Ονοματεπώνυμο φοιτήτριας: Αναστασία Μούσα
Ονοματεπώνυμο εισηγητή: Αντώνιος Σιδηρόπουλος
Ημερομηνία ανάληψης Π.Ε.: 15/10/2022
Ημερομηνία περάτωσης Π.Ε.: 10/09/2024

Βεβαιώνω ότι είμαι η 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 πτυχιακή 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

Επιστημονικό κίνητρο για την επιλογή και εκπόνηση του συγκεκριμένου θέματος αποτέλεσε η πραγματοποίηση της πρακτικής μου άσκησης πάνω στο κομμάτι του iOS Development, δηλαδή της ανάπτυξης εφαρμογών για συσκευές με το λειτουργικό σύστημα της Apple, η οποία ήταν και η έναρξη της ενασχόλησής μου με τον τομέα και με ενθάρρυνε στο να εμβαθύνω σε αυτόν και τις τεχνολογίες που τον διέπουν.

Επιπλέον, για την ενημέρωση των φοιτητών του τμήματος έχει ήδη διατεθεί client για Android συσκευές, οπότε θεώρησα πως θα μπορούσε να προσφερθεί μια αντίστοιχη εμπειρία χρήστη και στους φοιτητές που διαθέτουν iOS συσκευές.

Περίληψη

Η παρούσα εργασία μελετά μια προϋπάρχουσα υλοποίηση της εφαρμογής “IEE Aboard” για iOS συσκευές, η οποία βασισμένη στο σύστημα ανακοινώσεων “Aboard”, στοχεύει στην ενημέρωση των φοιτητών σχετικά με τις τελευταίες εξελίξεις που συμβαίνουν στο τμήμα και τους δίνει τη δυνατότητα να λαμβάνουν εξατομικευμένες ειδοποιήσεις. Σκοπός της εργασίας είναι ο εντοπισμός σημείων της εφαρμογής που επιδέχονται βελτίωση ώστε να είναι λειτουργική με βάση τον ιστοχώρο aboard.iee.ihu.gr και η ορθή επικοινωνία με το API του “Aboard” μέσω των κατάλληλων βιβλιοθηκών και εργαλείων που διαθέτει η γλώσσα προγραμματισμού Swift, η οποία χρησιμοποιείται για την ανάπτυξη iOS εφαρμογών.

«iOS Client for an open notifications' / updates'

system with RESTful API»

Anastasia Mousa

Abstract

This paper studies a pre-existing implementation of the "IEE Apps" application for iOS devices, which, based on the "Aboard" announcement system, aims to inform students about the latest announcements from the department and get them to receive personalized notifications. The purpose of the paper is to identify points of the application that can be improved so that it is fully functional corresponding to the website aboard.iee.ihu.gr and communicates correctly with the "Aboard" API through appropriate libraries and tools available in Swift programming language, which is used to develop iOS applications.

Ευχαριστίες

Η ολοκλήρωση της παρούσας πτυχιακής εργασίας, η οποία αποτελεί βασική προϋπόθεση για την περάτωση των προπτυχιακών σπουδών μου στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου Ελλάδος, θα ήταν αδύνατη χωρίς την υποστήριξη μερικών ατόμων.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα της εργασίας και αναπληρωτή καθηγητή του τμήματός μας, κ. Σιδηρόπουλο Αντώνιο για την εμπιστοσύνη του στην ανάθεση του θέματος της πτυχιακής εργασίας. Θερμές ευχαριστίες απευθύνω και σε όλους τους καθηγητές του τμήματος για τις γνώσεις που μου μετέδωσαν κατά τη διάρκεια των προπτυχιακών μου σπουδών.

Ένα μεγάλο ευχαριστώ οφείλω στους δικούς μου ανθρώπους για την κατανόηση και την υποστήριξη που μου έδειξαν απλόχερα, όχι μόνο κατά το διάστημα της εκπόνησης της πτυχιακής εργασίας αλλά και γενικότερα, στα χρόνια των σπουδών μου. Δε θα μπορούσα να μην αναφερθώ στους υπέροχους γονείς μου αλλά και στα δύο αδέρφια μου, που συνεχίζουν να με στηρίζουν απεριόριστα.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Αποσπασμάτων Κώδικα

Συντομογραφίες

Π.Ε.	Πτυχιακή Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος

Κεφάλαιο 1ο: Εισαγωγή

1.1 Πρόλογος/Περιγραφή του θέματος

Στο τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος λειτουργεί σύστημα ενημέρωσης των φοιτητών, που ονομάζεται “Aboard”. Η αναγκαιότητα διαρκούς ενημέρωσης των φοιτητών σχετικά με θέματα που αφορούν το τμήμα, έδωσε και την αφορμή για να δημιουργηθούν mobile clients, με πρώτο αυτό για το Android λειτουργικό σύστημα και κατ’ επέκταση και τον αντίστοιχο για κινητές συσκευές της Apple με iOS λειτουργικό σύστημα, τον οποίο πραγματεύεται και η παρούσα εργασία.

1.2 Περιγραφή της εφαρμογής

Η εφαρμογή που μελετήθηκε και με βάση την οποία έγινε ανακατασκευή και βελτίωση, αφορά την on-mobile εμπειρία χρήστη συσκευής με λειτουργικό σύστημα iOS, η οποία αφορά μια πλατφόρμα παρακολούθησης και εξατομικευμένης ενημέρωσης του φοιτητή για τις τελευταίες εξελίξεις στο τμήμα, με τη χρήση push notifications.

1.3 Διάρθρωση Πτυχιακής Εργασίας

- Στο πρώτο κεφάλαιο γίνεται εισαγωγή και σύντομη περιγραφή του θέματος που πραγματεύεται η εργασία και μια παρουσίαση του αποτελέσματος στο οποίο στοχεύει με την περάτωσή της.
- Στο δεύτερο κεφάλαιο παραθέτονται οι τεχνολογίες οι οποίες χρησιμοποιήθηκαν προκειμένου να καταστεί δυνατή η υλοποίηση των λειτουργιών της εφαρμογής που αναλύει αυτή η εργασία.
- Στο τρίτο κεφάλαιο γίνεται αναφορά στα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής και αναλύεται ο ρόλος που επιτέλεσαν.
- Στο τέταρτο κεφάλαιο παρουσιάζεται η εφαρμογή, η λειτουργικότητά της και η εμπειρία χρήστη που ικανοποιεί.
- Στο πέμπτο κεφάλαιο δίνονται κομμάτια και περιγραφές σχετικά με τον κώδικα που βρίσκεται πίσω από την υλοποίηση της εφαρμογής.
- Στο έκτο και τελευταίο κεφάλαιο που απαρτίζει αυτή την εργασία, γίνεται μια σύνοψη της όλης προσπάθειας και σημειώνονται κάποια συμπεράσματα που προκύπτουν από τη λειτουργία της εφαρμογής.

Κεφάλαιο 2ο: Περιγραφή τεχνολογιών

2.1 Λειτουργικό σύστημα iOS

2.1.1 Εισαγωγή

Η εφαρμογή που πραγματεύεται η παρούσα εργασία αφορά χρήστες που διαθέτουν Apple κινητές συσκευές οι οποίες περιέχουν iOS λειτουργικό σύστημα.

Το λειτουργικό σύστημα είναι ένα τμήμα λογισμικού το οποίο αποτελεί το σύνδεσμο μεταξύ του υλικού και του λογισμικού ενός υπολογιστή που εξασφαλίζει την αξιοποίηση του υλικού από τα εγκατεστημένα προγράμματα. Το ίδιο το λειτουργικό σύστημα μπορεί να θεωρηθεί επίσης ένα πρόγραμμα, με την ειδοποιό διαφορά πως τα υπόλοιπα προγράμματα του υπολογιστή βασίζονται σε αυτό για να εξασφαλίσουν την εύρυθμη λειτουργία τους. Σημαντικά σημεία στα οποία εξυπηρετεί είναι η διαχείριση πόρων, η κατανομή των διαφόρων εργασιών, η ομαλή εκτέλεση των εφαρμογών, η διαχείριση της λειτουργίας των συσκευών εισόδου και εξόδου, η οργάνωση και διαχείριση των αρχείων του συστήματος και η ανίχνευση πιθανών δυσλειτουργιών του συστήματος και ενημέρωση του χρήστη για αυτά. Το λειτουργικό σύστημα μπορεί να υπάρχει ήδη εγκατεστημένο σε μια συσκευή κατά την αγορά της, να υπάρχει κάποιο και να αλλάξει μεταγενέστερα από το χρήστη ή να εγκατασταθεί εξ' ολοκλήρου από το χρήστη μετέπειτα με βάση τα χαρακτηριστικά που υποστηρίζονται από το υλικό.

Τα επικρατέστερα λειτουργικά συστήματα γενικού σκοπού είναι τα εξής: Windows (Microsoft), MacOS (Apple) και οι διάφορες διανομές του Linux. Σχετικά με το κομμάτι των λειτουργικών συστημάτων κινητών συσκευών, χρησιμοποιούνται ευρέως: Android (Google), iOS (Apple), Chrome OS. Η διαφορά των δύο λειτουργικών συστημάτων που διανέμει η Google είναι ότι το Android είναι λογισμικό ανοικτού κώδικα, δηλαδή επιτρέπει τη συμβολή του κοινού στη διαμόρφωσή του διαθέτοντας τον κώδικα κατασκευής του δημόσια ενώ το Chrome OS είναι λογισμικό κλειστού κώδικα, δηλαδή απαγορεύεται οποιαδήποτε παρέμβαση. Το iOS, είναι λογισμικό κλειστού κώδικα που έχει αναπτύξει η Apple, με σκοπό να πλαισιώνει αποκλειστικά τις συσκευές που η ίδια κατασκευάζει και διανέμει στην αγορά.

2.1.2 Ιστορικό πλαίσιο

Το iOS λειτουργικό σύστημα για κινητές συσκευές κυκλοφόρησε από την Apple το Ιούνιο του 2007. Έκτοτε, κάθε χρόνο η εταιρεία κυκλοφορεί νέες εκδόσεις προσθέτοντας νέα χαρακτηριστικά και δυνατότητες για τους χρήστες. Οι εκδόσεις αυτές παρατίθενται παρακάτω:

- 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.

- iPhone OS 2 (2008)

Με τη δεύτερη αυτή έκδοση, εισήχθη η λειτουργία της συγκεντρωτικής πλατφόρμας διάθεσης εφαρμογών "App Store", στην οποία μπορούσαν πλέον να ανεβούν εφαρμογές και από άλλους δημιουργούς / προγραμματιστές / εταιρείες και να είναι ελεύθερες στους χρήστες για κατέβασμα, αφού περάσουν από διαδικασία ελέγχου από την Apple.

- iPhone OS 3 (2009)

Με την έλευση της τρίτης έκδοσης, 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)

Πλέον χρησιμοποιείται η συντομογραφία iOS για αναφορά στο λειτουργικό σύστημα κινητών συσκευών της Apple. Πέρα από την αλλαγή του ονόματος, υπάρχει και εκτεταμένη διαθεσιμότητα κάποιων λειτουργιών που δεν ήταν δυνατό να υποστηριχθούν σε κάποιες συσκευές (iPhone 1, iPhone 3G, iPod), όπως η προσαρμογή της οθόνης και η ταυτόχρονη χρήση παραπάνω από μίας εφαρμογής. Επίσης εισήχθησαν νέες native εφαρμογές της Apple.

- iOS 5 (2011)

Γίνεται προσθήκη της δυνατότητας λήψης ειδοποιήσεων από τις εφαρμογές, τόσο βασικών όσο και εξατομικευμένων, η διαθεσιμότητα νέων native εφαρμογών και η βοήθεια του χρήστη, η "Siri".

- iOS 6 (2012)

Με αυτή την ανανεωμένη έκδοση η Apple επεκτείνει το φάσμα των native εφαρμογών της, κάνοντας το λειτουργικό της ακόμα πιο μοναδικό. Εισάγει το Apple Maps το οποίο υποστηρίζει τη λειτουργία της Siri και έρχεται να αντικαταστήσει το Google Maps.

- iOS 7 (2013)

Γίνεται προσθήκη του AirDrop, σαν εργαλείο επικοινωνίας μεταξύ των Apple συσκευών για άμεσο διαμοιρασμό αρχείων με χρήση πρωτοκόλλων bluetooth.

- iOS 8 (2014)

Σε αυτή την έκδοση γίνεται ενεργοποίηση της "Siri" μέσω φωνητικών εντολών του χρήστη και ενίσχυση των εφαρμογών και των έξυπνων λειτουργιών των συσκευών της Apple.

- iOS 9 (2015)

Εισάγεται η λειτουργία του διαχωρισμού της οθόνης για την εξασφάλιση παράλληλης και ομαλής εκτέλεσης εφαρμογών στο ίδιο οπτικό πεδίο για τον χρήστη. Βελτιώνεται η κατανομή των διεργασιών και των πόρων του λειτουργικού συστήματος και της κινητής συσκευής, για μια καλύτερη εμπειρία χρήστη.

- iOS 10 (2016)

Βελτιώνεται το περιβάλλον και οι εφαρμογές που παρέχει η Apple στο χρήστη και καθίσταται ευκολότερη η επικοινωνία μεταξύ των εγκατεστημένων εφαρμογών που πλέον μπορεί να επιλέγει και αν επιθυμεί να καταργήσει ο χρήστης μετά από προεγκατάσταση.

- iOS 11 (2017)

Στη συγκεκριμένη έκδοση παρατηρείται μέριμνα και για το iPad όχι μόνο το iPhone, δεδομένου το ότι χρησιμοποιείται τόσο για ψυχαγωγία όσο και για εργασία καθώς υπάρχουν αξεσουάρ για την επεκτασιμότητά του, όπως για παράδειγμα πληκτρολόγιο. Εισάγονται το Dock με το οποίο εναλλάσσονται οι εφαρμογές που τρέχουν την παρούσα στιγμή, ο διαχειριστής αρχείων και το ARKit για ανάπτυξη και υποστήριξη εφαρμογών με λειτουργίες επαυξημένης πραγματικότητας.

- iOS 12 (2018)

Μεταβατική και διορθωτική έκδοση με βελτιώσεις σε εφαρμογές της Apple, στο ARKit και ενίσχυση της προσαρμοστικότητας των εφαρμογών και για ομαλή εκτέλεσή τους σε παλαιότερες συσκευές ώστε να μη γίνεται ώθηση σε αντικατάστασή τους από το χρήστη πρόωρα.

- iOS 13 (2019)

Στην παρούσα έκδοση γίνεται διαχωρισμός του λειτουργικού συστήματος κινητών συσκευών που παρέχει η Apple σε iOS για τα iPhone και iPadOS για τα iPads, για να επιτευχθεί εξατομίκευση και παρόμοια εμπειρία χρήστη. Επίσης συνεχίζεται η ανάπτυξη λειτουργιών γενικότερα όπως για παράδειγμα το Face ID σαν αναγνωριστικό για λόγους ασφαλείας.

- iOS 14 (2020)

Μια έκδοση στην οποία υπάρχουν βελτιώσεις σχετικά με την πολιτική απορρήτου και τη χρήση δεδομένων του χρήστη από τις εφαρμογές. Επίσης υπάρχουν γενικότερες βελτιώσεις στο σχεδιασμό της διεπαφής χρήστη.

- iOS 15 (2021)

Γίνονται βελτιώσεις και εισαγωγές νέων λειτουργιών στις προεγκατεστημένες εφαρμογές της Apple, όπως το FaceTime, το Maps, με οδηγό βάσει του ARKit, το iMessage και το Live Text το οποίο μπορεί πλέον να αναγνωρίζει κείμενο που απεικονίζεται σε μια φωτογραφία κ.α.

- iOS 16 (2022)

Η έκδοση του iOS που ανακοινώθηκε το Σεπτέμβριο του 2022, μαζί με την έλευση της σειράς του iPhone 14. Πρόκειται για μια ακόμη ενίσχυση της εμπειρίας χρήστη και της δυνατότητάς του για εξατομίκευση της συσκευής του. Σημεία όπως η οθόνη κλειδώματος, το iMessage, το Wallet και το Mail είναι κάποια ενδεικτικά αυτών των αλλαγών.

- iOS 17 (2023)

Είναι η τελευταία έκδοση του iOS που ανακοινώθηκε τον Ιούνιο του 2023. Η έκδοση αυτή φέρνει αναβαθμίσεις σε βασικές εφαρμογές iPhone όπως το iMessage, με "poster" εικόνες και Check-in λειτουργία. Υπάρχουν βελτιώσεις και στο AirDrop για τη μεταφορά αρχείων. Το πληκτρολόγιο διορθώνει πλέον την ορθογραφία πιο άμεσα με AI και το "Standby" μετατρέπει την αδρανή οθόνη της κατάστασης φόρτισης σε χρήσιμη. Το iOS 17 κυκλοφορεί μαζί με τα iPhone 15 από τον Σεπτέμβριο του 2023.

- iOS 18 (2024)

Τον Ιούνιο του 2024, ανακοινώθηκε η δέκατη όγδοη έκδοση του λειτουργικού iOS, όπως συνηθίζεται, στην επίσημη ετήσια παρουσίαση προϊόντων τεχνολογίας και λογισμικού που διοργανώνει η Apple και αναμένεται η κυκλοφορία του στο κοινό μέχρι το Νοέμβριο του ίδιου έτους.

2.1.3 Βασικά χαρακτηριστικά

Το iOS λειτουργικό σύστημα, που σχεδιάζεται και υλοποιείται από την Apple, προορίζεται για τις κινητές συσκευές της. Δεδομένης αυτής της αποκλειστικότητας, η Apple παρέχει την εμπειρία ενός ολοκληρωμένου οικοσυστήματος επιτυγχάνοντας της επικοινωνία μεταξύ iPhone, iPad, Apple Watch, Apple TV, Apple Home. Συνεπώς, κάθε αναβάθμιση του iOS έχει αντίκτυπο στη συνολική λειτουργία όσων προαναφέρθηκαν.

Σχετικά με το εσωτερικό του iOS, πρόκειται, όπως ήδη έχει προαναφερθεί για λογισμικό κλειστού κώδικα. Πρακτικά αυτό σημαίνει πως ο πηγαίος κώδικας που έχει γραφτεί για την υποστήριξη αυτού του λειτουργικού συστήματος είναι προσβάσιμος αποκλειστικά και μόνο από τους προγραμματιστές της Apple και δεν επιτρέπεται η αναπαραγωγή ή οποιαδήποτε παρέμβαση από τρίτα μέλη. Επιπλέον αυτό σημαίνει πως το συγκεκριμένο λειτουργικό σύστημα υποστηρίζεται αποκλειστικά από συσκευές που σχεδιάζει και διανέμει η Apple. Με αυτόν τον τρόπο η εταιρεία εξασφαλίζει το μονοπώλιο του λειτουργικού της και των μοναδικών χαρακτηριστικών που το διέπουν, στοχεύει σε συγκεκριμένο και αποκλειστικό κοινό και είναι η ίδια υπεύθυνη για τη διαρκή βελτιστοποίησή του.

Με το συνδυασμό λογισμικού κλειστού κώδικα και συγκεκριμένης γκάμας συσκευών, η Apple προσπαθεί να παρέχει εμπειρία χρήστη, εκμεταλλευόμενη τις δυνατότητες των συσκευών της στο μέγιστο βαθμό. Σε σχέση με το μεγάλο ανταγωνιστή της, το ανοικτού κώδικα λειτουργικό σύστημα Android της Google, ο σχεδιασμός και η διεπαφή του χρήστη εμφανίζονται απλούστερα, κάτι το οποίο μπορεί να κριθεί ως αρνητικό, αλλά από την άλλη πλευρά, μπορεί να καλύψει μεγαλύτερο φάσμα κοινού λόγω της ευχρηστίας του.

Πάνω στο κομμάτι των εφαρμογών που εκτελούνται, υπάρχει ένα συγκριτικό πλεονέκτημα στην πλευρά της Apple με το iOS. Το γεγονός ότι η ίδια η εταιρεία σχεδιάζει και υλοποιεί το λειτουργικό σύστημα που απαρτίζει τη συγκεκριμένη γκάμα συσκευών της κάνει πιο στοχευμένο τον σχεδιασμό των εφαρμογών με πολύ καλές επιδόσεις και περιορισμένα λάθη. Επιπλέον, σχετικά με την ανάπτυξη ιδιόκτητων εφαρμογών κλειστού κώδικα, η Apple έχει καταφέρει να φτάσει τις εφαρμογές της σε υψηλό επίπεδο χωρίς να υστερεί σε κάποιο γνώρισμα από τις αντίστοιχες, μαζικά χρησιμοποιούμενες εφαρμογές της Google για το Android λειτουργικό.

Εξίσου ιδιαίτερο ζήτημα, είναι και αυτό της ασφάλειας. Εδώ το κομμάτι του κλειστού κώδικα έρχεται να υπερισχύσει, καθώς με αυτόν τον τρόπο, το iOS μπορεί να προστατευθεί από κακόβουλες παρεμβάσεις σε μεγαλύτερο βαθμό από κάποιο άλλο, ανοικτού κώδικα λειτουργικό, όπως το Android. Επιπλέον, όσον αφορά την συλλογή και τη διαχείριση προσωπικών δεδομένων, η Apple φαίνεται να συμπεριφέρεται πιο φιλικά προς το χρήστη, καθώς δεν ακολουθεί την τακτική της Google, να βασίζεται στη συλλογή δεδομένων για την αύξηση των εσόδων.

2.1.4 Ανάπτυξη εφαρμογών για iOS

Δεδομένου το ότι οι πωλήσεις των συσκευών της Apple έχουν μια διαρκώς αυξανόμενη πορεία, εμφανίζεται ανάλογη ζήτηση και κινητικότητα και στον τομέα της ανάπτυξης εφαρμογών που απαρτίζουν το λειτουργικό iOS. Για να μπορέσει μια εφαρμογή να διατεθεί στο κοινό μέσω του AppStore, πρέπει να πληροί κάποια αυστηρά κριτήρια, τεχνικά και σχεδιασμού, τα οποία έχει θέσει η Apple προκειμένου να εξασφαλίσει την αξιοπιστία, την ασφάλεια και μια γενική ομοιομορφία, η οποία να καθιστά την εμπειρία χρήσης των εφαρμογών της ξεχωριστή.

Η ανάπτυξη των iOS εφαρμογών γίνεται σε περιβάλλον που έχει δομήσει η Apple για αυτό το σκοπό και με τη χρήση πληθώρας βιβλιοθηκών και σύγχρονων εργαλείων που κυκλοφορούν. Κάποια από αυτά θα αναλυθούν παρακάτω, καθώς χρησιμοποιούνται και στο πρακτικό κομμάτι που απαρτίζει την παρούσα εργασία. Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι η Swift, μια νέα σχετικά γλώσσα, υψηλού επιπέδου, με απλό και κατανοητό συντακτικό. Υπάρχει δυνατότητα προσομοίωσης και δοκιμών, η οποία είναι μια ελεγχόμενη διαδικασία, καθώς υπάρχει περιορισμένο πλήθος συσκευών.

2.2 Frameworks για iOS

2.2.1 Εισαγωγή

Με τον όρο framework αναφερόμαστε σε έναν σκελετό, στον οποίο μπορούμε να βασιστούμε προκειμένου να ενισχύσει με τα χαρακτηριστικά του τη διαδικασία ανάπτυξης λογισμικού που ακολουθούμε σε κάθε περίπτωση, εξυπηρετώντας τη γλώσσα προγραμματισμού που απαιτείται κάθε φορά.

Τα frameworks μπορούν να βελτιώσουν τη διαδικασία ανάπτυξης λογισμικού και να βοηθήσουν στην τήρηση χρονοδιαγραμμάτων, επιταχύνοντας διαδικασίες, ιδίως στην έναρξη υλοποίησης μιας εφαρμογής. Επιπλέον, δεν απαιτείται κάποια περαιτέρω διερεύνηση πριν τη χρήση τους, είναι σχεδιασμένα και διατίθενται κατευθείαν για χρήση και υπάρχει και η αντίστοιχη συντήρηση και υποστήριξή τους.

Συνοπτικά, η χρήση ενός framework στην προγραμματιστική διαδικασία, διευκολύνει τον προγραμματιστή, οδηγεί στην παραγωγή ασφαλέστερου και πιο “καθαρού” ποιοτικά κώδικα δίνει τη δυνατότητα προσαρμογής του ανάλογα με τις απαιτήσεις που πρέπει να καλυφθούν σε κάθε περίπτωση.

2.2.2 Κατηγορίες

Αυτή τη στιγμή υπάρχει διαθέσιμη μια πληθώρα frameworks που το καθένα βασίζεται σε μια γλώσσα προγραμματισμού και εξυπηρετεί συγκεκριμένες λειτουργικότητες, ανάλογα και με το αν αφορά web εφαρμογή, βάση δεδομένων ή εφαρμογή κινητών συσκευών. Με βάση το πεδίο εφαρμογής, διακρίνονται στις εξής κατηγορίες:

1. Web frameworks

   Πρόκειται για frameworks που χρησιμοποιούνται για την ανάπτυξη web εφαρμογών που χρησιμοποιούν διάφορα web services και APIs.

2. Front-end frameworks

   Είναι frameworks τα οποία παρέχουν σκελετούς και έτοιμα κομμάτια κώδικα από HTML, CSS και JavaScript προκειμένου να συμβάλλουν στη δημιουργία μιας διεπαφής για ιστοσελίδα ή web εφαρμογή.

3. Back-end frameworks

   Παρέχουν server-side γενικές λειτουργίες που μπορούν είτε να συνδυαστούν μεταξύ τους είτε να αποτελέσουν τα θεμέλια για την περαιτέρω ανάπτυξη. Η επιλογή του κατάλληλου framework, υπαγορεύεται από τη γλώσσα προγραμματισμού που έχει επιλεχθεί καθώς και από τις λειτουργίες που θα επιτελεί η εφαρμογή που δημιουργείται.

4. Mobile App Development Frameworks

   Πρόκειται για 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

   Τα Content Management Systems (CMS) είναι λογισμικά που επιτρέπουν στους χρήστες να δημιουργούν, να οργανώνουν και να τροποποιούν περιεχόμενο, όπως αναρτήσεις ιστολογίου, ηλεκτρονικά βιβλία, εφαρμογών για κινητά ή διαδίκτυο. Το αντίστοιχο framework, παρέχει επαναχρησιμοποιήσιμα κομμάτια για τη διαχείριση και την κοινή χρήση χαρακτηριστικών ενός web framework και ενός CMS.

6. Data Science Frameworks

   Στον τομέα της επιστήμης των δεδομένων παρατηρείται μια εκτεταμένη χρήση frameworks για ανάλυση, εξόρυξη δεδομένων και το σχεδιασμό αλγορίθμων.

Επίσης, τα frameworks ανάλογα με τον τύπο της ανάπτυξης της εφαρμογής που καλούνται να υποστηρίξουν κάθε φορά, χωρίζονται σε δύο κατηγορίες:

Native frameworks

Τα native frameworks προορίζονται για native εφαρμογές, δηλαδή για εφαρμογές οι οποίες σχεδιάζονται για ένα συγκεκριμένο λειτουργικό σύστημα. Οι εφαρμογές αυτές εγκαθίστανται στις συσκευές και λειτουργούν εκμεταλλευόμενες τους πόρους της συσκευής. Τα native frameworks μπορούν να περιπλέξουν τη διαδικασία ανάπτυξης εφαρμογών και να ανεβάσουν το κόστος τους, διότι απαιτείται περισσότερη εξατομίκευση, χρόνος και προσωπικό προγραμματιστών.

Hybrid frameworks

Οι hybrid εφαρμογές έχουν την ειδοποιό διαφορά ότι συνδυάζουν χαρακτηριστικά των native και των web εφαρμογών. Τα hybrid frameworks δίνουν τη δυνατότητα ανάπτυξης εφαρμογών που μπορούν να εγκατασταθούν και να χρησιμοποιηθούν σε διαφορετικά λειτουργικά συστήματα και πλατφόρμες. Με αυτόν τον τρόπο μπορεί να γενικευθεί η διαδικασία ανάπτυξης καθώς γίνεται ευρύτερη χρήση, που αφορά μεγαλύτερο μέρος κοινού και των συσκευών του και επίσης διευκολύνεται η εργασία των προγραμματιστών, κάτι το οποίο στέκεται σαν καθοριστικό κριτήριο για την επιλογή χρήσης κατάλληλου framework.

2.3 Web APIs

2.3.1 Εισαγωγή

Το API, γνωστό και ως διεπαφή εφαρμογών (Application Program Interface), αποτελεί το μέσο με το οποίο η εφαρμογή επικοινωνεί με το τμήμα που διαχειρίζεται τα δεδομένα της, δίνοντάς της τη δυνατότητα να συνεργάζεται με διάφορες υπηρεσίες. Στο πλαίσιο του Web API, αναφερόμαστε στον τρόπο με τον οποίο επιτυγχάνεται η πρόσβαση και η ανάκτηση των δεδομένων μιας δικτυακής εφαρμογής, μέσω αιτημάτων HTTP. Όταν ο χρήστης χρειάζεται συγκεκριμένες πληροφορίες, η εφαρμογή καθορίζει τον τρόπο πρόσβασης που απαιτείται. Στη συνέχεια, μεταβιβάζει το αίτημα στο API, το οποίο, με βάση τους καθορισμένους κανόνες επικοινωνίας, είτε ανακτά τις απαραίτητες πληροφορίες από το σύστημα διαχείρισης δεδομένων, είτε προβαίνει σε περαιτέρω ενέργειες για εξουσιοδότηση του χρήστη. Συνεπώς, το API αποτελεί ενδιάμεσο στοιχείο ανάμεσα στην εφαρμογή και τον διακομιστή δεδομένων, απαλλάσσοντας τον προγραμματιστή από την ανάγκη να υλοποιεί ο ίδιος ολόκληρη τη διαδικασία επικοινωνίας, κάτι που εξοικονομεί χρόνο κατά την προγραμματιστική διαδικασία.

2.3.2 Ανάπτυξη ενός API

Η ανάπτυξη ενός API μπορεί να διακριθεί σε πέντε φάσεις.

- Σχεδιασμός

Η πρώτη φάση αφορά το σχεδιασμό του API, κατά την οποία γίνονται πλάνα που αφορούν τις λειτουργίες του API και δημιουργούνται διαγράμματα χρήσης. Εξετάζονται οι πόροι και η ασφάλειά του και τα πρωτόκολλα και οι τεχνολογίες που θα χρησιμοποιηθούν καθορίζουν την αρχιτεκτονική του.

- Ανάπτυξη

Μετά το σχεδιασμό, ακολουθεί η φάση της ανάπτυξης του API. Αφού πληρούνται οι απαιτήσεις που καθορίστηκαν ήδη, γίνεται η υλοποίηση. Εδώ περιλαμβάνεται και η σύνταξη του οδηγού χρήσης από την ομάδα προγραμματιστών που αναπτύσσει το API, που αναλύει τα διάφορα τμήματα του και παρέχει οδηγίες για τη χρήση του από το κοινό.

- Δοκιμές

Στη φάση των δοκιμών, η ομάδα ελέγχει προσεκτικά το API για αδυναμίες και πιθανά λάθη που μπορεί να προκαλέσουν μη αναμενόμενη συμπεριφορά. Αυτό είναι ζωτικής σημασίας πριν την κυκλοφορία του λογισμικού και συνήθως δεν είναι το τελειωτικό. Οποιαδήποτε ανεπιθύμητη πτυχή γίνει αντιληπτή αναμένεται να διορθωθεί με μελλοντικές ενημερώσεις (patches).

- Διαχείριση

Εφόσον το API επιτύχει στις δοκιμές και επιβεβαιωθεί η συμβατότητά του με τον σχεδιασμό, είναι έτοιμο για κυκλοφορία. Η φιλοξενία του σε αξιόπιστες πλατφόρμες είναι ζωτικής σημασίας για λόγους ασφάλειας και ποιότητας. Μια πρώτη έκδοση (beta) επιτρέπει στους χρήστες να το δοκιμάσουν και να αναφέρουν προβλήματα που μπορεί να σημειωθούν. Μετά την κυκλοφορία, λαμβάνονται υπόψη στατιστικά για βελτίωσή του.

- Απόσυρση

Εάν για οποιοδήποτε λόγο δεν καθίσταται δυνατή η συνέχιση της διαχείρισης του API ή υπάρχει κάποιο σχέδιο αντικατάστασής του, είναι σημαντικό να μην εγκαταλειφθεί η υποδομή. Πρέπει να εξεταστούν εναλλακτικές για τους χρήστες που ήδη το χρησιμοποιούν. Ο απότομος τερματισμός ενός API μπορεί να προκαλέσει προβλήματα σε όλες τις εφαρμογές που το χρησιμοποιούν ως βάση, λειτουργούν ζωντανά και διαθέτουν πλήθος χρηστών.

Κεφάλαιο 3ο: Εργαλεία υλοποίησης

3.1 Xcode

Το Xcode αποτελεί το ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (IDE), που αναπτύσσει και διαθέτει η Apple και το οποίο προορίζεται για τη δημιουργία εφαρμογών που απαρτίζουν τα λειτουργικά της συστήματα και πλαισιώνουν τις συσκευές της. Διαθέτει όσα εργαλεία διευκολύνουν έναν προγραμματιστή στην ανάπτυξη κώδικα, όπως επίσης και προσομοίωση σε συσκευές της Apple που βρίσκονται την παρούσα στιγμή στην αγορά. Το συγκεκριμένο χαρακτηριστικό, αυτό της προσομοίωσης, είναι καθοριστικής σημασίας, καθώς πριν να κυκλοφορήσει μια εφαρμογή πρέπει να γίνεται και η απαραίτητη δοκιμή ώστε να είναι συμβατή και ομοιόμορφη σε όσες συσκευές υποστηρίζει. Η προσομοίωση το καθιστά απλούστερο αυτό, καθώς δεν υπάρχει η δυνατότητα ένας προγραμματιστής να έχει στην κατοχή του όποια συσκευή κυκλοφορεί.

3.2 Swift

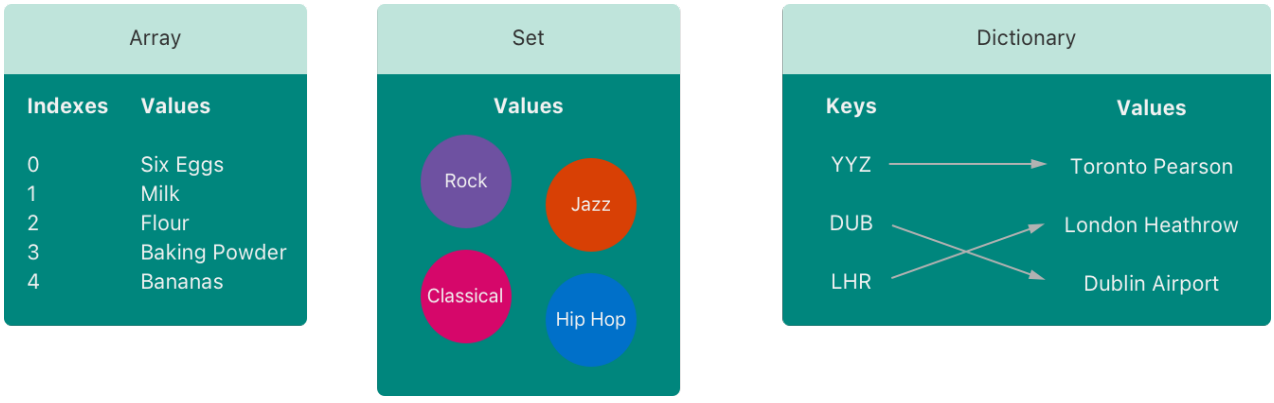
3.2.1 Γενικά

Η Swift είναι μια πρόσφατη γλώσσα προγραμματισμού υψηλού επιπέδου, με απλό συντακτικό, η οποία χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών που απαρτίζουν τα λειτουργικά συστήματα της Apple (iOS, macOS, watchOS, tvOS και πλέον visionOS), αλλά και σε διάφορες εκδόσεις του Linux, μιας και είναι με τη σειρά τους βασισμένες στο UNIX λειτουργικό.

Η γλώσσα προγραμματισμού Swift εμφανίστηκε το 2014 και λαμβάνοντας υπόψη τη διαρκώς αυξανόμενη πορεία των πωλήσεων iOS συσκευών, σημειώνει ραγδαία ανάπτυξη και ζήτηση προγραμματιστών. Η Swift ανήκει στις γλώσσες προγραμματισμού υψηλού επιπέδου, προσφέροντας στο χρήστη απλό και εύπεπτο συντακτικό που βοηθά στην εξοικείωση και τη χρήση από τον προγραμματιστή. Η ταχύτητα και η αξιοπιστία της την έχουν ανεβάσει αρκετά στις προτιμήσεις των υλοποιήσεων εφαρμογών τόσο για κινητές συσκευές όσο και μη φορητών.

3.2.2 Τύποι δεδομένων της Swift

Στη Swift χρησιμοποιούνται τύποι δεδομένων που αποτελούν τροποποιήσεις των ήδη πρωταρχικών τύπων που είναι γνωστοί και από άλλες γλώσσες προγραμματισμού που έχουν προηγηθεί χρονολογικά, δηλαδή int, double, float, string, bool. Ένας επιπλέον τύπος που διατίθεται είναι τα tuples, που αποτελούν έναν τρόπο αποθήκευσης δεδομένων διαφορετικών τύπων με τη χρήση μιας μόνο δομής. Επίσης συναντώνται οι τύποι Set, Dictionary και η διαδεδομένη δομή Array.



Εικόνα 1 : Collection Types [12]

Επιπρόσθετα στη Swift υπάρχει η δυνατότητα δήλωσης μεταβλητών ενός τύπου ως optional, προσδίδοντας τους δηλαδή και τη δυνατότητα διαχείρισης του ενδεχομένου μη αρχικοποίησης της μεταβλητής με κάποια τιμή. Τα Optionals αναφέροντα σε μεταβλητές που στην πορεία μπορεί να έχουν ή όχι μια τιμή.

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

Κώδικας 1: Σύντομος και εκτενής τρόπος δήλωσης optional μεταβλητών [13].

Πιο συγκεκριμένα, πίσω από μια Optional μεταβλητή, βρίσκεται ένα enum (enumeration) με δύο cases, περιπτώσεις, την Optional.none, που ισοδυναμεί με το nil και την Optional.some(value) που κρατά αποθηκευμένη μια τιμή, όπως φαίνεται και στο παρακάτω παράδειγμα κώδικα.

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

Κώδικας 2: Εξήγηση του enum των optional μεταβλητών [13]

Σε αντίθεση με τις περισσότερες γλώσσες προγραμματισμού, η Swift επιτρέπει την χρήση σχεδόν όλων των Unicode χαρακτήρων για την ονοματοδοσία μιας μεταβλητής. Επιτρέπεται επιπλέον (αν και δε συνιστάται) η χρήση δεσμευμένων λέξεων. Σε αυτή την περίπτωση το όνομα της μεταβλητής πρέπει να περικλείεται από μονά ανάποδα εισαγωγικά (`), προκειμένου να γίνει αποδεκτό. Παρόλα αυτά, διατηρούνται βασικοί κανόνες, όπως η απαγόρευση της χρήσης αριθμών στην αρχή του ονόματος, κενών χαρακτήρων ή μαθηματικών συμβόλων.

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

Εικόνα 2: Περίεργοι τρόποι δήλωσης μεταβλητών που επιτρέπονται στην Swift [20]

3.2.3 Χαρακτηριστικά της γλώσσας Swift

Παρακάτω παρατίθενται χαρακτηριστικά της Swift, κάνοντας χρήση της αγγλικής τους ορολογίας και επεξήγησή τους.

3.2.3.1 Multi-paradigm

Στη Swift υπάρχει η δυνατότητα υποστήριξης διαφόρων προγραμματιστικών προσεγγίσεων, όπως:

1. Object oriented

Δηλαδή, αντικειμενοστρέφεια. Με βάση αυτή την αρχή, διαχωρίζονται οι οντότητες ενός προγράμματος σε αντικείμενα, τα οποία ανήκουν σε μια κοινή ομάδα που καλείται ‘κλάση’. Τα αντικείμενα μιας κλάσης έχουν κοινά χαρακτηριστικά και αντίστοιχα κοινές μεθόδους διαχείρισης και επεξεργασίας αυτών. Με τη χρήση της αντικειμενοστρέφειας, ο κώδικας γίνεται πιο κατανοητός, ιδίως όταν πρόκειται για επαναχρησιμοποιούμενα κομμάτια κώδικα.

Στην περίπτωση της αντικειμενοστρέφειας, κάθε κλάση οφείλει να διαθέτει και να διαχειρίζεται όλες τις απαραίτητες λειτουργίες της, έναντι του κυρίου προγράμματος (dynamic dispatch). Η πρακτική αυτή ενισχύεται από το γεγονός ότι τα αντικείμενα και οι λειτουργίες που επικοινωνούν άμεσα είναι τα ορατά

στο υπόλοιπο πρόγραμμα. Άλλα αντικείμενα και λειτουργίες διαχειρίζονται εσωτερικά και δεν επηρεάζονται από γειτονικά στοιχεία του προγράμματος (δηλαδή, ενθυλάκωση).

Στον αντικειμενοστραφή προγραμματισμό, υπάρχει επίσης η έννοια της κληρονομικότητας, όπου τα αντικείμενα μπορούν να συσχετίζονται και να κληρονομούν χαρακτηριστικά και μεθόδους από άλλα αντικείμενα σε σχέση συγγένειας. Αυτό επιτρέπει την επέκταση των προηγούμενων εννοιών με νέα χαρακτηριστικά και μεθόδους.

Επιπλέον, ο πολυμορφισμός είναι μια ιδιότητα στον αντικειμενοστραφή προγραμματισμό που αναφέρεται στη δυνατότητα των συγγενικών αντικειμένων να εκτελούν τόσο τον κώδικα της κλάσης-γονέα όσο και του αντίστοιχου ανανεωμένου κώδικα που τα αφορά. Αυτό σημαίνει ότι τα αντικείμενα μπορούν να παρουσιάζουν διαφορετική συμπεριφορά ανάλογα με τον τύπο τους, προσφέροντας μεγαλύτερη ευελιξία στον προγραμματισμό.

2. Functional

Σε αντίθεση με την αντικειμενοστραφή προγραμματισμό, ο συναρτησιακός (functional) προγραμματισμός προσεγγίζει τα ζητούμενα ως αποτέλεσμα εκτέλεσης συναρτήσεων και επικεντρώνεται στην επίλυση προβλημάτων μέσω αλγοριθμικών προσεγγίσεων. Σε αντίθεση με την έμφαση στα αντικείμενα και τις μεθόδους, που χαρακτηρίζει την αντικειμενοστραφή προγραμματιστική παράδοση, ο συναρτησιακός προγραμματισμός χρησιμοποιεί μεταβλητές και συναρτήσεις. Αυτή η προσέγγιση παρέχει μια πιο άμεση, αλγοριθμική λύση στην επίλυση προβλημάτων, αναδεικνύοντας τη συνάρτηση ως βασική οντότητα εκτέλεσης κώδικα.

3. Imperative

Στον προστακτικό προγραμματισμό, η εκτέλεση των εντολών έχει ως άμεσο σκοπό την αλλαγή κατάστασης των αντικειμένων, αντίθετα με τον συναρτησιακό προγραμματισμό που προτιμά τα αμετάβλητα αντικείμενα. Ο προστακτικός προγραμματισμός είναι πιο περίπλοκος σε σχέση με τον συναρτησιακό και πιο δύσκολος στην υλοποίηση όταν απαιτείται παράλληλη εκτέλεση του προγράμματος. Σε αυτόν τον τύπο προγραμματιστικής λογικής, η επικεντρωμένη αλλαγή κατάστασης αντικειμένων δίνει τη δυνατότητα για άμεσο έλεγχο της ροής του προγράμματος, αλλά μπορεί να καθιστά πιο δύσκολη τη διαχείριση πολύπλοκων διεργασιών και πόρων, ιδίως σε περιβάλλοντα παράλληλης εκτέλεσης.

4. Block-structed

Ο ορισμός συνόλου εντολών εμφανίζεται συχνά σε προγράμματα με σειριακή εκτέλεση, όπου οι εντολές που εκτελούνται εξαρτώνται από κάποιον έλεγχο ή συνθήκη. Για παράδειγμα, μπορεί να υπάρχει η ανάγκη να εκτελεστούν συγκεκριμένες εντολές μόνον εάν μια συνθήκη είναι αληθής, ενώ η εκτέλεση του προγράμματος να συνεχίζεται μετά από μια συγκεκριμένη ομάδα εντολών, η οποία πρέπει σε αυτή την περίπτωση να αγνοηθεί. Σε αυτό το πλαίσιο, η χρήση αγκύλων ορίζει την πρώτη και τη δεύτερη ομάδα εντολών, με το πρόγραμμα να τις αντιμετωπίζει ως ξεχωριστές εντολές, εξασφαλίζοντας έτσι την επιθυμητή ροή εκτέλεσης.

3.2.3.2 Control Flow

Ο έλεγχος μέσω συνθηκών αποτελεί κοινή προγραμματιστική πρακτική, καθώς επιτρέπει την εκτέλεση συγκεκριμένων εντολών βάσει της ικανοποίησης ή μη κάποιας συνθήκης που έχει οριστεί, επιτυγχάνοντας έτσι αλγοριθμική αντιμετώπιση του προβλήματος. Επιπλέον, η επανάληψη αποτελεί συχνή ανάγκη στην προγραμματιστική διαδικασία, όπου ένα τμήμα του κώδικα επαναλαμβάνεται για συγκεκριμένο πλήθος επαναλήψεων ή μέχρις ότου μια συνθήκη ικανοποιηθεί. Η δομή επανάληψης

είναι ουσιαστικό στοιχείο στην πλειοψηφία των γλωσσών προγραμματισμού και ενσωματώνεται στις συνθήκες ελέγχου.

3.2.3.2.1 Συνθήκη ελέγχου if

Η συνθήκη ελέγχου "if" είναι ένα σύνηθες στοιχείο στον προγραμματισμό. Όπως υποδηλώνει το όνομά της, η "if" (αν) ελέγχει μια συνθήκη για την αληθή ή ψευδή κατάστασή της. Αν η συνθήκη είναι αληθής, τότε εκτελούνται οι αντίστοιχες εντολές που περικλείονται από τη συνθήκη. Αν, ωστόσο, η συνθήκη είναι ψευδής, δηλαδή δεν ισχύει, τότε οι εντολές της συνθήκης δεν εκτελούνται.

Σε ορισμένες περιπτώσεις, ο έλεγχος ενδέχεται να χρειαστεί να επεκταθεί σε περισσότερους έλεγχους "else if", προκειμένου να εξετάσει πολλαπλές πιθανές περιπτώσεις. Τελικά, αν κανένας από αυτούς οι έλεγχοι δεν είναι αληθής, μπορεί να χρησιμοποιηθεί μια τελική δήλωση "else" που θα εκτελέσει αντίστοιχες εντολές.

```
1. var temperatureInFahrenheit = 30
2. if temperatureInFahrenheit <= 32 {
3.     print("It's very cold. Consider wearing a scarf.")
4. }
5. // Prints "It's very cold. Consider wearing a scarf."
```

Κώδικας 3: Η συνθήκη ελέγχου if [15]

```
1. temperatureInFahrenheit = 40
2. if temperatureInFahrenheit <= 32 {
3.     print("It's very cold. Consider wearing a scarf.")
4. } else {
5.     print("It's not that cold. Wear a T-shirt.")
6. }
7. // Prints "It's not that cold. Wear a T-shirt."
```

Κώδικας 4: Η συνθήκη ελέγχου if με else διαχείριση αποτελέσματος συνθήκης [15]

```
1. temperatureInFahrenheit = 90
2. if temperatureInFahrenheit <= 32 {
3.     print("It's very cold. Consider wearing a scarf.")
4. } else if temperatureInFahrenheit >= 86 {
5.     print("It's really warm. Don't forget to wear sunscreen.")
6. } else {
7.     print("It's not that cold. Wear a T-shirt.")
8. }
9. // Prints "It's really warm. Don't forget to wear sunscreen."
```

Κώδικας 5: Η συνθήκη ελέγχου if με πρόσθετους else if ελέγχους [15]

3.2.3.2.2 Συνθήκη ελέγχου Switch

Η συνθήκη ελέγχου switch στον προγραμματισμό είναι παρόμοια με τη δομή "if...else if", με τη διαφορά ότι αντί να χρησιμοποιούμε επανειλημμένα το "else if", καθορίζουμε περιπτώσεις "case". Σε κάθε περίπτωση, αν η συνθήκη είναι αληθής, δηλαδή ισχύει, τότε εκτελούνται οι αντίστοιχες εντολές. Αν καμία από τις περιπτώσεις δεν ισχύει, τότε εκτελείται μια προκαθορισμένη περίπτωση "default" που περιέχει τις αντίστοιχες εντολές.

```
1. let someCharacter: Character = "z"
2. switch someCharacter {
3. case "a":
4.     print("The first letter of the Latin alphabet")
5. case "z":
```

```
 6.     print("The last letter of the Latin alphabet")
 7. default:
 8.     print("Some other character")
 9. }
10. // Prints "The last letter of the Latin alphabet"
```

Κώδικας 6: Η συνθήκη ελέγχου Switch [15]

3.2.3.3.3 Δομή επανάληψης for

Η δομή επανάληψης for χρησιμοποιείται για τον εύκολο και αποτελεσματικό προγραμματισμό επαναλαμβανόμενων κομματιών κώδικα. Στην ουσία, καθορίζεται μια συνθήκη που, εφόσον είναι αληθής, εκτελούνται οι αντίστοιχες εντολές. Η επανάληψη συνεχίζεται μέχρι τη στιγμή που η συνθήκη δεν ισχύει πλέον, και τότε ο κώδικας βγαίνει από τον βρόχο.

```
1. let names = ["Anna", "Alex", "Brian", "Jack"]
2. for name in names {
3. print("Hello, \(name)!")
4. }
5. // Hello, Anna!
6. // Hello, Alex!
7. // Hello, Brian!
8. // Hello, Jack!

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

1. for index in 1...5 {
2. print("\(index) times 5 is \(index * 5)")
3. }
4. // 1 times 5 is 5
5. // 2 times 5 is 10
6. // 3 times 5 is 15
7. // 4 times 5 is 20
8. // 5 times 5 is 25

1. let base = 3
2. let power = 10
3. var answer = 1
4. for _ in 1...power {
5. answer *= base
6. }
7. print("\(base) to the power of \(power) is \(answer)")
8. // Prints "3 to the power of 10 is 59049"

1. let minutes = 60
2. for tickMark in 0..<minutes {
3. // render the tick mark each minute (60 times)
4. }

1. let minuteInterval = 5
2. for tickMark in stride(from: 0, to: minutes, by: minuteInterval) {
3. // render the tick mark every 5 minutes (0, 5, ... 45, 50, 55)
```

Κώδικας 7: Η δομή επανάληψης for [15]

3.2.3.3 Functions

Η δυνατότητα ορισμού μιας συνάρτησης και η δυνατότητα παροχής κατάλληλων παραμέτρων επιτρέπει την επίτευξη επιθυμητών αποτελεσμάτων με επαναχρησιμοποιήσιμο κώδικα. Οι συναρτήσεις

αποτελούν τμήματα κώδικα που μπορούν να χρησιμοποιηθούν ξανά, προσφέροντας οργάνωση και αποτελεσματικότητα στον προγραμματισμό. Κάθε συνάρτηση έχει συγκεκριμένο τύπο δεδομένων, ανάλογα με την τιμή που επιστρέφει, παρόμοια με τις μεταβλητές.

```
1. func greet(person: String) -> String {
2. let greeting = "Hello, " + person + "!"
3. return greeting
4. }

1. func greetAgain(person: String) -> String {
2. return "Hello again, " + person + "!"
3. }
4. print(greetAgain(person: "Anna"))
5. // Prints "Hello again, Anna!"

1. func sayHelloWorld() -> String {
2. return "hello, world"
3. }
4. print(sayHelloWorld())
5. // Prints "hello, world"

1. func greet(person: String, alreadyGreeted: Bool) -> String {
2. if alreadyGreeted {
3. return greetAgain(person: person)
4. } else {
5. return greet(person: person)
6. }
7. }
8. print(greet(person: "Tim", alreadyGreeted: true))
9. // Prints "Hello again, Tim!"

1. func greet(person: String) {
2. print("Hello, \(person)!")
3. }
4. greet(person: "Dave")
5. // Prints "Hello, Dave!"
```

```
1. func minMax(array: [Int]) -> (min: Int, max: Int) {
2. var currentMin = array[0]
3. var currentMax = array[0]
4. for value in array[1..<array.count] {
5. if value < currentMin {
6. currentMin = value
7. } else if value > currentMax {
8. currentMax = value
9. }
10. }
11. return (currentMin, currentMax)
12. }
13. let bounds = minMax(array: [8, -6, 2, 109, 3, 71])
14. print("min is \(bounds.min) and max is \(bounds.max)")
15. // Prints "min is -6 and max is 109"
```

Κώδικας 8: Συναρτήσεις [17]

3.2.3.4 Generics

Οι γενικές συναρτήσεις (Generics) επιτρέπουν τη δημιουργία κώδικα που μπορεί να διαχειρίζεται διάφορους τύπους δεδομένων. Παρόμοια με τα πρότυπα (templates), οι γενικές συναρτήσεις

σχεδιάζονται έτσι ώστε να είναι ευέλικτες και να μπορούν να εκτελεστούν για οποιοδήποτε είδος δεδομένων. Αυτό συμβάλλει στην ανακύκλωση κώδικα, στη μείωση του όγκου του, και στην επαναχρησιμοποίησή του. Οι γενικές συναρτήσεις μπορούν επίσης να περιορίζονται σε συγκεκριμένους τύπους δεδομένων μέσω των περιορισμών (constraints), παρέχοντας ακόμα περισσότερη ευελιξία.

```
1. func swapTwoStrings(_ a: inout String, _ b: inout String) {
2. let temporaryA = a
3. a = b
4. b = temporaryA
5. }
6. func swapTwoDoubles(_ a: inout Double, _ b: inout Double) {
7. let temporaryA = a
8. a = b
9. b = temporaryA
10. }
11. func swapTwoValues<T>(_ a: inout T, _ b: inout T) {
12. let temporaryA = aA = a
13. a = b
14. b = temporaryA
15. }
16. var someInt = 3
17. var anotherInt = 107
18. swapTwoValues(&someInt, &anotherInt)
19. // someInt is now 107, and anotherInt is now 3
20. var someString = "hello"
21. var anotherString = "world"
22. swapTwoValues(&someString, &anotherString)
23. // someString is now "world", and anotherString is now "hello"
```

Κώδικας 9: Παραδείγματα Generics [18]

```
1. func findIndex<T: Equatable>(of valueToFind: T, in array:[T]) -> Int? {
2. for (index, value) in array.enumerated() {
3. if value == valueToFind {
4. return index
5. }
6. }
7. return nil
8. }
9. let doubleIndex = findIndex(of: 9.3, in: [3.14159, 0.1, 0.25])
10. // doubleIndex is an optional Int with no value, because 9.3 isn't in the array
11. let stringIndex = findIndex(of: "Andrea", in: ["Mike", "Malcolm", "Andrea"])
12. // stringIndex is an optional Int containing a value of 2
```

Κώδικας 10: Παραδείγματα Generics [18]

3.2.3.5 Structures & Classes

Ο αντικειμενοστραφής σχεδιασμός βασίζεται σε δομές και κλάσεις. Όπως και με τις κλάσεις, στις δομές ορίζονται χαρακτηριστικά και μέθοδοι που περιγράφουν ένα συγκεκριμένο σύνολο δεδομένων. Ωστόσο, οι δομές δεν υποστηρίζουν κληρονομικότητα και δεν διαθέτουν αυτόματες μεθόδους αποδέσμευσης μνήμης.

Μια σημαντική διαφορά είναι ότι τα Structs είναι τύπος αξίας δεδομένων, ενώ οι κλάσεις είναι τύπος αναφοράς. Τα Structs αντιγράφονται κατά την ανάθεσή τους σε μια νέα μεταβλητή, ενώ οι κλάσεις δημιουργούν μια αναφορά προς το αρχικό αντικείμενο. Αυτό σημαίνει ότι οποιαδήποτε αλλαγή σε μια κλάση επηρεάζει την αρχική και το αντίθετο.

```
1. struct SomeStructure {
 2. // structure definition goes here
 3. }
 4.
 5. class SomeClass {
 6. // class definition goes here
 7. }
 8.
 9. struct Resolution {
10. var width = 0
11. var height = 0
12. }
13.
14. class VideoMode {
15. var resolution = Resolution()
16. var interlaced = false
17. var frameRate = 0.0
18. var name: String?
19. }
```

Κώδικας 11: Σύγκριση Struct - Class [14]

```
1. enum CompassPoint {
2.   case north, south, east, west
3. mutating func turnNorth() {
4. self = .north
5. }
6. }
7. var currentDirection = CompassPoint.west
8. let rememberedDirection = currentDirection
9. currentDirection.turnNorth()
10. print("The current direction is \(currentDirection)")
11. print("The remembered direction is \(rememberedDirection)")
12. // Prints "The current direction is north"
13. // Prints "The remembered direction is west"
```

Κώδικας 12: Δομή Enum [31]

```
1. let tenEighty = VideoMode()
2. tenEighty.resolution = hd
3. tenEighty.interlaced = true
4. tenEighty.name = "1080i"
5. tenEighty.frameRate = 25.0
6. let alsoTenEighty = tenEighty
7. alsoTenEighty.frameRate = 30.0
8. print("The frameRate property of tenEighty is now \(tenEighty.frameRate)")
9. // Prints "The frameRate property of tenEighty is now 30.0"
1. let hd = Resolution(width: 1920, height: 1080)
2. var cinema = hd
3. cinema.width = 2048
4. print("cinema is now \(cinema.width) pixels wide")
5. // Prints "cinema is now 2048 pixels wide"
6. print("hd is still \(hd.width) pixels wide")
7. // Prints "hd is still 1920 pixels wide"
```

Κώδικας 13: Σύγκριση χειρισμού by value και by reference [14]

3.2.3.6 Error Handling

Η διαχείριση λαθών αναφέρεται στον τρόπο που ένα πρόγραμμα αντιμετωπίζει και αντιδρά σε αναπάντεχες καταστάσεις που μπορεί να προκύψουν κατά την εκτέλεσή του. Ακόμα και αν ο σχεδιασμός είναι άριστος, υπάρχουν πάντα περιπτώσεις που δεν μπορούν να προβλεφθούν.

Σε αυτές τις περιπτώσεις, χρησιμοποιούνται τεχνικές διαχείρισης λαθών που επικοινωνούν στον χρήστη την ύπαρξη ενός προβλήματος ή προσπαθούν να αντιμετωπίσουν αυτόματα το σφάλμα. Αντί να διακόπτεται η λειτουργία του προγράμματος, εφαρμόζονται διορθωτικά μέτρα που επιτρέπουν τη συνέχιση της εκτέλεσης.

```
1. func canThrowErrors() throws -> String
2. func cannotThrowErrors() -> String
```

Κώδικας 14: Διαχείριση λαθών [16]

```
1. let favoriteSnacks = [
 2.     "Alice": "Chips",
 3.     "Bob": "Licorice",
 4.     "Eve": "Pretzels",
 5. ]
 6. func buyFavoriteSnack(person: String, vendingMachine: VendingMachine) throws {
 7.     let snackName = favoriteSnacks[person] ?? "Candy Bar"
 8.     try vendingMachine.vend(itemNamed: snackName)
 9. }
```

```
 1. var vendingMachine = VendingMachine()
 2. vendingMachine.coinsDeposited = 8
 3. do {
 4.     try buyFavoriteSnack(person: "Alice", vendingMachine: vendingMachine)
 5.     print("Success! Yum.")
 6. } catch VendingMachineError.invalidSelection {
 7.     print("Invalid Selection.")
 8. } catch VendingMachineError.outOfStock {
 9.     print("Out of Stock.")
10. } catch VendingMachineError.insufficientFunds(let coinsNeeded) {
11.     print("Insufficient funds. Please insert an additional \(coinsNeeded) coins.")
12. } catch {
13.     print("Unexpected error: \(error).")
14. }
15. // Prints "Insufficient funds. Please insert an additional 2 coins."
```

```
 1. func someThrowingFunction() throws -> Int {
 2.     // ...
 3. }
 4.
 5. let x = try? someThrowingFunction()
 6.
 7. let y: Int?
 8. do {
 9.     y = try someThrowingFunction()
10. } catch {
11.     y = nil
12. }
```

```
1. let photo = try! loadImage(atPath: "./Resources/John Appleseed.jpg")
```

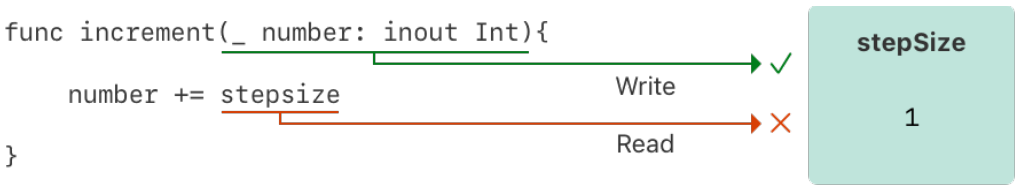
Κώδικας 15: Παραδείγματα συναρτήσεων για διαχείριση λαθών [16]

3.2.3.7. Memory management

Η διαχείριση μνήμης απαιτεί από τον προγραμματιστή να αντιμετωπίσει προβλήματα που προκύπτουν κατά την ανάθεση και απελευθέρωση μνήμης κατά την εκτέλεση ενός προγράμματος. Στην Swift, όπως και σε πολλές άλλες γλώσσες προγραμματισμού, υπάρχουν κανόνες για την προώθηση ασφαλών πρακτικών. Απαγορεύεται η χρήση μη αρχικοποιημένων μεταβλητών, η πρόσβαση σε απελευθερωμένη μνήμη και η υπέρβαση ορίων δομών όπως οι πίνακες και οι λίστες.

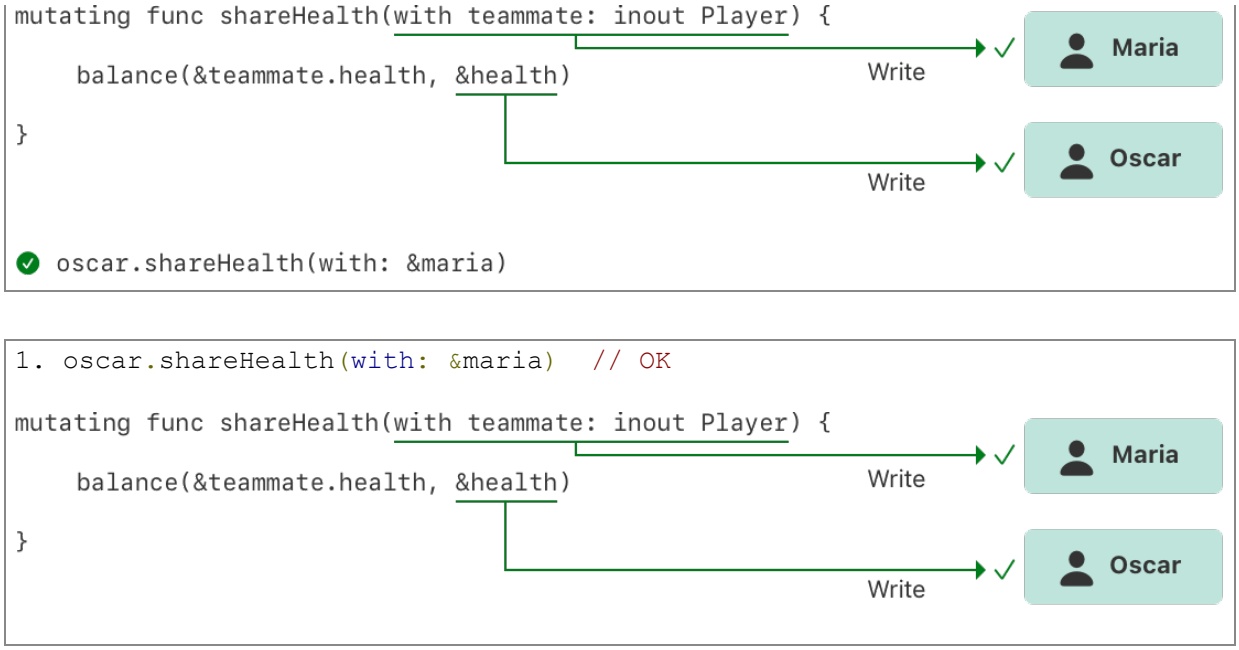
Σημαντικό είναι ότι η Swift αναλαμβάνει την αυτόματη διαχείριση μνήμης σε μεγάλο βαθμό, αντίθετα με γλώσσες όπως η C και η C++, όπου η διαχείριση μνήμης παραμένει υπό τον έλεγχο του προγραμματιστή. Αυτό μειώνει τον κίνδυνο εμφάνισης συχνών λαθών που σχετίζονται με τη μνήμη.

```
1. var stepSize = 1
2.
3. func increment(_ number: inout Int) {
4.     number += stepSize
5. }
6.
7. increment(&stepSize)
8. // Error: conflicting accesses to stepSize
```




```
1. // Make an explicit copy.
2. var copyOfStepSize = stepSize
3. increment(©OfStepSize)
4.
5. // Update the original.
6. stepSize = copyOfStepSize
7. // stepSize is now 2
```

```
 1. struct Player {
 2.     var name: String
 3.     var health: Int
 4.     var energy: Int
 5.
 6.     static let maxHealth = 10
 7.     mutating func restoreHealth() {
 8.         health = Player.maxHealth
 9.     }
10. }
11. extension Player {
12.     mutating func shareHealth(with teammate: inout Player) {
13.         balance(&teammate.health, &health)
14.     }
15. }
16.
17. var oscar = Player(name: "Oscar", health: 10, energy: 10)
18. var maria = Player(name: "Maria", health: 5, energy: 10)
19. oscar.shareHealth(with: &oscar)
20. // Error: conflicting accesses to Oscar
```

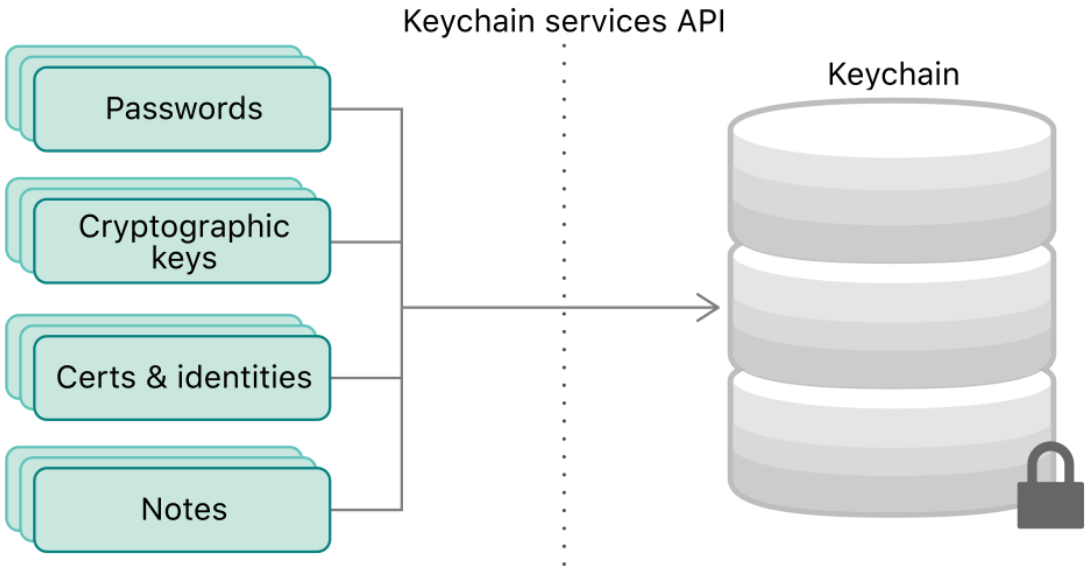


Κώδικας 16: Παραδείγματα λάθους διαχείρισης μνήμης και επίλυση [19]

3.3 Keychain

3.3.1 Γενικά

Το Keychain αποτελεί ένα API διαχείρισης κωδικών πρόσβασης που ανήκει στην Apple και εισήχθη με το macOS 8.6 το 1999. Από τότε, συνεχίζει να εξυπηρετεί τη διαχείριση κωδικών πρόσβασης για διάφορες χρήσεις. Το Keychain διαχειρίζεται κωδικούς πρόσβασης για ιστοσελίδες, FTP διακομιστές, SSH επικοινωνίες, κρυπτογράφηση δίσκων, δημιουργία ιδιωτικών κλειδιών, ψηφιακών πιστοποιητικών και πολλά άλλα. Η λειτουργία του εκτείνεται σε πολλούς τομείς ασφαλείας, προσφέροντας ένα κεντρικό μέρος για τη διαχείριση ευαίσθητων πληροφοριών.



Εικόνα 3: Keychain API [10]

3.3.2 Ιστορικά στοιχεία

Αρχικά, το Keychain αναπτύχθηκε ως βοηθητική υπηρεσία για το PowerTalk, το σύστημα ηλεκτρονικού ταχυδρομείου της Apple. Σε αυτό το πλαίσιο, χρησιμοποιήθηκε για τη διαχείριση των στοιχείων πρόσβασης σε πολλαπλούς λογαριασμούς ταυτόχρονα, όπως αυτοί που μπορεί να είχε ένας χρήστης στο PowerTalk.

Αντίστοιχα, σε άλλα συστήματα όπως το Gmail, δίνεται η επιλογή στους χρήστες να χρησιμοποιούν λογαριασμούς από διάφορες πλατφόρμες (π.χ., Yahoo, Outlook) ή ακόμα και να διαχειρίζονται πολλαπλούς λογαριασμούς της ίδιας πλατφόρμας. Αυτό οδηγεί στην ανάγκη απομνημόνευσης πολλαπλών στοιχείων πρόσβασης. Το Keychain πρωτοποριακά λύνοντας αυτό το πρόβλημα, ήταν μία από τις πρώτες υπηρεσίες στον χώρο που αντιμετώπισε αυτήν την πρόκληση και εξακολουθεί να χρησιμοποιείται μέχρι και σήμερα.

3.3.3 Ασφάλεια

Το Keychain είναι προεγκατεστημένο τόσο στα iOS όσο και στα macOS, αν και υπάρχουν διαφορετικές εκδόσεις για κάθε λειτουργικό σύστημα. Στο iOS, η έκδοση παρέχει λιγότερες λειτουργίες και δεν επιτρέπει τον διαμοιρασμό κωδικών μεταξύ διαφορετικών εφαρμογών, ενισχύοντας ωστόσο την ασφάλεια του συστήματος.

Ο τρόπος λειτουργίας του είναι ότι ο χρήστης ορίζει έναν ισχυρό κωδικό, που χρησιμοποιείται για την προστασία όλων των υπόλοιπων κωδικών του. Έτσι, αρκεί η απομνημόνευση ενός κωδικού για να διαχειριστεί αυτόματα τους υπόλοιπους. Αυτό είναι σημαντικό για την ασφάλεια, αφού δεν συνιστάται η χρήση του ίδιου κωδικού για πολλαπλές πλατφόρμες.

Οι κωδικοί αποθηκεύονται κρυπτογραφημένοι με τον αλγόριθμο AES-256-GCM. Οι χρόνοι κρυπτογράφησης, αποκρυπτογράφησης και συγχρονισμού με το iCloud διαφέρουν ανάλογα με τους τύπους δεδομένων. Η Apple παρέχει ένα εγχειρίδιο για την κατατόπιση των χρηστών στη χρήση του εργαλείου.

3.3.4 Κλείδωμα & ξεκλείδωμα

Το Keychain προσφέρει τη δυνατότητα χρήσης του κωδικού της συσκευής για τη διαχείριση των κωδικών, προσφέροντας έτσι επιπλέον άνεση. Αυτό μπορεί να οδηγήσει σε μειωμένο επίπεδο ασφάλειας, καθώς η χρήση του ίδιου κωδικού είναι πιο εύκολη, αλλά παρέχεται επίσης η επιλογή να οριστεί ξεχωριστός κωδικός για το Keychain.

Είναι απαγορευμένη η χρήση κενών κωδικών, ενώ η λειτουργία κλειδώματος μπορεί να ρυθμιστεί είτε να ενεργοποιείται αυτόματα, για παράδειγμα κατά περίοδο αδράνειας, είτε να ενεργοποιείται κατά βούληση του χρήστη.

3.3.5 Συγχρονισμός

Αν ο χρήστης επιλέξει να μοιραστεί τον κωδικό της συσκευής του με το Keychain, ο κωδικός του εργαλείου αυτοματοποιημένα ενημερώνεται με τον κωδικό του χρήστη. Ωστόσο, ενδέχεται να προκύψουν προβλήματα εάν η αλλαγή προέρχεται από συσκευή που δεν υποστηρίζεται από την Apple ή εάν η αλλαγή γίνεται με κάποιον τρόπο εκτός της προεπιλεγμένης γραφικής διεπαφής που διατίθεται για τον σκοπό αυτό.

3.4 Βιβλιοθήκες

3.4.1 Alamofire

Η Alamofire αποτελεί μια προηγμένη βιβλιοθήκη της Swift για τη διαχείριση HTTP δικτυακών επικοινωνιών σε περιβάλλοντα macOS και iOS της Apple. Αξιοποιώντας το URL Loading σύστημα της Apple, ενσωματώνει βασικά API όπως το URLSession και το URLSessionTask, παρέχοντας παράλληλα ευχρηστία και επιπλέον δυνατότητες για την ανάπτυξη εφαρμογών που χρησιμοποιούν το πρωτόκολλο επικοινωνίας HTTP.

Η Alamofire αναπτύχθηκε αποκλειστικά με Swift, αξιοποιώντας τις πιο πρόσφατες τεχνολογίες. Η βιβλιοθήκη διευκολύνει την επικοινωνία με διακομιστές HTTP και προσφέρει πρόσθετες δυνατότητες πέραν των βασικών λειτουργιών που παρέχονται από το Foundation framework της Apple. Με τη χρήση της Alamofire, οι προγραμματιστές επωφελούνται από μια ευέλικτη εργαλειοθήκη για τη διαχείριση των απαιτήσεων δικτύου στις εφαρμογές τους.

3.4.1.1 HTTP Μέθοδοι

1. GET - Ανάκτηση δεδομένων
2. HEAD - Ανάκτηση κεφαλίδας για δεικτοδότηση ή/και έλεγχο εγκυρότητας URL
3. POST - Αποστολή δεδομένων στον εξυπηρετητή
4. PUT - αντικατάσταση οντότητας με άλλη
5. DELETE - διαγραφή οντότητας/δεδομένων
6. CONNECT - ορισμός τρόπου επικοινωνίας με τον διακομιστή
7. OPTIONS - ρυθμίσεις για τον τρόπο επικοινωνίας
8. TRACE - παρακολούθηση εξέλιξης επικοινωνίας μέσω μηνυμάτων
9. PATCH - μερική τροποποίηση οντότητας

```swift
public struct HTTPMethod: RawRepresentable, Equatable, Hashable {
    public static let connect = HTTPMethod(rawValue: "CONNECT")
    public static let delete = HTTPMethod(rawValue: "DELETE")
    public static let get = HTTPMethod(rawValue: "GET")
    public static let head = HTTPMethod(rawValue: "HEAD")
    public static let options = HTTPMethod(rawValue: "OPTIONS")
    public static let patch = HTTPMethod(rawValue: "PATCH")
    public static let post = HTTPMethod(rawValue: "POST")
    public static let put = HTTPMethod(rawValue: "PUT")
    public static let query = HTTPMethod(rawValue: "QUERY")
    public static let trace = HTTPMethod(rawValue: "TRACE")

    public let rawValue: String

    public init(rawValue: String) {
        self.rawValue = rawValue
    }
}
```

Κώδικας 17: Alamofire – HTTP Μέθοδοι [30]

```swift
AF.request("https://httpbin.org/get")
AF.request("https://httpbin.org/post", method: .post)
```

```
3. AF.request("https://httpbin.org/put", method: .put)
4. AF.request("https://httpbin.org/delete", method: .delete)
```

Κώδικας 18: Ενδεικτικό παράδειγμα χρήσης HTTP μεθόδων [30]

3.4.1.2 Αιτήματα & παράμετροι

Σε συχνές περιπτώσεις, ένα αίτημα πρέπει να συνοδεύεται από επιπρόσθετες πληροφορίες που προσδιορίζουν τι ακριβώς αιτείται ή/και με ποιόν τρόπο ζητά απόκριση.

```
 1. struct Login: Encodable {
 2.     let email: String
 3.     let password: String
 4. }
 5.
 6. let login = Login(email: "test@test.test", password: "testPassword")
 7.
 8. AF.request("https://httpbin.org/post",
 9.            method: .post,
10.            parameters: login,
11.            encoder: JSONParameterEncoder.default).response { response in
12.     debugPrint(response)
13. }
```

Κώδικας 19: Ενδεικτικό παράδειγμα χρήσης HTTP μεθόδων με παραμέτρους [30]

```
1. let parameters = ["foo": "bar"]
2.
3. // All three of these calls are equivalent
4. AF.request("https://httpbin.org/get", parameters: parameters) // encoding defaults
to `URLEncoding.default`
5.    AF.request("https://httpbin.org/get",    parameters:    parameters,    encoder:
URLEncodedFormParameterEncoder.default)
6.    AF.request("https://httpbin.org/get",    parameters:    parameters,    encoder:
URLEncodedFormParameterEncoder(destination: .methodDependent))
7.
8. // https://httpbin.org/get?foo=bar
```

Κώδικας 20: Ενσωμάτωση παραμέτρων σε URLs [30]

3.4.1.3 Επικεφαλίδες

Με τις επικεφαλίδες εξασφαλίζεται η ανταλλαγή πληροφορίας σχετικά με το μήνυμα που αποστέλλεται.

```
1.let headers: HTTPHeaders = [
2.     "Authorization": "Basic VXNlcm5hbWU6UGFzc3dvcmQ=",
3.     "Accept": "application/json"
4.AF.request("https://httpbin.org/headers",headers:headers).responseDecodable(of:
DecodableType.self) { response in
7.     debugPrint(response)
8. }
```

Κώδικας 21: Επικεφαλίδες [30]

3.4.1.4 Επιβεβαίωση ανταπόκρισης

Αφού ο χρήστης κάνει ένα αίτημα, η Alamofire αναμένει μια απόκριση που μπορεί να πάρει διάφορες μορφές, όπως HTTP, JSON, Data, String, ή Decodable. Έπειτα, παρέχει διάφορα μέσα για την πρόσβαση σε αυτήν, όπως response, responseData, responseString, και responseDecodable.

```
1. AF.request("https://httpbin.org/get").response { response in
2.     debugPrint("Response: \(response)")
3. }
```

```
1. struct DecodableType: Decodable { let url: String }
2.
3. AF.request("https://httpbin.org/get").responseDecodable(of: DecodableType.self)
{ response in
4.     debugPrint("Response: \(response)")
5. }
```

```
1. AF.request("https://httpbin.org/get").responseString { response in
2.     debugPrint("Response: \(response)")
3. }
```

```
1. AF.request("https://httpbin.org/get").validate().responseData { response in
2.     debugPrint(response)
3. }
```

Κώδικας 22: Alamofire response types [30]

```
 1. AF.request("https://httpbin.org/get")
 2.     .validate(statusCode: 200..<300)
 3.     .validate(contentType: ["application/json"])
 4.     .responseData { response in
 5.         switch response.result {
 6.         case .success:
 7.             print("Validation Successful")
 8.         case let .failure(error):
 9.             print(error)
10.         }
11.     }
```

Κώδικας 23: Επιβεβαίωση ανταπόκρισης χειροκίνητα [30]

3.4.1.5 Αυθεντικοποίηση & Ταυτοποίηση

Η αυθεντικοποίηση και ταυτοποίηση μπορούν να επιτευχθούν είτε μέσω των επικεφαλίδων ή των τύπων δεδομένων URLCredential και URLAuthenticationChallenge, που διαθέτει η Alamofire.

```
1. let user = "user"
2. let password = "password"
3.
4. AF.request("https://httpbin.org/basic-auth/\(user)/\(password)")
5.     .authenticate(username: user, password: password)
6.     .responseDecodable(of: DecodableType.self) { response in
7.         debugPrint(response)
8.     }
```

Κώδικας 24: Ταυτοποίηση μέσω HTTP [30]

```
1. let user = "user"
2. let password = "password"
3.
4. let credential = URLCredential(user: user, password: password,
persistence: .forSession)
5.
6. AF.request("https://httpbin.org/basic-auth/\(user)/\(password)")
7.     .authenticate(with: credential)
8.     .responseDecodable(of: DecodableType.self) { response in
9.         debugPrint(response)
10.     }
```

Κώδικας 25: Ταυτοποίηση μέσω URLCredential [30]

```
let user = "user"
let password = "password"

let headers: HTTPHeaders = [.authorization(username: user, password: password)]

AF.request("https://httpbin.org/basic-auth/user/password", headers: headers)
    .responseDecodable(of: DecodableType.self) { response in
        debugPrint(response)
    }
```

Κώδικας 26: Χειροκίνητη ταυτοποίηση [30]

3.4.1.6 Λήψη & αποστολή αρχείου

Πέρα από τη διαχείριση των δεδομένων στη μνήμη, η Alamofire παρέχει δυνατότητες αποθήκευσης αρχείων στον δίσκο με μεθόδους όπως η Session.download, DownloadRequest, και DownloadResponse. Επιπλέον, προσφέρει επιλογές για τον έλεγχο της προόδου λήψης ή αποστολής αρχείων μέσω των .downloadProgress και .uploadProgress αντίστοιχα.

```
AF.download("https://httpbin.org/image/png").responseURL { response in
    // Read file from provided file URL.
}
```

Κώδικας 27: Session.download [30]

```
let destination: DownloadRequest.Destination = { _, _ in
    let documentsURL = FileManager.default.urls(for: .documentDirectory,
in: .userDomainMask)[0]
    let fileURL = documentsURL.appendingPathComponent("image.png")

    return (fileURL, [.removePreviousFile, .createIntermediateDirectories])
}

AF.download("https://httpbin.org/image/png", to: destination).response
{ response in
    debugPrint(response)

    if response.error == nil, let imagePath = response.fileURL?.path {
        let image = UIImage(contentsOfFile: imagePath)
    }
}
```

Κώδικας 28: DownloadRequest – Session.download [30]

```
AF.download("https://httpbin.org/image/png")
    .downloadProgress { progress in
        print("Download Progress: \(progress.fractionCompleted)")
    }
    .responseData { response in
        if let data = response.value {
            let image = UIImage(data: data)
        }
    }
```

Κώδικας 29: downloadProgress [30]

```
let fileURL = Bundle.main.url(forResource: "video", withExtension: "mov")

```

```
AF.upload(fileURL, to: "https://httpbin.org/post")
    .uploadProgress { progress in
        print("Upload Progress: \(progress.fractionCompleted)")
    }
    .downloadProgress { progress in
        print("Download Progress: \(progress.fractionCompleted)")
    }
    .responseDecodable(of: DecodableType.self) { response in
        debugPrint(response)
    }
```

Κώδικας 30: uploadProgress [30]

3.4.1.7 Μετρήσεις & στατιστικά

Κατά τη διάρκεια κάθε αλληλεπίδρασης, το URLSessionTaskMetrics καταγράφει σημαντικά στατιστικά για κάθε αίτημα, παρέχοντας λεπτομερείς πληροφορίες σχετικά με την απόκριση του συστήματος σε αυτό.

```
AF.request("https://httpbin.org/get").responseDecodable(of: DecodableType.self)
{ response in
    print(response.metrics)
}
```

Κώδικας 31: Response metrics [30]

3.4.1.8 cURL

Με τη χρήση της Alamofire, οι προγραμματιστές έχουν τη δυνατότητα να μετατρέψουν τα αιτήματά τους σε εντολές cURL. Αυτό παρέχει λεπτομερείς πληροφορίες σχετικά με το πώς διαμορφώνεται και πώς εξελίσσεται κάθε αίτημα, επιτρέποντάς τους να εντοπίζουν και να αντιμετωπίζουν προβλήματα.

```
AF.request("https://httpbin.org/get")
    .cURLDescription { description in
        print(description)
    }
    .responseDecodable(of: DecodableType.self) { response in
        debugPrint(response.metrics)
    }
```

Κώδικας 32: Μετατροπή αιτήματος σε cURL εντολή [30]

```
$ curl -v \
-X GET \
-H "Accept-Language: en;q=1.0" \
-H "Accept-Encoding: br;q=1.0, gzip;q=0.9, deflate;q=0.8" \
-H "User-Agent: Demo/1.0 (com.demo.Demo; build:1; iOS 15.0.0) Alamofire/1.0" \
"https://httpbin.org/get"
```

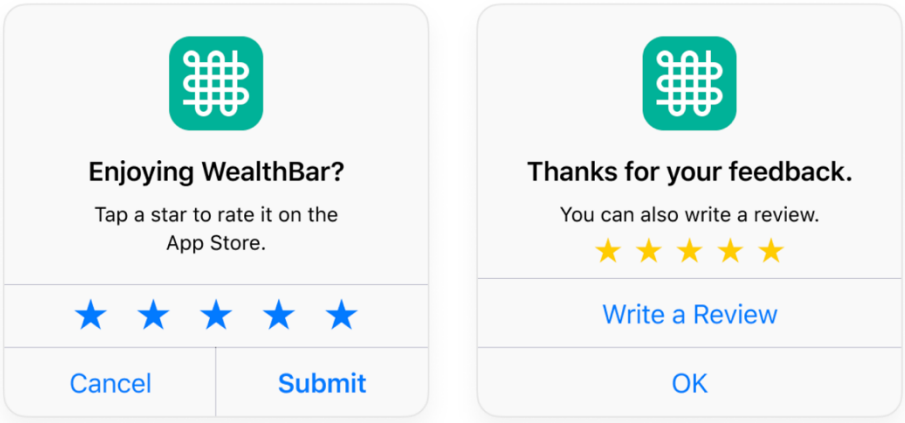
Κώδικας 33: Αποτέλεσμα του παραπάνω αιτήματος σε cURL εντολή [30]

3.4.2 StoreKit

Για την υλοποίηση της δυνατότητας αξιολόγησης της εφαρμογής, χρησιμοποιήθηκε η βιβλιοθήκη της Apple, που ονομάζεται StoreKit. Η βιβλιοθήκη αυτή υποστηρίζει αλληλεπιδράσεις με το App Store καθώς επίσης και αγορές που μπορούν να πραγματοποιηθούν εντός μιας εφαρμογής.

Συγκεκριμένα, στην παρούσα περίπτωση, αξιοποιήθηκε η κλάση SKStoreReviewController του StoreKit για να επιτραπεί στους χρήστες να υποβάλλουν αξιολογήσεις και κριτικές για την εφαρμογή

απευθείας στο App Store. Αυτή η προσέγγιση ευθυγραμμίζεται με τις βέλτιστες πρακτικές για την προτροπή των χρηστών να αξιολογήσουν την εφαρμογή, βελτιώνοντας έτσι τη διαδραστική εμπειρία τους και δίνοντάς τους τη δυνατότητα να παρέχουν πολύτιμο feedback για τη συνεχή μελλοντική βελτίωση της εφαρμογής.



Εικόνα 4: Παράδειγμα παραθύρου StoreKit [32]

3.5 Ανακεφαλαίωση εργαλείων υλοποίησης

Η εφαρμογή υλοποιήθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού Swift, με ελάχιστο ποσοστό συμμετοχής της γλώσσας Ruby, για να διευκολυνθεί η συγκρότηση του project και η εγκατάσταση εργαλείων/βιβλιοθηκών μέσω του CocoaPods.

Για την ανάπτυξη χρησιμοποιήθηκε το Xcode IDE, το περιβάλλον της Apple που παρέχει έναν ασφαλή και εξελιγμένο χώρο ανάπτυξης εφαρμογών, με δυνατότητες προσομοίωσής τους σε όλες τις διαθέσιμες Apple συσκευές.

Η εφαρμογή ενσωματώνει OAuth με το IEE Aboard API για εξακρίβωση των στοιχείων του χρήστη, το Keychain για ασφαλή αποθήκευση των credentials (στοιχείων εισόδου), και τη βιβλιοθήκη Alamofire για τη διαχείριση των αιτημάτων και των αποκρίσεων.

Κεφάλαιο 4ο: Ανάπτυξη Εφαρμογής

4.1 IEE Aboard

Κάνοντας μια πιο λεπτομερή περιγραφή της ανάπτυξης της εφαρμογής που συνοδεύει την παρούσα εργασία, γίνεται μια αναφορά στη δομή και τη συσχέτιση των αρχείων που την απαρτίζουν. Στο φάκελο-ρίζα, βρίσκονται, χωρισμένα σε φακέλους, τα αρχεία που αποτελούν την εφαρμογή και συμβάλλουν στη λειτουργικότητά της (UIViewControllers, swift class files etc.).

Η αρχιτεκτονική που έχει χρησιμοποιηθεί είναι αυτή του μοντέλου MVVM (Model, View, ViewModel), στην οποία διαχωρίζεται η επιχειρησιακή λογική (Model) από την εμφάνιση (View) μέσω του ViewModel, το οποίο διαχειρίζεται την επικοινωνία και τα δεδομένα μεταξύ τους, επιτρέποντας καθαρότερο κώδικα και κατ΄επέκταση, καλύτερο testing.

Η διάκριση και ομαδοποίηση των φακέλων και των επιμέρους αρχείων που εξυπηρετούν την αρχιτεκτονική που προαναφέρθηκε, διευκολύνει την πλοήγηση μέσα στο project και υποδηλώνει το ρόλο τους μέσα σε αυτό. Ενδεικτική διάκριση των φακέλων:

- *Shared: τα κοινά αρχεία, αυτά που χρησιμοποιούνται από πολλαπλές κλάσεις/οθόνες*
- *Scenes: Εμπεριέχονται οι φάκελοι με τα αρχεία κάθε οθόνης με αντίστοιχο όνομα*
- *Resources: φάκελος αρχείων εξωτερικού παράγοντα όπως γραμματοσειρά, assets δηλαδή εικόνες και γραφικά, χρώματα*
- *API: τα βασικά αρχεία που εξασφαλίζουν την επικοινωνία με το API*
- *Model: τα μοντέλα που ανταποκρίνονται στα πεδία του API που χρησιμοποιούνται στις οθόνες*

4.1.1 Scene Delegate

Η κλάση SceneDelegate διαχειρίζεται το περιβάλλον προβολής του χρήστη, δηλαδή των οθονών που θα μπορέσει να έχει πρόσβαση, ελέγχοντας αν έχει εκτελέσει είσοδο με τον αριθμό μητρώου και τον κωδικό του ή αν βρίσκεται στην αρχική σελίδα της εφαρμογής, όπου μπορεί να δει ό,τι είναι δημόσιο. Ένα flag στο keychain, με το κλειδί "FORCE_RELOAD_PRIVATE_ANNOUNCEMENTS," αποθηκεύεται για να κρατάει τον χρήστη συνδεδεμένο μετά από είσοδο ή να τον ανακατευθύνει στην αρχική σελίδα αν δεν έχει κάνει login κατά τις προηγούμενες χρήσεις της εφαρμογής.

```
 1. import UIKit
 2. import KeychainSwift
 3. import WebKit
 4.
 5. class SceneDelegate: UIResponder, UIWindowSceneDelegate {
 6.     let dependencyContainer = AppDependencyContainer()
 7.     var window: UIWindow?
 8.     let keychain = KeychainSwift()
 9.
10.     func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options
connectionOptions: UIScene.ConnectionOptions) {
11.         guard let windowScene = (scene as? UIWindowScene) else { return }
12.         let window = UIWindow(windowScene: windowScene)
13.
14.         let mainVC = dependencyContainer.makeHomeViewController()
15.         self.window = window
16.         window.frame = UIScreen.main.bounds
```

```
17.         let loggedIn =
keychain.get(DataReloadEnum.FORCE_RELOAD_PRIVATE_ANNOUNCEMENTS.rawValue)
18.         if loggedIn == "true" {
19.             let storyBoard: UIStoryboard = UIStoryboard(name: "LoggedInHome",
bundle: nil)
20.             let vc = storyBoard.instantiateViewController(withIdentifier:
"LoggedInHomeViewControllerIdentifier") as! LoggedInHomeViewController
21.             let loggedInHomeViewModel = LoggedInHomeViewModel()
22.             let loggedInHomeFlowCoordinator =
LoggedInHomeFlowCoordinator(loggedInHomeDIContainer: LoggedInHomeDIContainer(),
loggedInHomeViewController: vc)
23.             vc.viewModel = loggedInHomeViewModel
24.             vc.flowCoordinator = loggedInHomeFlowCoordinator
25.             let navigationController =
UINavigationController(rootViewController: vc)
26.             window.rootViewController = navigationController
27.         } else {
28.             window.rootViewController = mainVC
29.         }
30.         window.makeKeyAndVisible()
31.     }
32.
33.     func scene(_ scene: UIScene, openURLContexts URLContexts:
Set<UIOpenURLContext>) {
34.         if let urlContext = URLContexts.first {
35.             let url = urlContext.url
36.             if let deepLink = DeepLink(url: url) {
37.
dependencyContainer.deepLinkHandler.handleDeepLinkIfPossible(deepLink: deepLink)
38.             }
39.         }
40.     }
41. }
```

Κώδικας 34: Η κλάση SceneDelegate

4.2 API Router

Ο φάκελος Api_router περιλαμβάνει κλάσεις που χειρίζονται τον συγχρονισμό του Apps API με την εφαρμογή. Εδώ δημιουργούνται τα αιτήματα μέσω του Alamofire, είτε πρόκειται για GET είτε POST.

Στον Api_router υπάρχει ένα enum με διάφορες περιπτώσεις, όπου κάθε περίπτωση ορίζει τα headers, parameters, το request body, και τα endpoints που χρησιμοποιούνται ανάλογα.

Ο πλήρης οδηγός του API είναι διαθέσιμος στο https://aboard.iee.ihu.gr/documentation.

4.2.1 APIRouter

```
 1. enum APIRouter: URLRequestConvertible {
 2.
 3.     case getPublicAnnouncements(page: Int)
 4.     case getLoginAnnouncements(page: Int)
 5.     case getNotifications(page: Int)
 6.     case getToken(code: String)
 7.     case refreshToken(refreshToken: String)
 8.     case getUser
 9.     case getTags
10.     case getSubscriptions
11.     case postSubscriptions(id: [Int])
12.     case getNotAnn(id: Int)
```

```
13.  ……
```

Κώδικας 35: API Router functions enum

```
 1. var baseURL: String? {
 2.         switch self {
 3.             case .getPublicAnnouncements:
 4.                 return "https://aboard.iee.ihu.gr//api"
 5.             case .getLoginAnnouncements:
 6.                 return "https://aboard.iee.ihu.gr//api"
 7.             case .getNotifications:
 8.                 return "https://aboard.iee.ihu.gr//api/auth/user"
 9.             case .getUser:
10.                 return "https://aboard.iee.ihu.gr//api/auth"
11.             case .getToken,
12.                  .refreshToken:
13.                 return "https://login.iee.ihu.gr"
14.             case .getTags:
15.                 return "https://aboard.iee.ihu.gr//api"
16.             case .getSubscriptions:
17.                 return "https://aboard.iee.ihu.gr//api/auth"
18.             case .postSubscriptions:
19.                 return "https://aboard.iee.ihu.gr//api/auth"
20.             case .getNotAnn:
21.                 return "https://aboard.iee.ihu.gr//api"
22.         }
23.     }
```

Κώδικας 36: APIRouter baseURLs

```
 1. var path: String {
 2.         switch self {
 3.             case .getPublicAnnouncements:
 4.                 return "/announcements"
 5.             case .getLoginAnnouncements:
 6.                 return "/announcements"
 7.             case .getNotifications:
 8.                 return "/notifications"
 9.             case .getToken,
10.                 .refreshToken:
11.                 return "/token"
12.             case .getUser:
13.                 return "/user"
14.             case .getTags:
15.                 return "/tags"
16.             case .getSubscriptions:
17.                 return "/subscriptions"
18.             case .postSubscriptions:
19.                 return "/subscribe"
20.             case .getNotAnn(let id):
21.                 return "/announcements/\(id)"
22.         }
23.     }
```

Κώδικας 37: APIRouter paths

4.3 Data Context

Στο φάκελο DataContext συγκεντρώνονται τα αρχεία που χρησιμοποιούνται για την σύνδεση του APIRouter με το app, όπως Configuration και ClientRequest, τα οποία παρατίθενται παρακάτω.

4.3.1 Configuration

```
1. struct Configuration {
2.
3.     static let REMEMBER_TOKEN = "remember_token"
4.     static let REMEMBER_REFRESH_TOKEN = "remember_refresh_token"
5.
6. }
```

Κώδικας 38: Configuration tokens struct

4.3.2 ClientRequests

Το αρχείο ClientRequest περιλαμβάνει τη συνάρτηση που διαχειρίζεται το response (απάντηση) των αιτημάτων και τα δημιουργεί τα αντίστοιχα μοντέλα για εύκολα διαχείριση και προβολή των δεδομένων μέσω της διεπαφής χρήστη. Στο αρχείο DataContext+ClientRequest γίνεται ένωση του προηγούμενου με τα απαραίτητα στοιχεία ώστε να μπορεί να γίνει χρήση του από τους ViewControllers, δηλαδή τις οθόνες.

```
 1. func refreshToken(completion: @escaping (Bool) -> Void) {
 2.         guard let refreshToken =
keychain.get(Configuration.REMEMBER_REFRESH_TOKEN) else {
 3.             completion(false)
 4.             return
 5.         }
 6.         ClientRequests.refreshToken(refreshToken: refreshToken) {[weak self]
(authModel) in
 7.             guard let authModel = authModel else {
 8.                 completion(false)
 9.                 return
10.             }
11.             self?.keychain.set("\(authModel.access_token)", forKey:
Configuration.REMEMBER_TOKEN)
12.             self?.keychain.set("\(authModel.refresh_token)", forKey:
Configuration.REMEMBER_REFRESH_TOKEN)
13.             self?.refreshToken = authModel.refresh_token
14.             completion(true)
15.         }
16.     }
```

Κώδικας 39: refreshToken function

4.4 Dependencies

4.4.1 AppDIContainer

```
 1. import UIKit
 2.
 3. class AppDependencyContainer {
 4.
 5.     // MARK: - Properties
 6.
 7.     let deepLinkHandler = DeepLinkHandler()
 8.
 9.     // MARK: - Dependencies
10.
11.     lazy var oAuthConfig: OAuthConfig = {
12.         OAuthConfig(
```

```
13.             authorizationUrl: URL(string:
"https://login.iee.ihu.gr/authorization/")!,
14.             clientId: "62408ef084b2a60fc0ba856c",
15.             redirectUri: URL(string: "https://github.com/stefosAEL/IEEApp")!,
16.             responseType: "code",
17.             scope: ["announcements", "profile", "notifications",
"refresh_token", "edit_notifications"],
18.             clientSecret: "4mtxqivi27efteqcmkgzc7v7ex97o8ak4qjggack3jo07lfzaq"
19.         )
20.     }()
21.
22.     lazy var oAuthService: OAuthService = {
23.         let oAuthClient = RemoteOAuthClient(config: oAuthConfig, httpClient:
HTTPClient())
24.         return OAuthService(oauthClient: oAuthClient)
25.     }()
26.
27.     // MARK: - Methods
28.
29.     func makeHomeViewController() -> UINavigationController {
30.         let storyBoard: UIStoryboard = UIStoryboard(name: "Home", bundle: nil)
31.         let homeVC = storyBoard.instantiateViewController(withIdentifier:
"HomeViewControllerIdentifier") as! HomeViewController
32.         homeVC.oAuthService = oAuthService
33.         let homeFlowCoordinator = HomeFlowCoordinator(homeDIContainer:
HomeDIContainer(), appDIContainer: AppDependencyContainer(), homeViewController:
homeVC)
34.         let homeViewModel = HomeViewModel()
35.         homeVC.viewModel = homeViewModel
36.         homeVC.flowCoordinator = homeFlowCoordinator
37.         homeFlowCoordinator.homeViewController = homeVC
38.         let navigationController = UINavigationController(rootViewController:
homeVC)
39.         return navigationController
40.     }
41.
42.     func makeLoggedInHomeViewController() -> UINavigationController {
43.         let storyBoard: UIStoryboard = UIStoryboard(name: "LoggedInHome",
bundle: nil)
44.         let loggedInHomeVC =
storyBoard.instantiateViewController(withIdentifier:
"LoggedInHomeViewControllerIdentifier") as! LoggedInHomeViewController
45.         let flowCoordinator =
LoggedInHomeFlowCoordinator(loggedInHomeDIContainer: LoggedInHomeDIContainer(),
loggedInHomeViewController: loggedInHomeVC)
46.         let viewModel = LoggedInHomeViewModel()
47.         loggedInHomeVC.viewModel = viewModel
48.         loggedInHomeVC.flowCoordinator = flowCoordinator
49.         flowCoordinator.loggedInHomeViewController = loggedInHomeVC
50.         let navigationController = UINavigationController(rootViewController:
loggedInHomeVC)
51.         return navigationController
52.     }
53. }
```

Κώδικας 40: AppDIContainer

4.5 Infrastructure

4.5.1 DeeplinkHandler

Ο χρήστης μπορεί μέσω της οθόνης των ρυθμίσεων/πληροφοριών της εφαρμογής να επιλέξει να διαβάσει τον πηγαίο κώδικα, τους όρους χρήσης ή την πολιτική απορρήτου της. Σε κάθε μία από τις επιλογές, ο χρήστης ανακατευθύνεται από την ίδια την εφαρμογή, στην σελίδα της στο GitHub. Αυτό επιτυγχάνεται με την χρήση DeepLink, που επιτρέπει την επίσκεψη σε κάποια άλλη τοποθεσία εκτός της εφαρμογής, και τη μετέπειτα επιστροφή στην εφαρμογή. Καθίσταται δηλαδή, δυνατή η εναλλαγή και η συνεργασία διαφορετικών εφαρμογών (το GitHub είναι διαδικτυακή εφαρμογή).

```
let scheme = "https://github.com/anastasiamousa"

enum DeepLink: Hashable {
    case oAuth(URL)

    init?(url: URL) {
        let authLinkToDeepLink: (URL) -> DeepLink = { .oAuth($0) }

        let deepLinkMap: [String: (URL) -> DeepLink] = [
            "\(scheme)/IEEAboard": authLinkToDeepLink
        ]

        let deepLink = deepLinkMap.first(where:
{ url.absoluteString.hasPrefix($0.key) })?.value

        switch deepLink {
        case .some(let urlToDeepLink):
            self = urlToDeepLink(url)
        default:
            return nil
        }
    }

    func hash(into hasher: inout Hasher) {
        switch self {
        case .oAuth:
            return hasher.combine(1)
        }
    }

    static func ==(lhs: DeepLink, rhs: DeepLink) -> Bool {
      return lhs.hashValue == rhs.hashValue
    }
}
```

Κώδικας 41: Deeplink enum

4.5.2 RemoteOAuthClient

Για τον έλεγχο κα την επιβεβαίωση των στοιχείων πρόσβασης του χρήστη, χρησιμοποιείται η υπηρεσία OAuth.

```
struct OAuthConfig {
    let authorizationUrl: URL
    let clientId: String
    let redirectUri: URL
    let responseType:String
```

```
6.      let scope: [String]
7.      let clientSecret: String
8.  }
```

Κώδικας 42: OAuth struct

Συγκεντρώνονται τα απαραίτητα στοιχεία σε ένα struct, μια δομή τύπου OAuthConfig και αποστέλλονται ως αίτημα στην υπηρεσία για τον έλεγχο και την έγκρισή τους.

```
1.  protocol OAuthClient {
2.      func getAuthPageUrl(state: String) -> URL?
3.  }
4.
5.  class OAuthService {
6.
7.      enum OAthError: Error {
8.          case malformedLink
9.          case exchangeFailed
10.     }
11.
12.     private let oauthClient: OAuthClient
13.     private var state: String?
14.
15.     init(oauthClient: OAuthClient) {
16.         self.oauthClient = oauthClient
17.    }
18.
19.     func getAuthPageUrl(state: String = UUID().uuidString) -> URL? {
20.         self.state = state
21.         return oauthClient.getAuthPageUrl(state: state)
22.     }
23. }
```

Κώδικας 43: OAuth check

4.6 View Controller αρχεία

4.6.1 HomeViewController

Ο HomeViewController αντιπροσωπεύει την αρχική οθόνη της εφαρμογής και φιλοξενεί τη λειτουργικότητα που πρέπει να εκτελεστεί κατά την εκκίνησή της. Είναι η πρώτη οθόνη που θα εμφανιστεί σε έναν μη συνδεδεμένο χρήστη και μέσω αυτής, μπορεί να πλοηγηθεί στις λεπτομέρειες μιας δημόσιας ανακοίνωσης και να επιλέξει το κουμπί «ΣΥΝΔΕΣΗ» για να συνδεθεί στο λογαριασμό του και να δει τις υπόλοιπες ανακοινώσεις.

4.6.2 LoginViewController

Σε αυτόν το viewController, υπάρχει το webView που εμφανίζει τη διαδικτυακή εφαρμογή του Apps ώστε να μπορέσει ο χρήστης να συνδεθεί στο λογαριασμό του ή να προχωρήσει σε ανάκτηση κωδικού ή ενεργοποίησή του, αν υπάρχει τέτοιο ζήτημα. Η οθόνη διαθέτει κουμπί για τη επιστροφή του χρήστη στην προηγούμενη οθόνη.

4.6.3 LoggedInHomeViewController

Ο συγκεκριμένος viewController, ακολουθεί παρόμοια λογική με αυτή του HomeViewController, με τη διαφορά ότι στο use case που εμφανίζεται, ο χρήστης είναι συνδεδεμένος στο λογαριασμό του, συνεπώς, έχει περισσότερες δυνατότητες. Μπορεί να γίνει πλοήγηση του χρήστη στις λεπτομέρειες των

ανακοινώσεων, να δει τις λεπτομέρειες του προφίλ του, τις ειδοποιήσεις και να μεταφερθεί στην οθόνη των ρυθμίσεων.

4.6.4 WebViewController

Με τον WebViewController, εμφανίζεται στο χρήστη όποια οθόνη χρειάζεται κάποιο URL για να ανοίξει, καθώς το περιεχόμενό της βρίσκεται εκτός της εφαρμογής, στο διαδίκτυο. Ουσιαστικά, φιλοξενεί ένα view μέσα στο οποίο φορτώνει το URL και μπορεί ο χρήστης να αλληλεπιδράσει με το περιεχόμενο της ιστοσελίδας. Τέλος, διαθέτει κουμπί για τη επιστροφή του χρήστη στην προηγούμενη οθόνη.

4.6.5 AnnouncementDetailsViewController

Πρόκειται για οθόνη που ανοίγει από την αρχική σελίδα είτε ο χρήστης είναι συνδεδεμένος είτε όχι, δηλαδή και από τον HomeViewController και από τον LoggedInHomeViewController, και προβάλλει λεπτομέρειες για την κάθε ανακοίνωση. Φαίνεται ο τίτλος της ανακοίνωσης, το όνομα του συγγραφέα, πότε προστέθηκε και αν υπάρχει κάποιο συνημμένο. Ακόμη, διαθέτει κουμπί για τη επιστροφή του χρήστη στην προηγούμενη οθόνη καθώς επίσης και ένα κουμπί πάνω δεξιά για αντιγραφή του συνδέσμου URL που καταλήγει στην ανακοίνωση, όπως αυτή εμφανίζεται στο aboard.iee.ihu.gr. Με την αντιγραφή του συνδέσμου, ο χρήστης μπορεί να πραγματοποιήσει επικόλλησή του όπου επιθυμεί και να τη μοιραστεί.

4.6.6 UserDetailsViewController

Ο UserDetailsViewController ανοίγει από την αρχική οθόνη, εφόσον ο χρήστης έχει συνδεθεί στο λογαριασμό του. Στην οθόνη αυτή, βλέπει το όνομά του, το όνομα χρήστη στην διαδικτυακή εφαρμογή Apps και το email που έχει δηλώσει στα στοιχεία επικοινωνίας του. Επίσης, διαθέτει κουμπί για τη επιστροφή του χρήστη στην προηγούμενη οθόνη.

4.6.7 NotificationsViewController

Ο χρήστης έχει πρόσβαση σε αυτή την οθόνη, μετά την επιτυχή σύνδεσή του στο λογαριασμό του. Σε αυτή , προβάλλονται οι ειδοποιήσεις των ανακοινώσεων που ανήκουν σε κάποια από τις ετικέτες που έχει επιλέξει να παρακολουθεί. Αφού ενημερωθεί από τις ειδοποιήσεις, ο χρήστης μπορεί να γυρίσει πίσω στην προηγούμενη οθόνη, με το πάτημα του αντίστοιχου κουμπιού.

4.6.8 TagsSubscriptionController

Πρόκειται για ακόμη μια οθόνη που αφορά τον συνδεδεμένο χρήστη. Στο συγκεκριμένο viewController, εμφανίζεται η λίστα με όλες τις διαθέσιμες ετικέτες που μπορεί να επιλέξει ο χρήστης ώστε να έχει μια πιο εξατομικευμένη εμπειρία σχετικά με την ενημέρωσή του για τις ανακοινώσεις. Ο χρήστης επιλέγει όποια ετικέτα επιθυμεί να συμπεριληφθεί στις προτιμήσεις του και μόλις αλλάξει κάτι στις ήδη αποθηκευμένες ετικέτες, το κουμπί της αποθήκευσης ενεργοποιείται και αποθηκεύονται οι νέες επιλογές.

4.6.9 SettingsViewController

Στο συγκεκριμένο viewController, προβάλλονται οι διαθέσιμες ρυθμίσεις της εφαρμογής. Ο χρήστης μπορεί να επιλέξει, να δει και να τροποποιήσει οτιδήποτε τις αφορά και με το αντίστοιχο κουμπί να γυρίσει πίσω στην αρχική σελίδα των ανακοινώσεων.

4.6.10 SearchViewController

Η συγκεκριμένη οθόνη παρουσιάζει τα φίλτρα αναζήτησης ανακοινώσεων. Με βάση τον τίτλο, με βάση κάποιο λεκτικό που περιέχεται στις λεπτομέρειες της ανακοίνωσης, επιλέγοντας το όνομα του συγγραφέα ή κάποια πιθανή ετικέτα που να ανήκει η ανακοίνωση, καθώς και κάποιο χρονικό πλαίσιο στο οποίο μπορεί να έχει δημοσιευτεί. Με το πάτημα του κουμπιού της αναζήτησης, εμφανίζονται όσες ανακοινώσεις ανταποκρίνονται στα φίλτρα, εάν υπάρχουν και με κύλιση του view προς τα κάτω, γίνεται επιστροφή στην αρχική οθόνη.

4.7 Models

Με τη χρήση μοντέλων, ορίζονται με αντικειμενοστρεφή τρόπο οι δομές του μοντέλου ταυτοποίησης, των δημόσιων και των απευθυνόμενων προς τους συνδεδεμένους φοιτητές ανακοινώσεων, το μοντέλο των ειδοποιήσεων, οι εγγραφές σε αυτές μέσω της επιλογής των ετικετών, καθώς και τα μοντέλα για τις ίδιες τις ετικέτες και τους χρήστες. Τα μοντέλα δηλαδή αφορούν τα βασικά χαρακτηριστικά όλων των οντοτήτων της εφαρμογής.

```
1. struct User: Codable {
2.     var subscriptions: [Subscriptions]?
3.     let name: String
4.     let email: String
5.     let uid: String
6. }
7.
8. struct Subscriptions: Codable {
9.     let title: String
10. }
```

```
1. struct Users: Codable {
2.   let data: User
3. }
```

Κώδικας 44: Μοντέλα User, Users

```
1. struct Announcement: Codable {
2.     let author: Teacher
3.     let title: String
4.     let body: String
5.     let created_at: String
6.     let tags: [Tags]
7.     let id: Int
8.     let attachments: [Attachments]?
9. }
10.
11. struct Teacher: Codable {
12.     let name: String
13. }
14.
15. struct Attachments: Codable {
16.     let attachment_url: String
17. }
18.
19. struct Tags:Codable {
20.     let title: String
21. }
```

```
1. struct AnnouncementsList: Codable {
2.   let data: [Announcement]
3.   let meta: Meta?
4. }
5.
6. struct Meta: Codable {
7.     let last_page: Int
8. }
```

Κώδικας 45: Μοντέλα Announcement, AnnouncementsList

```
1. struct Tag: Codable {
2.   let data: [TagsArray]
3. }
```

```
1. struct TagsArray: Codable {
2.     let title : String
3.     var id : Int
4.     let isSelected : Bool?
```

```
5. }
```

Κώδικας 46: Μοντέλα Tag, TagsArray

```
1. struct Notification: Codable {
2.     let data: Datas?
3.     let created_at: String
4. }
5.
6. struct Datas: Codable {
7.     let type: String
8.     let user: String
9.     let id: Int
10. }
```

```
1. struct Notifications: Codable {
2.   let data: [Notification]
3.   let meta : Metas?
4. }
5.
6. struct Metas: Codable {
7.     let last_page: Int
8. }
```

Κώδικας 47: Μοντέλα Notification, Notifications

```
1. struct AuthModel: Codable {
2.     let access_token: String
3.     let user: String
4.     let refresh_token: String
5.     let error: Error?
6.
7.     struct Error: Codable{
8.         let code: Int?
9.     }
10. }
```

```
1. struct Subscription: Codable {
2.     let pivot: Pivot?
3.
4.     struct Pivot: Codable {
5.         let tagID: Int
6.
7.         enum CodingKeys: String, CodingKey
{
8.             case tagID = "tag_id"
9.         }
10.     }
11. }
```

Κώδικας 48: Μοντέλα AuthModel, Subscription

Κεφάλαιο 5ο: Παρουσίαση Λειτουργίας Εφαρμογής

5.1 Εμφάνιση και συμπεριφορά στο περιβάλλον του λειτουργικού συστήματος

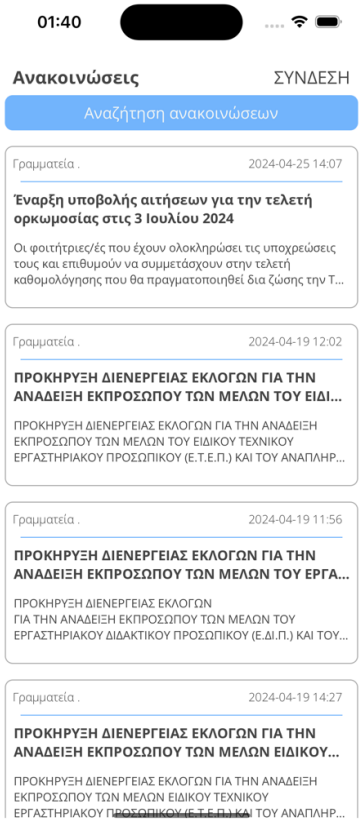
Αρχικά η εφαρμογή, όπως είναι ορατή με το app icon της, δηλαδή το εικονίδιο που την αντιπροσωπεύει, βρίσκεται στην αρχική οθόνη της φορητής συσκευής iPhone ή/και iPad του χρήστη. Οι δυνατότητες που δίνονται μέσω παρατεταμένου πατήματος στο εικονίδιο, είναι η μετακίνησή του και η απεγκατάσταση της εφαρμογής.

Εικόνα 5: Στιγμιότυπα οθόνης από Xcode simulator, iPhone 15 Pro

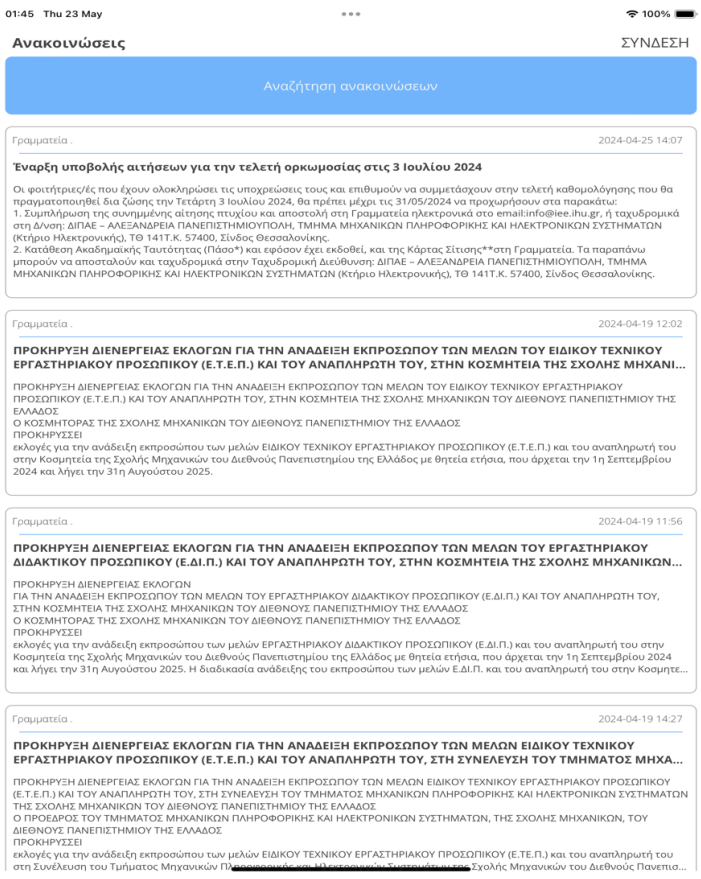
Η εφαρμογή επίσης υποστηρίζει notifications για ενημέρωση του χρήστη όποτε αναρτηθεί νέα ανακοίνωση που ανήκει σε κάποια από τις ετικέτες που έχει δηλώσει στις ρυθμίσεις της εφαρμογής.

5.2 Αρχική σελίδα εφαρμογής

Με την είσοδο στην αρχική σελίδα της εφαρμογής, ο χρήστης βλέπει τις δημόσιες ανακοινώσεις. Οι ανακοινώσεις αυτές, είτε αφορούν το τμήμα γενικότερα είτε έχουν δημοσιευτεί για ειδικότερα θέματά του, όπως ένα μάθημα του προγράμματος σπουδών, αλλά οι δημιουργοί τους έχουν επιλέξει τη δημόσια κοινοποίησή τους. Οι ανακοινώσεις παρουσιάζονται μέσω ενός πίνακα τύπου UITableView και στο κάθε κελί του πίνακα είναι ορατά ο συγγραφέας της ανακοίνωσης, η ημερομηνία και η ώρα που κοινοποιήθηκε, ο τίτλος της ανακοίνωσης και οι πρώτες γραμμές του βασικού κειμένου της εφαρμογής.

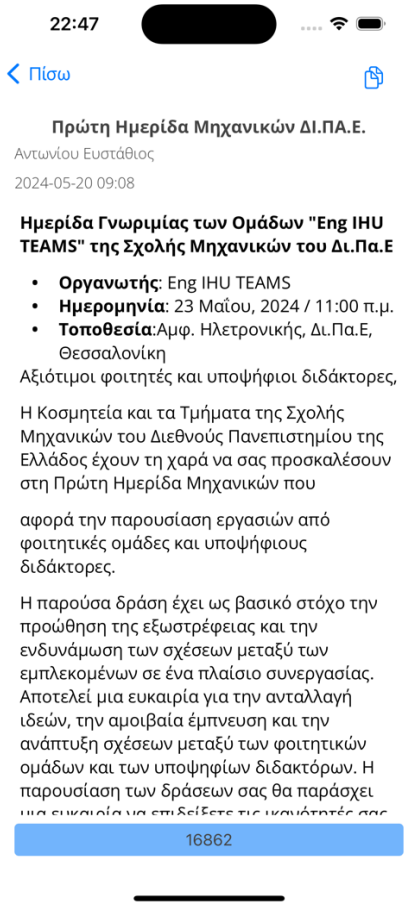


Εικόνα 6: Αρχική οθόνη εφαρμογής (μη συνδεδεμένος χρήστης)



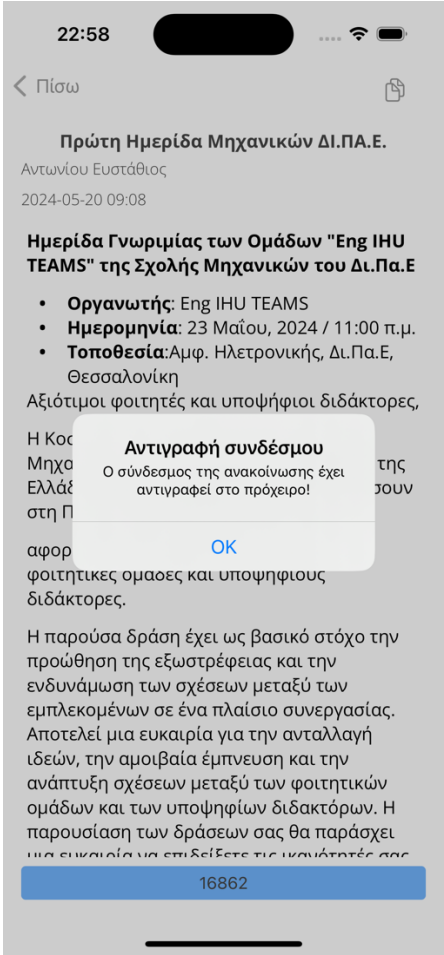
Εικόνα 7: Αρχική οθόνη εφαρμογής (μη συνδεδεμένος χρήστης) σε συσκευή tablet, iPad

Αν ο χρήστης επιλέξει μια από τις ανακοινώσεις, ανοίγει μια επόμενη οθόνη, η οποία δείχνει τις περαιτέρω λεπτομέρειές της. Η οθόνη με τις λεπτομέρειες διαθέτει ScrollView, ώστε ανάλογα με το μέγεθος της οθόνης του κινητού του χρήστη, να μπορεί με κύλιση προς τα κάτω να διαβάσει τις πληροφορίες που αναφέρει η ανακοίνωση. Αν η προβαλλόμενη ανακοίνωση διαθέτει κάποιο συνημμένο, προστίθεται στο κάτω μέρος της οθόνης κουμπί με το όνομα του κάθε αρχείου και αν ο χρήστης το πατήσει, εμφανίζεται ένα native prompt στο οποίο ο χρήστης έχει επιλογές όπως να αποθηκεύσει το αρχείο, να το στείλει κάπου, να το εκτυπώσει και άλλες.



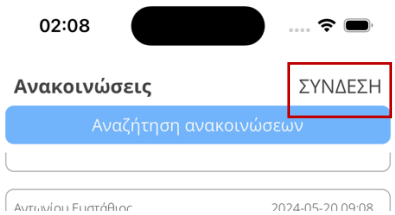
Εικόνα 8: Οθόνη λεπτομερειών ανακοίνωσης

Επιπλέον, πάνω δεξιά στην οθόνη των λεπτομερειών, υπάρχει ένα κουμπί που με το πάτημά του πραγματοποιείται αντιγραφή του συνδέσμου της ανακοίνωσης, δηλαδή του URL που καταλήγει στο σημείο που φιλοξενείται η ανακοίνωση στο aboard.iee.ihu.gr. Με την περάτωση της αντιγραφής του συνδέσμου, ο χρήστης ενημερώνεται για αυτό με την εμφάνιση alertView. Πατώντας το «OK», το alertView κλείνει.



Εικόνα 9: AlertView αντιγραφής συνδέσμου

Εάν ο χρήστης επιθυμεί να συνδεθεί στην εφαρμογή και να δει και τις υπόλοιπες, μη δημόσιες ανακοινώσεις, πατά το κουμπί «ΣΥΝΔΕΣΗ», πάνω δεξιά της αρχικής οθόνης, προκειμένου να μεταβεί στη σελίδα εισόδου στις υπηρεσίες του τμήματος.



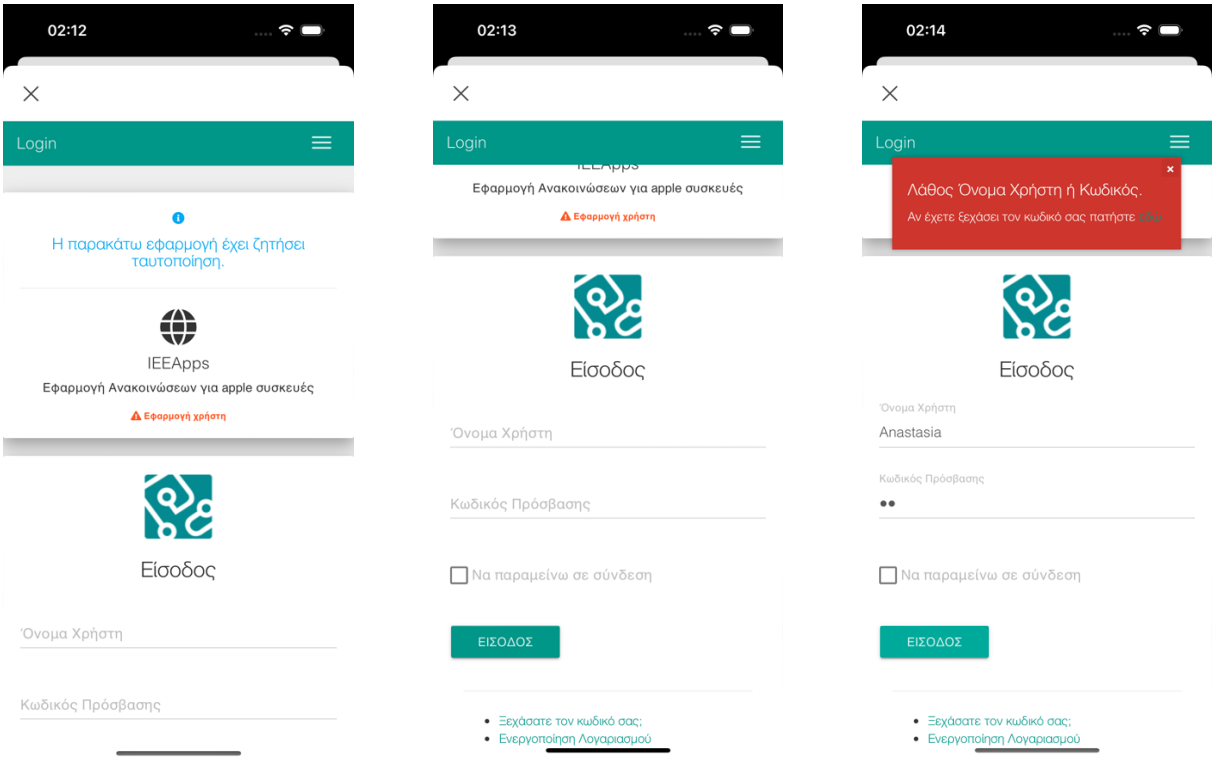
Εικόνα 10: Στιγμιότυπο οθόνης με το κουμπί για σύνδεση του χρήστη

5.3 Σελίδα σύνδεσης/εισόδου του χρήστη

5.3.1 Ταυτοποίηση χρήστη

Με το άνοιγμα της σελίδας εισόδου, που γίνεται μέσω webView, ο χρήστης εμφανώς παροτρύνεται να εισάγει τα στοιχεία του (όνομα χρήστη, κωδικός πρόσβασης) ώστε να πραγματοποιηθεί η ταυτοποίησή του και να αποκτήσει πρόσβαση στην εξατομικευμένη έκδοση της εφαρμογής. Αν εισαχθούν λάθος στοιχεία σύνδεσης ή αν ο χρήστης δε διαθέτει ακόμη ενεργό λογαριασμό, υπάρχει κατάλληλο πεδίο της

σελίδας εισόδου, που θα τον καθοδηγήσει ανάλογα. Η κατεύθυνση και υλοποίηση της εκάστοτε λύσης, γίνεται μέσα από τη διαδικτυακή εφαρμογή Apps.



Εικόνα 11: Οθόνη σύνδεσης, simulator iPhone 15 Pro

5.3.2 Λοιπές λειτουργίες υποστήριξης

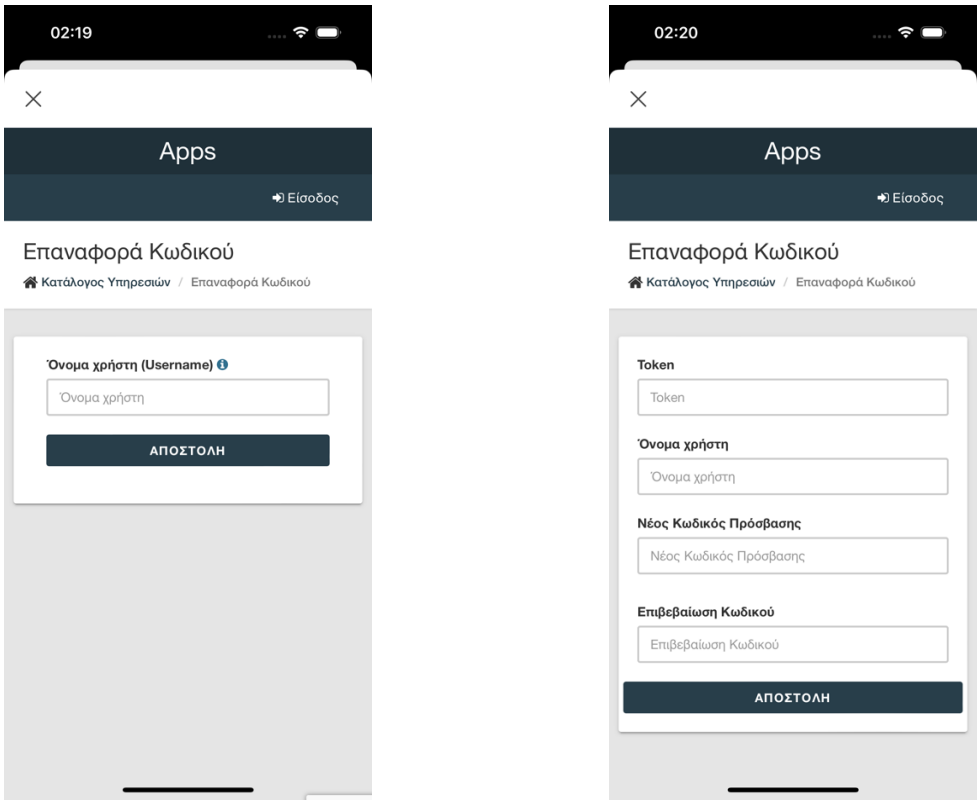
Οι λειτουργίες υποστήριξης που προσφέρονται από την διαδικτυακή εφαρμογή Apps σχετικά με τη διαχείριση του λογαριασμού του χρήστη είναι οι παρακάτω:

- Επαναφορά κωδικού λογαριασμού

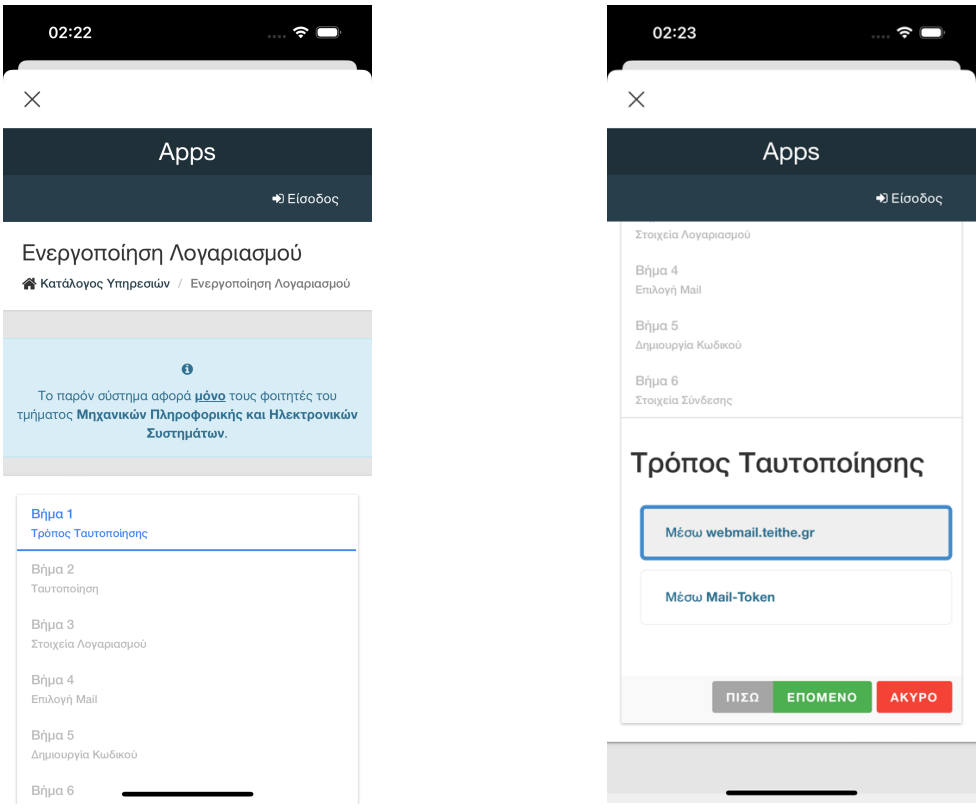
Για να επαναφέρει τον κωδικό πρόσβασης, ο χρήστης πρέπει να εισάγει το όνομα χρήστη που χρησιμοποιεί στην εφαρμογή Apps. Μετά από αυτό, θα του σταλεί ένα ειδικό token εξουσιοδότησης στη δηλωμένη διεύθυνση ηλεκτρονικού ταχυδρομείου του. Στη συνέχεια, ο χρήστης πρέπει να συμπληρώσει μια νέα φόρμα με το token, το ίδιο όνομα χρήστη και τον νέο κωδικό πρόσβασης που επιθυμεί να χρησιμοποιήσει.

- Ενεργοποίηση λογαριασμού

Η ταυτοποίηση γίνεται μέσω του ακαδημαϊκού λογαριασμού του χρήστη και στη συνέχεια καταχωρούνται τα στοιχεία του νέου χρήστη στη βάση δεδομένων της εφαρμογής Apps.



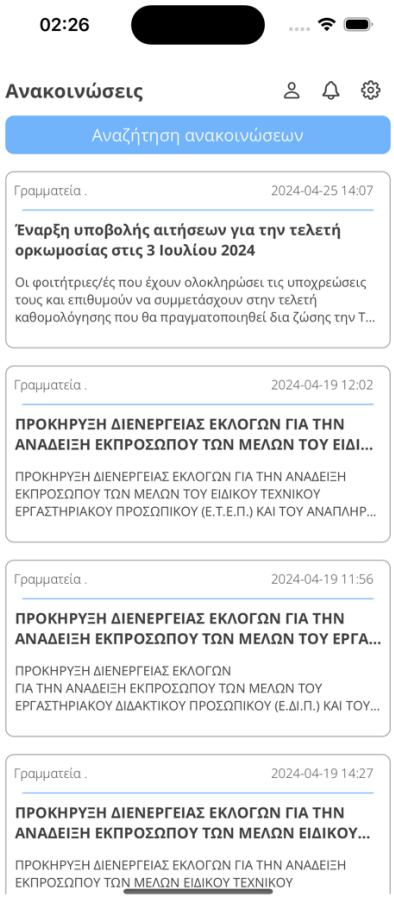
Εικόνα 12: Επαναφορά κωδικού λογαριασμού, simulator iPhone 15 Pro



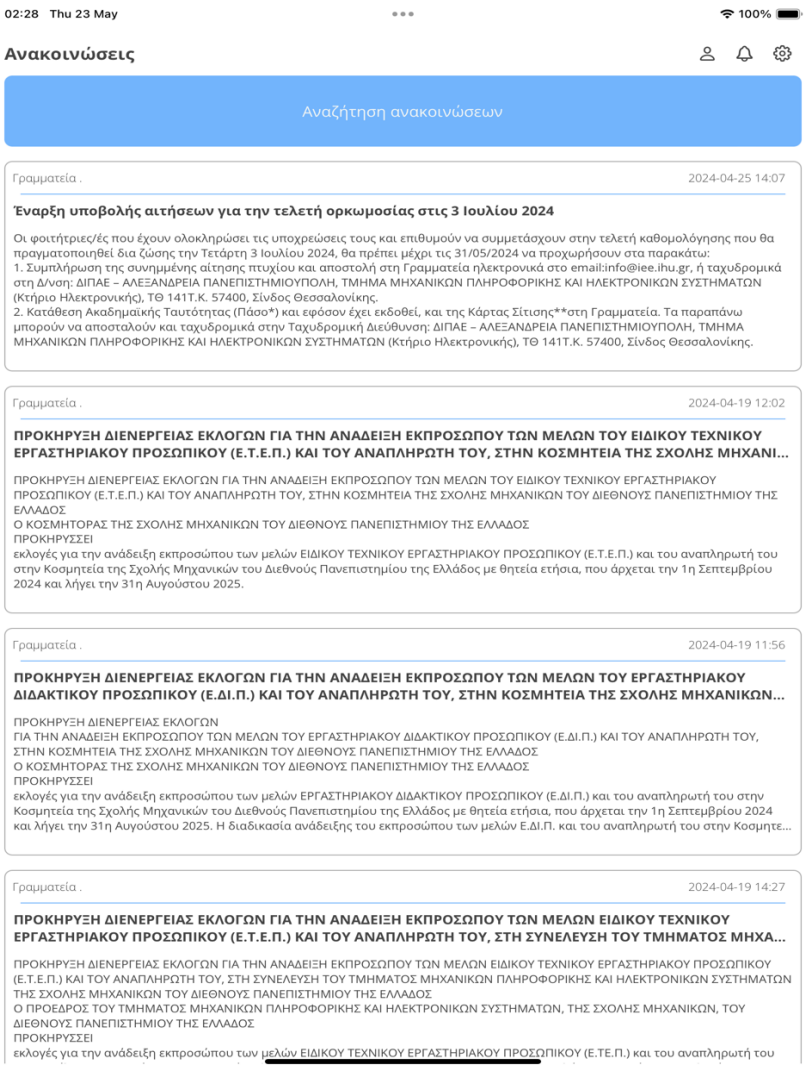
Εικόνα 13: Ενεργοποίηση λογαριασμού, simulator iPhone 15 Pro

5.4 Αρχική σελίδα συνδεδεμένου χρήστη

Στην αρχική σελίδα, που ακολουθεί της επιτυχούς σύνδεσης του χρήστη, εμφανίζονται όλες οι ανακοινώσεις και παρέχεται πρόσβαση στις υπόλοιπες λειτουργίες της εφαρμογής. Στη σελίδα αυτή, υπάρχει μια λίστα με ανακοινώσεις, όμοια με αυτή στην αρχική σελίδα της εφαρμογής πριν τη σύνδεση του χρήστη. Σε αντίθεση με την πρώτη, εδώ είναι διαθέσιμες επιπλέον και όλες οι ανακοινώσεις που απευθύνονται σε συγκεκριμένα μαθήματα ή εξάμηνα. Παρόμοια με την προηγούμενη σελίδα, όταν ο χρήστης πατήσει πάνω σε μια ανακοίνωση, θα ανοίξει η επόμενη οθόνη, αυτή με τις λεπτομέρειες της ανακοίνωσης, και ο χρήστης θα έχει πρόσβαση σε όσα στοιχεία προαναφέρθηκαν.



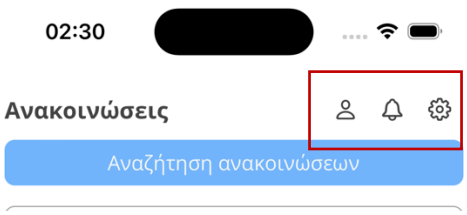
Εικόνα 14: Αρχική οθόνη εφαρμογής (συνδεδεμένος χρήστης), simulator iPhone 15 Pro



Εικόνα 15: Αρχική οθόνη εφαρμογής (συνδεδεμένος χρήστης), iPad

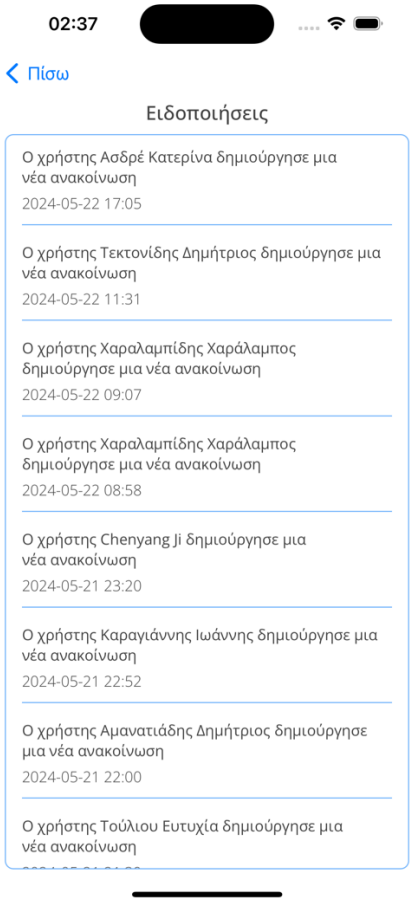
5.5 Ειδοποιήσεις χρήστη

Στην αρχική οθόνη, πάνω δεξιά, υπάρχουν τρία εικονίδια, το προφίλ του χρήστη, οι ειδοποιήσεις του και οι ρυθμίσεις.



Εικόνα 16: Στιγμιότυπο οθόνης με τα εικονίδια, simulator iPhone 15 Pro

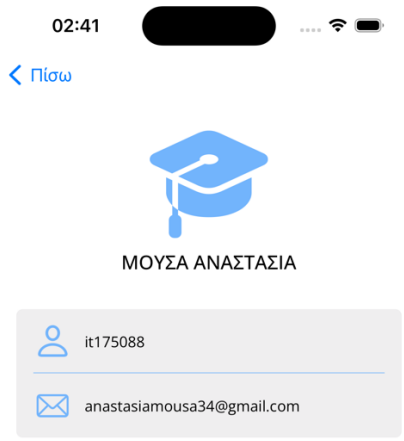
Αν ο χρήστης πατήσει το κουμπί με το καμπανάκι, εμφανίζεται μια λίστα, ένα UITableView με κελιά με τις ειδοποιήσεις που λαμβάνει, ανάλογα με τις προτιμήσεις του. Η τροποποίηση των προτιμήσεων δεν επηρεάζει τις υπάρχουσες ειδοποιήσεις, αλλά καθορίζει τη λήψη των ενημερώσεων στο μέλλον. Πατώντας το κουμπί «Πίσω» στην πάνω αριστερή γωνία, ο χρήστης επιστρέφει ξανά στην κύρια σελίδα της εφαρμογής, όπου βρίσκονται συγκεντρωμένες όλες οι ανακοινώσεις.



Εικόνα 17: Οθόνη ειδοποιήσεων, simulator iPhone 15 Pro

5.6 Προφίλ χρήστη

Αναφορικά με το εικονίδιο του προφίλ του χρήστη, δηλαδή το εικονίδιο με το ανθρωπάκι δίπλα στην καμπάνα των ειδοποιήσεων, ο χρήστης μεταφέρεται στο προφίλ του, που περιλαμβάνει δικές του πληροφορίες. Οι πληροφορίες που εμφανίζονται είναι το ονοματεπώνυμό, το όνομα χρήστη του λογαριασμού στη διαδικτυακή εφαρμογή Apps και το δηλωμένο email για επικοινωνία και λήψη των ενημερώσεων μέσω email.



Εικόνα 18: Οθόνη προφίλ χρήστη, simulator iPhone 15 Pro

5.7 Ρυθμίσεις εφαρμογής

Με την επιλογή του κουμπιού με το σχήμα γραναζιού, ο χρήστης αποκτά πρόσβαση στην οθόνη των ρυθμίσεων. Σε αυτή την οθόνη παρουσιάζονται δυνατότητες για το χρήστη:

- Αποσύνδεση

Ο χρήστης μπορεί να αποσυνδεθεί από το λογαριασμό του. Αν το επιλέξει, γυρνά στην αρχική οθόνη και έχει πρόσβαση στις δημόσιες ανακοινώσεις.

- Τροποποίηση προτιμήσεων

Σε αυτή την οθόνη, ο χρήστης μπορεί να επεξεργαστεί τις προτιμήσεις του σχετικά με τη λήψη ειδοποιήσεων για ανακοινώσεις.

- Λήψη ειδοποιήσεων

Επιλέγοντας αυτό το πεδίο, ο χρήστης ορίζει αν θέλει να λαμβάνει push notifications κάθε φορά που προστίθεται μια ανακοίνωση που ανήκει σε κάποια από τις ετικέτες που έχει δηλώσει στις προτιμήσεις του.

- Αλλαγή γλώσσας

Με το πάτημα αυτής της επιλογής ρυθμίσεων, ο χρήστης μπορεί να επιλέξει τη γλώσσα εμφάνισης της εφαρμογής.

- Αξιολόγηση εφαρμογής

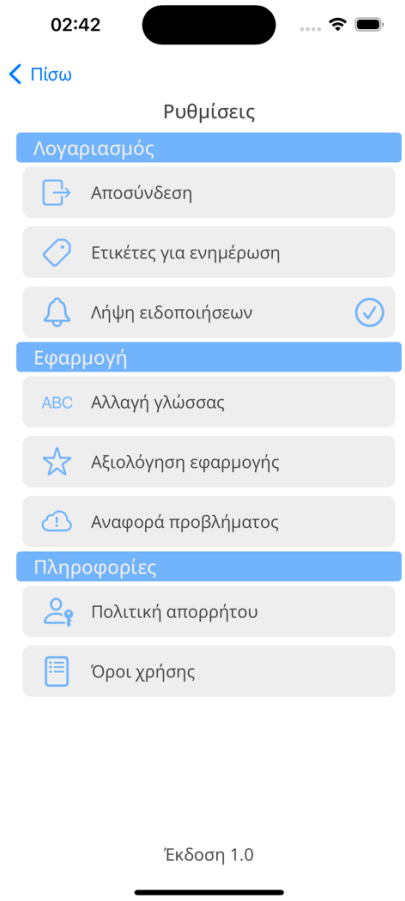
Ο χρήστης μπορεί να εκφράσει την άποψή του σχετικά με την εφαρμογή, στο AppStore.

- Αναφορά προβλήματος

Σε αυτή την επιλογή, ο χρήστης μπορεί να αναφέρει κάποιο τεχνικό πρόβλημα (bug) της εφαρμογής.

- Πολιτική Απορρήτου / όροι χρήσης

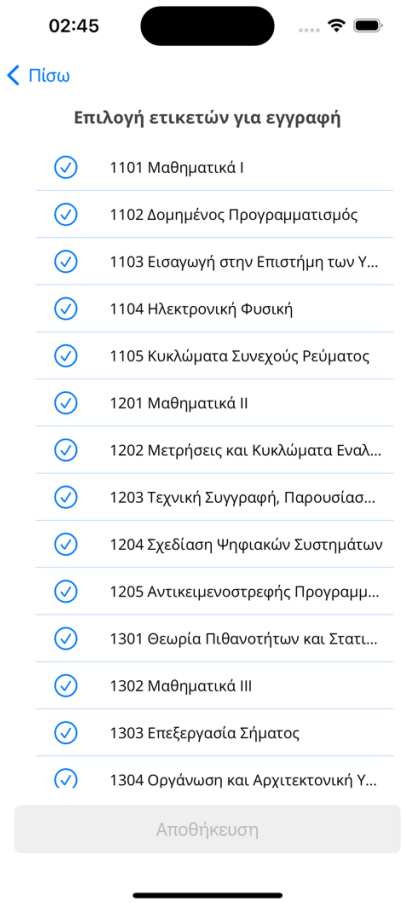
Επιλέγοντας κάποιο από αυτά τα δύο πεδία ο χρήστης μπορεί να διαβάσει την πολιτική απορρήτου και τους όρους χρήσης της εφαρμογής. Ανοίγει ένα WebView και καταλήγει σε link στο GitHub repository όπου βρίσκεται το αρχείο με την εκάστοτε πληροφορία.



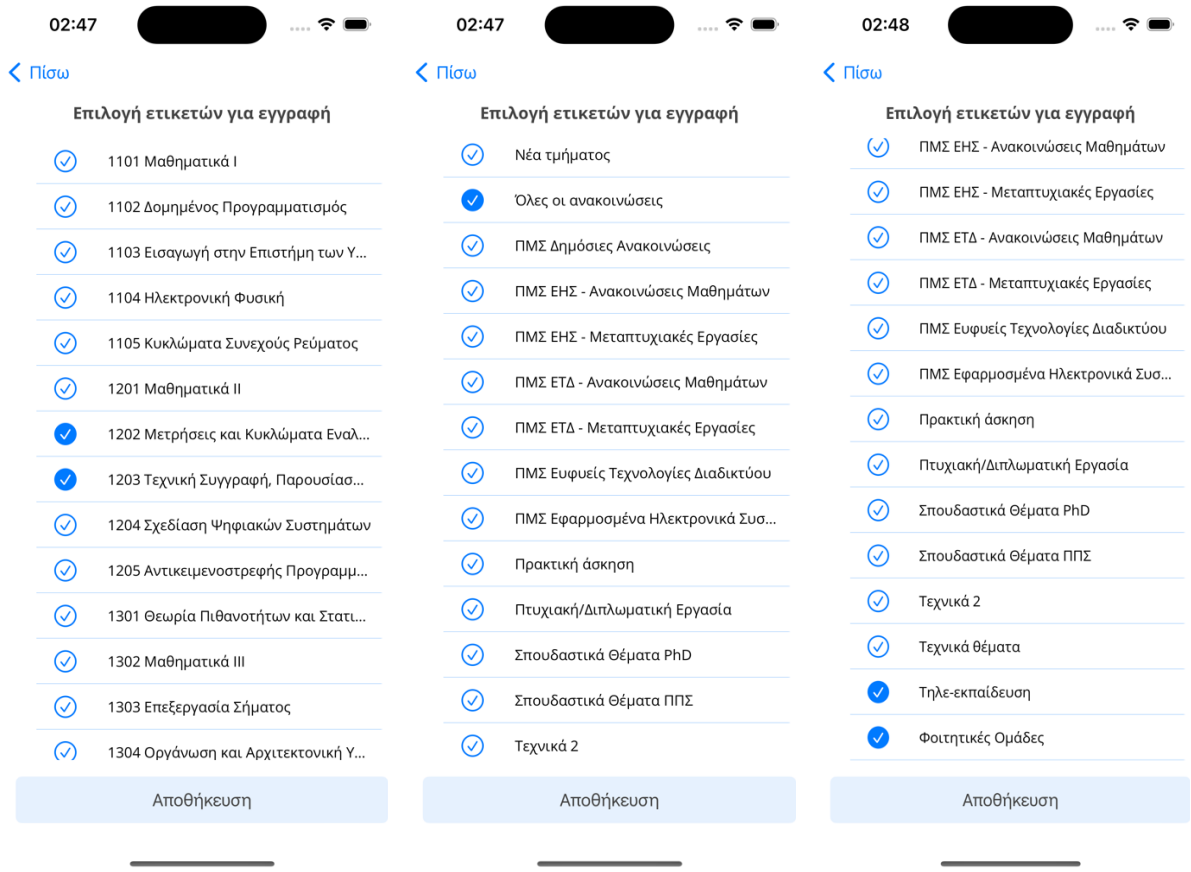
Εικόνα 19: Οθόνη ρυθμίσεων , simulator iPhone 15 Pro

5.7.1 Ετικέτες για ενημέρωση

Εδώ ο χρήστης μπορεί να επιλέξει τις ετικέτες των ανακοινώσεων για τις οποίες επιθυμεί να ενημερώνεται. Οι επιλογές περιλαμβάνουν διαφορετικά μαθήματα, εξάμηνα, ανακοινώσεις για θέματα γενικού ενδιαφέροντος, καθώς και την επιλογή να λαμβάνει ειδοποιήσεις για όλες τις ανακοινώσεις που αναρτώνται. Κάθε φορά που ο χρήστης κάνει αλλαγές, οι επιλογές του αποθηκεύονται.



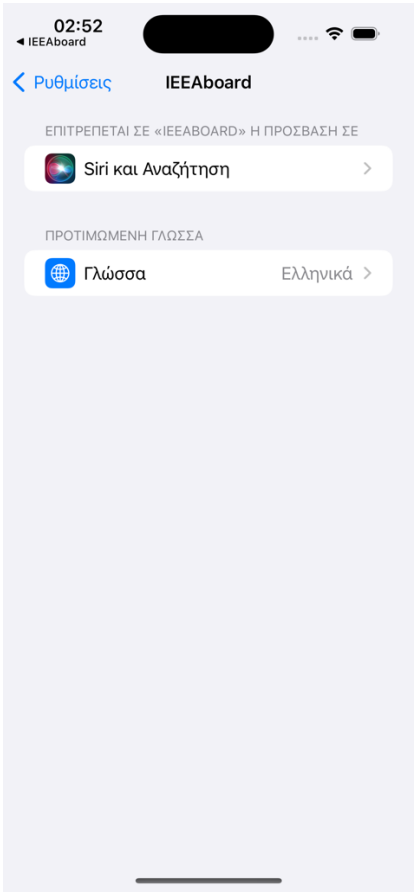
Εικόνα 20: Οθόνη ετικετών , simulator iPhone 15 Pro



Εικόνα 21: Κάποιοι δυνατοί συνδυασμοί επιλογής ετικετών, simulator iPhone 15 Pro

5.7.2 Αλλαγή γλώσσας

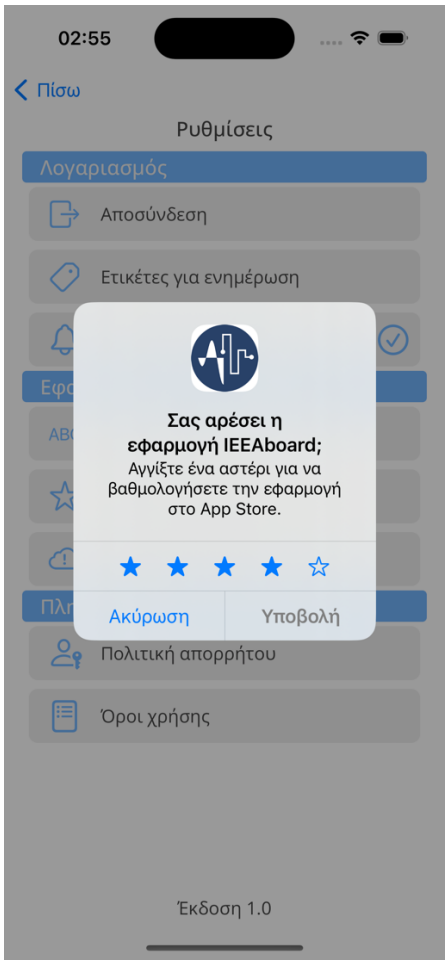
Μόλις ο χρήστης επιλέξει την αλλαγή γλώσσας της εφαρμογής, μεταφέρεται στις ρυθμίσεις της συσκευής που αφορούν τη συγκεκριμένη εφαρμογή. Εκεί, μπορεί να επιλέξει σε ποια γλώσσα θέλει να βλέπει την εφαρμογή. Η βασική γλώσσα τόσο της εφαρμογής, όσο και των ανακοινώσεων, είναι τα ελληνικά. Αν ο χρήστης επιλέξει την αγγλική γλώσσα, οι βασικοί τίτλοι και η πλοήγηση μέσα στην εφαρμογή θα υποστηρίζονται στα αγγλικά, ενώ οι λεπτομέρειες της ανακοίνωσης, θα συνεχίσουν να φαίνονται όπως έρχονται από τα αιτήματα που γίνονται, στα ελληνικά δηλαδή.



Εικόνα 22: Ρυθμίσεις λειτουργικού που αφορούν την εφαρμογή, simulator iPhone 15 Pro

5.7.3 Αξιολόγηση εφαρμογής & Αναφορά προβλήματος

Με την επιλογή του πεδίου της αξιολόγησης της εφαρμογής, ενεργοποιείται εμφάνιση του παραθύρου του StoreKit, δηλαδή της βιβλιοθήκης της Apple που συνδέει την εφαρμογή με την καταχώρισή της στο AppStore. Στην παρούσα φάση, που η εφαρμογή δεν έχει δημοσιευτεί, ο χρήστης μπορεί να δει την προτροπή για αξιολόγηση και τη βαθμολόγηση των πέντε αστεριών, αλλά δε μπορεί να την υποβάλει, ώστε να καταχωρηθεί. Μετά το πέρας της δημοσίευσης στης εφαρμογής, η αξιολόγηση με αστέρια ή/και κάποιο σχόλιο, θα μπορεί να καταχωρηθεί και θα είναι ορατή στη σελίδα της εφαρμογής στο AppStore. Αντίστοιχα, η αναφορά προβλήματος θα μπορεί να καταχωρηθεί μόλις δηλωθούν στοιχεία επικοινωνίας και υποστήριξης της εφαρμογής, κατά την προετοιμασία της για δημοσίευση στο AppStore.



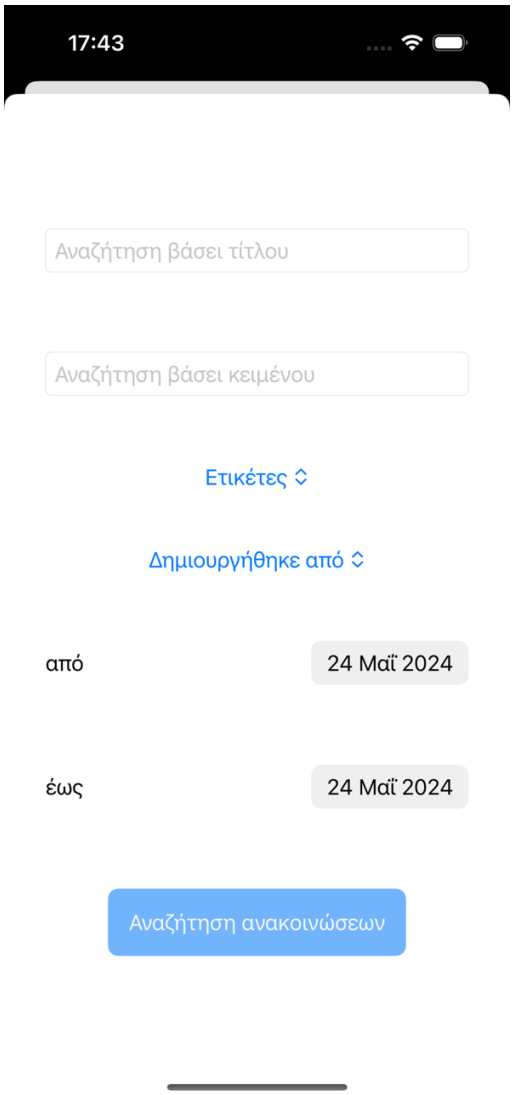
Εικόνα 23:Εμφάνιση παραθύρου StoreKit, simulator iPhone 15 Pro

5.7.4 Πολιτική απορρήτου & όροι χρήσης

Τα δύο αυτά αρχεία, ανοίγουν στο χρήστη με τη μορφή webView. Η πολιτική απορρήτου αποτελεί δήλωση του προγραμματιστή σχετικά με τη χρήση των δεδομένων που ο χρήστης εισάγει στην εφαρμογή, όπως στοιχεία πρόσβασης, ετικέτες ειδοποιήσεων, στατιστικά χρήσης και cookies. Οι όροι χρήσης προσδιορίζουν τις προϋποθέσεις υπό τις οποίες διανέμεται η εφαρμογή και καθορίζουν τη θέση του προγραμματιστή απέναντι σε τυχόν νομικά θέματα που μπορεί να προκύψουν.

5.8 Αναζήτηση ανακοινώσεων

Πρόκειται για την οθόνη που παρουσιάζει τα φίλτρα αναζήτησης ανακοινώσεων. Με βάση τον τίτλο, με βάση κάποιο λεκτικό από τις λεπτομέρειες της ανακοίνωσης, με βάση το όνομα του συγγραφέα ή κάποια ετικέτα που να ανήκει η ανακοίνωση, καθώς και ένα χρονικό πλαίσιο που μπορεί να έχει δημοσιευτεί. Με το πάτημα του κουμπιού της αναζήτησης, εμφανίζονται ανακοινώσεις που ανταποκρίνονται στα φίλτρα, εάν υπάρχουν και με κύλιση του view προς τα κάτω, γίνεται επιστροφή στην αρχική οθόνη (dismiss).



Εικόνα 24:Οθόνη φίλτρων αναζήτησης, simulator iPhone 15 Pro

Εικόνα 25:Οθόνη αποτελεσμάτων αναζήτησης, simulator iPhone 15 Pro

Κεφάλαιο 6ο: Επίλογος/Συμπεράσματα

Συνοψίζοντας, η εφαρμογή iOS του aboard αναμένεται να εξυπηρετήσει πολλούς φοιτητές του τμήματος που διαθέτουν τις ανάλογες συσκευές. Όπως αναφέρθηκε και στην αρχή της εργασίας, η αντίστοιχη εφαρμογή σε Android είναι ήδη σε κυκλοφορία και εξυπηρετεί πάνω από 1000 χρήστες.

Η εφαρμογή παρέχει τις απαραίτητες βασικές λειτουργίες για την ενημέρωση των φοιτητών. Παρέχεται άμεση πρόσβαση στις δημόσιες ανακοινώσεις χωρίς να είναι απαραίτητη κάποια ενέργεια, απλά και μόνο με το άνοιγμα της εφαρμογής, ενώ με την σύνδεσή στον ακαδημαϊκό τους λογαριασμό, τους γνωστοποιούνται και οι λειτουργίες που αφορούν αποκλειστικά τους φοιτητές του τμήματος. Ο χρήστης ενημερώνεται με ειδοποίηση κάθε φορά που γίνεται ανάρτηση νέας ανακοίνωσης στον πίνακα με την ετικέτα που τον ενδιαφέρει, καθώς επίσης μπορεί και να επεξεργαστεί αυτές τις προτιμήσεις. Δεν αντιμετωπίστηκε κάποιο ιδιαίτερο πρόβλημα κατά την ανάπτυξή της εφαρμογής, καθώς η Swift είναι μια εύχρηστη και ευέλικτη γλώσσα.

Η εφαρμογή είναι ανοιχτού κώδικα, καθώς φιλοξενείται στο GitHub, ώστε να μπορούν με τη σειρά τους να συμβάλλουν όσοι φοιτητές το επιθυμούν στο μέλλον για τη βελτίωση και αναβάθμισή της. Προσωπικά, μου έδωσε τη δυνατότητα να εξασκηθώ και να εξελιχθώ περαιτέρω σε μια γλώσσα εύχρηστη, που αγγίζει ανταγωνιστικό τομέα της ανάπτυξης εφαρμογών και συμβάλλει σημαντικά σε αυτόν.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] «Welcome to Swift.org» [Ηλεκτρονικό]. Available: https://www.swift.org/ (last accessed: 05/07/2024).

[2] «Releases» [Ηλεκτρονικό]. Available: https://developer.apple.com/news/releases/ (last accessed: 20/06/2024).

[3] «Featured» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/ (last accessed: 12/02/2024).

[4] «iOS» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/IOS (last accessed: 21/11/2023).

[5] «The History of iOS and its Evolution» [Ηλεκτρονικό]. Available: https://www.mobileappdaily.com/history-of-ios (last accessed: 21/11/2023).

[6] «Application Framework» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Application_framework (last accessed: 27/11/2023).

[7] «What is a Framework?» [Ηλεκτρονικό]. Available: https://www.codecademy.com/resources/blog/what-is-a-framework/ (last accessed: 27/11/2023).

[8] «API» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/API (last accessed: 30/11/2023).

[9] «What is an API?» [Ηλεκτρονικό]. Available: https://www.codecademy.com/resources/blog/what-to-know-about-apis/ (last accessed: 30/11/2023).

[10] «Keychain Services» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/security/keychain_services (last accessed: 3/12/2023).

[11] «Keychain (software)» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/security/keychain_services/ (last accessed: 3/12/2023).

[12] «Collection Types» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/collectiontypes/ (last accessed: 15/12/2023).

[13] «Optional» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/swift/optional (last accessed: 15/12/2023).

[14] «Structures and Classes» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/classesandstructures (last accessed: 15/12/2023).

[15] «Control Flow» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/controlflow (last accessed: 15/12/2023).

[16] «Error Handling» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/errorhandling (last accessed: 19/12/2023).

[17] «Functions» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/functions (last accessed: 19/12/2023).

[18] «Generics» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/generics (last accessed: 19/12/2023).

[19] «Memory Safety» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/memorysafety (last accessed: 19/12/2023).

[20] «The Basics» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/thebasics (last accessed: 04/01/2024).

[21] «cURL Command Output» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#curl- command-output (last accessed: 04/01/2024).

[22] «Statistical Metrics» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#statistical-metrics (last accessed: 04/01/2024).

[23] «Downloading Data to a File» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#downloading-data-to-a-file (last accessed: 04/01/2024).

[24] «Authentication» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#authentication (last accessed: 10/01/2024).

[25] «Uploading Data to a Server» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#uploa ding-data-to-a-server (last accessed: 10/01/2024).

[26] «Response Validation» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#response-validation (last accessed: 10/01/2024).

[27] «Response Handling» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#response-handling (last accessed: 10/01/2024).

[28] «HTTP Headers,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#http-headers (last accessed: 15/04/2024).

[29] «Request Parameters» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#request-parameters-and-parameter-encoders (last accessed: 15/04/2024).

[30] «HTTP Methods» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#http-methods (last accessed: 15/04/2024).

[31] «Enumerations» [Ηλεκτρονικό]. Available:

https://docs.swift.org/swift-book/documentation/the-swift-programming-language/enumerations (last accessed: 12/07/2024).

[32] «StoreKit» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/storekit/ (last accessed: 12/07/2024).