ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«ΣΥΣΤΗΜΑ ΕΝΗΜΕΡΩΣΗΣ ΑΝΑΚΟΙΝΩΣΕΩΝ ΓΙΑ IOS»

Του φοιτητή:
Στέφανου Καυκαλιά
Αρ. Μητρώου: 164695

Επιβλέπων
Ονοματεπώνυμο: Αντώνης Σιδηρόπουλος
Βαθμίδα: Επίκουρος Καθηγητής

Ημερομηνία: 9/06/2022

Τίτλος Δ.Ε: Σύστημα ενημέρωσης ανακοινώσεων για IoS
Ονοματεπώνυμο φοιτητή: Στέφανος Καυκαλιάς
Ονοματεπώνυμο εισηγητή: Στέφανος Καυκαλιάς
Ημερομηνία ανάληψης Δ.Ε. : 23/09/2021
Ημερομηνία περάτωσης Δ.Ε.: 18/06/2022

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

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

«Στην Οικογένεια μου»

Πρόλογος

Επιστημονικά κίνητρα για την επιλογή και ενασχόληση του συγκεκριμένου θέματος υπήρξε η διάθεση για την εκμάθηση και εξοικείωση μιας νέας τεχνολογίας, η οποία είχε πολλά να με διδάξει και να μου προσφέρει σημαντικά εφόδια για το μέλλον στην αναζήτηση εργασίας.

Επιπλέον η γνώση πως υπήρχε εφαρμογή για τους χρήστες των Android συνέβαλε καθοριστικά στην επιλογή του θέματος, δίνοντάς μου το αναγκαίο κίνητρο να δημιουργήσω κάτι νέο και να μάθω μέσω της πτυχιακής μου μια νέα τεχνολογία. Παράλληλα, μου έδωσε την δυνατότητα να συγκρίνω ομοιότητες και διαφορές και να εμβαθύνω σε έννοιες και συστήματα ενδιαφέροντα αλλά σε μεγάλο βαθμό άγνωστα σε εμένα μέχρι πρότινος.

Περίληψη

Η παρούσα εργασία αναλύει την ανάπτυξη και την χρησιμότητα της εφαρμογής “IEE Apps”, η οποία είναι μια iOS εφαρμογή βασισμένη στο καινούριο σύστημα ανακοινώσεων του τμήματός μας, “Apps”. Το πρόγραμμα δίνει στον χρήστη άμεση πρόσβαση στις ανακοινώσεις του τμήματος, παρέχοντας παράλληλα τις κατάλληλες επιλογές ώστε ο φοιτητής να λαμβάνει αποκλειστικά τις ειδοποιήσεις που έχει δηλώσει ότι τον αφορούν.

«Application development:

IoS Announcements’ Notification System»

«Stefanos Kafkalias»

Abstract

This paper analyzes the development and usefulness of the "IEE Apps" application, which is an iOS application based on the new notification system of our department, "Apps". The program gives the user direct access to the announcements of the department, while providing the appropriate options so that the student receives only the notifications that he has stated concern him.

Ευχαριστίες

Μέσα από αυτές τις λίγες γραμμές, θα ήθελα να ευχαριστήσω όλους όσους με βοήθησαν και με στήριξαν για την εκπόνηση και την ολοκλήρωση της πτυχιακής μου εργασίας. Η εκπόνηση της εργασίας αυτής ήταν μια πρόκληση για μένα διότι είναι η βασική προϋπόθεση για την ολοκλήρωση του κύκλου σπουδών μου στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου Ελλάδος.

Πρώτα από όλα θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον υπεύθυνο καθηγητή μου, κ. Σιδηρόπουλο, για την εμπιστοσύνη που μου έδειξε και την ανάθεση της παραπάνω πτυχιακής εργασίας. Το αμείωτο ενδιαφέρον, οι υποδείξεις, η καθοδήγηση, η προθυμία του και η συμπαράστασή του κατά τη συγγραφή της εργασίας, ήταν καθοριστική. Θερμές ευχαριστίες απευθύνω σε όλους τους καθηγητές που είχα όλα τα χρόνια της ακαδημαϊκής μου ζωής, για τις γνώσεις που μου μετέδωσαν και με έκαναν καλύτερο άνθρωπο.

Ένα μεγάλο και εγκάρδιο ευχαριστώ στους καρδιακούς μου φίλους για τη στήριξη, τη συμπαράσταση και την κατανόησή τους, όπως επίσης, σε όλους όσους συνέβαλαν με οποιονδήποτε τρόπο στην επιτυχή εκπόνηση αυτής της πτυχιακής εργασίας. Τέλος ένα τεράστιο ευχαριστώ αξίζουν δύο ήρωες της καθημερινότητάς μου, οι γονείς μου, που με στηρίζουν όλα αυτά τα χρόνια, δίνοντάς μου κουράγιο να προχωρώ και τελικά να επιτύχω τους στόχους μου, αλλά και ο αδερφός μου που πάντα στέκεται δίπλα μου.

Περιεχόμενα

Κατάλογος Αποσπάσματων Κώδικα

Συντομογραφίες

ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία

Κεφάλαιο 1ο: Εισαγωγή

1.1 Περιγραφή του Προβλήματος/Πρόλογος

Το τμήμα μας (Τμήμα Μηχανικών Πληροφορικής και Υπολογιστικών Συστημάτων) πρόσφατα διέθεσε ένα νέο σύστημα ανακοινώσεων, το λεγόμενο "Apps". Αυτή η κίνηση ευεργέτησε τους φοιτητές και έτσι προέκυψε η ιδέα για την ανάπτυξη της παρούσας εφαρμογής ως συμπληρωματική του νέου συστήματος ανακοινώσεων. Στόχος της δημιουργίας του "IEEApps" είναι η διευκόλυνση της πρόσβασης των χρηστών Apple (iOS) συσκευών στο σύστημα ανά πάσα στιγμή, με την επιπρόσθετη δυνατότητα φιλτραρίσματος των ενημερώσεων βάση εξαμήνου, ώστε η ροή πληροφορίας να είναι σχετική με τα μαθήματα που παρακολουθεί.

1.2 Περιγραφή της εφαρμογής

Το πρόγραμμα που αναπτύχθηκε για την πτυχιακή πρόκειται για μια iOS εφαρμογή που παρέχει μια πλατφόρμα παρακολούθησης των ανακοινώσεων του τμήματος, και παράλληλα ενημερώνει τον χρήστη για νέες ανακοινώσεις μέσω push notification.

1.3 Διάρθρωση της Πτυχιακής

- Στο πρώτο κεφάλαιο εισάγονται οι γενικές έννοιες που διατρέχουν την πτυχιακή στη μορφή σύντομης περιγραφής, οι στόχοι και η συνεισφορά που αποσκοπεί να πετύχει με το παρόν θέμα.
- Στο δεύτερο κεφάλαιο αναλύονται με τεχνικές λεπτομέρειες οι τεχνολογίες πάνω στις οποίες βασίστηκε η εν λόγω εφαρμογή.
- Στο τρίτο κεφάλαιο γίνεται αναφορά στα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη και την λειτουργία του προγράμματος και τον ρόλο τους.
- Στο τέταρτο κεφάλαιο παρουσιάζεται η εφαρμογή και οι λειτουργίες της, όπως τα βιώνει ο χρήστης.
- Στο πέμπτο κεφάλαιο αναλύεται ο κώδικας του προγράμματος.
- Στο έκτο και τελευταίο κεφάλαιο πραγματοποιείται μια γενική ανακεφαλαίωση και ακολουθούν κάποια γενικά σχόλια σχετικά με τα αποτελέσματα της εργασίας.

Κεφάλαιο 2ο: Θεωρητικό υπόβαθρο/Τεχνολογίες

2.1 Λειτουργικό σύστημα: iOS

2.1.1 Εισαγωγή

Το αντικείμενο της πτυχιακής είναι μία εφαρμογή που απευθύνεται σε λειτουργικά συστήματα τύπου iOS.

Το λειτουργικό σύστημα είναι ένα κομμάτι λογισμικού το οποίο παρέχει όλες τις απαραίτητες υπηρεσίες ώστε να ελέγχονται και να αξιοποιούνται οι φυσικοί πόροι του συστήματος, ή αλλιώς το υλικό, από τα υπόλοιπα εγκατεστημένα προγράμματα. Το λειτουργικό μπορεί να θεωρηθεί και το ίδιο ένα πρόγραμμα, με την διαφορά ότι τα υπόλοιπα προγράμματα βασίζονται σε αυτό για την λειτουργία τους. Χαρακτηριστικές υπηρεσίες που προσφέρει είναι ο συγχρονισμός των διάφορων επεξεργαστικών μονάδων, η ανάθεση προσωρινής και μη μνήμης και η διαχείριση εισαγωγής και εξαγωγής δεδομένων (input/output), που συμπεριλαμβάνει την επικοινωνία με τυχόν συνδεδεμένες συσκευές (πληκτρολόγιο, ποντίκι, ηχεία κλπ). Το λειτουργικό μπορεί να παρέχεται προεγκατεστημένο από τον κατασκευαστή της συσκευής ή να εγκατασταθεί μετέπειτα εφόσον υποστηρίζεται από το υλικό.

Τα πιο δημοφιλή λειτουργικά συστήματα γενικού σκοπού είναι τα Windows της Microsoft(76%), το macOS της Apple (17%) και οι διάφορες παραλλαγές του Linux (2%). Το MacOSκαι το Linux είναι βασισμένα στο πρότυπο λειτουργικών συστημάτων UNIX. Όσον αφορά τα συστήματα κινητών συσκευών, την πρώτη θέση έχει το Android (72%) (ανοιχτό λογισμικό της Google) που ακολουθείται από το iOS (κλειστό λογισμικό της Apple), το Chrome OS (κλειστό λογισμικό της Google) κ.α.. Το λογισμικό διαχωρίζεται σε ανοιχτού και κλειστού κώδικα ανάλογα με την διαθεσιμότητά του στο ευρύ κοινό. Ο ανοιχτός κώδικας είναι δημοσίως διαθέσιμος και επιτρέπονται αντιγραφές και τροποποιήσεις, ενώ ο κλειστός είναι ιδιοκτησία που προστατεύεται από πνευματικά δικαιώματα που θέτουν παράνομη την αναπαραγωγή του από τρίτους. Το iOS είναι λογισμικό κλειστού κώδικα, δηλαδή ανήκει στην εταιρεία Apple Inc. και χρησιμοποιείται αποκλειστικά στις συσκευές της, όπως είναι το iPhone, iPad κλπ από το 2007.

Σύμφωνα με την εταιρεία, το iOS (i Operating System) χαρακτηρίζεται από τις λέξεις “individual, instruct, inform, inspire) ως έννοιες που καθορίζουν τον στόχο του.

2.1.2 Ιστορική αναδρομή

Με την πρώτη του κυκλοφορία να είναι το 2007, το iOS εξελίσσεται κάθε χρόνο, υιοθετώντας νέα ονόματα και χαρακτηριστικά. Συγκεκριμένα, οι επίσημες εκδόσεις του είναι οι εξής:

(2007 - 2010) iPhone OS 1
Η πρώτη κυκλοφορία του λειτουργικού συστήματος, βασισμένη στο λειτουργικό σύστημα προσωπικών υπολογιστών mac OS X. Υποστήριζε την λειτουργία του πρώτου iPhone της αγοράς. Έφερε στην αγορά τις τότε πρωτότυπες εφαρμογές Multitouch Gestures, Mobile Safari, Visual Voicemail, Maps, iTunes Sync, Multitouch Keyboard, iTunes.

(2008) iPhone OS 2
Κυριότερη αναβάθμιση ήταν η προσθήκη του κέντρου λογισμικού “App Store”, που πλέον επέτρεπε την εγκατάσταση λογισμικού που είχε δημιουργηθεί από τρίτους, αφού είχε εγκριθεί από την Apple. Υποστήριζε την λειτουργία του iPhone 3G.

(2009 - 2012) iPhone OS 3
Σε αυτή την έκδοση προστέθηκαν πολλά καινούρια χαρακτηριστικά, όπως η αντιγραφήεπικόλληση κειμένου και φωτογραφιών, η βιντεοσκόπηση, η μεταφορά δεδομένων κινητής τηλεφωνίας (tethering) μέσω USB και Bluetooth, η υποστήριξη MMS, το Spotlight Search αλλά και εφαρμογές όπως οι Ringtone Downloads, Find My iPhone κ.α. Η τελευταία έκδοση που υποστήριζε συσκευές όπως το πρώτο iPhone και το iPod Touch (το οποίο αντικαταστάθηκε από το iPad με την έκδοση iOS 3.2). Υποστήριζε την λειτουργία του iPhone 3GS.

(2010 - 2014) iOS 4
Η επόμενη σημαντική αναβάθμιση από το iPhone OS 2, όπου εισάχθηκαν λειτουργίες όπως η ταυτόχρονη εκτέλεση πολλαπλών εφαρμογών και η εξατομίκευση της οθόνης, που δεν υποστηρίζονταν από συσκευές παλιότερων αρχιτεκτονικών (iPhone 1, iPhone 3G, iPod Touch). Εφαρμογές που προστέθηκαν είναι το FaceTime, το Airplay, το AirPrint, το iBooks, το Game Center, ενώ σημειώθηκαν αναβαθμίσεις στην υηρεσία ηλεκτρονικού ταχυδρομείου.

(2011 - 2014) iOS 5
Τα κύρια νέα χαρακτηριστικά αυτής της έκδοσης ήταν η προσθήκη συστήματος ειδοποιήσεων, η εφαρμογή επικοινωνίας iMessage, το iCloud κ.α.. Σε προχωρημένες εκδόσεις πρωτοκυκλοφορεί και η ”βοηθός” του iOS Siri μαζί με την συσκευή iPhone 4S.

(2012 - 2015) iOS 6
Αρκετά σημαντική έκδοση μετά το iOS 4, παρατηρούνται αναβαθμίσεις στις περισσότερες εφαρμογές. Αντικατάσταση του Google Maps με το Apple Maps, παρόμοιας εφαρμογής η οποία επιπλέον χρησιμοποιεί την Siri (η οποία σε αυτή την έκδοση είναι βελτιωμένη) και κάποια ακόμα επιπρόσθετα χαρακτηριστικά, προσθήκη δημοφιλών εφαρμογών όπως το YouTube στο App Store (αντί να είναι αναπόσπαστο μέρος του συστήματος). Πλέον κυκλοφορούν τα iPad 3, iPod Touch 5 και iPhone 5. 12.

(2013) iOS 7
Προσθήκη πολλαπλών χαρακτηριστικών και ρυθμίσεων που βελτιώνουν την διεπαφή χρήστη-συσκευής. Προσθήκη γρήγορης πρόσβασης σε διάφορες βασικές λειτουργίες. Εισαγωγή του AirDrop.

(2014 - 2016) iOS 8
Νέες “έξυπνες” λειτουργίες όπως το Health (παρακολούθηση υγείας και καθημερινής συμπεριφοράς του χρήστη), το Quick Type (αυτόματη πρόβλεψη δαχτυλογράφησης) κ.α.. Βελτιστοποίηση της επικοινωνίας μεταξύ διαφορετικών Apple συσκευών. Εισαγωγή των εφαρμογών Family Sharing, QuickType, iCloud Drive, HealthKit, HomeKit. Ενεργοποίηση της Siri μέσω φωνητικών εντολών.

(2015 - 2018) iOS 9
Νέα εφαρμογή γενικής ενημέρωσης (News), βελτιώσεις στις προ υπάρχουσες εφαρμογές όπως οι Notes, Apple Maps, Siri, και εισαγωγή της λειτουργίας διαχωρισμού οθόνης (split screen). Ιδιαίτερη προσοχή δόθηκε στην επιτάχυνση της λειτουργίας των συσκευών και την καλύτερη αξιοποίηση των πόρων γενικότερα.

(2016 - 2019) iOS 10
Περισσότερες ελευθερίες εξατομίκευσης, επιλογή αφαίρεσης κάποιων προεγκατεστημένων εφαρμογών. Επικοινωνία μεταξύ διαφορετικών εφαρμογών, κινούμενα γραφικά στην γραπτή επικοινωνία.

(2017) iOS 11 Υποστήριξη επαυξημένης πραγματικότητας, ατομικός διαχειριστής αρχείων (Files). Λειτουργία drag&drop. Εισαγωγή της λειτουργίας Dock μέσω της οποίας ο χρήστης μπορεί να εναλλάσσει γρήγορα και εύκολα τις εφαρμογές που χρησιμοποιεί. Εστίασε κυρίως σε βελτιστοποιήσεις για το iPad.

(2018) iOS 12
Προσαρμογές που επέτρεψαν σε παλιότερες συσκευές να αποδίδουν καλύτερα. Ομαδικές βιντεοκλήσεις (Group FaceTime) και διορθώσεις στην λειτουργία της επαυξημένης πραγματικότητας.

(2019) iOS 13
Λήξη υποστήριξης για την εφαρμογή iTunes Store (κατάστημα μουσικής). Διαφοροποίηση iOS και iPadOS. Χρήσιμες λειτουργίες όπως το dark mode, η αναγνώριση προσώπου κ.α..

(2020) iOS 14
Βελτίωση της διεπαφής χρήστη. Οργάνωση των εφαρμογών μέσω της βιβλιοθήκης App Library. Προσθήκη οδηγιών για ποδηλάτες και τουριστικών οδηγών στο Maps. Ενίσχυση της διαφάνειας σχετικά με την χρήση δεδομένων και συσκευών από τις εφαρμογές.

(2021) iOS 15
Βελτιώσεις στο FaceTime και στο σύστημα ανακοινώσεων. Μια νέα εφαρμογή, το Shared With You, εντοπίζει κι εμφανίζει όσα άρθρα, φωτογραφίες και άλλο υλικό έχει διαμοιραστεί μέσω της εφαρμογής Messages. Το Maps πλέον προσφέρει τρισδιάσταση ξενάγηση μέσω οδηγού επαυξημένης πραγματικότητας. Η εφαρμογή Live Text πλέον αναγνωρίζει κείμενο μέσα από φωτογραφίες, ενώ προστέθηκαν ρυθμίσεις ιδιωτικότητας στις εφαρμογές Siri, Mail κ.α.

2.1.3 Χαρακτηριστικά

Το iOS υποστηρίζει μια σειρά προϊόντων της Apple (Apple Watch, Apple TV, iPad, HomePod). Με αυτό τον τρόπο, κάθε βελτίωση του iOS βελτιώνει την συνολική εμπειρία με όλες τις παραπάνω συσκευές.

Όπως αναφέρθηκε, το iOS είναι λογισμικού κλειστού κώδικα. Αυτό σημαίνει ότι πρόσβαση στον πηγαίο κώδικά του έχουν μόνο οι προγραμματιστές της εταιρείας και ως αποτέλεσμα απαγορεύεται η αναπαραγωγή του από τρίτα μέλη. Παρομοίως με το Chrome OS και αντιθέτως με το Android, αυτό σημαίνει ότι υποστηρίζει και λειτουργεί αποκλειστικά σε συσκευές της εταιρείας (Apple Inc.). Έτσι επιτυγχάνεται μονοπώλιο των μοναδικών χαρακτηριστικών του καθώς και η βελτιστοποίηση του λογισμικού, αφού αφορά μόνο συγκεκριμένες αρχιτεκτονικές συσκευών που αποτελούν επίσης αποκλειστική ιδιοκτησία της Apple, αλλά μειώνεται το εύρος χρηστών που προσεγγίζει. Παράλληλα παρατηρείται συγκριτικά μικρότερο πλήθος προγραμματιστών που ασχολούνται με την ανάπτυξη εφαρμογών που απευθύνονται στο iOS.

Έχοντας δημιουργηθεί αποκλειστικά για συγκεκριμένες συσκευές, το iOS περιέχει βελτιστοποιήσεις και είναι ειδικά σχεδιασμένο ώστε να εκμεταλλευτεί τις δυνατότητες της συσκευής στον έπακρον. Γενικά περιέχει μια συγκριτικά πιο απλή διεπαφή χρήστη σε σχέση πχ. με το Android, κάτι που κρίνεται αρνητικά από πολλούς αλλά προσφέρει μεγαλύτερη αξιοπιστία, ευκολότερη πρόσβαση στις λειτουργίες και αφήνει μικρότερο περιθώριο για λάθη. Επίσης με το να διατηρούνται κάποιες κύριες διαφορές με το Android, ενισχύεται η εικόνα του ως μια μοναδική εμπειρία.

Οι κατασκευαστές του iOS τείνουν να περιμένουν αρκετά ώστε μια καινούρια τεχνολογία να περάσει τα πρώιμα στάδιά της πριν αποφασίσουν να την ενσωματώσουν στα συστήματά τους, πάντα με στόχο την βέλτιστη επίδοση και αξιοπιστία της υπηρεσίας, επιτρέποντας όμως στον ανταγωνισμό να αξιοποιήσει τις νέες τεχνολογίες πριν από αυτούς.

Όσον αφορά τις εφαρμογές, λόγω της αποκλειστικότητας και λόγω του περιορισμένου εύρους συσκευών που τρέχουν iOS σε σύγκριση με αυτές που τρέχουν Android, παρατηρείται ότι το ίδιο λογισμικό συνήθως σημειώνει πολύ καλύτερες επιδόσεις σε μια Apple συσκευή, αφού είναι πολύ πιο εύκολο να προβλέψεις την ακριβή αρχιτεκτονική της συσκευής κατά την σχεδίαση της εφαρμογής.

Επίσης, οι εφαρμογές που έχουν σχεδιαστεί από την ίδια την εταιρεία, όπως συνηθίζεται με το λογισμικό κλειστού κώδικα, σημειώνουν πλεονεκτήματα έναντι των αντίστοιχων εφαρμογών ανοιχτού κώδικα που κυκλοφορούν με τις υπόλοιπες συσκευές. Η Apple προσπαθεί να ανταγωνιστεί την Google, η οποία έχει τον πρώτο λόγο στις περισσότερες εφαρμογές που χρησιμοποιούνται καθημερινά, όπως οι Google Maps, το Google Chrome κ.α. κυκλοφορώντας εφαρμογές ίδιου σκοπού (Maps, Safari) που αν και λιγότερο διαδεδομένες δεν υστερούν σε κάποιον τομέα. Συχνά το πετυχαίνουν. Γενικότερα η Apple έχει καθιερώσει την θέση της ως μια εταιρεία που σχεδιάζει και παρέχει εφαρμογές του υψηλότερου δυνατού επιπέδου.

Ακόμη ένα σημαντικό χαρακτηριστικό είναι η ασφάλεια της εκάστοτε συσκευής. Το iOS προστατεύεται από κακόβουλο λογισμικό σε αρκετά μεγαλύτερο βαθμό από το Android, ενώ η φύση του και η φύση των εφαρμογών του ως λογισμικό κλειστού κώδικα δίνουν λιγότερες ευκαιρίες προς εκμετάλλευση από εγκληματίες. Επίσης, είναι γνωστό ότι η λειτουργία της Google, και κατ' επέκταση του Android, βασίζεται στην συλλογή δεδομένων. Τα προϊόντα της διατίθενται φαινομενικά δωρεάν, αλλά οι πληροφορίες του χρήστη γίνονται προϊόν της εταιρείας. Αντιθέτως, η Apple δεν χρειάζεται να βασιστεί σε αυτό για τα έσοδά της, κι έτσι δείχνει μεγαλύτερο σεβασμό προς τα προσωπικά δεδομένα.

2.1.4 Ανάπτυξη iOS εφαρμογών

Η ανάπτυξη iOS εφαρμογών είναι μια συνεχώς αυξανόμενη τάση. Δεδομένου της διάδοσης των Apple συσκευών, η ανάπτυξη εφαρμογών για αυτές είναι ζητούμενο από πολλές εταιρείες και μια δυνητικά πολύ κερδοφόρα ασχολία. Οι iOS εφαρμογές χαρακτηρίζονται από αξιοπιστία και ασφάλεια, αφού καλούνται να τηρήσουν αυστηρούς κανόνες και να πληρούν συγκεκριμένες τόσο σχεδιαστικές όσο και τεχνικές προϋποθέσεις ώστε να γίνουν δεκτές στο AppStore. Με αυτόν τον τρόπο επιτυγχάνεται σχετική ομοιομορφία μεταξύ των διαφορετικών εφαρμογών και διατηρείται ένας συγκεκριμένος χαρακτήρας στο "οικοσύστημα" της Apple. Οι προδιαγραφές μπορεί να είναι εν μέρη περιοριστικές, όμως προτείνουν ένα καθαρό πλάνο ανάπτυξης που επισπεύδει κατά πολύ την διεργασία.

Υπάρχουν πολλά εργαλεία και παράγοντες που κάνουν την ανάπτυξη εφαρμογών για iOS μια πολύ ενδιαφέρουσα εμπειρία. Κάποιες από τις τεχνολογίες θα αναλυθούν και στις επόμενες ενότητες σε μεγαλύτερο βάθος. Η γλώσσα προγραμματισμού που χρησιμοποιείται για την ανάπτυξη των εφαρμογών είναι η Swift, μια πολύ σταθερή και μοντέρνα γλώσσα υψηλού επιπέδου, που σημαίνει ότι είναι κοντά στην φυσική γλώσσα και συνεπώς εύκολη ως προς την συγγραφή και την κατανόηση. Κατά την ανάπτυξη των εφαρμογών και για τις δοκιμές τους επιβάλλεται η χρήση προσομοίωσης, και τα εργαλεία που υπάρχουν για αυτό το σκοπό και αφορούν το iOS είναι ανώτερου επιπέδου από τα αντίστοιχα για Android. Για την εν λόγω ανάπτυξη μπορεί να χρησιμοποιηθεί το XCode, το οποίο και πάλι είναι πολύ καλύτερο από το Android Studio στους περισσότερους τομείς. Επιπλέον, το μειωμένο πλήθος συσκευών στις οποίες απευθύνεται το λογισμικό, εξασφαλίζει την επίτευξη ομαλής λειτουργίας της εφαρμογής με λιγότερη προσπάθεια.

2.2 Frameworks:

2.2.1 Γενικές πληροφορίες

Ως framework ορίζεται μια δομή που υποστηρίζει την ανάπτυξη λογισμικού πάνω σε αυτή, λειτουργώντας ως σκελετός που περιέχει κάποια βασικά, απαραίτητα προγραμματιστικά χαρακτηριστικά ανάλογα με το είδος της εφαρμογής που πρόκειται να αναπτυχθεί. Συνήθως είναι σχεδιασμένα έτσι ώστε να υποστηρίζουν κάποια συγκεκριμένη προγραμματιστική γλώσσα.

Η χρήση των frameworks είναι μια σίγουρη δίοδος εξοικονόμησης χρόνου, αφού προσφέρει έτοιμα τα αρχικά στάδια ανάπτυξης της εφαρμογής. Επίσης αφαιρεί την ευθύνη από τον εκάστοτε προγραμματιστή να εκτελέσει τους απαραίτητους εκτενείς ελέγχους ασφάλειας και αξιοπιστίας. Τα frameworks, όπως κάθε άλλο είδος λογισμικού, είναι σχεδιασμένα από ομάδες ατόμων που εξειδικεύονται στην ανάπτυξή τους και συντηρούνται έτσι ώστε να είναι έτοιμα προς χρήση οποιαδήποτε στιγμή.

Εν ολίγοις, ένα framework προσφέρει στον προγραμματιστή πιο ασφαλή και καλύτερης ποιότητας κώδικα, διευκολύνει την περάτωση δοκιμών πάνω στην εφαρμογή, επιτρέπει στον προγραμματιστή να επικεντρωθεί στην ανάπτυξη της ίδιας της εφαρμογής πέρα των τυπικών (αλλά απαραίτητων) πρώτων βημάτων, και τέλος, το framework δίνει την δυνατότητα επέκτασης και προσαρμογής του σε εξειδικευμένες ανάγκες. Εάν πρόκειται για λογισμικό κλειστού κώδικα, συνήθως η κίνηση τροφοδοτείται από κάποιο εμπορικό ενδιαφέρον, ενώ αν πρόκειται για λογισμικό ανοιχτού κώδικα, συναντώνται περισσότερες επεκτάσεις και παραλλαγές, συχνά όμως χωρίς κάποια ιδιαίτερη εγγύηση ποιότητας.

2.2.2 Κατηγορίες

Τα frameworks χωρίζονται σε δύο κύριες κατηγορίες, ανάλογα με τον τύπο εφαρμογής που υποστηρίζουν:

1. Native

   Οι native εφαρμογές είναι ειδικά σχεδιασμένες για ένα συγκεκριμένο λειτουργικό σύστημα (iOS, Android, ChromeOS, Windows, Linux κλπ.). Πρόκειται για εφαρμογές που είναι εγκατεστημένες στην συσκευή και βασίζονται στο υλικό της για την λειτουργία τους. Η ανάπτυξη native framework και εφαρμογών είναι σχετικά πιο δύσκολη, και έχει μικρότερη απήχηση, λόγω του ότι πρόκειται για μια διαδικασία που χρειάζεται περισσότερες προγραμματιστικές γνώσεις και είναι χρονοβόρα, αφού πρέπει να ληφθούν υπόψη τα χαρακτηριστικά της εκάστοτε συσκευής. Αυτό για μια εταιρεία σημαίνει αυξημένα έξοδα.

2. Hybrid

   Οι hybrid εφαρμογές συνδυάζουν χαρακτηριστικά των native, με των web εφαρμογών. Οι web εφαρμογές είναι ειδικά σχεδιασμένες για να εξυπηρετούν τον σκοπό τους μέσω ενός φυλλομετρητή (Safari, Mozilla Firefox, Google Chrome κλπ.). Πρόκειται για προγράμματα που φιλοξενούνται σε κάποιο διακομιστή, και η λειτουργία τους βασίζεται στους πόρους του διακομιστή, και την ποιότητα της σύνδεσης. Η ανάπτυξη hybrid framework δίνει την δυνατότητα ανάπτυξης εφαρμογών που απευθύνονται σε περισσότερες από μία πλατφόρμες ή λειτουργικά συστήματα. Με αυτό τον τρόπο συγκεντρώνεται μεγαλύτερο ενδιαφέρον προς την ανάπτυξη, αφού αφορά ευρύτερο κοινό καταναλωτών, και είναι συνήθως πιο εύκολη και γρήγορη διαδικασία. Επιπλέον, λόγω της ευρείας χρησιμότητάς τους, είναι πιο πιθανό οι προγραμματιστές της ομάδας να έχουν προηγούμενη εμπειρία με το εργαλείο, ενδεχόμενο το οποίο είναι σημαντικός παράγοντας κατά την επιλογή του κατάλληλου framework.

2.3 Web APIs

2.3.1 Γενικές πληροφορίες

Το API (Application Program Interface) είναι μια διεπαφή μέσω της οποίας η εφαρμογή επικοινωνεί με το τμήμα που διαχειρίζεται τα δεδομένα (application backend). Επιτρέπει την συνεργασία διαφορετικών υπηρεσιών. Συγκεκριμένα με τον όρο Web API αναφερόμαστε στον τρόπο με τον οποίο γίνεται προσπέλαση και χρήση των δεδομένων μιας δικτυακής εφαρμογής, συνήθως μέσω μιας σειράς HTTP αιτημάτων. Ο χρήστης χρειάζεται πρόσβαση σε κάποια πληροφορία, η εφαρμογή αποφασίζει τον τύπο πρόσβασης που θα χρειαστεί, μεταφέρει το αίτημα στο API, και το API βάση των κανόνων επικοινωνίας που ορίζει είτε αποσπά την πληροφορία από το σύστημα διαχείρισης δεδομένων ή προχωρά σε περαιτέρω ενέργειες εξουσιοδότησης. Το API είναι ένα ενδιάμεσο στάδιο ανάμεσα στην εφαρμογή και τον διακομιστή δεδομένων (server) ώστε να μην χρειαστεί να αναλάβει η εφαρμογή (και κατ΄ επέκταση ο προγραμματιστής) εξ ολοκλήρου τον τρόπο επικοινωνίας.

2.3.2 Διαδικασία ανάπτυξης

Η διαδικασία ανάπτυξης ενός API μπορεί να διαχωριστεί σε πέντε χρονολογικά στάδια.

1. Σχεδιασμός
   Το αρχικό στάδιο είναι, φυσικά, ο σχεδιασμός του. Σε αυτό το στάδιο συγκεντρώνονται ιδέες περί των λειτουργιών που θα τελεί, και τους πόρους που απαιτούνται για την κάθε ενέργεια. Δημιουργούνται σχολαστικά διαγράμματα περιπτώσεων χρήσης του API ώστε να υπάρχει μια ξεκάθαρη εικόνα πριν αρχίσει η υλοποίησή του. Η εικόνα θα πρέπει να περιέχει όλες τις δυνατές λειτουργίες του API, βάση των αναγκών που αποσκοπεί να καλύψει, καθώς και τα δεδομένα που εμπλέκονται και τον τρόπο που επεξεργάζονται. Μέτρα ασφαλείας και η ορατότητα των δεδομένων παίζουν καθοριστικό ρόλο. Τα πρωτόκολλα επικοινωνίας, οι τεχνολογίες που θα χρησιμοποιηθούν και άλλοι κανόνες ορίζουν την τελική αρχιτεκτονική του.

2. Ανάπτυξη
   Την δημιουργία του πλάνου ακολουθεί το στάδιο της ανάπτυξης του API. Εφόσον έχουν προσδιοριστεί και ικανοποιηθεί οι προαπαιτήσεις που ορίστηκαν προηγουμένως, η ομάδα ανάπτυξης εστιάζει στην υλοποίηση βάση των σχεδίων. Σε αυτό το στάδιο συμπεριλαμβάνεται και η σύνταξη του σχετικού οδηγού χρήσης, όπου επεξηγούνται τα τμήματα από τα οποία αποτελείται και οι διάφοροι τρόποι χρήσης του από το κοινό στο οποίο θα απευθυνθεί. Η ύπαρξη ενός καλού οδηγού είναι εξίσου σημαντική με το ίδιο το λογισμικό, ιδιαίτερα όταν πρόκειται για εμπορικό προϊόν.

3. Σχολαστικές δοκιμές
   Κατά το στάδιο των δοκιμών, η υπεύθυνη ομάδα καλείται να ελέγξει σχολαστικά το API για αδυναμίες στον τομέα της απόδοσης, όσο και για πιθανά λογικά ή προγραμματιστικά λάθη που οδηγούν σε απρόσμενες και ανεπιθύμητες συμπεριφορές. Αποτελεί απαραίτητο βήμα πριν την κυκλοφορία του λογισμικού, και όπως για κάθε είδους πρόγραμμα, σχεδόν ποτέ δεν είναι αρκετό. Έτσι, αναμενόμενο είναι ότι θα υπάρξουν ενημερώσεις (patches) που θα διορθώνουν τα λάθη που εντοπίζονται μετέπειτα της κυκλοφορίας του.

4. Διαχείριση
   Αφού το API έχει περάσει επιτυχώς από τους απαραίτητους ελέγχους, και βρεθεί ότι λειτουργεί σύμφωνα με τον σχεδιασμό και τους οδηγούς, πλέον θεωρείται έτοιμο προς διάθεση Σημαντικός παράγοντας για την επιτυχία του είναι η φιλοξενία του σε αξιόπιστες πλατφόρμες που μπορούν να εγγυηθούν στους καταναλωτές για την ασφάλεια και την ποιότητά του ως προϊόν. Όπως είναι σύνηθες με τα νέα προγράμματα, συχνά ένα ενδιάμεσο στάδιο μεταξύ των δοκιμών και της κυκλοφορίας, είναι η διανομή μιας πρώιμης έκδοσης (beta version), όπου ο χρήστης καλείται να χρησιμοποιήσει το API και να αναφέρει τυχόν δυσκολίες ή προβλήματα που συναντά. Ακολουθώντας την επίσημη κυκλοφορία, η ομάδα της διαχείρισης καλείται να συλλέγει και να αναλύει στατιστικά όπως οι επιδόσεις, τα προβλήματα και η ικανοποίηση των χρηστών ώστε να βελτιωθούν οι υπηρεσίες και η εμπειρία που προσφέρει ως πρόγραμμα. Ανάλογα με την φύση του API, υπάρχουν και περιπτώσεις όπου χτίζεται μια κοινότητα γύρω από αυτό, γίνεται ανταλλαγή ιδεών, προσφέρεται βοήθεια με συχνά προβλήματα, κ.α..

5. Απόσυρση
   Αν για οποιοδήποτε λόγο δεν μπορεί να συνεχιστεί η διαχείριση του API, ή έχει προγραμματιστεί η αντικατάστασή του από κάποιο καινούριο πρόγραμμα, είναι σημαντικό να μην εγκαταλειφθεί εν ψυχρό η υπάρχουσα υποδομή. Πρέπει να βρεθούν εναλλακτικές για τους χρήστες που βασίζονται στην χρήση του ώστε να γίνει σταδιακή απεξάρτησή τους από το λογισμικό. Ο ξαφνικός τερματισμός λειτουργίας ενός API, μπορεί να οδηγήσει σε αδυναμία λειτουργίας όλων των εφαρμογών που βασίζονται σε αυτό.

Κεφάλαιο 3ο: Εργαλεία

3.1 XCode

Το XCode είναι ένα εργαλείο ανάπτυξης και προσομοίωσης εφαρμογών προοριζόμενες για τα λειτουργικά συστήματα της Apple. Θεωρείται ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (IDE), δηλαδή εκτός της παροχής εργαλείων που διευκολύνουν την συγγραφή κώδικα, παρέχει και όλες τις κατάλληλες υπηρεσίες για την ανάπτυξη εφαρμογών, όπως είναι ο μεταγλωττιστής (compiler) και μια διεπαφή χρήστη όπου γίνονται οι δοκιμές της εφαρμογής πριν φτάσει στο στάδιο όπου μπορεί να χρησιμοποιηθεί και να κυκλοφορήσει. Δίνει επιπλέον την λειτουργία προσομοίωσης συσκευών. Αυτό είναι ένα άκρως απαραίτητο χαρακτηριστικό ειδικά για την ανάπτυξη εφαρμογών κινητών τηλεφώνων, αφού εξυπακούεται ότι δεν γίνεται κατά την ανάπτυξη να δοκιμαστεί το προϊόν στις αντίστοιχες αληθινές συσκευές. Ο προγραμματιστής πρέπει να μπορεί να αναπτύξει την εφαρμογή μόνο με την χρήση του προσωπικού του υπολογιστή.

3.2 Γλώσσα προγραμματισμού: Swift

3.2.1 Γενικές πληροφορίες

Η Swift είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου η οποία έχει βάσεις στις σύγχρονες γλώσσες προγραμματισμού (Python, C/C++, Java κλπ), απ' όπου παρουσιάζει διάφορα χαρακτηριστικά, με ιδιαίτερη έμφαση στην ασφάλεια. Χρησιμοποιείται ως επί των πλείστων στην ανάπτυξη εφαρμογών που απευθύνονται στα λειτουργικά συστήματα της Apple (iOS, macOS, watchOS, tvOS), αλλά και στις διάφορες εκδόσεις του Linux, μιας και είναι επίσης βασισμένες στο UNIX.

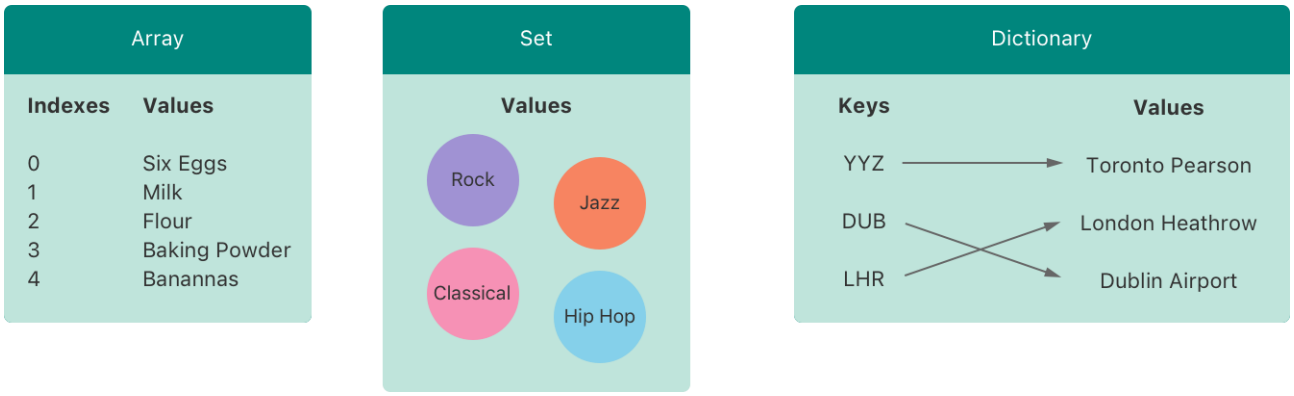
Η γλώσσα εμφανίστηκε το 2014, και σημειώνει ραγδαία αυξανόμενη βάση προγραμματιστών από τότε. Αυτό οφείλεται στα πολυπληθή πλεονεκτήματα που προσφέρει. Ως γλώσσα υψηλού επιπέδου, η Swift είναι φιλική προς τον προγραμματιστή, είτε αυτό αφορά την εξοικείωση είτε την χρήση της. Επιπλέον, αυτό επιτυγχάνεται χωρίς να συμβιβαστεί η ταχύτητα και η αξιοπιστία της, κάτι που την καθιστά μία από τις πιο ισχυρές γλώσσες προγραμματισμού στις μέρες μας. Όλα τα παραπάνω έχουν βοηθήσει την Swift να καθιερωθεί ως μια γλώσσα κατάλληλη τόσο για ερασιτέχνες και φοιτητές, όσο και επαγγελματίες, ενώ χρησιμοποιείται για την ανάπτυξη εφαρμογών φορητών και μη συσκευών. Τέλος, αν και σπανιότερα, μπορεί να χρησιμοποιηθεί και για Unix ή Windows λειτουργικά συστήματα, κάτι που σε συνδυασμό με τα προηγούμενα προτερήματα αυξάνει ακόμα περισσότερο το ενδιαφέρον και την χρησιμότητά της.

Ως μοντέρνα γλώσσα προγραμματισμού, η Swift καλείται και επιτυγχάνει να αντιμετωπίσει κοινά, συχνά προγραμματιστικά προβλήματα.

- Απαγορεύεται η χρήση μη-αρχικοποιημένων μεταβλητών.
- Ελέγχεται το ενδεχόμενο υπερχείλισης ακεραίων μεταβλητών.
- Η πρόσβαση σε δείκτες μιας λίστας ελέγχεται ώστε να μην γίνει απόπειρα πρόσβασης σε εκτός ορίων, απροσδιόριστη θέση μνήμης.
- Αυτόματη διαχείριση μνήμης
- Διαχείριση λαθών

3.2.2 Τύποι δεδομένων

Οι τύποι δεδομένων που χρησιμοποιούνται από την Swift είναι τροποποιημένες εκδοχές των γνωστών int, double, float, bool, string που χρησιμοποιούνται στην βασική και Objective C. Ένας επιπρόσθετος, ισχυρός τύπος δεδομένων είναι τα tuples, τα οποία επιτρέπουν την αποθήκευση δεδομένων διαφορετικών τύπων σε μία ενιαία δομή. Ακόμη, ως τύποι δεδομένων παρέχονται οι δομές συλλογής δεδομένων Array, Set και Dictionary.



Εικόνα 1: Collection Types
(https://docs.swift.org/swift-book/LanguageGuide/CollectionTypes.html)

Τέλος, η Swift παρέχει optional τύπους δεδομένων, ώστε να διαχειριστεί το ενδεχόμενο μη-αρχικοποίησης με τιμή. Οι optionals αναφέρονται σε μεταβλητές που μπορεί αλλά και μπορεί να μην έχουν μια τιμή (παρόμοια έννοια με το NULL της C και το nil της Objective-C).

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

Κώδικας 1: Τρόποι δήλωσης optional μεταβλητών
(https://developer.apple.com/documentation/swift/optional)

```
let number: Int? = Optional.some(42)
let noNumber: Int? = Optional.none
print(noNumber == nil)
// Prints "true"
```

Κώδικας 2: Μερικοί ακόμα τρόποι δήλωσης optional μεταβλητών
(https://developer.apple.com/documentation/swift/optional)

Σε αντίθεση με τις περισσότερες προγραμματιστικές γλώσσες, η Swift επιτρέπει την χρήση σχεδόν όλων των Unicode χαρακτήρων κατά την ονομασία μιας μεταβλητής. Επιτρέπεται επιπλέον (αν και δεν προτείνεται) η χρήση δεσμευμένων λέξεων. Σε αυτή την περίπτωση το όνομα της μεταβλητής πρέπει να περικλείεται από μονά ανάποδα εισαγωγικά (`). Διατηρούνται κάποιοι βασικοί κανόνες, δηλαδή απαγορεύεται η χρήση αριθμών στην αρχή του ονόματος, κενών (whitespace) 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
(https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html)

Όπως και οι προκάτοχοί της, η Swift έχει ρυθμισμένες δικλείδες ασφαλείας ώστε κατά την μεταγλώττισή της να προβλέπονται και να αποτρέπονται κοινά προγραμματιστικά λάθη όπως η ανάθεση λάθος τύπου δεδομένου σε μια μεταβλητή.

3.2.3 Χαρακτηριστικά

Ως γλώσσα γενικού σκοπού η Swift πληροί τα παρακάτω προγραμματιστικά χαρακτηριστικά (χρησιμοποιούνται οι αγγλικές ορολογίες):

3.2.3.1 Multi-paradigm

Η Swift υποστηρίζει διάφορες προγραμματιστικές προσεγγίσεις, όπως:

1. Object oriented
   Αντικειμενοστρέφεια. Η αντικειμενοστρέφεια είναι μια έννοια που διαχωρίζει τις οντότητες του προγράμματος σε αντικείμενα, τα οποία ανήκουν σε μια κοινή οικογένεια που ονομάζεται κλάση. Τα αντικείμενα μιας κλάσης χαρακτηρίζονται από κοινά χαρακτηριστικά και μεθόδους επεξεργασίας των εν λόγω χαρακτηριστικών. Η χρήση αντικειμενοστρέφειας επιτρέπει για πιο κατανοητό και μικρότερου μεγέθους κώδικα, ειδικά όταν πρόκειται για επαναλαμβανόμενα μοτίβα.
   Χαρακτηριστικά της αντικειμενοστρέφειας είναι η ότι κάθε κλάση καλείται να παρέχει και να διαχειρίζεται όλες τις απαραίτητες λειτουργίες που την αφορούν, έναντι του κύριου προγράμματος (dynamic dispatch). Η πρακτική αυτή ενισχύεται από το γεγονός ότι στον αντικειμενοστραφή προγραμματισμό ορατά είναι μόνο τα αντικείμενα και οι λειτουργίες που επικοινωνούν άμεσα με τα υπόλοιπα αντικείμενα και λειτουργίες του προγράμματος, ενώ τα υπόλοιπα διαχειρίζονται εσωτερικά και δεν αφορούν ή επηρεάζονται από γειτονικά στοιχεία του προγράμματος (ενθυλάκωση). Στην αντικειμενοστρέφεια έχουμε επίσης κληρονομικότητα και πολυμορφισμό. Τα αντικείμενα μπορούν να συσχετίζονται με άλλα αντικείμενα με σχέση συγγένειας και να κληρονομούν τα στοιχεία τους, επεκτείνοντας στην παλιότερη έννοια με νέα επιπρόσθετα χαρακτηριστικά και μεθόδους. Ο πολυμορφισμός αναφέρεται στην ιδιότητα των συγγενικών αντικειμένων να μπορούν να εκτελέσουν τόσο τον κώδικα της κλάσης-γονέα όσο και του αντίστοιχου ανανεωμένου κώδικα που τα αφορά.

2. Functional
   Σε αντίθεση με την αντικειμενοστρέφεια, ο συναρτησιακός προγραμματισμός αντιμετωπίζει τα ζητούμενα ως αποτέλεσμα εκτέλεσης συναρτήσεων και δεν δίνει βάση τόσο στην ομαδοποίηση της πληροφορίας σε αντικείμενα, όσο δίνει στην επίλυση των προβλημάτων μέσω των συναρτήσεων. Αντί για αντικείμενα και μεθόδους, συναντώνται μεταβλητές και συναρτήσεις. Αποτελεί μια πιο άμεση, αλγοριθμική προσέγγιση.

3. Imperative
   Προστακτικός προγραμματισμός. Σε αντίθεση με τον συναρτησιακό προγραμματισμό, που θέλει τα αντικείμενα ως επί το πλείστων αμετάβλητα, εδώ η εκτέλεση των εντολών έχει ως άμεσο σκοπό την αλλαγή κατάστασης των αντικειμένων. Είναι πιο περίπλοκος από τον συναρτησιακό, και πιο δύσκολος στην υλοποίηση όταν χρειάζεται παράλληλη εκτέλεση του προγράμματος.

4. Block-structured
   Ορισμός συνόλου εντολών. Όταν υπάρχει σειριακή εκτέλεση ενός προγράμματος, συχνά οι εντολές που εκτελούνται εξαρτώνται από κάποιον έλεγχο ή συνθήκη. Για παράδειγμα, μπορεί δεδομένου μιας συνθήκης να πρέπει να εκτελεστούν συγκεκριμένες εντολές, και η εκτέλεση του προγράμματος να συνεχιστεί έπειτα από μια συγκεκριμένη ομάδα εντολών που πρέπει σε αυτή την περίπτωση να αγνοηθούν. Σε αυτή την περίπτωση, με την χρήση των αγκύλων ορίζεται η πρώτη και η δεύτερη ομάδα εντολών, και αντιμετωπίζονται από το πρόγραμμα ως ξεχωριστές εντολές.

3.2.3.2 Control flow

Συνθήκες ελέγχου. Όπως αναφέρθηκε προηγουμένως, μία κοινή προγραμματιστική τεχνική είναι η εκτέλεση συγκεκριμένων εντολών, ανάλογα με την τήρηση ή μη κάποιας συνθήκης. Αυτό επιτρέπει την αλγοριθμική προσέγγιση του προβλήματος. Επίσης, συχνά χρειάζεται να επαναληφθεί ένα τμήμα του κώδικα για συγκεκριμένο πλήθος επαναλήψεων ή μέχρις ότου να ικανοποιηθεί μια συνθήκη. Η επανάληψη είναι ένα ουσιαστικά απαραίτητο προγραμματιστικό χαρακτηριστικό που συναντάται στην πλειοψηφία των γλωσσών προγραμματισμού, και εμπεριέχεται στις συνθήκες ελέγχου.

3.2.3.2.1 Συνθήκη ελέγχου if

Την συνθήκη ελέγχου if θα την συναντήσουμε πολλές φορές στα προγράμματα μας . Όπως δηλώνει και το όνομα της “if” (Αν) ,ελέγχει μια συνθήκη αν είναι αληθής η ψευδής. Αν είναι αληθής , εκτελούνται και οι αντίστοιχες εντολές που περικλείονται από την συνθήκη, αν είναι ψευδής , δηλαδή δεν ισχύει , τότε δεν εκτελούνται οι εντολές της συνθήκης.

Σε κάποιες περιπτώσεις ο έλεγχος θα πρέπει να επεκταθεί και σε περισσότερους ελέγχους “else if” και να καταλήξει εάν όλοι οι έλεγχοι είναι ψευδής σε μια τελική “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
4.  }
5.  // Prints "It's very cold. Consider wearing a scarf."
```

```
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."
```

```
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 3: Η συνθήκη ελέγχου if
(https://docs.swift.org/swift-book/LanguageGuide/ControlFlow.html)

3.2.3.2.2 Συνθήκη ελέγχου switch

Η συνθήκη ελέγχου switch είναι αντίστοιχη της “if .. else if” με τη διαφορά ότι αντί να χρησιμοποιούμαι πολλές φορές το “else if” δίνουμε περιπτώσεις “case” που αν είναι αληθής δηλαδή ισχύει μια περίπτωση τότε θα εκτελεστούν οι ανάλογες εντολές . Εάν δεν ισχύει καμιά περίπτωση τότε δηλώνουμε μια προκαθορισμένη “default” περίπτωση που θα εκτελέσει τις ανάλογες εντολές .

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

```
let anotherCharacter: Character = "a"
switch anotherCharacter {
case "a": // Invalid, the case has an empty body
case "A":
print("The letter A")
default:
print("Not the letter A")
}
// This will report a compile-time error.
```

```
let anotherCharacter: Character = "a"
switch anotherCharacter {
case "a", "A":
print("The letter A")
default:
print("Not the letter A")
}
// Prints "The letter A"
```

Κώδικας 4: Η συνθήκη ελέγχου switch

(https://docs.swift.org/swift-book/LanguageGuide/ControlFlow.html)

3.2.3.3.3 Δομή επάναληψης for

Η δομή επανάληψης for χρησιμοποιείται για να μην χρειάζεται να επαναλαμβάνουμε πολλές φορές τα ίδια κομμάτια κώδικα . Υπάρχει μια συνθήκη που όσο ισχύει αυτή η συνθήκη θα εκτελούνται οι αντίστοιχες εντολές.

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

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

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

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

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

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

Κώδικας 5: Η δομή επανάληψης for

(https://docs.swift.org/swift-book/LanguageGuide/Functions.html)

3.2.3.3 Functions

Επιτρέπει τον ορισμό μιας συνάρτησης και την ανατροφοδότηση των κατάλληλων παραμέτρων σε αυτή ώστε να έχουμε το επιθυμητό αποτέλεσμα. Πρόκειται για τμήματα κώδικα που μπορούν να επαναχρησιμοποιηθούν. Όπως και οι μεταβλητές, κάθε συνάρτηση είναι συγκεκριμένου τύπου δεδομένων, ανάλογα με το αντικείμενο που επιστρέφει.

```
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 6: Συναρτήσεις

(https://docs.swift.org/swift-book/LanguageGuide/Functions.html)

3.2.3.4 Generics

Επιτρέπουν στην αναπαραγωγή κώδικα που μπορεί να απευθύνεται σε διαφορετικούς τύπους δεδομένων. Παρομοίως με τα templates, οι generic συναρτήσεις είναι σχεδιασμένες έτσι ώστε να έχουν προβλέψει και να μπορούν να εκτελεστούν για οποιοδήποτε τύπο δεδομένων. Αυτό βοηθά στην ανακύκλωση του κώδικα, την μείωση του όγκου του, και την επαναχρησιμοποίησή του. Τα generics έχουν ακόμα ως λειτουργία τον περιορισμό σε κάποια συγκεκριμένη ομάδα τύπου δεδομένων μέσω των constraints.

```
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 7: Παράδειγμα χρήσης generic συνάρτησης

(https://docs.swift.org/swift-book/LanguageGuide/Functions.html)

```
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 8: Επιπλέον παράδειγμα χρήσης generic συνάρτησης με constraints

(https://docs.swift.org/swift-book/LanguageGuide/Functions.html)

3.2.3.5 Structures and classes

Δομές και κλάσεις, βασίζονται στον αντικειμενοστραφή σχεδιασμό. Όπως και με τις κλάσεις, στις δομές ορίζονται κάποια χαρακτηριστικά και μέθοδοι που χαρακτηρίζουν μια συγκεκριμένη ομάδα οντοτήτων. Οι δομές δεν παρουσιάζουν κληρονομικότητα και δεν έχουν αυτόματες μεθόδους αποδέσμευσης μνήμης, οπότε ουσιαστικά είναι μια απλουστευμένη εκδοχή των κλάσεων. Μία ακόμα σημαντική διαφορά είναι ότι, τα structs είναι τύπος αξίας δεδομένων, ενώ οι κλάσεις τύπος αναφοράς. Τα structs αντιγράφονται με την ανάθεσή τους σε μια νέα μεταβλητή, ενώ για τις κλάσεις δημιουργείται μια αναφορά στο αρχικό αντικείμενο, με αποτέλεσμα οποιαδήποτε αλλαγή να επηρεάσει και το ίδιο.

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

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

Κώδικας 9: Σύγκριση δομής και κλάσης

(https://docs.swift.org/swift-book/LanguageGuide/ClassesAndStructures.html)

```
enum CompassPoint {
case north, south, east, west
mutating func turnNorth() {
self = .north
   }
   }
var currentDirection = CompassPoint.west
let rememberedDirection = currentDirection
currentDirection.turnNorth()
print("The current direction is \(currentDirection)")
print("The remembered direction is \(rememberedDirection)")
// Prints "The current direction is north"
// Prints "The remembered direction is west"
```

Κώδικας 10: Η δομή enum

(https://docs.swift.org/swift-book/LanguageGuide/ClassesAndStructures.html)

```
let tenEighty = VideoMode()
tenEighty.resolution = hd
tenEighty.interlaced = true
tenEighty.name = "1080i"
tenEighty.frameRate = 25.0
let alsoTenEighty = tenEighty
alsoTenEighty.frameRate = 30.0
print("The frameRate property of tenEighty is now \(tenEighty.frameRate)")
// Prints "The frameRate property of tenEighty is now 30.0"
```

```
let hd = Resolution(width: 1920, height: 1080)
var cinema = hd
cinema.width = 2048
print("cinema is now \(cinema.width) pixels wide")
// Prints "cinema is now 2048 pixels wide"
print("hd is still \(hd.width) pixels wide")
// Prints "hd is still 1920 pixels wide"
```

Κώδικας 11: Σύγκριση χειρισμού by value και by reference

(https://docs.swift.org/swift-book/LanguageGuide/ClassesAndStructures.html)

3.2.3.6 Error Handling

Διαχείριση λαθών. Όσο καλά και να έχει σχεδιαστεί ένα πρόγραμμα, πάντα θα προκύπτουν σενάρια τα οποία δεν μπορεί να διαχειριστεί. Σε αυτές τις περιπτώσεις, χρησιμοποιούνται τεχνικές διαχείρισης αυτών των σεναρίων οι οποίες επισημαίνουν στον χρήστη ότι προέκυψε κάποιο λάθος ή προσπαθούν να το διορθώσουν ανακατευθύνοντας το πρόγραμμα αντί να διακοπεί η λειτουργία του.

```
func canThrowErrors() throws -> String
func cannotThrowErrors() -> String
```

Κώδικας 12: Διαχείριση λαθών

(https://docs.swift.org/swift-book/LanguageGuide/ErrorHandling.html)

```
let favoriteSnacks = [
"Alice": "Chips",
"Bob": "Licorice",
"Eve":"Pretzels",
]
func buyFavoriteSnack(person: String,
vendingMachine: VendingMachine) throws {
let snackName = favoriteSnacks[person] ?? "Candy
Bar"
try vendingMachine.vend(itemNamed: snackName)}
```

```
var vendingMachine = VendingMachine()
vendingMachine.coinsDeposited = 8
do {
try buyFavoriteSnack(person: "Alice", vendingMachine: vendingMachine)
print("Success! Yum.")
} catch VendingMachineError.invalidSelection {
print("Invalid Selection.")
} catch VendingMachineError.outOfStock {
print("Out of Stock.")
} catch VendingMachineError.insufficientFunds(let coinsNeeded) {
print("Insufficient funds. Please insert an additional \(coinsNeeded)
coins.")
} catch {
print("Unexpected error: \(error).")
}
// Prints "Insufficient funds. Please insert an additional 2 coins."
```

```
func someThrowingFunction() throws -> Int {
// ...
}
Dsdsd
let x = try? someThrowingFunction()
Sd
let y: Int?
do {
    y = try someThrowingFunction()Sd
} catch {
     y = nil
}
```

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

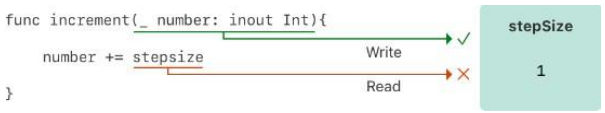
Κώδικας 13: Παραδείγματα συναρτήσεων που διαχειρίζονται λάθη

(https://docs.swift.org/swift-book/LanguageGuide/ErrorHandling.html)

3.2.3.7 Memory management

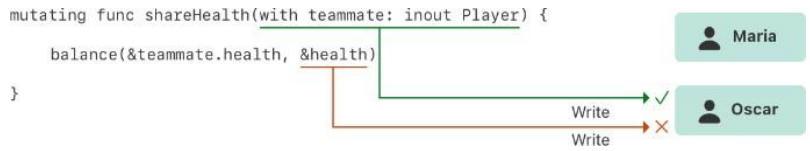
Διαχείριση μνήμης. Όπως και με την διαχείριση λαθών, έτσι και εδώ ο προγραμματιστής καλείται να προβλέψει και να αποφύγει όσο το δυνατόν περισσότερα πιθανά προβλήματα που μπορεί να προκύψουν με την ανάθεση μνήμης κατά την εκτέλεση του προγράμματος. Όπως πολλές άλλες μοντέρνες γλώσσες, έτσι και η Swift επιβάλει κάποιους κανόνες για να ενισχύσει την εφαρμογή ασφαλών πρακτικών όσον αφορά την μνήμη. Για παράδειγμα, απαγορεύεται η χρήση μεταβλητών που δεν έχουν αρχικοποιηθεί, η πρόσβαση σε μνήμη που έχει αποδεσμευτεί, καθώς και η υπέρβαση των ορίων δομών όπως οι πίνακες και οι λίστες. Επιπλέον, η Swift, όπως και η Java, διαχειρίζεται αυτόματα την μνήμη σε μεγάλο βαθμό. Αυτό αποτρέπει προγραμματιστικά λάθη που εμφανίζονται πολύ συχνά σε άλλες γλώσσες που αφήνουν την διαχείριση της μνήμης στον προγραμματιστή, όπως είναι η C και η C++.

```swift
var stepSize = 1
func increment(_ number: inout Int) {
number += stepSize
}
increment(&stepSize)
// Error: conflicting accesses to stepSize
```

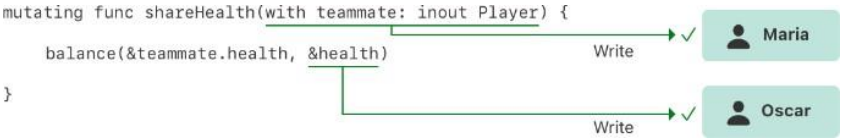



```swift
// Make an explicit copy.
var copyOfStepSize = stepSize
increment(&copyOfStepSize)
// Update the original.
stepSize = copyOfStepSize
// stepSize is now 2
```

```swift
struct Player {
var name: String
var health: Int
var energy: Int
static let maxHealth = 10
mutating func restoreHealth() {
health = Player.maxHealth
}
}
extension Player {
mutating func shareHealth(with teammate: inout Player) {
balance(&teammate.health, &health)
}
}
var oscar = Player(name: "Oscar", health: 10, energy: 10)
var maria = Player(name: "Maria", health: 5, energy: 10)
oscar.shareHealth(with: &oscar)
// Error: conflicting accesses to Oscar
```




```swift
oscar.shareHealth(with: &maria) // OK
```



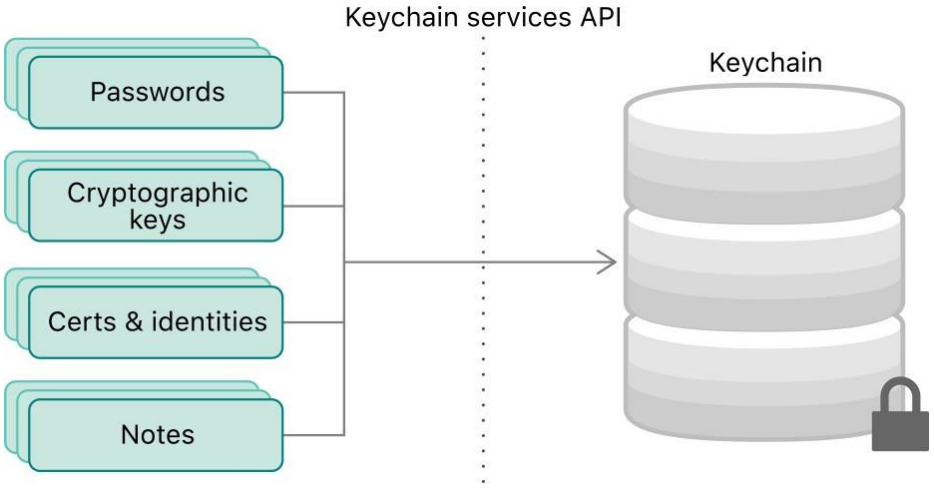
Κώδικας 14: Παραδείγματα λάθους διαχείρισης μνήμης και τρόποι επίλυσης

(https://docs.swift.org/swift-book/LanguageGuide/MemorySafety.html)

3.3 Keychain

3.3.1 Γενικές πληροφορίες

Το Keychain είναι ένα API διαχείρισης των κωδικών πρόσβασης, και ανήκει στην Apple. Εισάχθηκε με το macOS 8.6 το 1999 και η χρήση του συνεχίζεται από τότε. Το Keychain διαχειρίζεται τους κωδικούς πρόσβασης για ιστοσελίδες, FTP διακομιστές, SSH επικοινωνίες, κρυπτογράφηση δίσκων, δημιουργία ιδιωτικών κλειδιών και ψηφιακών πιστοποιητικών, κ.α..



Εικόνα 3: Το Keychain API
(https://developer.apple.com/documentation/security/keychain_services)

3.3.2 Ιστορική αναδρομή

Το Keychain ξεκίνησε ως βοηθητική υπηρεσία για το PowerTalk, το σύστημα ηλεκτρονικού ταχυδρομείου της Apple, όπου χρησιμοποιούταν για την διαχείριση των στοιχείων πρόσβασης στους διάφορους ταυτόχρονους λογαριασμούς που μπορεί να είχε ένας χρήστης. Όπως παρατηρείται και σε άλλα συστήματα, όπως το Gmail, υπάρχει η επιλογή χρήσης της πλατφόρμας για λογαριασμούς από άλλες εταιρείες (Yahoo, Outlook κτλ.) ή και πολλαπλούς λογαριασμούς της ίδιας εταιρείας. Φυσικά τότε δημιουργείται η ανάγκη απομνημόνευσης των διαφορετικών στοιχείων. Το Keychain ήταν το πρώτο στον χώρο του από πολλές παρόμοιες υπηρεσίες που εμφανίστηκαν μετέπειτα.

3.3.3 Ασφάλεια

Το συγκεκριμένο εργαλείο είναι προεγκατεστημένο τόσο στα iOS όσο και τα macOS, αν και πρόκειται για διαφορετικές εκδόσεις. Η έκδοση του iOS έχει λιγότερες λειτουργίες, και δεν επιτρέπεται ο διαμοιρασμός των κωδικών ανάμεσα σε διαφορετικές εφαρμογές. Αυτό ίσως αποτελεί εμπόδιο σε μερικές περιπτώσεις χρήσης, όμως ενισχύει την ασφάλεια του συστήματος. Γενικότερα ο τρόπος λειτουργίας του είναι ότι ο χρήστης ορίζει έναν κωδικό μέσω του οποίου προστατεύει όλους τους υπόλοιπους κωδικούς του. Έτσι αρκεί η απομνημόνευση ενός αρκετά ισχυρού κωδικού, για να διαχειριστούν αυτόματα όλοι οι υπόλοιποι. Αυτό είναι πολύ σημαντικό αφού δεν ενδείκνυται η χρήση του ίδιου κωδικού για πολλαπλές πλατφόρμες, και έτσι ο χρήστης μεν θυμάται μόνο τον κωδικό του keychain, αλλά εκείνο έχει αποθηκευμένους όλους τους υπόλοιπους κωδικούς. Οι κωδικοί αποθηκεύονται αφού κρυπτογραφηθούν με τον αλγόριθμο AES-256-GCM. Οι χρόνοι κρυπτογράφησης και αποκρυπτογράφησης, καθώς και αν η πληροφορία συγχρονίζεται με το iCloud διαφέρουν ανάλογα των τύπων δεδομένων. Η Apple διαθέτει ένα σχετικό εγχειρίδιο για την κατατόπιση των χρηστών του εργαλείου.

3.3.4 Κλείδωμα/Ξεκλείδωμα

Το Keychain δίνει την δυνατότητα να χρησιμοποιηθεί ο κωδικός της συσκευής και για την διαχείριση των κωδικών. Αυτό έχει ως πλεονέκτημα μια επιπλέον άνεση, με αρνητικό αντίκτυπο στο επίπεδο της ασφάλειας, οπότε αν και το πρώτο είναι η αρχική επιλογή, δίνεται και η δυνατότητα ορισμού ξεχωριστού κωδικού για το Keychain. Οι κενοί κωδικοί απαγορεύονται. Το κλείδωμα μπορεί να ρυθμιστεί είτε να γίνεται αυτόματα, π.χ. όταν υπάρχει περίοδος αδράνειας, ή να ενεργοποιηθεί κατά βούληση του χρήστη.

3.3.5 Συγχρονισμός κωδικών πρόσβασης

Αν χρησιμοποιηθεί η προαναφερθείσα δυνατότητα, και ο χρήστης μοιράζεται τον κωδικό της συσκευής με το Keychain, τότε ο κωδικός του εργαλείου ενημερώνεται αυτόματα με τον κωδικό χρήστη, αν και σημειώνονται προβλήματα αν η αλλαγή γίνει από συσκευή που δεν υποστηρίζεται από την Apple, ή η αλλαγή γίνει με κάποιο τρόπο πέρα της προεπιλεγμένης γραφικής διεπαφής για τον σκοπό.

3.4 Βιβλιοθήκες

3.4.1 Alamofire

Η Alamofire είναι μια βιβλιοθήκη της Swift, ειδικά σχεδιασμένη για HTTP δικτυακές επικοινωνίες στα λειτουργικά συστήματα της Apple macOS και iOS. Αποτελεί έναν εύχρηστο τρόπο επικοινωνίας μέσω HTTP αιτημάτων. Ως βάση χρησιμοποιεί το URL Loading σύστημα της Apple, το οποίο εμπεριέχεται στο Foundation framework. Η Alamofire βασίζεται σε API όπως το URLSession και URLSessionTask, προσφέροντας έναν πιο άνετο τρόπο χρήσης των εν λόγω τεχνολογιών, καθώς και πολλές επιπλέον λειτουργίες σχετικές με την ανάπτυξη δικτυακών εφαρμογών βασισμένων στο πρωτόκολλο HTTP. Η Alamofire έχει κατασκευαστεί εξ' ολοκλήρου με την γλώσσα Swift και περιέχει μόνο τις πιο μοντέρνες τεχνολογίες.

3.4.1.1 HTTP μέθοδοι

1. GET (ανάκτηση δεδομένων),
2. HEAD (παρόμοιο με το GET),
3. POST (αποστολή δεδομένων στον εξυπηρετητή, συνήθως ως παράμετρος τροποποίησης κάποιας οντότητας),
4. PUT (αντικατάσταση συγκεκριμένης οντότητας με κάποια άλλη),
5. DELETE (διαγραφή οντότητας/δεδομένων),
6. CONNECT (ορισμός τρόπου επικοινωνίας με τον διακομιστή),
7. OPTIONS (ρυθμίσεις σχετικά με τον τρόπο επικοινωνίας),
8. TRACE (παρακολούθηση της εξέλιξης της επικοινωνίας μέσω μηνυμάτων)
9. PATCH (εν μέρη τροποποίηση μιας οντότητας)

```
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 15: Οι HTTP μέθοδοι στην Alamofire

(https://developer.apple.com/documentation/swift/optional)

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

Κώδικας 16: Χρήση των HTTP μεθόδων

(https://developer.apple.com/documentation/swift/optional)

3.4.1.2 Παράμετροι αιτημάτων

Τα αιτήματα συχνά χρειάζεται να συνοδευτούν από περαιτέρω πληροφορίες.

```
struct Login:
  Encodable { let
  email: String
  let password: String
}
let login = Login(email: "test@test.test", password:
"testPassword") AF.request("https://httpbin.org/post",
     method:
     .post,
     parameters:
     login,
     encoder: JSONParameterEncoder.default).response {
  response in debugPrint(response)
}
```

Κώδικας 17: Παράμετροι HTTP αιτημάτων

(https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html.)

```
let parameters = ["foo": "bar"]

// All three of these calls are equivalent
AF.request("https://httpbin.org/get", parameters:
parameters)
// encoding defaults to `URLEncoding.default`
AF.request("https://httpbin.org/get", parameters:
parameters,
encoder: URLEncodedFormParameterEncoder.default)
```

```
AF.request("https://httpbin.org/get", parameters: parameters, encoder:
URLEncodedFormParameterEncoder(destination: .methodDependent))
// https://httpbin.org/get?foo=bar
```

Κώδικας 18: Ενσωμάτωση παραμέτρων σε URLs
(https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html.)

3.4.1.3 Επικεφαλίδες

Οι επικεφαλίδες επιτρέπουν την ανταλλαγή πληροφορίας σχετικά με το μήνυμα που αποστέλλεται.

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

Κώδικας 19: HTTP Επικεφαλίδες
(https://docs.swift.org/swift-book/LanguageGuide/CollectionTypes.html.)

3.4.1.4 Επιβεβαίωση ανταπόκρισης

Αφού γίνει κάποιο αίτημα από τον χρήστη, δέχεται πίσω μια απάντηση η οποία περιέχει τα δεδομένα που αιτήθηκε ή κάποιο σχετικό μήνυμα επιτυχίας ή λάθους. Η απάντηση μπορεί να είναι μια τυπική HTTP απάντηση ή να πάρει την μορφή δεδομένου JSON, Data, String, Decodable. Η Alamofire αρχικά προσφέρει απαντήσεις τύπου response, responseData, responseString και responseDecodable.

```
AF.request("https://httpbin.org/get").response { response in

    debugPrint("Response: \(response)")
}
```

```
struct DecodableType: Decodable { let url: String }


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

    debugPrint("Response: \(response)")
    }
```

```
AF.request("https://httpbin.org/get").responseString { response in

     debugPrint("Response: \(response)")

    }
```

```
AF.request("https://httpbin.org/get").responseData { response in

     debugPrint("Response: \(response)")
    }
```

Κώδικας 20: Alamofire response types (https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#statistical-metrics)

```
AF.request("https://httpbin.org/get")

    .validate(statusCode: 200..<300)

    .validate(contentType: ["application/json"])

    .responseData { response in

        switch response.result {

        case .success:

            print("Validation Successful")

        case let .failure(error):

            print(error)

        }

    }
```

Κώδικας 21: Εξατομικευμένη επιβεβαίωση ανταπόκρισης (https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#statistical-metric)

3.4.1.5 Ταυτοποίηση

Η αυθεντικοποίηση και ταυτοποίηση μπορεί να επιτευχθεί είτε μέσω των επικεφαλίδων ή των τύπων δεδομένων της Alamofire, URLCredential και URLAuthenticationChallenge.

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

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

Κώδικας 22: Ταυτοποίηση μέσω HTTP

(https://developer.apple.com/documentation/)

```
let user = "user"
let password = "password"
let credential = URLCredential(user: user, password: password, persistence:
.forSession)
AF.request("https://httpbin.org/basic-auth/\(user)/\(password)")
    .authenticate(with: credential)
    .responseDecodable(of: DecodableType.self) { response in
        debugPrint(response)
    }
```

Κώδικας 23: Ταυτοποίηση μέσω URLCredential

(https://developer.apple.com/documentation/)

```
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 24: Χειροκίνητη ταυτοποίηση

(https://developer.apple.com/documentation/)

3.4.1.6 Λήψη και αποστολή αρχείων

Πέρα της λήψης και αποστολής δεδομένων που απευθύνονται στην μνήμη που χειρίζονται τα προγράμματα του χρήστη και του εξυπηρετητή, δίνεται επίσης η δυνατότητα αποθήκευσης αρχείων στον δίσκο με τις μεθόδους Session.download, DownloadRequest και DownloadResponse.

Η Alamofire επίσης δίνει την επιλογή παρακολούθησης της διαδικασίας 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 25: Session.download

(https://www.codecademy.com/resources/blog/what-is-a-framework/)

```
let      destination      =      DownloadRequest.suggestedDownloadDestination(for:
.documentDirectory)
AF.download("https://httpbin.org/image/png", to: destination)
```

Κώδικας 26: DownloadRequest σε συνδυασμό με το Session.download

(https://www.codecademy.com/resources/blog/what-is-a-framework/)

```
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 27: downloadProgress

(https://www.codecademy.com/resources/blog/what-is-a-framework/)

```
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 28: uploadProgress

(https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#response-handling.)

3.4.1.7 Στατιστικά

Τα δεδομένα στατιστικού ενδιαφέροντος που προκύπτουν από κάθε επικοινωνία συγκεντρώνονται μέσω του URLSessionTaskMetrics για κάθε αίτημα.

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

Κώδικας 29: Alamofire Statistics
(https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#downloading-data-to-a-file)

3.4.1.8 cURL

Η Alamofire δίνει την δυνατότητα μετατροπής των αιτημάτων στις αντίστοιχες cURL εντολές ώστε να παρέχει περισσότερες πληροφορίες σχετικά με το κάθε αίτημα και την εξέλιξή του.

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

Κώδικας 30: Μετατροπή αιτήματος σε cURL εντολή
(https://en.wikipedia.org/wiki/Application_framework)

```
$ 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 31: Αποτέλεσμα αντίστοιχης cURL εντολής
(https://en.wikipedia.org/wiki/Application_framework)

3.5 Ανακεφαλαίωση/ρόλοι τεχνολογιών στην εφαρμογή

Η εφαρμογή είναι γραμμένη εξ' ολοκλήρου σε Swift, με εξαίρεση μερικές σειρές εντολών σε Ruby που χρειάστηκαν για την συγκρότηση του project και την εγκατάσταση των απαραίτητων εργαλείων με το μέσο εγκατάστασης CocoaPods. Η Swift είναι μια μοντέρνα γλώσσα υψηλού επιπέδου με πολλές βελτιστοποιήσεις, και δεν κάνει συμβιβασμούς προς την λειτουργικότητα ή την ευκολία χρήσης της.

Ως περιβάλλον ανάπτυξης της εφαρμογής επιλέχθηκε το XCode, το native περιβάλλον ανάπτυξης λογισμικού που απευθύνεται σε Apple συσκευές. Το XCode παρέχει ένα πλήρη, ασφαλή και εν μέρη αυτοματοποιημένο γραφικό τρόπο ανάπτυξης, με δυνατότητες προσομοίωσης της εφαρμογής σε όλες τις συσκευές και τα λειτουργικά συστήματα που έχουν κυκλοφορήσει και υποστηρίζονται ακόμα από την εταιρεία.

Ακόμα ως μέρος της εφαρμογής ενσωματώθηκε ένας αριθμός εργαλείων. Χρησιμοποιείται το OAuth σε συνεργασία με το Aboard API του τμήματος για την εξακρίβωση των στοιχείων του χρήστη και την τροφοδότηση με ανακοινώσεις αντίστοιχα. Επίσης, γίνεται χρήση του Keychain για την απομνημόνευση των στοιχείων, και της βιβλιοθήκης Alamofire για την επικοινωνία, μέσω της οποίας υλοποιείται η διαχείριση όλων των αιτημάτων από και προς την εφαρμογή.

Κεφάλαιο 4ο: Ανάπτυξη εφαρμογής

4.1 IEEApps

Στον ριζικό φάκελο της εφαρμογής βρίσκονται όλα τα αρχεία που αφορούν άμεσα την διεπαφή χρήστη (UI, ViewCells).

4.1.1 SceneDelegate

Με την κλάση SceneDelegate εκφράζεται το σκηνικό που απεικονίζεται για τον χρήστη, ανάλογα με το αν έχει προβεί σε είσοδο με τα στοιχεία του ή βρίσκεται στην αρχική σελίδα της εφαρμογής. Υπάρχει ένα flag που αποθηκεύτε μέσα στο keychain το "FORCE_RELOAD_PRIVATE_ANNOUNCEMENTS" που αν ο χρήστης έκανε login τη τελευταία φορά που χρησιμοποιείσαι την εφαρμογή τότε θα τον κράτηση login αλλιώς θα των παραπέμψει στην αρχική σελίδα.

```
class SceneDelegate: UIResponder,
    UIWindowSceneDelegate { let dependencyContainer =
    AppDependencyContainer() var window: UIWindow?
    let keychain = KeychainSwift()

    func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options
connectionOptions: UIScene.ConnectionOptions) {
        guard let windowScene = (scene as? UIWindowScene) else { return
        } let window = UIWindow(windowScene: windowScene)

        let mainVC = dependencyContainer.makeMainViewController()
        self.window = window
        window.frame =
        UIScreen.main.bounds let loggedIn
        =
keychain.get(DataReloadEnum.FORCE_RELOAD_PRIVATE_ANNOUNCEMENTS.rawVal
        ue)    if loggedIn == "true" {
            let storyBoard : UIStoryboard = UIStoryboard(name: "Main",
            bundle:nil) let viewcontroller =
            storyBoard.instantiateViewController(withIdentifier:
"PrivateAnnouncementsVC") as! PrivateAnnouncementsVC
            let navigationController = UINavigationController(rootViewController:
viewcontroller)
            navigationController.modalPresentationStyle = .fullScreen
            window.rootViewController = navigationController
        } else {
            window.rootViewController  =  mainVC
        }

        window.makeKeyAndVisible()
    }

    func scene(_ scene: UIScene, openURLContexts URLContexts:
        Set<UIOpenURLContext>) { if let urlContext = URLContexts.first {
            let url = urlContext.url
            if let deepLink = DeepLink(url: url) {
                dependencyContainer.deepLinkHandler.handleDeepLinkIfPossible(deepLink:
deepLink)
            }
        }
    }
```

Κώδικας 32: Η κλάση SceneDelegate

4.2 Api_router

Στον φάκελο Api_router βρίσκονται όσες κλάσεις είναι υπεύθυνες για τον συγχρονισμό του Apps API με την εφαρμογή. Εδώ δηλαδή δημιουργούνται όλα τα request μέσο του Alamofire (get η post αναλόγως) .

Στον Api_router υπάρχει ένα enum με διάφορες περιπτώσεις όπου στη κάθε περίπτωση θα χρησιμοποιήσει αναλόγως headers , parameters , request body και endpoints .

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

4.2.1 APIRouter

```
    enum APIRouter: URLRequestConvertible {
    case getPublicAnnouncments(page:Int)
    case getLoginAnnouncments(page:Int)
    case  getNotifications(page:Int)
    case getToken(code: String)
    case refreshToken(refreshToken: String)
    case getUser
    case getTags
    case getSubscriptions
    case postSubscripe(id:[Int])
    case getNotAnn(id:Int)
…
```

Κώδικας 33: APIRouter Use Cases

```
    var baseURL: String? {
    switch self {
        case .getPublicAnnouncments:
            return "https://aboard.iee.ihu.gr//api"
        case .getLoginAnnouncments:
            return "https://aboard.iee.ihu.gr//api"
        case .getNotifications:
            return "https://aboard.iee.ihu.gr//api/auth/user"
        case .getUser:
            return "https://aboard.iee.ihu.gr//api/auth"
        case .getToken,
             .refreshToken:
            return "https://login.iee.ihu.gr"
        case .getTags:
            return "https://aboard.iee.ihu.gr//api"
        case .getSubscriptions:
            return "https://aboard.iee.ihu.gr//api/auth"
        case .postSubscripe:
            return "https://aboard.iee.ihu.gr//api/auth"
        case .getNotAnn:
            return "https://aboard.iee.ihu.gr//api"
    }
}
```

Κώδικας 34: APIRouter baseURLs

```
var path: String {
    switch self {
        case .getPublicAnnouncments:
            return "/announcements"
        case .getLoginAnnouncments:
            return "/announcements"
        case .getNotifications:
            return "/notifications"
        case .getToken,
            .refreshToken:
            return "/token"
        case .getUser:
            return "/user"
        case .getTags:
            return "/tags"
        case .getSubscriptions:
            return "/subscriptions"
        case .postSubscripe:
            return "/subscripe"
        case .getNotAnn(let id):
            return "/announcements/\(id)"
    }
}
```

Κώδικας 35: APIRouter paths

4.2.2 GetPublicAnnModel

Εδώ συγκεντρώνονται, μέσω αιτήματος προς το API, όλες οι δημοσιευμένες ανακοινώσεις.

```
extension MainViewController{
    func fetchPublicAnn(){
        let headers :HTTPHeaders=[.contentType("application/json")]
        let request =
AF.request("https://aboard.iee.ihu.gr//api/announcements",headers: headers)
            request.responseDecodable(of: PublicAnns.self) { (response) in
                guard let publicAnns = response.value else { return }
                print(publicAnns.all[0].tags)
            }
    }
}
```

Κώδικας 36: Η συνάρτηση fetchPublicAnn

4.3 DataContext

Στο φάκελο DataContext συγκεντρώνονται όλα τα αρχεία που χρησιμοποιούνται για την ένωση του Api_router μαζί με το app για παράδειγμα το Configuration , DataContext+ClientRequest που αναλύονται παρακάτω .

4.3.1 Configuration

Ορίζονται τα tokens ώστε να αποθηκεύονται τα δεδομένα του χρήστη για όλες τις συνεδρίες.

```
Struct Configuration {

    static let REMEMBER_TOKEN = "remember_token"
    static let REMEMBER_REFRESH_TOKEN = "remember_refresh_token"
}
```

Κώδικας 37: Configuration Struct

4.3.2 DataContext+ClientRequests

Στο αρχείο ClientRequest του φακέλου Api_router δημιουργείται η συνάρτηση που διαχειρίζεται το response των αιτημάτων και τα μοντελοποιεί. Στο αρχείο DataContext+ClientRequest γίνεται ένωση του προηγούμενου με τα απαραίτητα στοιχεία ώστε να μπορεί να γίνει χρήση του από τα ViewControllers.

Με κάθε ανανέωση της σελίδας, φορτώνονται τα δεδομένα ανάλογα με τις αποθηκευμένες προτιμήσεις και τα cookies του χρήστη.

```
Func refreshToken(completion: @escaping (Bool) -> Void) { guard
    let refreshToken =
keychain.get(Configuration.REMEMBER_REFRESH_TOKEN) else {
        completion(false)
        return
    }
    ClientRequests.refreshToken(refreshToken: refreshToken) {[weak self]
(authModel) in
        guard let authModel = authModel else { completion(false)
            return
        }
        self?.keychain.set("\(authModel.access_token)", forKey:
Configuration.REMEMBER_TOKEN)
        self?.keychain.set("\(authModel.refresh_token)", forKey:
Configuration.REMEMBER_REFRESH_TOKEN)
        self?.refreshToken = authModel.refresh_token
        completion(true)
    }
}
```

Κώδικας 38: Η συνάρτηση refreshToken

4.4 Dependencies

4.4.1 AppDependenciesContainer

Στο AppDependenciesContainer δημιουργούμε το Url που θα ανοίξει με το που θα πατήσει το κουμπί "ΣΥΝΔΕΣΗ" ο χρήστης για να γίνει αναγνώριση και να συνδεθεί στην εφαρμογή ακολουθώντας το μοντέλο αναγνώρισης oAuth.

```
class AppDependencyContainer {
    let deepLinkHandler = DeepLinkHandler()

    func makeMainViewController() -> UIViewController {
        let redirectUri = URL(string: "https://github.com/stefosAEL/IEEApp")!
        let oAuthConfig = OAuthConfig(authorizationUrl: URL(string:
"https://login.iee.ihu.gr/authorization/")!,
                                      clientId: "62408ef084b2a60fc0ba856c",
                                      redirectUri: redirectUri,
                                      responseType: "code",

scope:["announcements","profile","notifications","refresh_token","edit_notificati
ons"],
                                      clientSecret:
"4mtxqivi27efteqcmkgzc7v7ex97o8ak4qjggack3jo07lfzaq")

        let oAuthClient = RemoteOAuthClient(config: oAuthConfig, httpClient:
HTTPClient())
        let oAuthService = OAuthService(oauthClient: oAuthClient)

        let storyBoard : UIStoryboard = UIStoryboard(name: "Main", bundle:nil)
        let loginVC = storyBoard.instantiateViewController(withIdentifier:
"MainStoryboardID") as! MainViewController
        loginVC.oAuthService = oAuthService
        loginVC.makeHomeViewController = makeHomeViewController
        let navigationController = UINavigationController(rootViewController:
loginVC)
        return navigationController


    }

    func makeHomeViewController() -> UITabBarController {
        let tabBarController = UITabBarController()
        let repoVC = UIViewController()
        let profileVC = UIViewController()
        tabBarController.viewControllers = [repoVC, profileVC].map {
UINavigationController(rootViewController: $0)}

        return tabBarController
    }
}
```

Κώδικας 39: Οι συναρτήσεις του AppDependenciesContainer

4.5 Infrastructure

4.5.1 DeepLinkHandler

Όταν ο χρήστης επιλέγει να διαβάσει τον πηγαίο κώδικα, τους όρους χρήσης ή την πολιτική απορρήτου, ανακατευθύνεται από την εφαρμογή στην σελίδα της στο GitHub, stefosAEL/IEEApp (github.com). Αυτό επιτυγχάνεται με την χρήση DeepLink, που επιτρέπει την επίσκεψη σε κάποια άλλη τοποθεσία εκτός της εφαρμογής, και την επιστροφή στην εφαρμογή από το ίδιο περιβάλλον. Γίνεται δηλαδή δυνατή η εναλλαγή και η συνεργασία διαφορετικών εφαρμογών (μιας και το GitHub πρόκειται για δικτυακή εφαρμογή).

```
let scheme = "https://github.com/stefosAEL"
enum DeepLink: Hashable {
    case oAuth(URL)

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

        let deepLinkMap: [String: (URL) -> DeepLink] = [
            "\(scheme)/IEEApp": 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 40: Github DeepLink

4.5.2 RemoteOAuthClient

Για 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
    let scope: [String]
    let clientSecret: String
}
```

Κώδικας 41: Δομή τύπου OAuthConfig

Η εφαρμογή συγκεντρώνει τα απαραίτητα στοιχεία σε μια δομή τύπου OAuthConfig και τα αποστέλλει ως αίτημα στην υπηρεσία για την έγκρισή τους.

```
private let config: OAuthConfig
private let httpClient: HTTPClient

init(config: OAuthConfig, httpClient: HTTPClient) {
    self.config = config
    self.httpClient = httpClient
}

func getAuthPageUrl(state clientId : String) -> URL? {
    let params = AuthParms(client_id: config.clientId,
                           redirect_uri:
config.redirectUri.absoluteString,response_type: config.responseType,
                           scope: config.scope.joined(separator: ","))
    let httpRequest = HttpRequest(baseUrl: config.authorizationUrl,
                                  method: .get,
                                  params: params,
                                  headers: [:])
    return httpRequest.asURLRequest()?.url
}
```

Κώδικας 42: Χρήση της υπηρεσίας RemoteOAuth

```
protocol OAuthClient {
    func getAuthPageUrl(state: String) -> URL?
}

class OAuthService {
    enum OAthError: Error {
        case malformedLink
        case exchangeFailed
    }
    private let oauthClient: OAuthClient
    private var state: String?

    init(oauthClient: OAuthClient) {
        self.oauthClient = oauthClient
   }

    func getAuthPageUrl(state: String = UUID().uuidString) -> URL? {
        self.state = state
        return oauthClient.getAuthPageUrl(state: state)
    }
}
```

Κώδικας 43: Συνέχεια της υλοποίησης OAuth ταυτοποίησης, από τον φάκελο Domain

4.6 ViewControllers

4.6.1 MainViewController

Στο MainViewController χτίζεται η λογική βάση της οποίας λειτουργεί η εφαρμογή με την εκκίνησή της. Μέσω αυτού καλούνται τα controllers αφορούν τις δημόσιες ανακοινώσεις, δηλαδή το περιεχόμενο της σελίδας πριν την σύνδεση σε λογαριασμό, καθώς φυσικά και το ViewController που αφορά την σελίδα εισόδου.

4.6.2 PrivateAnnouncementsVC

Εδώ χειρίζονται όλες οι ανακοινώσεις που έχουν κοινοποιηθεί συγκεκριμένα στον φοιτητή, δηλαδή τις ανακοινώσεις στις οποίες έχει πρόσβαση εφόσον ολοκληρωθεί επιτυχής ταυτοποίηση του χρήστη. Πρόκειται για την κύρια σελίδα μετά την σύνδεση σε λογαριασμό, απ' όπου παρέχεται πρόσβαση σε όλες τις υπόλοιπες σελίδες της εφαρμογής.

4.6.3 LoggInWebViewVC

Χειρίζεται την σελίδα από την οποία, σε συνδυασμό με το OAuth API γίνεται η αυθεντικοποίηση των στοιχείων και η είσοδος του χρήστη στην εφαρμογή. Μέσω του ViewController παίρνουμε το code από το Url αν κάνει επιτυχής σύνδεση ο χρήστης και το στέλνουμε στο request για να πάρουμε το access token και στη συνέχεια να το χρησιμοποιήσουμε για τα αιτήματα προς το API .

4.6.4 SettingViewController

Χειρίζεται ότι έχει να κάνει σχέση με τα Settings της εφαρμογής και όπως επίσης όλα τα links για της πληροφορίες που χρειάζεται να ξέρει ο χρήστης για παράδειγμα που μπορούν να δουν τους όρους χρήσεις και το απόρρητο .Παρόλα αυτά μια από τις πιο σημαντικές του ειδικότητες είναι οι παρακολούθηση πινάκων που με πατώντας στην ετικέτα του εμφανίζετε ο πίνακας με τα μαθήματα που θέλει ο χρήστης να παίρνει ειδοποιήσεις όταν μπει μια καινούρια ανακοίνωση.

4.6.5 TagsViewController

Χειρίζεται των πίνακα μαθημάτων που θέλει ο χρήστης να παίρνει ειδοποιήσεις . Φορτώνει μέσω αιτήματος όλα τα μαθήματα , εξάμηνα κλπ. Μαζί με τα μαθήματα υπάρχει ένα checkbox που όταν πατηθεί γεμίζει η αδειάζει ανάλογα ένα πίνακα με id ο οποίος στέλνετε μέσω μεθόδου σε post request με το που θα πατήσει το κουμπί "Αποθήκευση".

4.7 Model

Εδώ ορίζονται με αντικειμενοστραφή τρόπο οι δομές για το μοντέλο ταυτοποίησης, τα μοντέλα των αρχικών και των δημοσιευμένων προς τους φοιτητές ανακοινώσεων, το μοντέλο που ακολουθούν οι ειδοποιήσεις, οι εγγραφές σε αυτές μέσω επιλογής ετικετών, καθώς και τα μοντέλα για τις ίδιες τις ετικέτες και τους χρήστες. Αφορά δηλαδή τα βασικά χαρακτηριστικά όλων των οντοτήτων της εφαρμογής.

```
struct User: Codable {
    let subscriptions: [Subscriptions]?
    let name: String
    let email: String
    let uid: String
}
struct Subscriptions: Codable {
    let title: String
}
```

```
struct Users: Codable {
  let data: User
}
```

Κώδικας 44: Μοντέλα User, Users

```
struct Tag: Codable {
  let data: [TagsArray]
}
```

```
struct TagsArray: Codable {
    let title : String
    var id : Int
    let isSelected : Bool?
}
```

Κώδικας 45: Μοντέλα Tag, Tags

```
struct PublicAnn: Codable {
    let  author:  Teacher
    let title: String
    let body: String
    let created_at: String
    let tags: [Tags]
    let id: Int
}

struct Teacher: Codable {
    let name: String
}

struct Tags:Codable {
    let title: String
}
```

```
struct PublicAnns: Codable {
  let data: [PublicAnn]
  let meta : Meta?
}

struct Meta:Codable{
    let last_page:Int
}
```

Κώδικας 46: Μοντέλα PublicAnn, PublicAnns

```
struct Notification: Codable {
    let data : Datas?
    let created_at:String
}

struct Datas:Codable{
    let type: String
    let user: String
    let id : Int
}
```

```
struct Notifications: Codable {
  let data: [Notification]
  let meta : Metas?
}

struct Metas:Codable{
    let last_page:Int
}
```

Κώδικας 47: Μοντέλα Notification, Notifications

```
struct LogginAnns: Codable {
    let data: [PublicAnn]
    let meta : Meta?
}

struct Meta:Codable{
    let last_page:Int
}
```

```
struct NotAnn: Codable {
  let data: PublicAnn
}
```

Κώδικας 48: Μοντέλα LogginAnns, NotAnn

```
struct AuthModel: Codable {
    let access_token: String
    let user: String
    let refresh_token: String
    let error:Error?

    struct Error:Codable{
        let code:Int?
    }
}
```

```
enum PostRequest : URLRequestConvertible
{
    func asURLRequest() throws -> URLRequest {
        <#code#>
    }
}
```

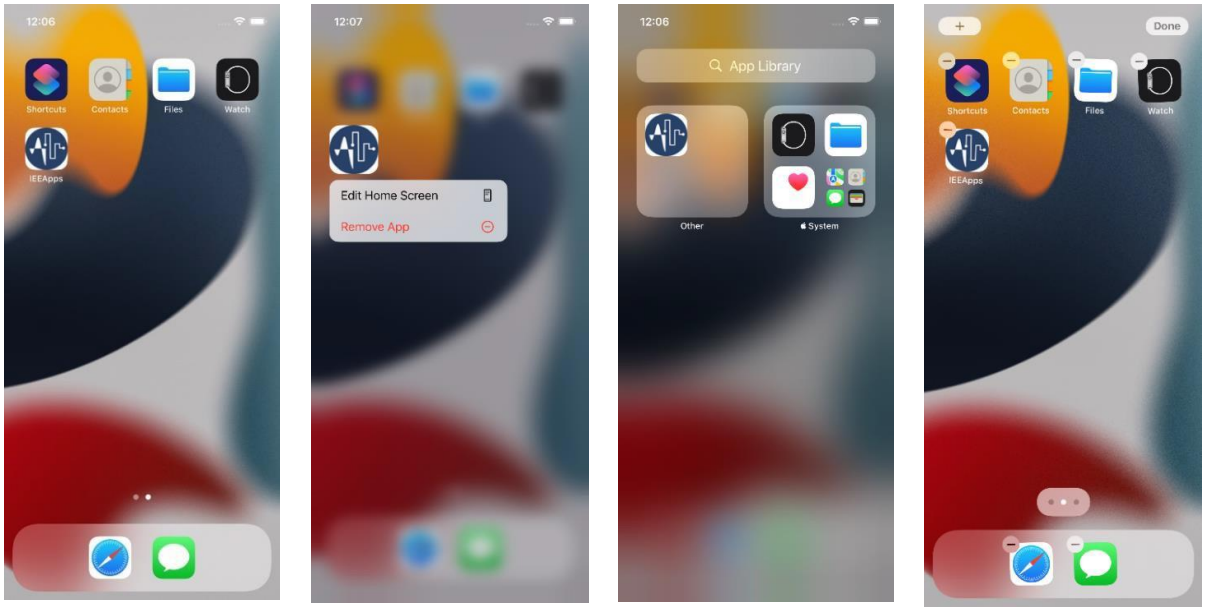
```
struct Subscription: Codable {
    let pivot : Pivot?

    struct Pivot: Codable {
        let tagID: Int

        enum CodingKeys: String, CodingKey {
            case tagID = "tag_id"
        }
    }
}
```

Κώδικας 49: Μοντέλα AuthModel, PostRequest, Subscription

Κεφάλαιο 5ο: Παρουσίαση λειτουργίας εφαρμογής

5.1 Συμπεριφορά στο περιβάλλον του συστήματος

Η εφαρμογή αρχικά εντοπίζεται ως εικονίδιο στην οθόνη της φορητής συσκευής iPhone, iPad ή iPod του χρήστη. Οι μόνες δυνατότητες που δίνονται είναι η μετακίνηση του εικονιδίου και η απεγκατάσταση της εφαρμογής.

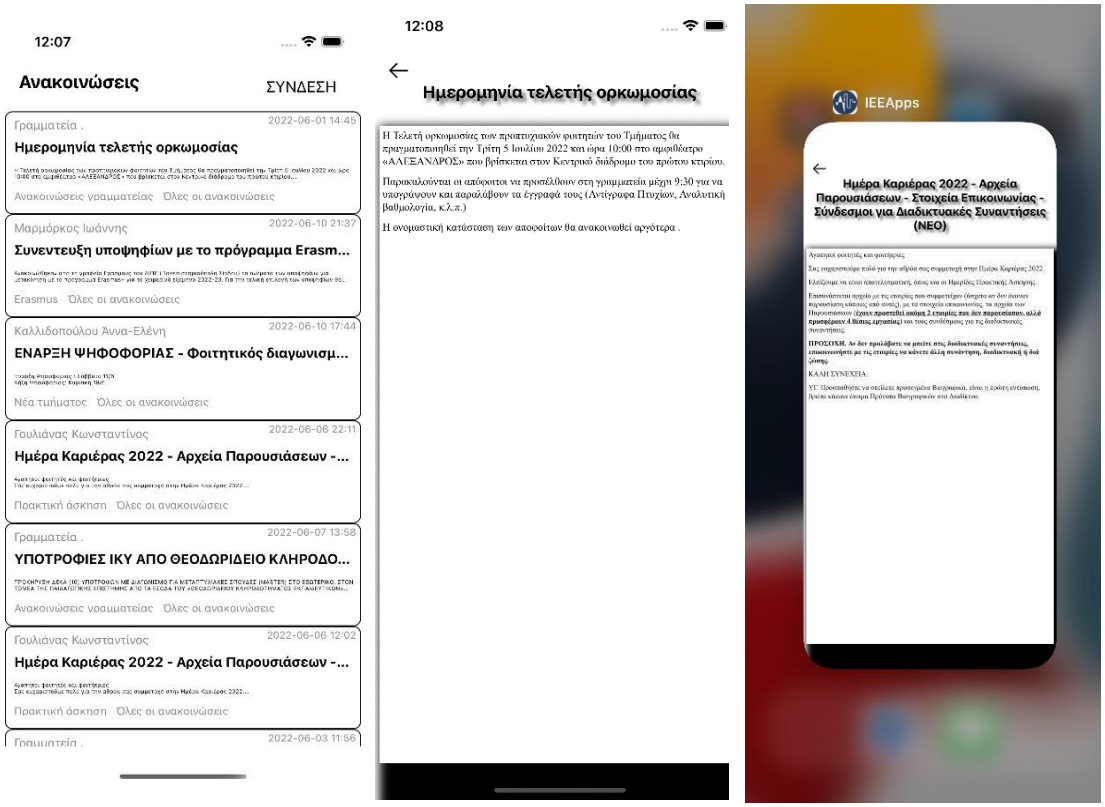


Εικόνα 4: Εικόνες από iPhone 13 Pro

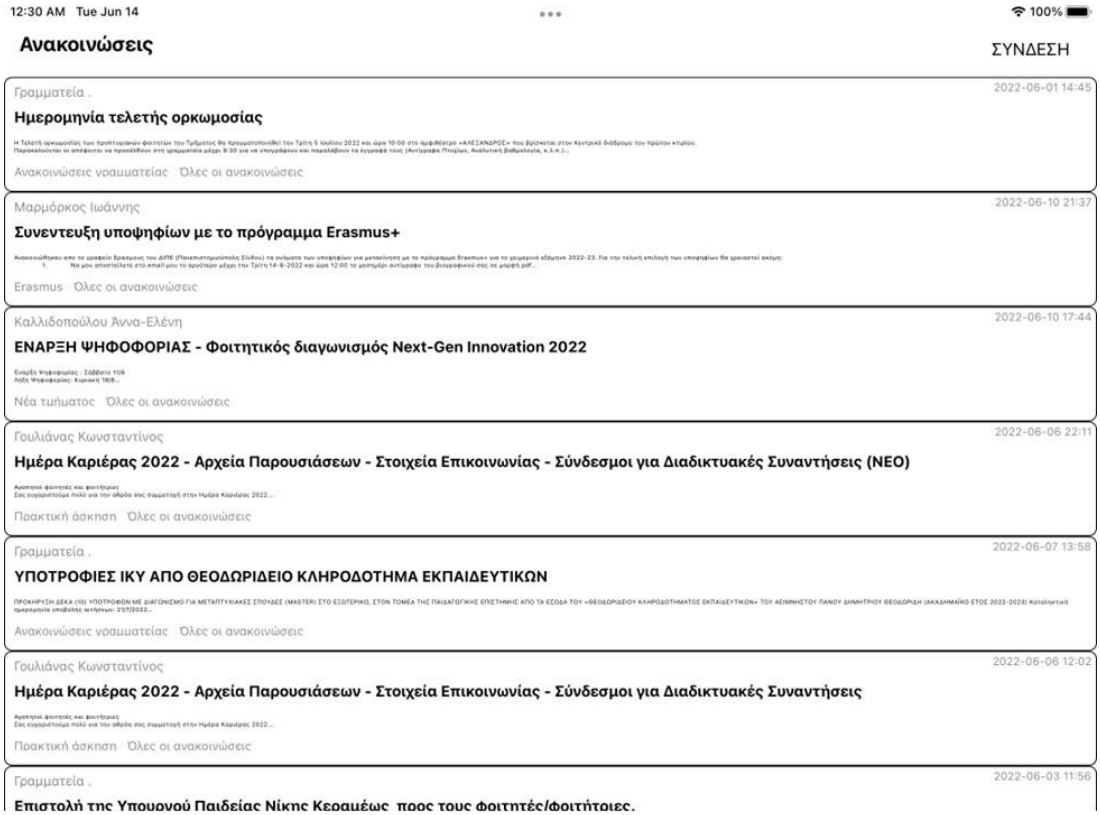
Η εφαρμογή επίσης ενημερώνει τον χρήστη με push notifications όποτε αναρτηθεί νέα ενημέρωση με κάποια από τις ετικέτες που έχει δηλώσει στο εσωτερικό της εφαρμογής, μέσω των ρυθμίσεων.

5.2 Αρχική σελίδα

Με την είσοδο στην εφαρμογή, υπάρχει άμεση πρόσβαση σε όλες τις δημόσιες ανακοινώσεις του τμήματος, δηλαδή όσες ανακοινώσεις αφορούν το τμήμα γενικότερα ή, ενώ αφορούν συγκεκριμένο μάθημα, οι υπεύθυνοι του μαθήματος αποφάσισαν να τις κοινοποιήσουν ευρέως. Η την επιλογή μιας συγκεκριμένης ανακοίνωσης, ανακοίνωση απομονώνεται και επεκτείνεται το περιεχόμενό της σε όλη την έκταση της οθόνης. Η ανακοίνωση παραμένει ορατή και με την ελαχιστοποίησή της. Αν ο χρήστης χρειάζεται πρόσβαση στις ανακοινώσεις που τον αφορούν προσωπικά, καλείται να συνδεθεί μέσω του κουμπιού «ΣΥΝΔΕΣΗ» στην πάνω δεξιά γωνία της εφαρμογής, ώστε να μεταβεί στην σελίδα εισόδου.



Εικόνα 5: Αρχική σελίδα, iPhone 13 Pro

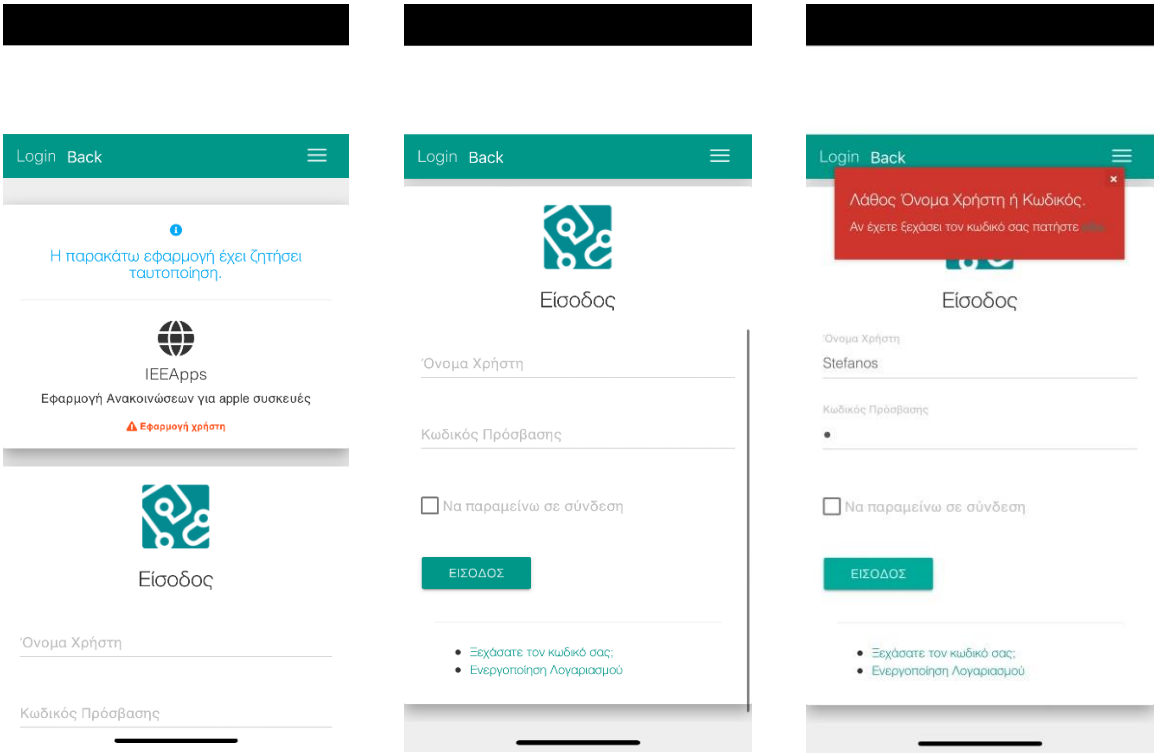


Εικόνα 6: Αρχική σελίδα, iPad 9th gen, οριζόντια προβολή

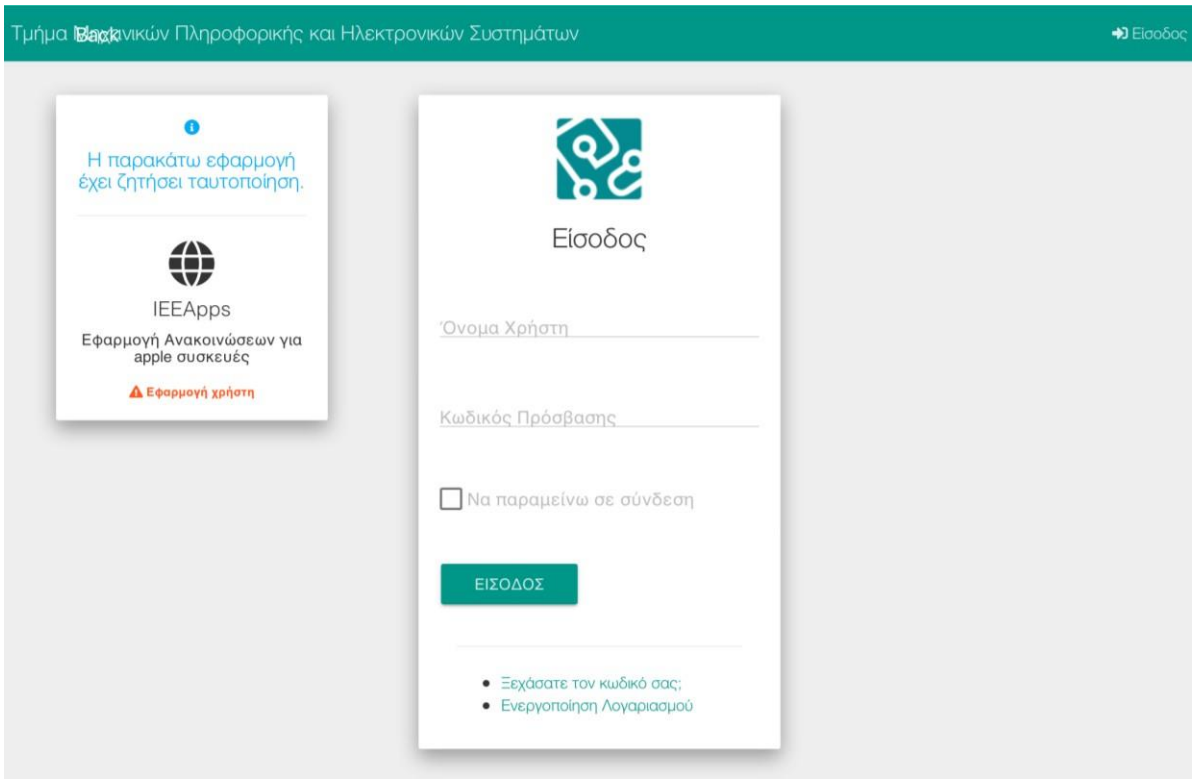
5.3 Σελίδα εισόδου

5.3.1 Ταυτοποίηση

Με την είσοδο στην σελίδα της εισόδου, η πιο εμφανής επιλογή προς τον χρήστη είναι η εισαγωγή των στοιχείων του (όνομα χρήστη, κωδικός πρόσβασης) ώστε να πραγματοποιηθεί η ταυτοποίηση και να αποκτήσει πρόσβαση στην εξατομικευμένη για εκείνον έκδοση της εφαρμογής. Σε περίπτωση εισαγωγής λάθος στοιχείων ή αν ο χρήστης επιθυμεί να ενεργοποιήσει τον ακαδημαϊκό λογαριασμό του, περαιτέρω βοήθεια είναι διαθέσιμη μέσω της δικτυακής εφαρμογής του τμήματος Apps, πάνω στην οποία βασίστηκε και η εφαρμογή που αναπτύχθηκε για την εργασία.



Εικόνα 7: Σελίδα εισόδου, iPhone 13 Pro



Εικόνα 8: Σελίδα εισόδου, iPad 9th gen, οριζόντια προβολή

5.3.2 Λοιπές λειτουργίες

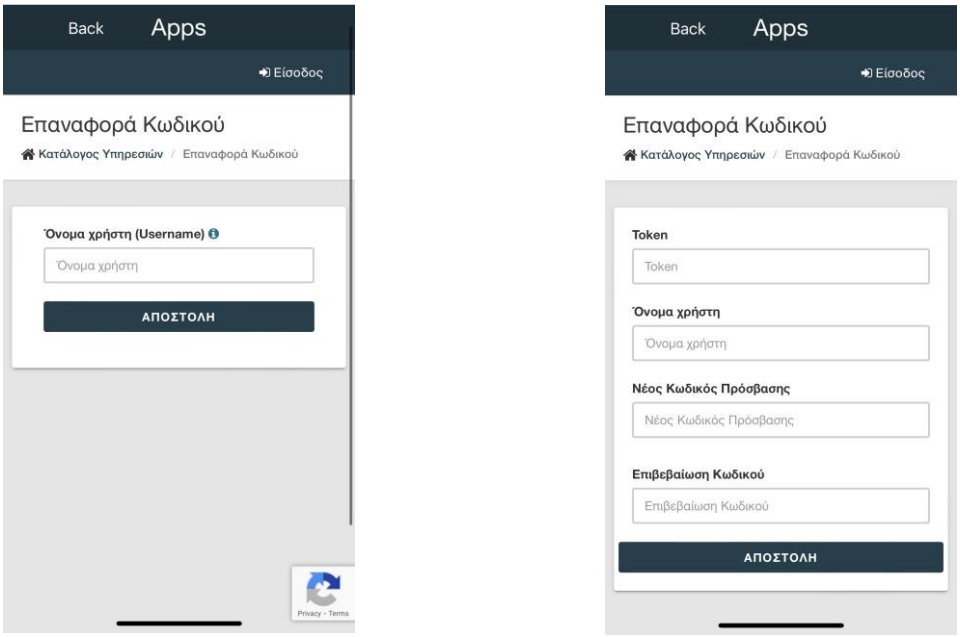
Οι υπηρεσίες που προσφέρονται από την δικτυακή εφαρμογή Apps όσον αφορά τα στοιχεία του χρήστη είναι οι εξής:

- Επαναφορά κωδικού

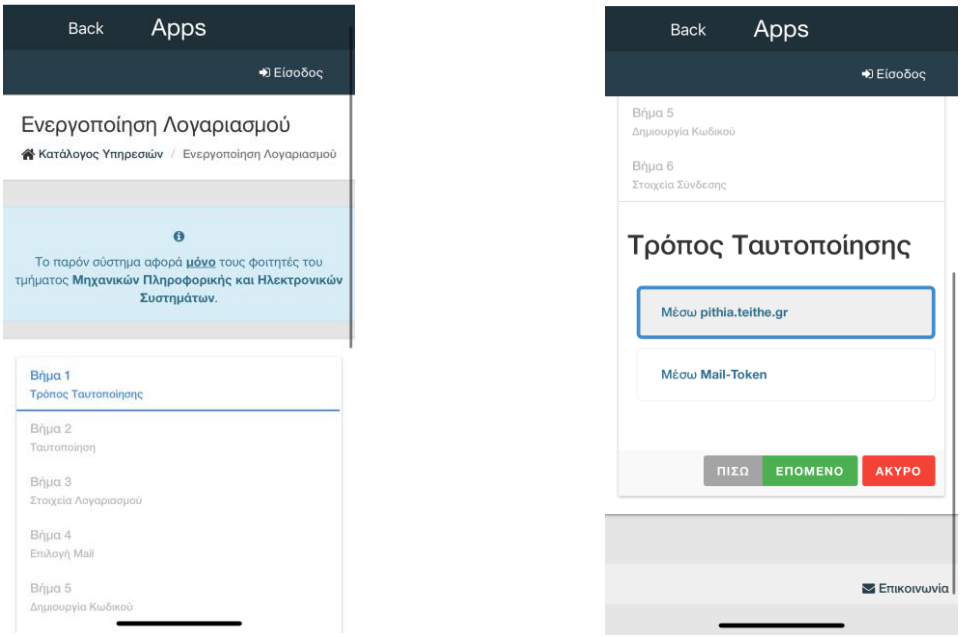
  Για να αποκαταστήσει τον κωδικό πρόσβασης ο χρήστης καλείται να συμπληρώσει το όνομα που χρησιμοποιεί στην εφαρμογή Apps, ώστε να του αποσταλεί ένα ειδικό token εξουσιοδότησης στην διεύθυνση ηλεκτρονικού ταχυδρομείου που έχει δηλώσει. Έπειτα πρέπει να συμπληρώσει μια νέα φόρμα με το token, το ίδιο όνομα χρήστη και τον νέο κωδικό που επιθυμεί να χρησιμοποιήσει.

- Ενεργοποίηση λογαριασμού

  Πραγματοποιείται ταυτοποίηση μέσω του ακαδημαϊκού λογαριασμού του χρήστη, κι έπειτα συμπληρώνονται τα στοιχεία του νέου χρήστη ώστε να κατοχυρωθεί στην βάση δεδομένων του Apps.



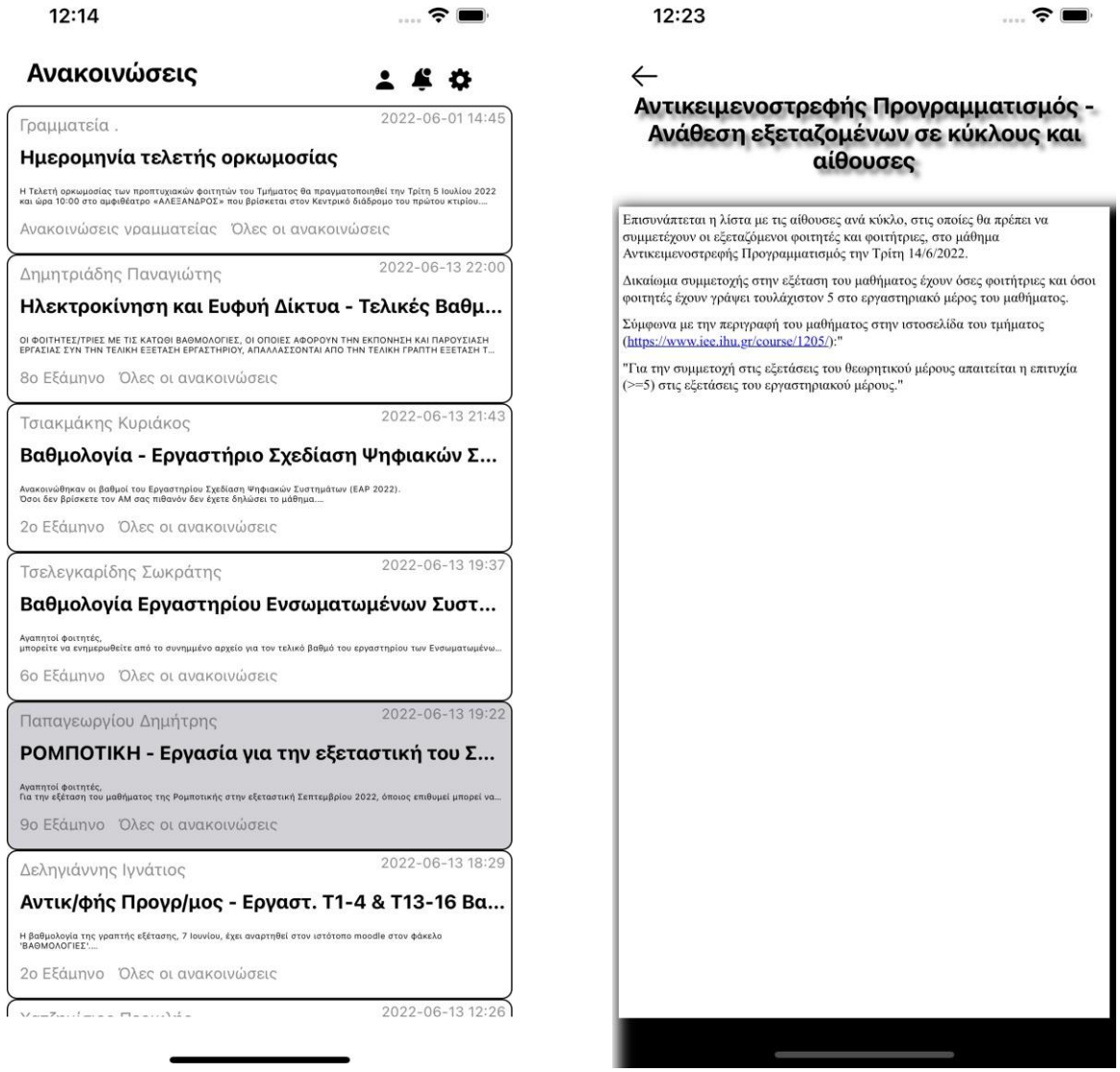
Εικόνα 9: Επαναφορά κωδικού, iPhone 13 pro



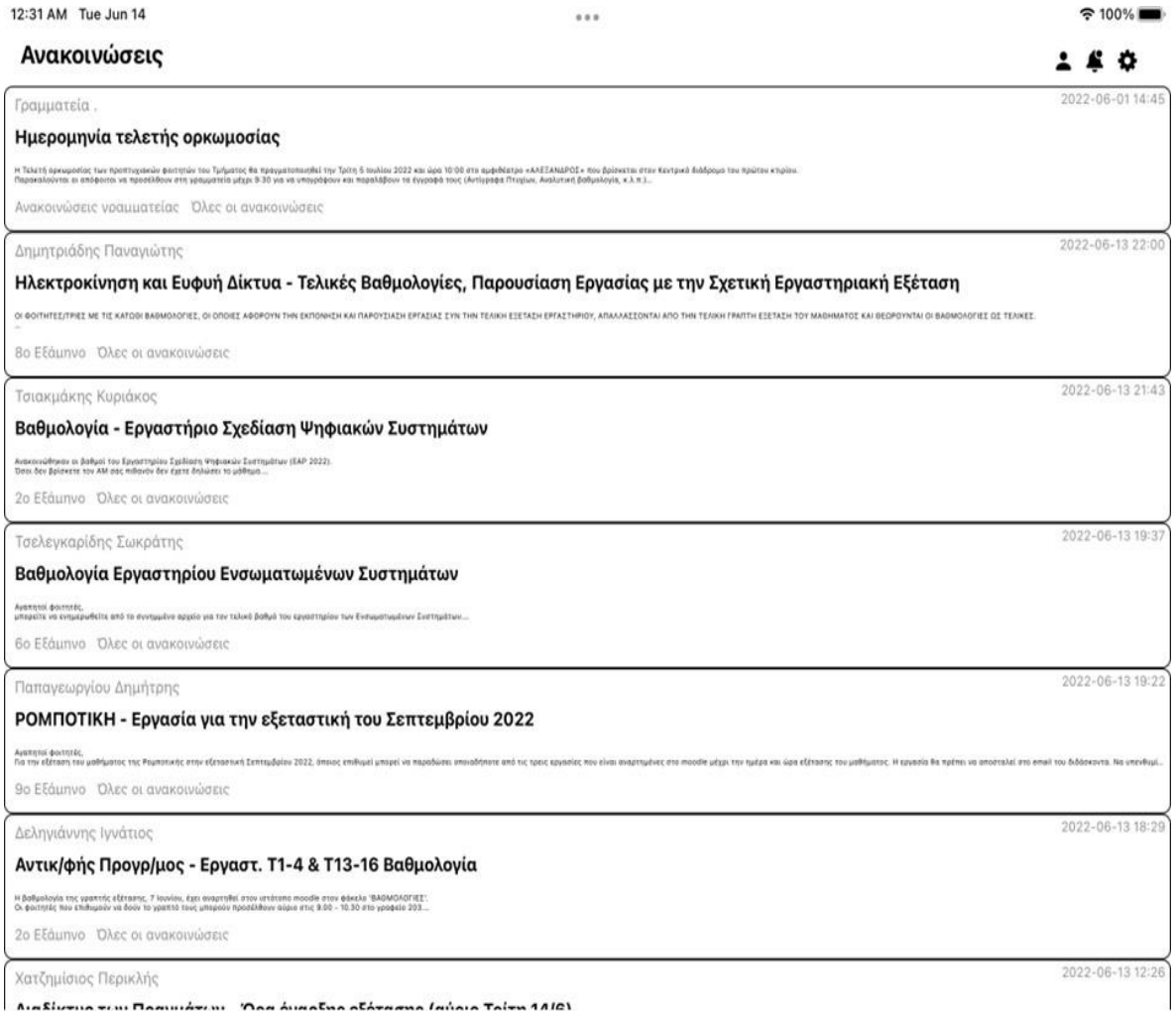
Εικόνα 10: Ενεργοποίηση λογαριασμού, iPhone 13 Pro

5.4 Ανακοινώσεις/Main Interface

Με την είσοδο στην εφαρμογή, ο χρήστης τοποθετείται στην κύρια σελίδα της εφαρμογής, απ' όπου του παρέχεται πρόσβαση σε όλες τις υπόλοιπες λειτουργίες του προγράμματος. Στην κύρια σελίδα, ή αλλιώς σελίδα ανακοινώσεων, υπάρχει μια λίστα με ανακοινώσεις παρόμοια με αυτή στην αρχική σελίδα της εφαρμογής. Σε αντίθεση με την αρχική σελίδα, εδώ είναι διαθέσιμες και όλες οι ανακοινώσεις που απευθύνονται σε συγκεκριμένα μαθήματα ή εξάμηνα. Όπως και προηγουμένως, αν επιλεχθεί κάποια ειδοποίηση, απομονώνεται ώστε να γίνει πλήρως ορατό το περιεχόμενό της.



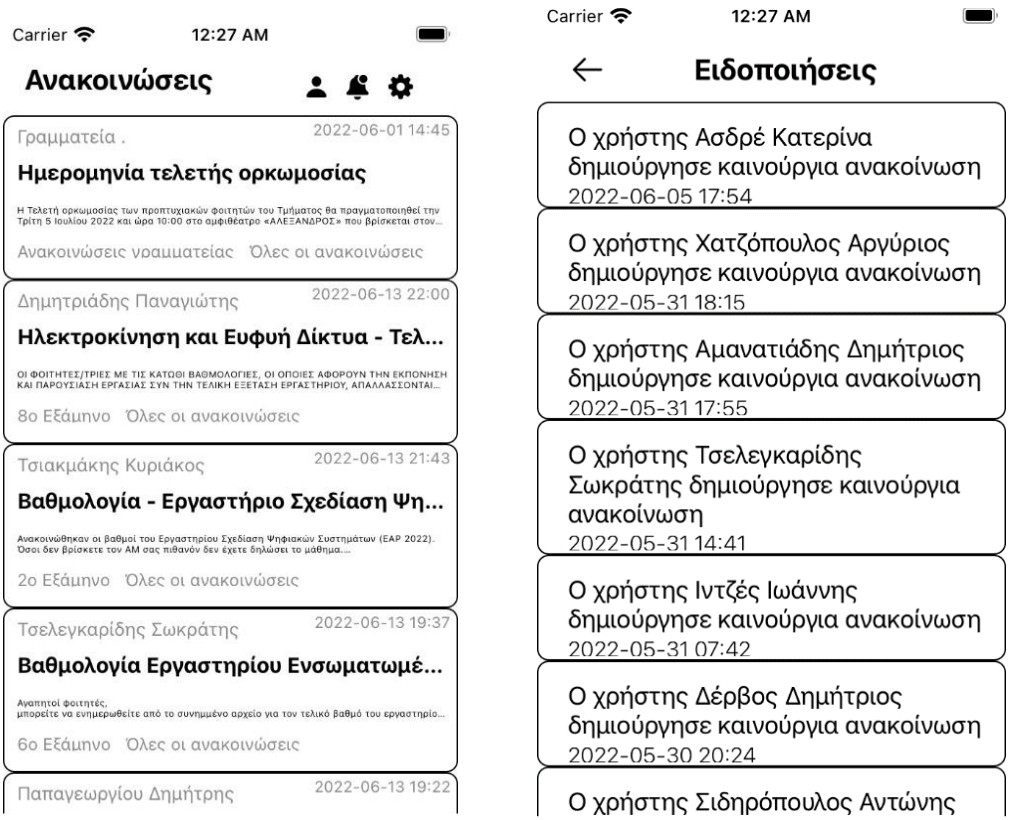
Εικόνα 11: Εικόνες από iphone 13 pro



Εικόνα 12: Εικόνα από ipad 9th gen, οριζόντια προβολή

5.5 Ειδοποιήσεις

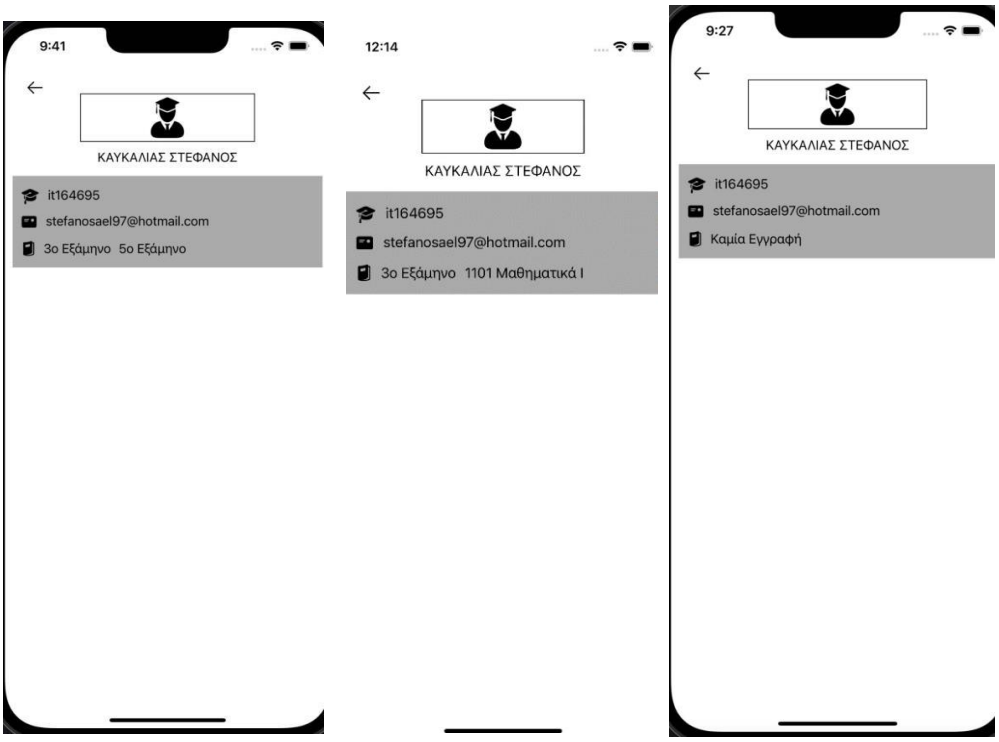
Με το πάτημα στο κουμπί με το καμπανάκι, εμφανίζεται μια λίστα με τις ειδοποιήσεις που έχει λάβει ο χρήστης, σύμφωνα με τις προτιμήσεις του. Η αλλαγή κάποιας προτίμησης δεν επηρεάζει τις ήδη υπάρχουσες ειδοποιήσεις, αλλά ρυθμίζει για ποιες ανακοινώσεις θέλει να ενημερώνεται στο μέλλον. Με την επιλογή του βέλους πάνω αριστερά, ο χρήστης επιστρέφει στην κύρια σελίδα της εφαρμογής όπου βρίσκονται οι ανακοινώσεις.



Εικόνα 13: Εικόνες από ipod touch 7th gen

5.6 Προφίλ χρήστη

Επιλέγοντας το εικονίδιο με το ανθρωπάκι, δίπλα στην καμπάνα των ειδοποιήσεων, γίνεται μετάβαση στο προφίλ του χρήστη όπου αναγράφονται το όνομα, το όνομα χρήστη, το δηλωμένο email και τα μαθήματα για τα οποία λαμβάνει ειδοποιήσεις.



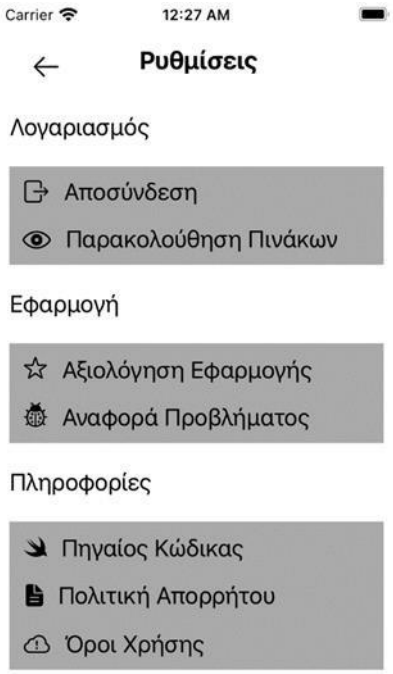
Εικόνα 14: Στιγμιότυπα του προφίλ με διαφορετικές δηλωμένες ετικέτες, iPhone 13 Pro

5.7 Ρυθμίσεις

5.7.1 Το μενού

Μέσω του κουμπιού με το σχήμα γραναζιού, ο χρήστης αποκτά πρόσβαση στις ρυθμίσεις.

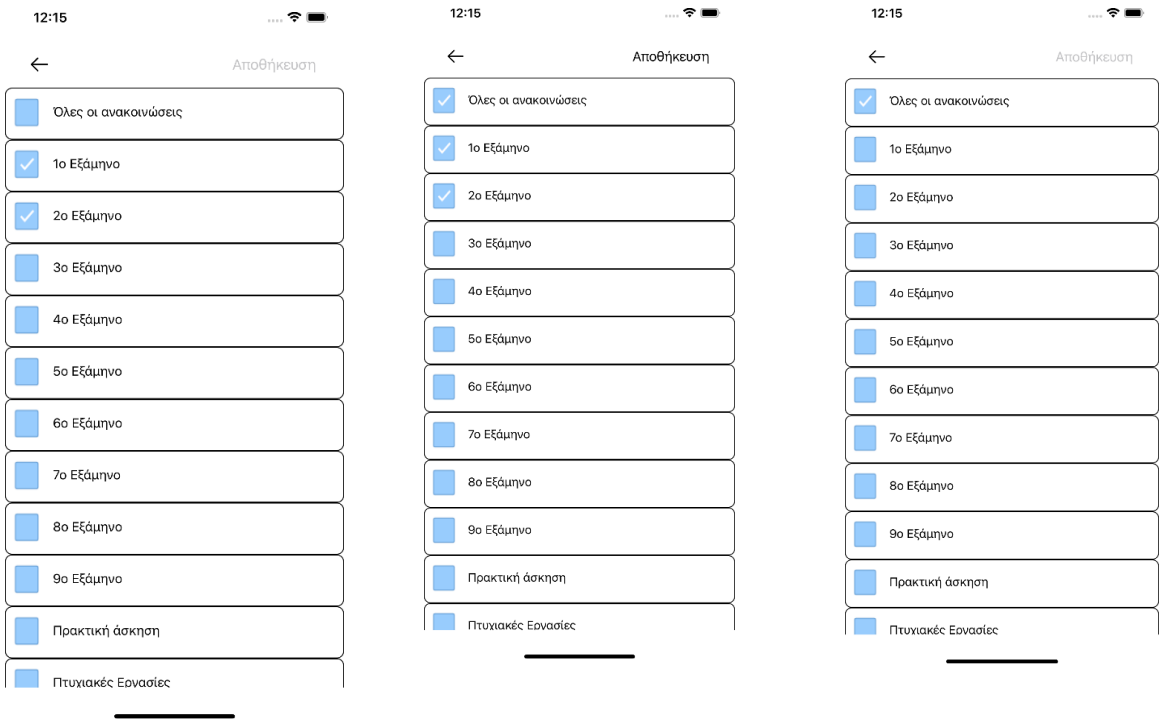
Στις ρυθμίσεις ο χρήστης μπορεί να αποσυνδεθεί, να τροποποιήσει για ποιες ανακοινώσεις θα λαμβάνει ενημερώσεις, να αξιολογήσει την εφαρμογή ή να αναφέρει κάποιο τεχνικό πρόβλημα (bug), όπως επίσης και να διαβάσει σχετικά χρήσιμα έγγραφα όπως η πολιτική απορρήτου και οι όροι χρήσης. Υπάρχει ακόμα σύνδεσμος που ανακατευθύνει στον πηγαίο κώδικα της εφαρμογής.



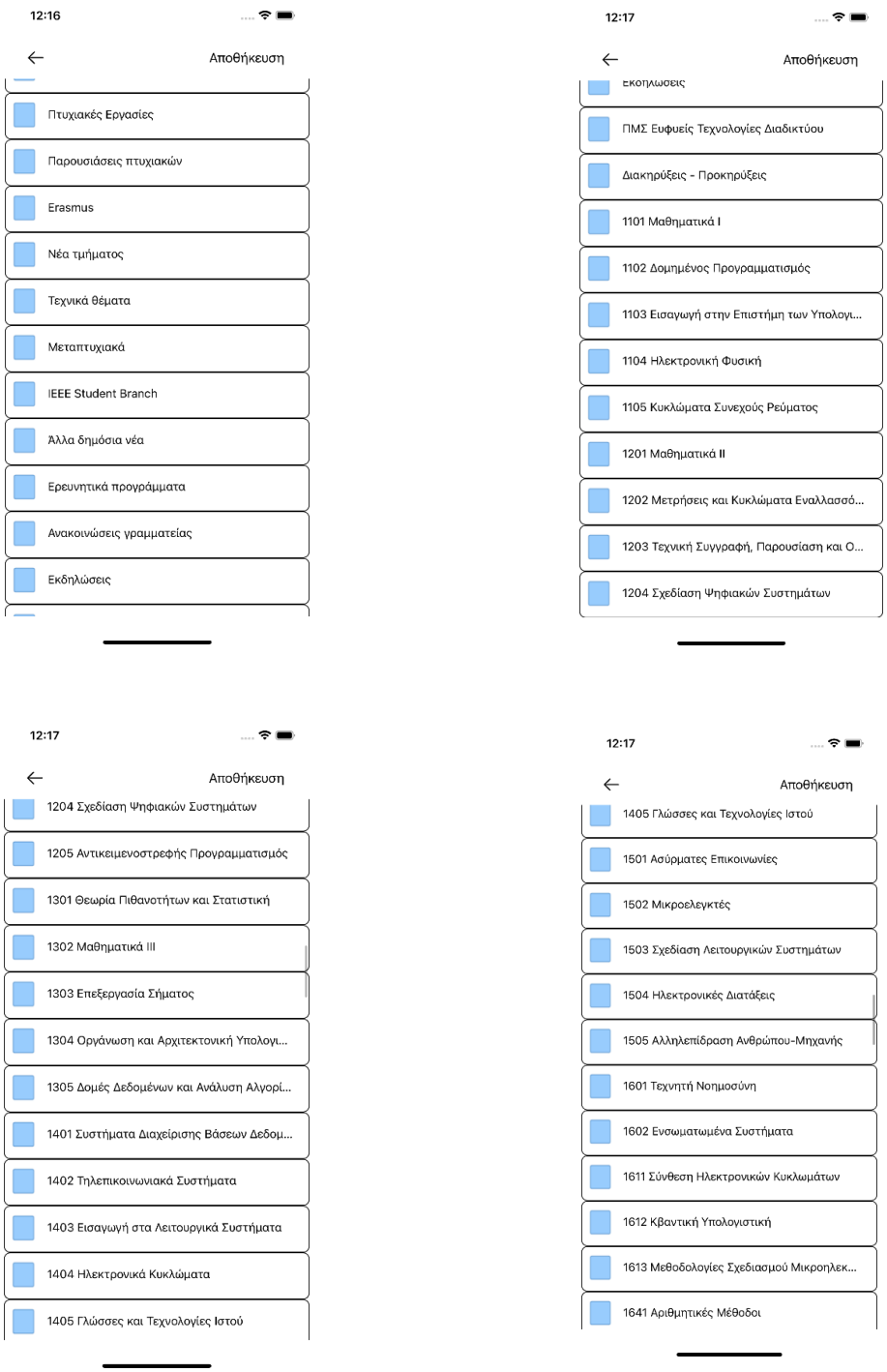
Εικόνα 15: Ρυθμίσεις, iPod Touch 7th gen

5.7.2 Παρακολούθηση πινάκων

Εδώ ο χρήστης μπορεί να επιλέξει τις ετικέτες των ανακοινώσεων για τις οποίες θα ενημερώνεται. Μεταξύ των επιλογών, υπάρχουν τα διαφορετικά μαθήματα, εξάμηνα, θέματα γενικού ενδιαφέροντος, καθώς και η επιλογή να λαμβάνει ειδοποιήσεις για όλες τις ανακοινώσεις που αναρτώνται. Κάθε φορά ενημερώνονται και οι αντίστοιχες πληροφορίες στο προφίλ του χρήστη.



Εικόνα 16: Μερικοί δυνατοί συνδυασμοί ετικετών, iPhone 13 Pro



Εικόνα 17: Όλες οι διαθέσιμες ετικέτες των ενημερώσεων, iPhone 13 Pro

5.8 Αξιολόγηση Εφαρμογής & Αναφορά Προβλήματος

Η αξιολόγηση της εφαρμογής και η αναφορά προβλήματος πρόκειται για λειτουργίες άχρηστες στο παρόν στάδιο, αφού η εφαρμογή δεν έχει κυκλοφορήσει. Ως εκ τούτου, προς το παρόν δεν γίνεται τίποτα με την επιλογή τους. Με την κυκλοφορία της εφαρμογής, η αξιολόγηση θα παραπέμπει στον αντίστοιχο χώρο του AppStore και η αναφορά προβλήματος θα χρησιμοποιείται ώστε οι χρήστες να μπορούν να γνωστοποιήσουν τα τυχόν προβλήματα που αντιμετωπίζουν, για να διορθώνονται στις επόμενες εκδόσεις της εφαρμογής.

5.9 Πληροφορίες

Οι πληροφορίες διακρίνονται σε 3 ενότητες, και φιλοξενούνται στην χώρο της εφαρμογής στο GitHub, όπου και παραπέμπουν οι σύνδεσμοι.

1. Πηγαίος κώδικας

   Παραπέμπει στον ιστότοπο όπου φιλοξενείται ο πηγαίος κώδικας της εφαρμογής. Καθώς η εφαρμογή πρόκειται για open source project, άλλοι προγραμματιστές μπορούν τόσο να πάρουν έμπνευση από τον πηγαίο κώδικα, όσο και να προτείνουν βελτιώσεις ή παραλλαγές. Επιπλέον, με την διάθεση του πηγαίου κώδικα στο κοινό υπάρχει πλήρης διαφάνεια όσον αφορά την λειτουργία της εφαρμογής, την διαχείριση των προσωπικών δεδομένων κλπ..

2. Πολιτική απορρήτου

   Πρόκειται για δήλωση του προγραμματιστή σχετικά με την χρήση των δεδομένων που τροφοδοτούν στο πρόγραμμα οι χρήστες (στοιχεία πρόσβασης, ετικέτες ειδοποιήσεων, στατιστικά χρήσης, cookies).

3. Όροι χρήσης

   Ως όροι χρήσης προσδιορίζονται οι προϋποθέσεις υπό τις οποίες διανέμεται η εφαρμογή, και ορίζεται η θέση του προγραμματιστή απέναντι σε τυχόν νομικά θέματα που μπορεί να προκύψουν.

Κεφάλαιο 6ο: Επίλογος/Συμπεράσματα

Συνοψίζοντας, η εφαρμογή αναμένεται να χρησιμεύσει σε πολλούς φοιτητές του τμήματος. Όπως αναφέρθηκε και στην αρχή της εργασίας, ήδη κυκλοφορεί η αντίστοιχη εφαρμογή σε Android η οποία έχει εξυπηρετήσει πάνω από 1000 χρήστες.

Η εφαρμογή παρέχει όλες τις απαραίτητες βασικές λειτουργίες. Οι φοιτητές έχουν άμεση πρόσβαση στις δημόσιες ανακοινώσεις χωρίς να είναι απαραίτητη κάποια ενέργεια, ενώ με την σύνδεσή τους στο σύστημα τους κοινοποιούνται και οι εφαρμογές που αφορούν συγκεκριμένους τομείς και έχουν ως κοινό αποκλειστικά τους σπουδαστές του τμήματος. Ο χρήστης ενημερώνεται με push ειδοποίηση κάθε φορά που γίνεται ανάρτηση νέας ανακοίνωσης, και μπορεί να ρυθμίσει τις ειδοποιήσεις που θα λαμβάνει σύμφωνα με τις αντίστοιχες ετικέτες. Δεν αντιμετωπίστηκε κάποιο ιδιαίτερο πρόβλημα κατά την ανάπτυξή της, καθώς η Swift πρόκειται για 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. Προσωπικά μου έδωσε το κίνητρο για να μάθω μια νέα, δημοφιλή και εξελισσόμενη γλώσσα προγραμματισμού που θα είναι αδιαμφισβήτητα χρήσιμο εφόδιο στην σταδιοδρομία μου μετά την σχολή.

Επιπλέον, με γοήτευσε η δυνατότητα που δίνεται να ανέβει μελλοντικά η εφαρμογή στο app store, αλλά και να βελτιωθεί, ενσωματώνοντας νέα χαρακτηριστικά όπως για παράδειγμα αναζήτησης, φιλτράρισμα, ανάρτηση ανακοινώσεων εάν ο χρήστης είναι καθηγητής, τα οποία θα την εξελίξουν περαιτέρω βελτιστοποιώντας το υπάρχον εργαλείο και διευκολύνοντας τους φοιτητές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] «Welcome to Swift.org,» [Ηλεκτρονικό]. Available: https://www.swift.org/.

[2] «Releases,» [Ηλεκτρονικό]. Available: https://developer.apple.com/news/releases/.

[3] «Featured,» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/.

[4] «Themes - iOS Human Interface Guidelines,» [Ηλεκτρονικό]. Available: https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/.

[5] «iOS,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/IOS.

[6] «The History of iOS and its Evolution,» [Ηλεκτρονικό]. Available: https://www.mobileappdaily.com/history-of-ios.

[7] «Application Framework,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Application_framework.

[8] «What Is a Framework?,» [Ηλεκτρονικό]. Available: https://www.codecademy.com/resources/blog/what-is-a-framework/.

[9] «API,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/API.

[10] «What is an API?,» [Ηλεκτρονικό]. Available: https://www.codecademy.com/resources/blog/what-to-know-about-apis/.

[11] «Keychain Services,» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/security/keychain_services.

[12] «Keychain (software),» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Keychain_(software).

[13] «Collection Types,» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/LanguageGuide/CollectionTypes.html.

[14] «Optionals,» [Ηλεκτρονικό]. Available: https://developer.apple.com/documentation/swift/optional.

[15] «Classes and Structures,» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/LanguageGuide/ClassesAndStructures.html.

[16] «Control Flow,» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/LanguageGuide/ControlFlow.html.

[17] «Error Handling,» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/LanguageGuide/ErrorHandling.html.

[18] «Functions,» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/LanguageGuide/Functions.html.

[19] «Generics,» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/LanguageGuide/Functions.html.

[20] «Memory Safety,» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/LanguageGuide/MemorySafety.html.

[21] «Swift Basics,» [Ηλεκτρονικό]. Available: https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html.

[22] «cURL Command Output,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#curl-command-output.

[23] «Statistical Metrics,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#statist ical-metrics.

[24] «Downloading Data to a File,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#downl oading-data-to-a-file.

[25] «Authentication,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#authe ntication.

[26] «Uploading Data to a Server,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#uploa ding-data-to-a-server.

[27] «Response Validation,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#respo nse-validation.

[28] «Response Handling,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#respo nse-handling.

[29] «HTTP Headers,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#http- headers.

[30] «Request Parameters,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#reque st-parameters-and-parameter-encoders.

[31] «HTTP Methods,» [Ηλεκτρονικό]. Available: https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#http-methods.