ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εφαρμογή διαχείρισης χρόνου για συσκευές λογισμικού IOS

Διπλωματική Εργασία

ΚΑΤΣΙΑΝΑΣ ΑΛΕΞΑΝΔΡΟΣ

Επιβλέπων: ΓΕΩΡΓΙΟΣ ΣΤΑΜΟΥΛΗΣ

Ιανουάριος 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εφαρμογή διαχείρισης χρόνου για συσκευές λογισμικού IOS

Διπλωματική Εργασία

ΚΑΤΣΙΑΝΑΣ ΑΛΕΞΑΝΔΡΟΣ

Επιβλέπων: ΓΕΩΡΓΙΟΣ ΣΤΑΜΟΥΛΗΣ

Ιανουάριος 2023

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Time management application for IOS devices

Diploma Thesis

KATSIANAS ALEXANDROS

Supervisor: GEORGE STAMOULIS

January 2023

Περίληψη

Η εργασία αυτή έχει ως θέμα τη μελέτη και δημιουργία της εφαρμογής TODOLIST για κινητές συσκευές IOS. Στην εργασία μελετώνται οι native εφαρμογές τόσο για android όσο και για IOS. Επίσης αναλύεται το λειτουργικό σύστημα Macintosh και το περιβάλλον ανάπτυξης κώδικα (Xcode), οι τεχνολογίες βάσης δεδομένων σε περιβάλλον IOS, η σύγκριση τεχνολογιών βάσης δεδομένων για τις κινητές εφαρμογές IOS και άλλες παρόμοιες εφαρμογές toDoList σε περιβάλλον IOS. Ακόμη, μελετάται η αρχιτεκτονική του συστήματος στις κινητές συσκευές IOS, τα επίπεδα της αρχιτεκτονικής και ο κύκλος ζωής ενός UIView Controller. Σε επίπεδο υλοποίησης της εφαρμογής, η εργασία παρουσιάζει τη δημιουργία της εφαρμογής, την εμφάνιση των δεδομένων μέσα σε μία λίστα (UITableView), την προσθήκη των δεδομένων στη λίστα της εφαρμογής, τη διαγραφή και τροποποίηση των δεδομένων μέσα στη λίστα, καθώς και την αποθήκευση των δεδομένων στην τοπική βάση (CoreData).

Λέξεις – Κλειδιά: *εφαρμογή, δημιουργία, κινητές συσκευές, IOS*

Abstract

This thesis is about the study and creation of the application TODOLIST for mobile devices IOS. In this thesis, native applications for both android and IOS are studied. It also analyzes the Macintosh operating system and code development environment (Xcode), database technologies in IOS environment, comparison of database technologies for IOS mobile applications and other similar toDoList applications in IOS environment is also analyzed. Furthermore, the system architecture on mobile IOS devices, the layers of the architecture and the life cycle of a UIView Controller are studied. At the application implementation level, the paper presents the creation of the application, the display of data within a list (UITableView), the addition of data to the application list, the deletion and modification of data within the list, and the storage of data in the local database (CoreData).

Keywords: *app, creation, creation, mobile devices, IOS*

Πίνακας περιεχομένων

Κατάλογος Πινάκων

1. Εισαγωγή

Η ανάπτυξη κινητών συστημάτων και εφαρμογών απαιτεί κατάλληλες γλώσσες προγραμματισμού και εργαλεία ανάπτυξης. Αρχικά, η Objective-C ήταν η κύρια γλώσσα για το iOS. Τα τελευταία χρόνια, η Objective-C αποδείχθηκε ότι έπαψε να ανταποκρίνεται στις σύγχρονες εξελίξεις.

Γι' αυτό η Apple έχει αρχίσει να αναπτύσσει τη νέα γλώσσα προγραμματισμού Swift. Πρόκειται για μια μεταγλωττισμένη γλώσσα προγραμματισμού για εφαρμογές iOS, watchOS, tvOS, macOS και Linux. Δημιουργήθηκε για πιο άνετη και ασφαλέστερη δημιουργία κώδικα, αλλά και ευκολότερη εκμάθηση προγραμματισμού. Παρουσιάστηκε για πρώτη φορά στο Παγκόσμιο Συνέδριο Προγραμματιστών της Apple το 2014 (Brunner, Blöchlinger, Toffetti, Spillner & Bohnert, 2015).

Η γλώσσα έχει υποστεί δυναμική ανάπτυξη, και αυτό έχει κατά καιρούς οδηγήσει σε προβλήματα. Η Swift φέρνει απλούστερη σύνταξη, συντομότερο κώδικα και ασφαλέστερα προγράμματα. Η γλώσσα δεν είναι μια αμιγώς αντικειμενοστραφής γλώσσα αλλά είναι υβριδική. Η Apple αναφέρει ότι η Swift είναι 2,6 φορές ταχύτερη από την Objective-C και 8,4 φορές ταχύτερη από την Python. Η Swift, μαζί με την Kotlin, είναι μία από τις ταχύτερα αναπτυσσόμενες γλώσσες προγραμματισμού (Danielsson, 2016).

Η Swift είναι μια υβριδική γλώσσα προγραμματισμού που χρησιμοποιεί αντικειμενοστραφή, διαδικαστικά και ορισμένα λειτουργικά χαρακτηριστικά. Επί του παρόντος, φαίνεται να είναι η καταλληλότερη γλώσσα για την ανάπτυξη εγγενών εφαρμογών και συστημάτων για κινητά τηλέφωνα.

Παρά την ταχεία ανάπτυξη και τα περιστασιακά προβλήματα συμβατότητας προς τα πίσω. Η εμπειρία και οι μετρήσεις δείχνουν ότι η Swift είναι μια γρήγορη και ασφαλής γλώσσα. Ο κώδικας Swift είναι έως και ένα τρίτο μικρότερος από την Objective-C, γεγονός που οδηγεί σε πιο γρήγορη, σαφέστερη αλλά και ασφαλέστερη δημιουργία κώδικα (Charland & Leroux, 2011).

Πειράματα με φοιτητές δείχνουν ότι η Swift είναι πιο εύκολη στην ανάγνωση για τους φοιτητές από την Objective-C ή τη C++. Λόγω της σύντομης ύπαρξης της Swift, υπάρχουν, βέβαια, προβλήματα με την ταχεία ανάπτυξη και τις αλλαγές της γλώσσας, λιγότερη υποστήριξη και πληθώρα βιβλιοθηκών, πλαισίων και περιβαλλόντων ανάπτυξης.

Η Swift δεν φέρνει νέες προσεγγίσεις προγραμματισμού, αλλά οι δημιουργοί της προσπάθησαν να την ενσωματώσουν και να συνδυάσουν τα καλύτερα χαρακτηριστικά που μπορούμε να έχουμε σε ένα συγκεκριμένο εύρος σε άλλες γλώσσες προγραμματισμού, για παράδειγμα, Closures (JavaScript), Generics (Java), Tuples (Python), Operator Overloading (C++), Optional types (Haskell), Protocols (Java) κ.ά (Kratzke & Peinl, 2016).

Η εμπειρία δείχνει ότι η γλώσσα προγραμματισμού Swift είναι κατάλληλη για τη διδασκαλία του προγραμματισμού. Η απλότητα και η δομή της γλώσσας προγραμματισμού συμβάλλουν στη γρήγορη κατανόηση της γλώσσας.

2. Θεωρητικό Μέρος

2.1. Native Εφαρμογή και Cross Platform

Οι τεχνολογικές τάσεις συνεχίζουν να εξελίσσονται και να αλλάζουν με την πάροδο του χρόνου. Μέσω της χρήσης των κινητών τηλεφώνων και των υπολογιστών, η τεχνολογία έχει αλλάξει σημαντικά τον πλανήτη. Έχουν μεταμορφώσει σημαντικά τον σύγχρονο πολιτισμό. Η ζωή έχει επηρεαστεί τρομερά από τις τεχνολογικές εξελίξεις στην υγεία, την επιστήμη, την επικοινωνία, τις μεταφορές, την εκπαίδευση και τις ανθρώπινες προσωπικότητες (Kratzke & Quint, 2017).

Τόσο το υλικό όσο και οι εφαρμογές λογισμικού έχουν αλλάξει δραστικά με την πάροδο του χρόνου. Όλες οι συσκευές παρέχουν διάφορους τύπους ενσωματωμένων λειτουργιών, όπως πρόσβαση στη μητρική συσκευή, κάμερα, GPS και πολλά άλλα. Αυτά τα πρωτοποριακά χαρακτηριστικά και οι λειτουργίες παρέχονται κυρίως από τα λειτουργικά τους συστήματα, όπως τα Windows, το macOS, το Android, το iOS, το Linus και μερικά άλλα.

Το Android, το γνωστό λειτουργικό σύστημα 4 5 (OS) για κινητές συσκευές της Google καλύπτει το 42,61% και τα windows της Microsoft κατέχουν το 30,66% του συνολικού μεριδίου αγοράς των λειτουργικών συστημάτων παγκοσμίως. Από την άλλη πλευρά, το λειτουργικό σύστημα του τεχνολογικού γίγαντα Apple για κινητά (iOS) και υπολογιστές (MacOS) κατέχει το 16,55% και το 6,55% του μεριδίου αγοράς παγκοσμίως (Sihag, Vardhan & Singh, 2021).

Αυτά τα λειτουργικά συστήματα είναι πολύ δημοφιλή για το γεγονός ότι παρέχουν διάφορα μοναδικά χαρακτηριστικά και λειτουργίες για τον χρήστη τους. Ο λόγος για την παροχή διαφορετικών μοναδικών χαρακτηριστικών και λειτουργιών είναι η διαφορετική αρχιτεκτονική και η υποστήριξη γλώσσας προγραμματισμού στο λειτουργικό σύστημα. Για να διατηρήσουν τη δημοτικότητά τους, προσθέτουν πάντα διάφορα νέα χαρακτηριστικά στο λειτουργικό σύστημα, τα οποία διευρύνουν πάντα το φάσμα των εφαρμογών (Kratzke & Quint, 2017).

Λόγω αυτού του ανταγωνισμού της μοναδικότητας, η εφαρμογή που αναπτύσσεται για μια πλατφόρμα δεν είναι συμβατή με άλλες πλατφόρμες. Έτσι, οι προγραμματιστές αναγκάζονται να αναπτύσσουν την ίδια εφαρμογή για διαφορετικές πλατφόρμες. Η παροχή μιας εξειδικευμένης για όλες τις πλατφόρμες native λύσης μπορεί να μην προτιμάται πάντα λόγω του κόστους, των πόρων και του χρόνου που σχετίζονται με τη δραστηριότητα ανάπτυξης για κάθε πλατφόρμα.

Όταν μια εφαρμογή είναι αφιερωμένη σε περισσότερες από μία πλατφόρμες και υπάρχουν χρονικοί και οικονομικοί περιορισμοί, συνιστώνται ιδιαίτερα οι λύσεις

πολλαπλών πλατφορμών. Υπάρχουν διάφορες προσεγγίσεις που βασίζονται στην αρχή "Ανάπτυξη μία φορά και εκτέλεση οπουδήποτε". Έτσι, η προσέγγιση cross-platform επιτρέπει σε μια ενιαία βάση κώδικα να απευθύνεται σε πολλαπλές πλατφόρμες (Brunner, Blöchlinger, Toffetti, Spillner & Bohnert, 2015).

Τα σημαντικότερα πλεονεκτήματα αυτών των προσεγγίσεων είναι η εξοικονόμηση κόστους και χρόνου για τη στόχευση πολλαπλών πλατφορμών. Άλλες προσεγγίσεις ανάπτυξης, όπως η υβριδική προσέγγιση, στέκονται ανάμεσα στη διαδικτυακή και την εγγενή μεθοδολογία, οι οποίες έχουν εμφανιστεί για την παροχή μη εγγενών εφαρμογών πολλαπλών πλατφορμών.

Οι υβριδικές εφαρμογές χρησιμοποιούν τεχνολογίες ιστού, όπως HTML5, CSS3, JavaScript και plugins για την πρόσβαση στην πλατφόρμα API της συσκευής, και ένα γενικό API JavaScript για τη γεφύρωση όλων των αιτημάτων υπηρεσιών από τον κώδικα που βασίζεται στον ιστό προς το API της αντίστοιχης πλατφόρμας. Υπάρχουν διάφορες προσεγγίσεις, μία από τις δημοφιλείς προσεγγίσεις ονομάζεται cross-compiled approach, κατά την οποία ο προγραμματιστής γράφει κώδικα σε μια ενιαία γλώσσα και ο cross compiler μεταγλωττίζει τον κώδικα σε συγκεκριμένο εγγενή κώδικα για την εγγενή συσκευή (Guo, Sun, Huang, Wang & Gao, B2007).

Κάθε προσέγγιση έχει τα δικά της πλεονεκτήματα και μειονεκτήματα. Όλες οι διασταυρούμενες πλατφόρμες ακολουθούν διαφορετικές προσεγγίσεις και παρέχουν μοναδικά χαρακτηριστικά για την εφαρμογή ώστε να είναι σε πολλαπλές πλατφόρμες.

Υπάρχουν τέσσερα από τα κυριότερα διαπλατφορμιστικά πλαίσια που είναι διαθέσιμα για την ανάπτυξη εφαρμογών. Στην επόμενη ενότητα θα δοθεί μια σύντομη περιγραφή καθεμιάς από αυτές τις επιλεγμένες πλατφόρμες (Guo, Sun, Huang, Wang & Gao, 2007).

Το React-native είναι ένα συναρπαστικό πλαίσιο ανοικτού κώδικα που δημιουργήθηκε από την Facebook Inc και κυκλοφόρησε αρχικά το 2015. Αυτό δίνει τη δυνατότητα στους προγραμματιστές ιστού να δημιουργούν ισχυρές εφαρμογές για Android, Android TV, iOS, macOS, tvOSs, windows, UWP χρησιμοποιώντας τις υπάρχουσες γνώσεις τους σε Java Script.

Ο στόχος του React-Native είναι να παράγει ειδικές για κάθε πλατφόρμα εκδόσεις των στοιχείων, έτσι ώστε μια ενιαία βάση κώδικα να μπορεί να διαμοιραστεί σε πολλές πλατφόρμες. Τα συστατικά του React τυλίγουν τον υπάρχοντα εγγενή κώδικα και επικοινωνούν με τα εγγενή API χρησιμοποιώντας το δηλωτικό παράδειγμα UI του React και τη JavaScript. Αυτό ανοίγει την ανάπτυξη εγγενών εφαρμογών σε εντελώς νέες ομάδες προγραμματιστών, καθώς και επιτρέπει στις υπάρχουσες ομάδες εγγενών εφαρμογών να εργάζονται σημαντικά ταχύτερα (Piper, 2010).

Το React Native χρησιμοποιεί εγγενή στοιχεία τόσο για το Android όσο και για το iOS. Αυτό σημαίνει ότι οι χρήστες της υβριδικής εφαρμογής iOS ή Android θα έχουν την ίδια εμπειρία που θα είχαν με την εγγενή έκδοση.

Το Flutter είναι ένα κιτ ανάπτυξης λογισμικού UI ανοικτού κώδικα που δημιουργήθηκε από την Google. Χρησιμοποιείται για την ανάπτυξη εφαρμογών για Android, iOS, Linux, Mac, Windows, Google Fuchsia και τον ιστό από μια ενιαία βάση κώδικα. Το Flutter χρησιμοποιεί ένα ευέλικτο σύστημα που επιτρέπει να καλείται APIs συγκεκριμένης πλατφόρμας, είτε αυτά είναι διαθέσιμα σε κώδικα Kotlin ή Java στο Android, είτε σε κώδικα Swift ή Objective-C στο iOS (Javed & Estep, 2019).

Η ενσωματωμένη υποστήριξη API συγκεκριμένων πλατφορμών του Flutter βασίζεται σε ένα ευέλικτο στυλ διέλευσης μηνυμάτων και όχι στη δημιουργία κώδικα. Το Package Pigeon, από την άλλη πλευρά, μπορεί να χρησιμοποιηθεί για τη δημιουργία κώδικα για την αποστολή δομημένων typesafe μηνυμάτων.

Το Ionic είναι ένα πλήρες SDK ανοικτού κώδικα για την ανάπτυξη υβριδικών εφαρμογών που δημιουργήθηκε από τους Max Lynch, Ben Sperry και Adam Bradley της Drifty Co. το 2013. Η πρώτη έκδοση, η οποία κυκλοφόρησε το 2013, βασιζόταν στο AngularJS και το Apache Cordova.

Η πιο πρόσφατη έκδοση, ωστόσο, ξαναχτίστηκε ως ένα σύνολο Web Components, επιτρέποντας στον χρήστη να χρησιμοποιήσει οποιοδήποτε πλαίσιο διεπαφής χρήστη, συμπεριλαμβανομένων των Angular, React και Vue.js. Επιτρέπει επίσης τη χρήση στοιχείων Ionic χωρίς καθόλου πλαίσιο διεπαφής χρήστη (Sihag, Vardhan & Singh, 2021).

Το Ionic παρέχει εργαλεία και υπηρεσίες για την ανάπτυξη υβριδικών εφαρμογών για κινητά, επιτραπέζιες και προοδευτικές εφαρμογές ιστού που βασίζονται σε σύγχρονες τεχνολογίες και πρακτικές ανάπτυξης ιστού, χρησιμοποιώντας τεχνολογίες ιστού όπως CSS, HTML5 και Sass. Ειδικότερα, οι εφαρμογές για κινητά μπορούν να αναπτυχθούν χρησιμοποιώντας αυτές τις τεχνολογίες Web και στη συνέχεια να δημοσιευτούν μέσω των native app stores για εγκατάσταση σε συσκευές χρησιμοποιώντας το Cordova ή το Capacitor.

Η Xamarin είναι μια εταιρεία λογισμικού με έδρα το Σαν Φρανσίσκο που ανήκει στη Microsoft και ιδρύθηκε τον Μάιο του 2011, η Xamarin Android (πρώην Mono for Android) και η Xamarin.iOS (πρώην MonoTouch), οι οποίες είναι διαπλατφορμικές υλοποιήσεις της Υποδομής Κοινής Γλώσσας (CLI) και των Προδιαγραφών Κοινής Γλώσσας (που συχνά αποκαλούνται Microsoft .NET) (Teng & Helps, 2010).

Με μια κοινή βάση κώδικα σε C#, οι προγραμματιστές μπορούν να χρησιμοποιούν τα εργαλεία Xamarin για να γράφουν εγγενείς εφαρμογές Android, iOS και Windows με εγγενείς διεπαφές χρήστη και να μοιράζονται κώδικα σε πολλές πλατφόρμες, συμπεριλαμβανομένων των Windows, macOS και Linux.

Η Xamarin ισχυρίζεται ότι τα προϊόντα της χρησιμοποιούνται από πάνω από 1,4 εκατομμύρια προγραμματιστές σε 120 χώρες σε όλο τον κόσμο. Το Xamarin επεκτείνει την πλατφόρμα .NET με εργαλεία και βιβλιοθήκες ειδικά για τη δημιουργία εφαρμογών σε iOS, Android, macOS και άλλα. Οι εφαρμογές Xamarin μεταγλωττίζονται για εγγενείς επιδόσεις και εκμεταλλεύονται την επιτάχυνση υλικού για συγκεκριμένες πλατφόρμες. Μειώστε το χρόνο εκκίνησης, ενισχύστε τον διαμοιρασμό μνήμης και βελτιώστε τις επιδόσεις εκτελώντας πλήρη μεταγλώττιση πριν από το χρόνο (AOT) στις εφαρμογές (Sihag, Vardhan & Singh, 2021).

Το σημαντικότερο κριτήριο επιλογής ενός πλαισίου Cross-platform είναι ο αριθμός των πλατφορμών που καλύπτει. Είναι επίσης σημαντική η δυνατότητα χρήσης των υφιστάμενων δεξιοτήτων. Το ζητούμενο είναι η εξοικονόμηση χρόνου και ο στόχος είναι να φτάσουμε σε επιπλέον πλατφόρμες χωρίς να δαπανήσουμε περισσότερο χρόνο για την εκμάθηση μιας νέας γλώσσας προγραμματισμού.

Έτσι, η αρχιτεκτονική του πλαισίου, συμπεριλαμβανομένης της γλώσσας ανάπτυξης, των δυνατοτήτων API και του εύρους των πλατφορμών που υποστηρίζονται από το CPF, είναι ένα πολύ σημαντικό κριτήριο επιλογής για τη διατήρηση του χαμηλού κόστους και της εύκολης εκμάθησης. Η ταχεία διαδικασία ανάπτυξης είναι ένα άλλο σημαντικό στοιχείο κατά την επιλογή του CPF (Anderson, 2006).

Ο πρωταρχικός σκοπός του CPF είναι η μείωση του κόστους και της προσπάθειας. Έτσι, η ταχεία ανάπτυξη και η απρόσκοπτη διαδικασία εντοπισμού σφαλμάτων είναι ύψιστης προτεραιότητας για τους προγραμματιστές. Δεν λειτουργούν όλα τα συστήματα κατασκευής με τον ίδιο τρόπο. Κατά την επιλογή ενός CPF, οι προγραμματιστές θέλουν πάντα να βεβαιώνονται ότι υπάρχει καλή τεκμηρίωση γύρω από την κατασκευή, την αποσφαλμάτωση και την απελευθέρωση για όλες τις πλατφόρμες-στόχους (Sihag, Vardhan & Singh, 2021)..

Τελικά, το IPA και το APK θα δημιουργηθούν χρησιμοποιώντας τα εγγενή εργαλεία κατασκευής, αλλά οι προγραμματιστές θέλουν να βεβαιωθούν ότι το CPF έχει τη δυνατότητα να αλληλεπιδρά με αυτά σε κάποιο επίπεδο για να κάνει τη ζωή του προγραμματιστή ευκολότερη.

Η ενεργή υποστήριξη της κοινότητας για τους προγραμματιστές είναι επίσης πολύ σημαντική για την ταχεία διαδικασία ανάπτυξης. Έτσι, το τοπίο υποστήριξης της γλώσσας προγραμματισμού ή το ίδιο το CPF είναι πολύ σημαντικό για τους προγραμματιστές κατά την επιλογή ενός CPF (Malhotra, Kumar & Gupta, 2020).

Τέλος, η εμπειρία του χρήστη, η εμφάνιση και η αίσθηση της διεπαφής χρήστη και η απόδοση του τελικού προϊόντος/εφαρμογής. Έτσι, μετά τη συζήτηση, θα αναλύσουμε και θα συγκρίνουμε τα CPF ανάλογα με την αρχιτεκτονική του πλαισίου, την ανάπτυξη της εφαρμογής, την ανάπτυξη, την υποστήριξη των προγραμματιστών και την εμπειρία του χρήστη, τα οποία έχουν αναγνωριστεί ως βασικά ζητήματα στην ανάπτυξη εφαρμογών CPF (Sihag, Vardhan & Singh, 2021).

2.2. Native Εφαρμογές

Οι εγγενείς εφαρμογές δημιουργούνται με τη χρήση της εγγενούς γλώσσας προγραμματισμού της συσκευής για την οποία πρέπει να δημιουργηθούν. Εάν μια εφαρμογή δημιουργείται για το iOS, πρέπει να είναι γραμμένη σε Objective-C ή στη νέα γλώσσα, Swift. Οι εφαρμογές για το Android χρησιμοποιούν τη μητρική του γλώσσα Java (Niveditha & Ananthan, 2019).

Οι εγγενείς εφαρμογές παρέχουν ένα περιβάλλον ανάπτυξης με εργαλεία και widgets για τη δημιουργία των επιθυμητών διεπαφών με εγγενή εμπειρία αλληλεπίδρασης με τον χρήστη, τα οποία δεν υπάρχουν ακόμη στην περίπτωση των εργαλείων ανάπτυξης υβριδικών εφαρμογών.

Κάθε εγγενής εφαρμογή προσφέρει καλύτερη συνολική εμπειρία για τον χρήστη. Ορισμένες από τις τυπικές διαδικασίες που θα επεξεργαζόταν η εγγενής εφαρμογή είναι multi-touch, ταχύτερα γραφικά APIs, ρευστή κίνηση, ενσωματωμένα στοιχεία και ευκολία χρήσης.

Τα χαρακτηριστικά πολλαπλής αφής της εγγενούς εφαρμογής καθιστούν δυνατή την αλληλεπίδραση του χρήστη με τη συσκευή με σύνθετες χειρονομίες UI (διεπαφή χρήστη). Για παράδειγμα, οι χρήστες μπορούν να κάνουν διπλό πάτημα για να κάνουν ζουμ. Pinch-spread και άλλες προηγμένες χειρονομίες

Ανάλογα με τα διάφορα χαρακτηριστικά της συσκευής, οι εγγενείς εφαρμογές παρέχουν γρήγορο γραφικό API (Anderson, 2006). Κινούμενα σχέδια, τα οποία είναι απαραίτητα όταν παρέχεται εμπειρία παιχνιδιού στη συσκευή. Χρειάζεται επίσης για άκρως διαδραστικές αναφορές και σύνθετους υπολογιστικούς αλγορίθμους.

2.2.1. Android

Το Android είναι ένα δωρεάν λειτουργικό σύστημα ανοικτού κώδικα που προορίζεται για κινητές συσκευές. Παρουσιάστηκε το 2007 από την Open Handset Alliance η οποία υποστηρίζεται εμπορικά από την Google. Το Android βασίζεται στον πυρήνα Linux με πολλές βιβλιοθήκες και APIs γραμμένα σε C και πολλά πλαίσια εφαρμογών που περιλαμβάνουν βιβλιοθήκες συμβατές με Java (Niveditha & Ananthan, 2019).

Το Android εξακολουθεί να υποστηρίζει εγγενώς τον προγραμματισμό σε Java API, αλλά τελευταία έχουν γίνει πολλά βήματα για να ενθαρρυνθούν οι προγραμματιστές να στραφούν στην Kotlin. Σε αντίθεση με τη δημοφιλή πεποίθηση, το Android δεν χρησιμοποιεί JVM (Java Virtual Machine). Αντ' αυτού, χρησιμοποιεί το ART (Android Runtime), το οποίο μετατρέπει τον bytecode της εφαρμογής σε εγγενείς εντολές. Μετά από αυτό, οι εντολές μπορούν να εκτελεστούν από το περιβάλλον

εκτέλεσης Android. Η Java και η Kotlin μπορούν να συνυπάρχουν στο ίδιο έργο, αλλά η Kotlin έχει πολλά φιλικά προς το χρήστη χαρακτηριστικά και βελτιώσεις στο συντακτικό που δεν έχει η Java (Malhotra, Kumar & Gupta, 2020).

Η Kotlin είναι μια γλώσσα προγραμματισμού ανοικτού κώδικα που αναπτύχθηκε από την JetBrains8 (τσεχική εταιρεία ανάπτυξης λογισμικού), γνωστή ως γλώσσα προγραμματισμού εφαρμογών για συσκευές Android. Παρόλο που η Google αρχικά υποστήριζε μόνο Java για εφαρμογές Android, στα τέλη του 2017 στράφηκε προς την Kotlin και έκτοτε προωθεί και υποστηρίζει τη χρήση της στο Android Studio.

Παρά το γεγονός αυτό, η Kotlin είναι κυρίως γνωστή ως γλώσσα προγραμματισμού για την ανάπτυξη Android. Μπορεί επίσης να χρησιμοποιηθεί για την ανάπτυξη ιστοσελίδων, την επιστήμη δεδομένων και την ανάπτυξη στην πλευρά του διακομιστή. Η Kotlin υποστηρίζει επίσης πολλά παρόμοια φιλικά προς τον προγραμματιστή χαρακτηριστικά όπως και η Swift (Malhotra, Kumar & Gupta, 2020).

Για παράδειγμα, την εξαγωγή συμπερασμάτων τύπου κατά τη διάρκεια εκτέλεσης που δεν αναγκάζει τον προγραμματιστή να καθορίσει τον τύπο της έκφρασης κατά τη μεταγλώττιση, ή τις εξαιρέσεις μηδενικού δείκτη και την έμφαση στη συνολική ασφάλεια του κώδικα. Παρόλο που η ιδέα που κρύβεται από πίσω είναι θεμελιωδώς κοινή μεταξύ αυτών των δύο γλωσσών, η εκτέλεση του κώδικα διαφέρει. Χάρη στο Kotlin Mobile SDK, η Kotlin μπορεί επίσης να χρησιμοποιηθεί για τη δημιουργία μιας εφαρμογής iOS και μπορεί ακόμη και να χρησιμοποιήσει το εγγενές API της, και έτσι μπορεί να θεωρηθεί για μια cross-platform ανάπτυξη (Holla & Katti, 2012).

Το Android Studio είναι ένα εγγενές IDE που δημιουργήθηκε από την JetBrains ειδικά για την ανάπτυξη Android. Είναι διαθέσιμο σε λειτουργικά συστήματα macOS, Linux και Windows. Ο έμμεσος προκάτοχός του ήταν ένα plugin για το Eclipse IDE με την ονομασία Eclipse Android Development Tools (E-ADT), το οποίο σταμάτησε το 2015 (Malhotra, Kumar & Gupta, 2020).

Το Android Studio προσφέρει πολλά χαρακτηριστικά που είναι δημοφιλή σε άλλα IDE, όπως η επισήμανση σύνταξης για πολλές γλώσσες προγραμματισμού, η συμπλήρωση κώδικα ή ο έλεγχος έκδοσης κώδικα μέσω του Git. Διαθέτει επίσης μια εξαιρετική ενσωμάτωση εικονικής συσκευής Android (εξομοιωτή) με ενσωματωμένο περιβάλλον χρήστη, όπου ο προγραμματιστής μπορεί να επιλέξει από πολλές συσκευές της Google και τις εκδόσεις της για να εκτελέσει και να αποσφαλματώσει την εφαρμογή (Niveditha & Ananthan, 2019).

Ο εξομοιωτής υποστηρίζει την προσομοίωση πολλών χαρακτηριστικών που είναι προσβάσιμα μόνο σε φυσικές συσκευές, όπως η προσομοίωση σημείων θέσης σαν να χρησιμοποιείται η συσκευή σε αυτοκίνητο, η προσομοίωση ειδικών προκαθορισμένων ακολουθιών κλίσης της συσκευής, η προσομοίωση αργού δικτύου ή ακόμη και η πρόσβαση στην κάμερα (στην προκειμένη περίπτωση σε μια κάμερα, αν αυτή είναι προσβάσιμη στον υπολογιστή) (Malhotra, Kumar & Gupta, 2020).

Το Android Studio υποστηρίζει επίσης πολλές δημοφιλείς γλώσσες προγραμματισμού (π.χ. Java, C++, Kotlin και Go). Όπως το Xcode, έτσι και το Android Studio διαθέτει το εργαλείο του για την οπτική αναπαράσταση των στοιχείων και της σχέσης τους με άλλα στοιχεία.

Ονομάζεται visual layout editor και προσφέρει στον προγραμματιστή τη δυνατότητα να δει (πριν από τη μεταγλώττιση) πώς θα φαινόταν η συγκεκριμένη οθόνη σε μια συγκεκριμένη συσκευή. Προσφέρει επίσης τη δυνατότητα ορισμού περιορισμών για κάθε στοιχείο (π.χ. το πλάτος, το ύψος του ή περιορισμούς που σχετίζονται με άλλα στοιχεία, όπως το περιθώριο) (Niveditha & Ananthan, 2019).

2.2.2. IOS

2.2.2.1. Objective

Η Objective-C είναι μια γλώσσα προγραμματισμού που υιοθετήθηκε από την Apple το 1996. Αυτό συνέβη επειδή η Apple εξαγόρασε τη NeXT με σκοπό τη χρήση του λειτουργικού της συστήματος στους υπολογιστές της. Η Apple ωθεί αργά αλλά σταθερά την Objective-C εκτός χρήσης πλέον (Malhotra, Kumar & Gupta, 2020).

Η γλώσσα προγραμματισμού εξακολουθεί να υποστηρίζεται, αλλά δίνεται μεγαλύτερη έμφαση στην εγγενή ανάπτυξη με τη χρήση του Swift3 , το οποίο εισήχθη το 2014. Η Objective-C και η Swift μπορούν να συνυπάρξουν στο ίδιο έργο, αλλά η Swift έχει πολλά φιλικά προς τη σύνταξη χαρακτηριστικά και βελτιώσεις σε σχέση με την Objective-C.

2.2.2.2. Swift

Το Swift είναι μια γλώσσα προγραμματισμού ανοιχτού κώδικα που αναπτύχθηκε από την Apple, γνωστή κυρίως ως γλώσσα που χρησιμοποιείται για τη δημιουργία εφαρμογών και λογισμικού για τα προϊόντα της. Αναπτύχθηκε ως αντικαταστάτης της Objective-C λόγω της έλλειψης σύγχρονων γλωσσικών χαρακτηριστικών (Niveditha & Ananthan, 2019).

Ορισμένα από τα χαρακτηριστικά της Swift επηρεάστηκαν φυσικά από την Objective-C, όπως η δυναμική αποστολή ή η καθυστερημένη δέσμευση. Παρόλο που η Swift σχεδιάστηκε για να είναι συγκρίσιμη σε απόδοση με τις γλώσσες που βασίζονται στη C, δίνει επίσης μεγάλη έμφαση στο κομμάτι της ασφάλειας της γλώσσας.

Για παράδειγμα, οι μεταβλητές αρχικοποιούνται πάντα πριν από τη χρήση, οι πίνακες ελέγχονται για υπερχείλιση και η διαχείριση της μνήμης γίνεται αυτόματα. Ένα από

τα χαρακτηριστικά ασφαλείας της Swift είναι η χρήση του συμβολισμού nil (Annuzzi, Darcey & Conder, 2014).

Αντικείμενα με τιμή nil δεν μπορούν να έχουν πρόσβαση στις ιδιότητες και τις μεθόδους τους. Αυτό θα έχει ως αποτέλεσμα ένα σφάλμα σε χρόνο μεταγλώττισης. Ωστόσο, το Swift διαθέτει ένα χαρακτηριστικό που είναι γνωστό ως optionals. Αν χρησιμοποιήσουμε ένα αντικείμενο nil, μας αναγκάζει να υποδείξουμε στον μεταγλωττιστή ότι είμαστε σίγουροι και κατανοούμε τη συμπεριφορά μιας τέτοιας ενέργειας.

2.3. Λειτουργικό Σύστημα Macintosh και Περιβάλλον Ανάπτυξης Κώδικα (Xcode)

Όπως αναφέρθηκε, το Xcode είναι το κύριο IDE για την ανάπτυξη λογισμικού για όλες τις συσκευές και τα λειτουργικά συστήματα της Apple (macOS, tvOS, watchOS, iPadOS). Το Xcode υποστηρίζει πολλές δημοφιλείς γλώσσες προγραμματισμού, όπως C++, Java, Python, αλλά κυρίως Objective-C και Swift (Holla & Katti, 2012).

Διαθέτει επίσης μια ολοκληρωμένη διαχείριση πηγαίου κώδικα (version control) βασισμένη στο Git4 , λειτουργίες profiling και debugging, η οποία επιτρέπει στον προγραμματιστή να ελέγχει την έκδοση του πηγαίου κώδικα του από το περιβάλλον χρήστη και να βλέπει τις αλλαγές απευθείας στα αρχεία κώδικα (Sarkar, Goyal, Hicks, Sarkar & Hazra, 2019).

Το Xcode προσφέρει πολλές δυνατότητες για τα εγγενή έργα ανάπτυξης που μπορεί να μην παρέχονται σε άλλα δημοφιλή IDE. Για παράδειγμα, ένας οργανωτής συντριβών επιτρέπει την επιθεώρηση μεμονωμένων συντριβών των εφαρμογών που είναι διαθέσιμες στο App Store. Οι παρεχόμενες πληροφορίες περιλαμβάνουν τη στοίβα νημάτων, συμπεριλαμβανομένων των υποκείμενων σφαλμάτων και πρόσθετων στατιστικών στοιχείων, όπως τα ποσοστά συντριβής σε διαφορετικές εκδόσεις και συσκευές iOS.

Αυτές οι αναφορές συντριβών μπορούν επίσης να διερευνηθούν σαν να συνέβη το σφάλμα κατά την ανάπτυξη της εφαρμογής. Προσφέρει επίσης ένα εργαλείο δημιουργίας διεπαφών με την ονομασία Interface Builder , το οποίο παρέχει έναν εύκολο, φιλικό προς τον χρήστη τρόπο σχεδιασμού της διεπαφής χρήστη της εφαρμογής, ενισχυμένο με χαρακτηριστικά όπως η βασική αλληλεπίδραση μεταξύ των στοιχείων της διεπαφής (π.χ. προώθηση συγκεκριμένης προβολής στη στοίβα πλοήγησης όταν πατηθεί ένα κουμπί) (Sarkar, Goyal, Hicks, Sarkar & Hazra, 2019).

Επιπλέον, η έκδοση 11 του Xcode εισήγαγε το SwiftUI6 , το οποίο πηγαίνει τη δημιουργία διεπαφής ακόμη πιο μακριά και παρέχει μια ζωντανή προεπισκόπηση της εφαρμογής. Αυτό είναι ιδιαίτερα χρήσιμο επειδή δεν χρειάζεται πλέον να μεταγλωττίζουμε την εφαρμογή για να αλληλεπιδράσουμε με τη διεπαφή χρήστη.

2.4. Τεχνολογίες Βάσης Δεδομένων σε Περιβάλλον IOS

Η Apple χωρίζει τη δομή της σε τέσσερα κύρια επίπεδα:

- Core OS
- Core Services
- Media
- Cocoa Touch

Κάθε στρώμα έχει συγκεκριμένο σκοπό και φιλοξενεί διάφορα πλαίσια για την αλληλεπίδραση με αυτά. Το στρώμα Core OS είναι υπεύθυνο για την επικοινωνία με το υλικό της υποκείμενης συσκευής, όπως το γυροσκόπιο, το Bluetooth και τα εξωτερικά αξεσουάρ.

Αυτό το στρώμα χειρίζεται επίσης τη διαχείριση ενέργειας, την ασφάλεια και πολλές από τις λειτουργίες προγραμματισμού χαμηλότερου επιπέδου. Το Core Services είναι υπεύθυνο για την παροχή λειτουργιών δικτύου, αυτόματης καταμέτρησης αναφορών (γνωστή σε άλλες γλώσσες και ως garbage collector), λειτουργιών συμβολοσειρών και άλλων λειτουργιών μορφοποίησης δεδομένων (Annuzzi, Darcey & Conder, 2014).

Παρέχει επίσης στους προγραμματιστές δυνατότητες πρόσβασης σε αρχεία και νήματα. Το επίπεδο πολυμέσων χειρίζεται την αναπαραγωγή πολυμέσων καθώς και την εγγραφή και την επεξεργασία οπτικοακουστικών μέσων. Είναι επίσης υπεύθυνο για την απόδοση 2D και 3D γραφικών καθώς και για την κινούμενη εικόνα.

Το Cocoa Touch είναι ένα στρώμα στο οποίο συμβαίνει το μεγαλύτερο μέρος της αλληλεπίδρασης με τον χρήστη, χειρίζεται τις χειρονομίες, τα συμβάντα αφής που ενεργοποιούνται από τον χρήστη, τη συνολική πτυχή της οπτικοποίησης της εφαρμογής και τη συμπεριφορά της (Sarkar, Goyal, Hicks, Sarkar & Hazra, 2019).

2.4.1. Σύγκριση τεχνολογιών βάσης δεδομένων για τις κινητές εφαρμογές IOS

Ποια βάση θα χρησιμοποιηθεί για την αποθήκευση των δεδομένων.

Το θέμα της βάσης δεδομένων που θα χρησιμοποιηθεί στην ανάπτυξη του λογισμικού IOS είναι αρκετά περίπλοκο και υπάρχουν διάφορες επιλογές.Τρείς από αυτές είναι οι πιο δημοφιλές:

- COREDATA
- REALM
- SQLITE

COREDATA:

Αυτή είναι η βάση που προτεινόμενη από την εταιρεία της APPLE.Καταλάμβάνει περισσότερο χώρο σε σύγκριση με τις υπόλοιπες τεχνολογίες αποθήκευσης δεδομένων σε κινητές συσκευές.Εχει αυξημένη ταχύητητα εισαγωγής δεδομένων αλλά δεν παρέχει την δυνατότητα περιορισμού αποθήκευσης δεδομένων.Κάθε φορά πρέπει να γίνει μεταφορά δεδομένων από τον δίσκο στην κύρια μνήμη διότι εκτελείται ως γνωστόν στην κύρια μνήμη.Για την εκτέλεση των λειτουργιών διαγραφής και ενημέρωσης είναι απαραίτητη η μεταφορά της από τον δίσκο στην μνήμη.

SQLITE:

Αυτή η βάση δεδομένων ανοιχτού κώδικα(opensource)χρησιμοποιείται πολύ συχνά τόσο στο Android όσο και στο IOS.Δεδομένου ότι έιναι ένα απλό και χρήσιμο προγραμμα,παρέχει άνεση στους χρήστες.Παρέχει τη δυνατότητα εκτέλεσης πολλαπλών λειτουργειών παράλληλα.Προσφέρει την δυνατότητα καθορισμού του μεγέθους των δεδομένων που πρέπει να αποθηκευτούν στην τοπική βάση μας.Για την

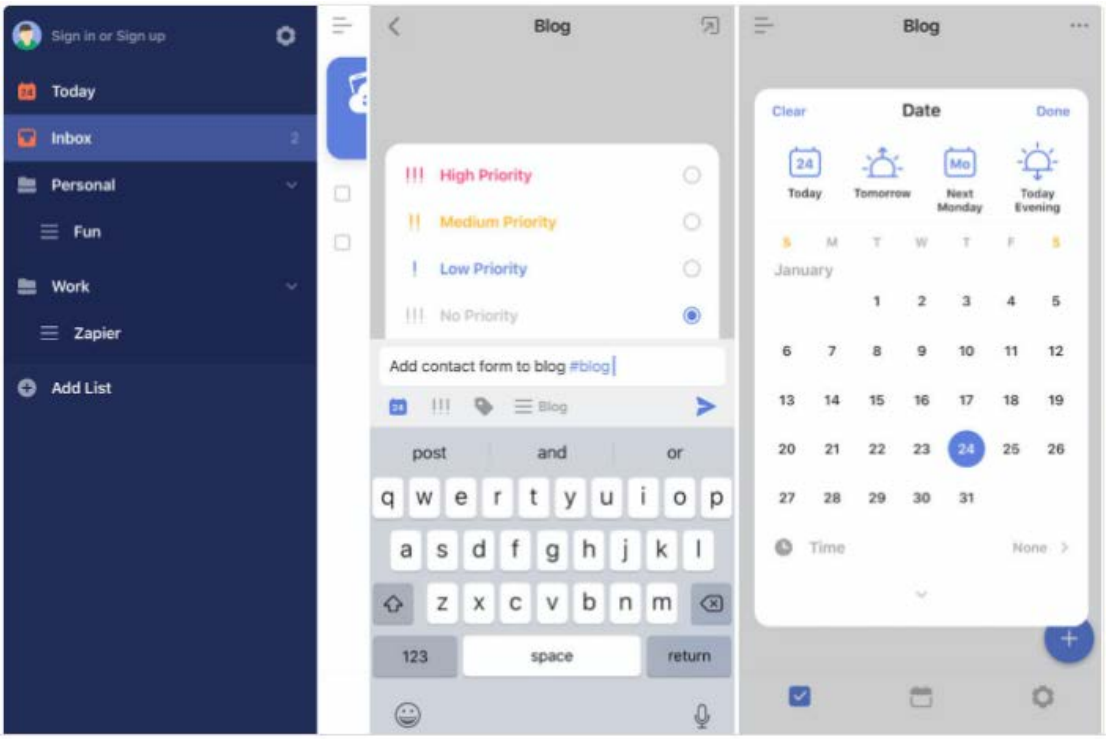
εκτέλεση των βασικών λειτουργίων (π.χ ενημέρωση, διαγραφή) δεν χρειάζεται η μεταφορά της βάσης από τον δίσκο στην μνήμη.

REALM:

Αυτή η βάση είναι μια επιλογή που προσφέρεται από την Apple.Καταλαμβάνει περισσότερο χώρο από τη βάση SQLITE.Δεν παρέχει τη δυνατότητα περιορισμού δεδομένων.Στις εγγραφές των δεδομένων έχει αρκετά υψηλή ταχύτητα.Η εκτέλεση της βάσης γίνεται στην μνήμη οπότε κάθε φορά χρειάζεται η μεταφορά της βάσης από τον δίσκο.Για να εκτελέσετε λειτουργίες όπως ενημέρωση και διαγραφή ,είναι απαράιτητο να φορτώσετε ξανά το πρόγραμμα δεδομένων.

2.4.2. Παρόμοιες εφαρμογές toDoList σε περιβάλλον IOS

TickTick:

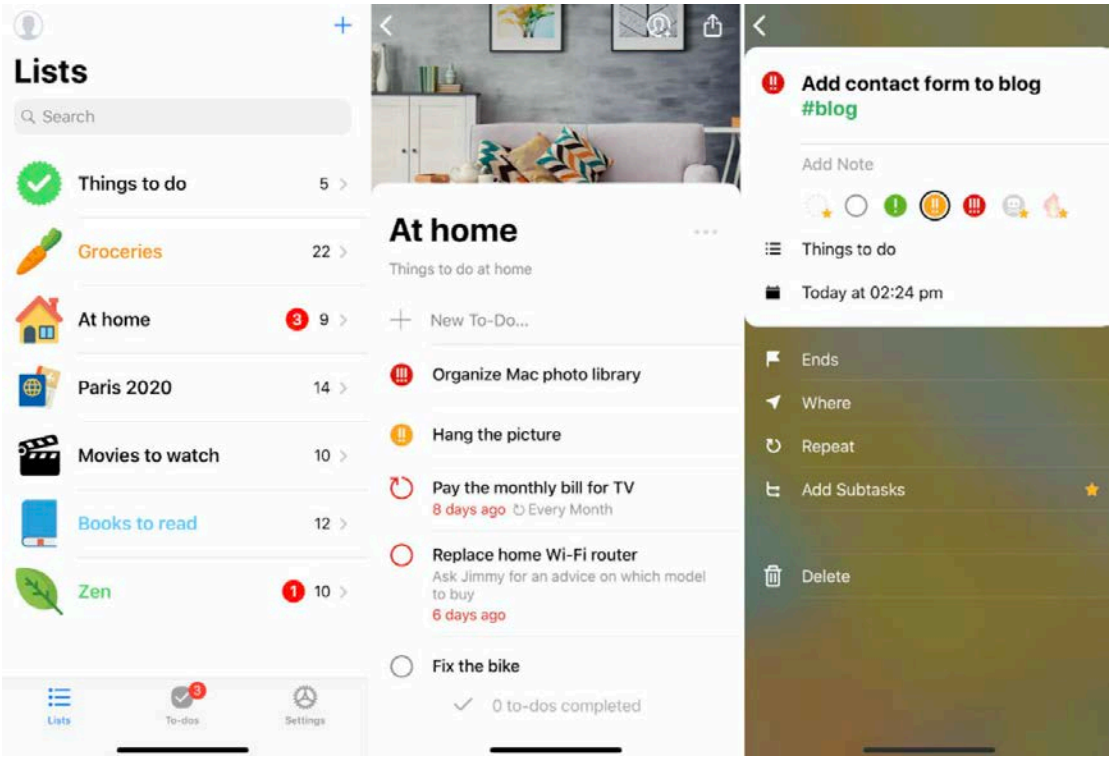


Το TickTick είναι ένα άλλο ισχυρό σύστημα διαχείρισης λίστας εκκρεμοτήτων που σας παρασύρει με μια συναρπαστική δωρεάν επιλογή. Εάν αποφασίσετε να κάνετε

αναβάθμιση, η ετήσια συνδρομή προσφέρει καλή αξία λαμβάνοντας υπόψη τις δυνατότητες που θα λάβετε.Η προσθήκη εργασιών στην εφαρμογή είναι γρήγορη χάρη στην ανάλυση φυσικής γλώσσας για ημερομηνίες και συντομεύσεις για τον ορισμό προτεραιότητας, την προσθήκη ετικετών και την ανάθεση των εργασιών σας σε μια συγκεκριμένη λίστα. Πατήστε το κουμπί Επιστροφή για να προσθέσετε μια εργασία στα εισερχόμενά σας, στη συνέχεια συνεχίστε να πληκτρολογείτε και πατήστε ξανά Επιστροφή για να προσθέσετε άλλη.

Προσθέστε μια λίστα ελέγχου για να δημιουργήσετε δευτερεύουσες εργασίες, να ανεβάσετε συνημμένα ή φωτογραφίες, να προσθέσετε τοποθεσίες ετικετών ή να προσθέσετε μια ωραία μεγάλη περιγραφή σε κάθε καταχώριση.

Η συνεργασία είναι ενσωματωμένη ώστε να μπορείτε να μοιράζεστε τις λίστες σας TickTick, καθώς και ένα χρονόμετρο Pomodoro που σας βοηθά να εστιάσετε.



Pocket Lists:

Ο Pocket Lists περιγράφει τον εαυτό του ως την πιο φιλική εφαρμογή λίστας στον κόσμο και δεν είναι δύσκολο να καταλάβει κανείς γιατί. Ακριβώς όπως το WeDo, το Pocket Lists αποτινάσσει το μονόχρωμο σχέδιο που μοιράζονται τόσοι πολλοί διοργανωτές υπέρ των χρωμάτων, των emoji και των θεμάτων. Είναι μια καθαρή εφαρμογή δημιουργίας λιστών από την αρχή, πλήρης με έκδοση εφαρμογής web και επεκτάσεις iMessage και Apple Watch για premium χρήστες.

Προσθέστε εργασίες απευθείας στη σχετική λίστα και, στη συνέχεια, ορίστε μια ημερομηνία λήξης (ή χρησιμοποιήστε ανάλυση φυσικής γλώσσας), καθορίστε μια τοποθεσία και ορίστε το επίπεδο προτεραιότητας. Κάθε λίστα μπορεί να προσαρμοστεί με το δικό της εικονίδιο, με πάνω από 400 να διαλέξετε. Αυτό κάνει το Pocket Lists μια λύση δημιουργίας λιστών για σχεδόν οποιοδήποτε σκοπό, από την καθημερινή οργάνωση έως τον μακροπρόθεσμο προγραμματισμό.

Είναι δυνατό να προβάλετε όλα τα επερχόμενα στοιχεία σας σε ένα πρόγραμμα, ώστε να γνωρίζετε τι πρέπει να κάνετε πρώτα. Στη συνέχεια, μπορείτε να ταξινομήσετε αυτήν τη λίστα κατά υψηλή προτεραιότητα, κοντινές εργασίες, επαναλαμβανόμενες εργασίες κ.λπ.

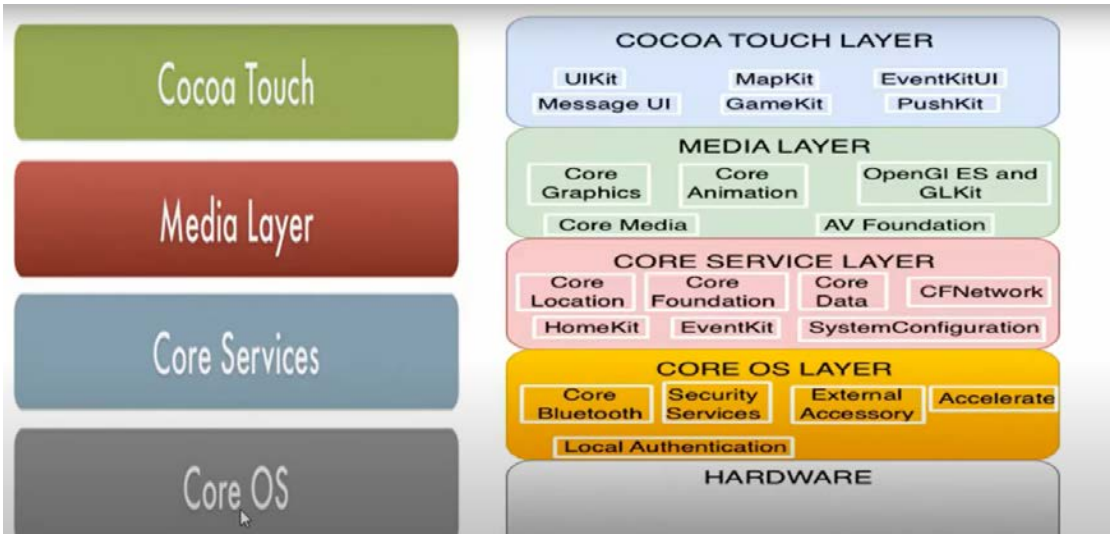
3. Η Αρχιτεκτονική του συστήματος στις κινητές συσκευές IOS

3.1. Τα επίπεδα της αρχιτεκτονικής

Η αρχιτεκτονική του συστήματος για κινητές συσκευές IOS αποτελείται από 4 επίπεδα:

- Cocoa Touch
- Media Player
- Core Services
- CoreOS

Η αρχιτεκτονική των κινητών συσκευών IOS είναι μια πολυεπίπεδη αρχιτεκτονική.Οι βιβλιοθήκες του τελευταίου επιπέδου λειτουργούν ως ενδιάμεσος κρίκος μεταξύ του υλικού του συστήματος και των εφαρμογών που δημιουργούμε εμείς.Οι εφαρμογές δεν επικοινωνούν με τις βιβλιοθήκες του χαμηλότερου επιπέδου.Μέσω των διεπαφών(Interfaces) οι εφαρμογές έχουν την δυνατότητα της πρόσβασης σε κάποιες δυνατότητες του υλικού(Bluetooth).Στις χαμηλότερο επίπεδο της αρχιτεκτονικής παρέχει τις βασικές υπηρεσίες ενώ στο τελευταίο επίπεδο της αρχιτεκτονικής υπάρχουν υπηρεσίες με σκοπό να καλύψουν τις αναγκες για το σχεδιασμό των εφαρμογών.Σε κάθε επίπεδο υπάρχουν πλαίσια(frameworks) με σκοπό να χρησιμοποιήσει ο προγραμματιστής για να αναπτύξει τις εφαρμογές του.



Εικόνα 1

Αυτό είναι το διάγραμμα που δείχνει την αρχιτεκτονική του IOS. Η αρχιτεκτονική του IOS αποτελείται από 4 επίπεδα.

Coco Touch: Είναι το επίπεδο που διαθέτει τα κατάλληλα γραφικά μέσα που αλληλεπιδρούν με τον χρήστη είτε για την εισαγωγή των δεδομένων είτε για την εμφάνιση.

Media: Διαθέτει τα κατάλληλα εργαλεία για ήχο, γραφικά,

Core Services: Διαθέτει τα κατάλληλα εργαλεία για την ομαλή λειτουργία των παραπάνω επιπέδων. Π.χ. Μια τοπική βάση σε αυτό το επίπεδο δημιουργείται. Όπως επίσης εντοπισμός της τοποθεσίας ανήκουν σε αυτή την κατηγορία. Όλα εκτελούνται εκτός ελέγχου το προγραμματιστή.

Core OS: Περιέχει το λειτουργικό σύστημα, ασφάλεια, οδηγοί του λειτουργικού συστήματος. Έχει σαν επίπεδο μια αυτόνομη λειτουργία και δεν μπορούν να επέμβουν οι προγραμματιστές.

Το χαμηλότερο επίπεδο ονομάζεται ο πυρήνας του λειτουργικού συστήματος (Core OS). Το υψηλότερο επίπεδο στην αρχιτεκτονική ονομάζεται (Cocoa Touch).Σε αυτό το επίπεδο ο χρήστης αλληλεπιδρά απευθείας με την εφαρμογή.

Πάμε τώρα να μελετήσουμε την κάθε βαθμίδα με περισσότερη λεπτομέρεια.

Στο κατώτερο επίπεδο υπάρχει το υλικό του συστήματος.Το υλικό αυτό μπορεί να είναι ένας επιτραπέζιος υπολογιστής η ένας φορητός υπολογιστής πάνω σε αυτό σχεδιάζετε και αναπτύσσεται η εφαρμογή.

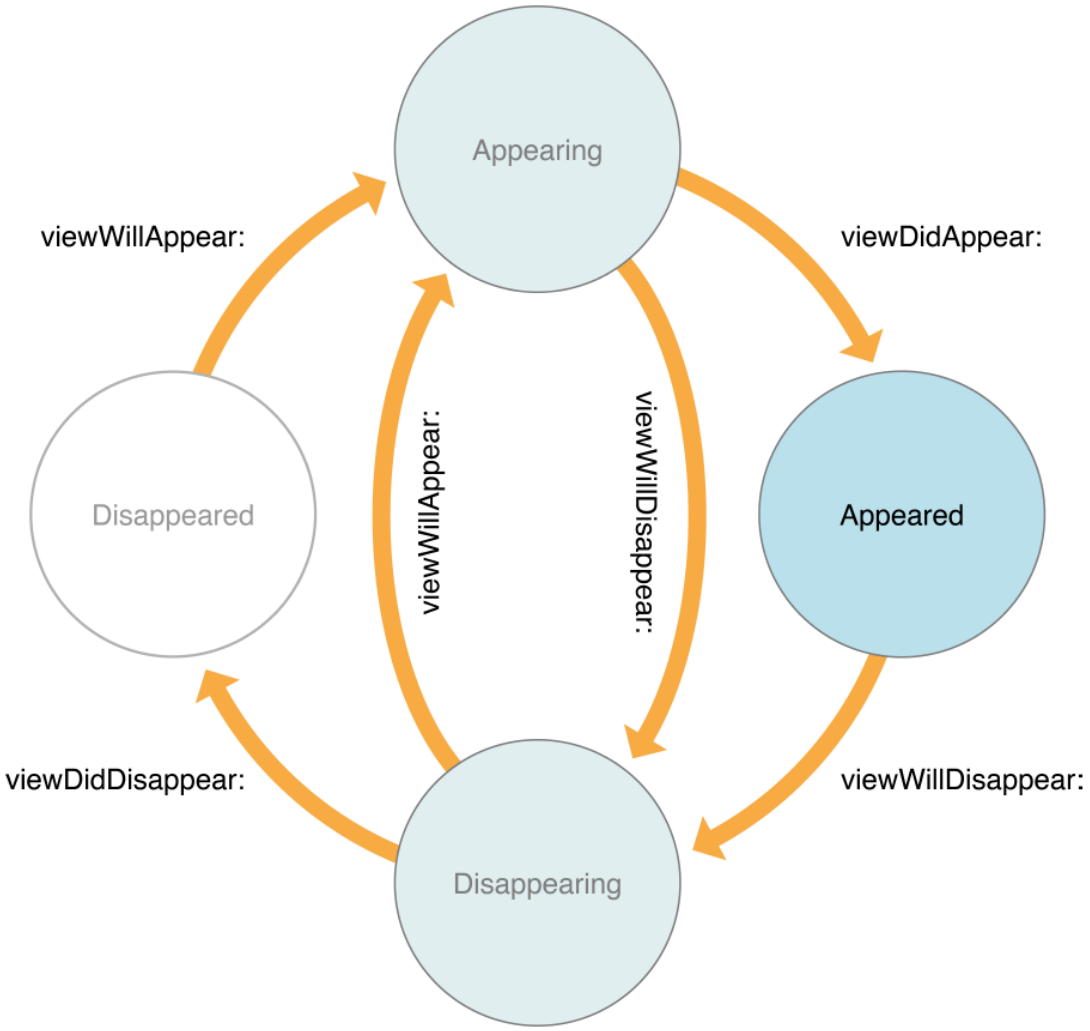
Πάνω από το υλικό υπάρχει ένα βασικό επίπεδο λειτουργικού συστήματος(Core OS).Αυτό το επίπεδο περιέχει κάποιες υπηρεσίες όπως σύνδεση με άλλες συσκευές μέσω Core Bluetooth ,έλεγχος υπηρεσιών ασφάλειας(Security Services) .Εξωτερική πρόσβαση (External Acessory), επιτρέπει στο σύστημα σας να συνδεθεί με εξωτερικές συσκευές της συσκευής σας(SD Cart).

Το επόμενο επίπεδο είναι το επίπεδο υπηρεσίων (Core Service Layer). Σε αυτό επίπεδο βρίσκεται η τοπική βάση CoreData για την αποθήκευση των δεδομένων στην τοπική βάση.

Επάνω στο βασικό επίπεδο υπηρεσίας είναι το επίπεδο πολυμέσων (MEDIA LAYER).Καθώς το όνομα του είναι οι υπηρεσίες που σχετίζονται με τα πολυμέσα σας που μπορεί να είναι ο ήχος/βίντεο. Μπορεί να είναι αρχεία ήχου η αρχεία βίντεου, κινούμενα σχέδια γραφικών. Όπως επίσης το πλαίσιο OpenGI Es and GLkit υποστηρίζει 2d η 3d γραφικά.

Το τελευταίο στρώμα ονομάζεται COCOA TOUCH LAYER που αυτό τώρα περιέχει πλαίσια τύπου εντοπισμού χάρτη η επικοινωνία μέσω εφαρμογών.

3.2. Ο κύκλος ζωής ενός UIView Controller



Εικόνα 2

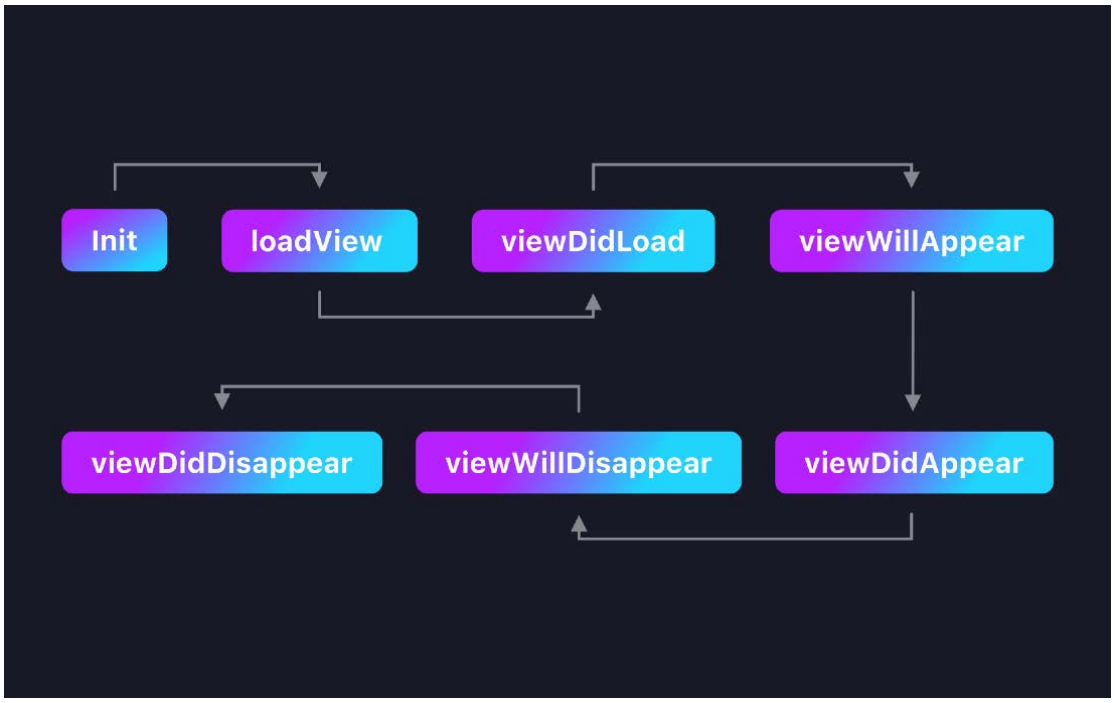
UIView Controller είναι μια κλάση της βιβλιοθήκης UIKit και είναι η κλάση που καθορίζει τις λειτουργίες που θα έχει μια γραφική διεπαφή. Ο βασικός ρόλος της κλάσης είναι να καθορίσει τις λειτουργίες που θα έχει κάθε γραφικό μέσο για την

εισαγωγή η την αναπαράσταση των δεδομένων στο τελικό χρήστη του δίνοντας την δυνατότητα της αλληλεπίδρασης.

Η κύριες λειτουργίες της κλάσης UIView Controller:

Ανανεώνει να τα δεδομένα με βάσει τις αλλαγές που έχουν συμβεί στα αντικείμενα της γραφικής διεπαφής.

Διαχειρίζεται την αλληλεπίδραση μεταξύ των αντικειμένων UIView και του χρήστη.



Εικόνα 3

viewDidLoad: αυτή η συνάρτηση κατά τη διάρκεια του κύκλου ζωής ενός UIViewController εκτελείται μόνο μια φορά. Όταν εκτελείται αυτή η συνάρτηση όλα τα IBOutlet έρχονται σε κατάσταση ετοιμότητας για αναπαράσταση δεδομένων.

Επίσης είναι η κατάλληλη συνάρτηση για φόρτωση αρχικών δεδομένων η πρόσβαση δεδομένων από μια απομακρυσμένη βάση.

viewWillAppear: Προτού να εμφανιστή η γραφική διεπαφή μέσω του UIViewController στον τελικό χρήστη εκτελείται αυτό το στάδιο. Στο στάδιο αυτό εκτελούνται οι διεργασίες για την απόκρυψη στοιχείων στην γραφική διεπαφή.

viewDidAppear: Καλείται αμέσως μετά την εμφάνιση των στοιχείων της κλάσης UIViewController. Είναι το κατάλληλο σημείο για φόρτωση των δεδομένων από ένα μεσάζων εργαλείο πληροφόρησης(API) και για το χειρισμό των κινούμενων γραφικών.

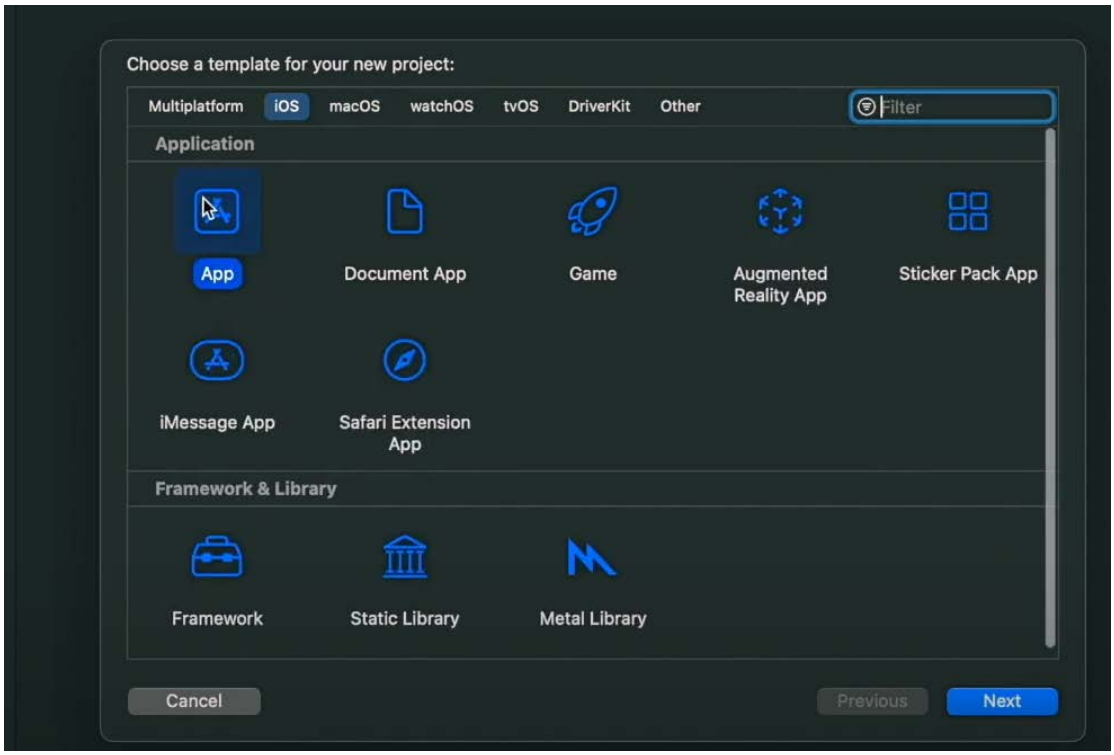
viewWillDisappear: Παρόμοια με τη μέθοδο viewWillAppear, αυτή η συνάρτηση εκτελείται ακριβώς πριν εξαφανιστεί ένα αντικείμενο UIViewController από την

οθόνη. Όπως το viewWillAppear,αυτή η συνάρτηση μπορεί να εκτελεστεί πολλές φορές κατά τη διάρκεια του κύκλου ζωής ενός αντικειμένου UIViewController. Λειτουργεί όταν ο Χρήστης θέλει να μεταβαίνει σε διαφορετική σελίδα.

viewDidDisappear: Εκτελείται αυτή η συνάρτηση όταν έχει ολοκληρώσει τις λειτουργίες του ένα αντικείμενο τύπου UIViewController. Τις διεργασίες που δεν θέλω να εκτελούνται στο παρασκήνιο κάνοντας υπερφόρτωση μεθόδων (Override)καθορίζουμε την οριστική λήξη τους.

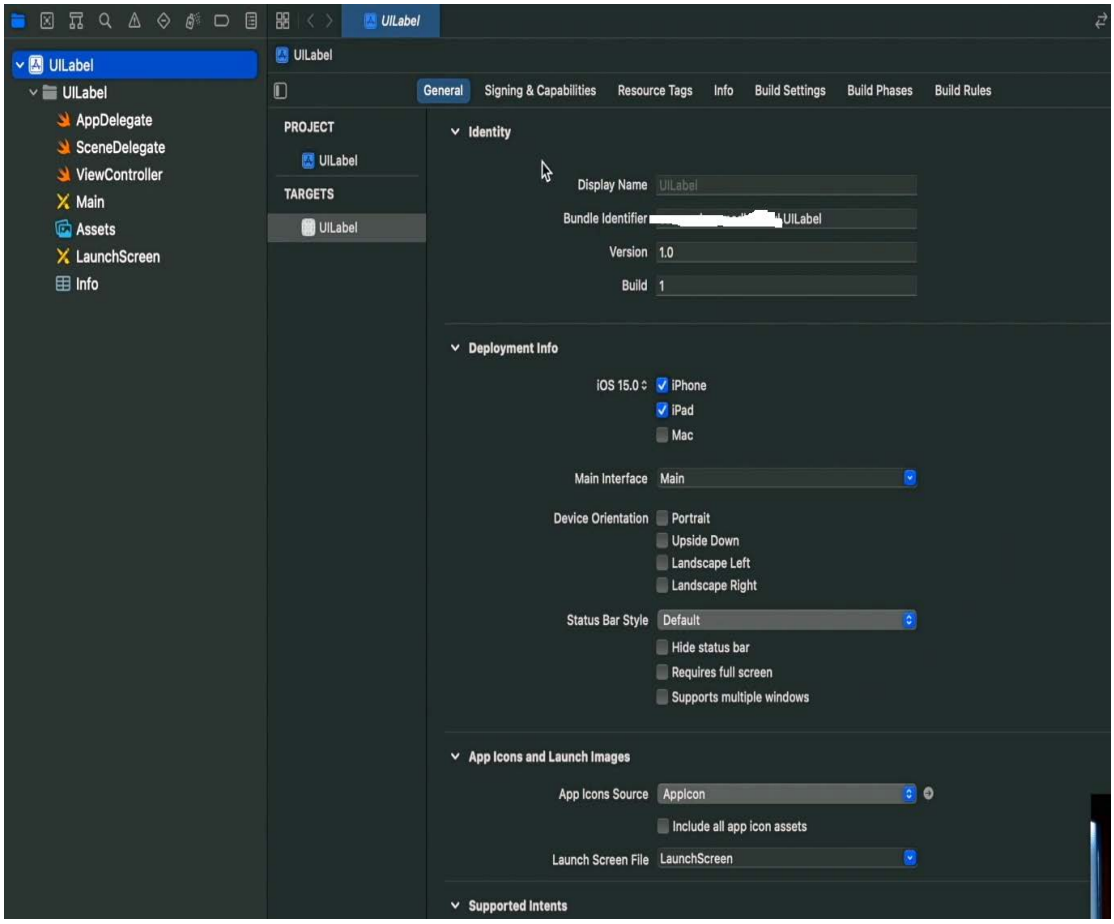
4. Υλοποίηση εφαρμογής

4.1. Δημιουργία της εφαρμογής μας



Εικόνα 4

Για την δημιουργία της εφαρμογής μας επιλέγουμε το πεδίο “IOS” ->App και μας εμφανίζεται η φόρμα που εμφανίζεται στην παρακάτω εικόνα 3.

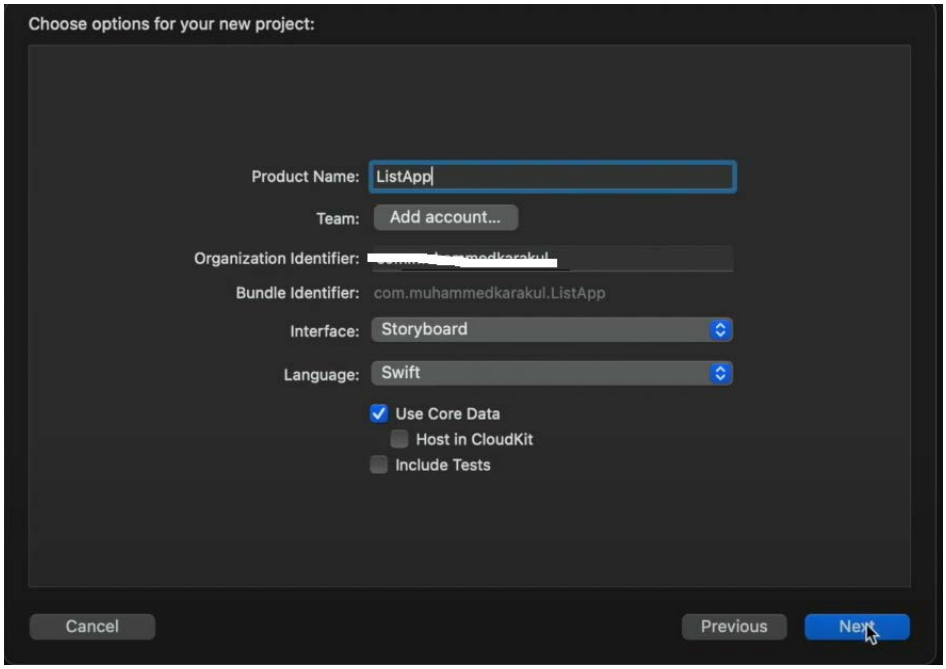


Εικόνα 5

Display Name: ονομασία της εφαρμογής μας

Builder Identifier: το μονοπάτι του δημιουργού.

Version:1.0



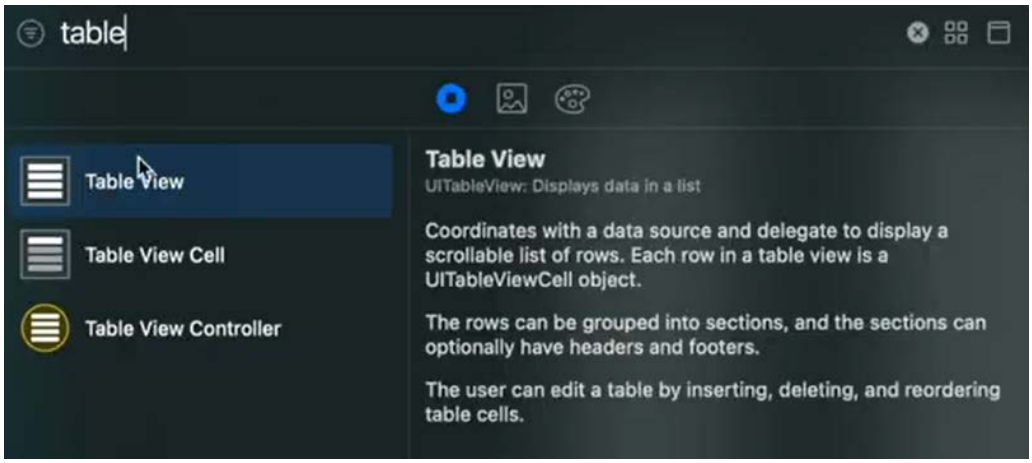
Εικόνα 6

Product Name: όνομα προϊόντος

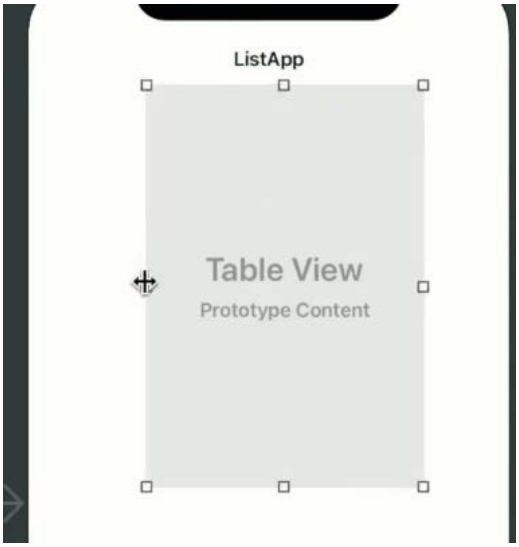
Interface: κανονική διεπαφή

Language: σε τι γλώσσα θα αναπτυχθεί η εφαρμογή.

4.2. Εμφάνιση των δεδομένων μέσα σε μία λίστα (UITableView)



Εικόνα 7



Εικόνα 8

Προσθέτουμε στην γραφική διεπαφή της εφαρμογής μας τον πίνακα για εμφάνιση των δεδομένων.

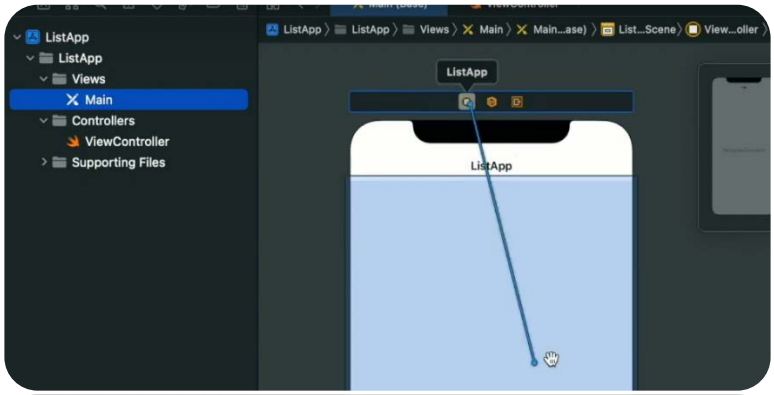
```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loa
    }


}
```

Εικόνα 9

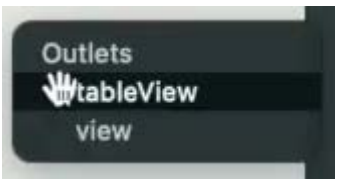
1. @IBOutlet wek var tableview:UITableView!

Προσθέτουμε τον παραπάνω κώδικα στην κλάση μας και ορίζουμε την μεταβλητή για τον πίνακα στοιχείων κάνοντας την κατάλληλη συνδεσμολογία με τον πίνακα που είχαμε ορίσει στο πεδίο “Views” για την εμφάνιση των δεδομένων στον χρήστη με δυναμικό τρόπο.

Στην εικόνα 25 βλέπετε την διαδικασία την συσχέτισης των γραφικών εργαλείων με τους μεταβλητές στην κλάση View Controller.Σέρνοντας την γραμμή πάνω στον πίνακα που είχαμε σχεδιάσει νωρίτερα εμφανίζεται παρακάτω εμφανίζεται

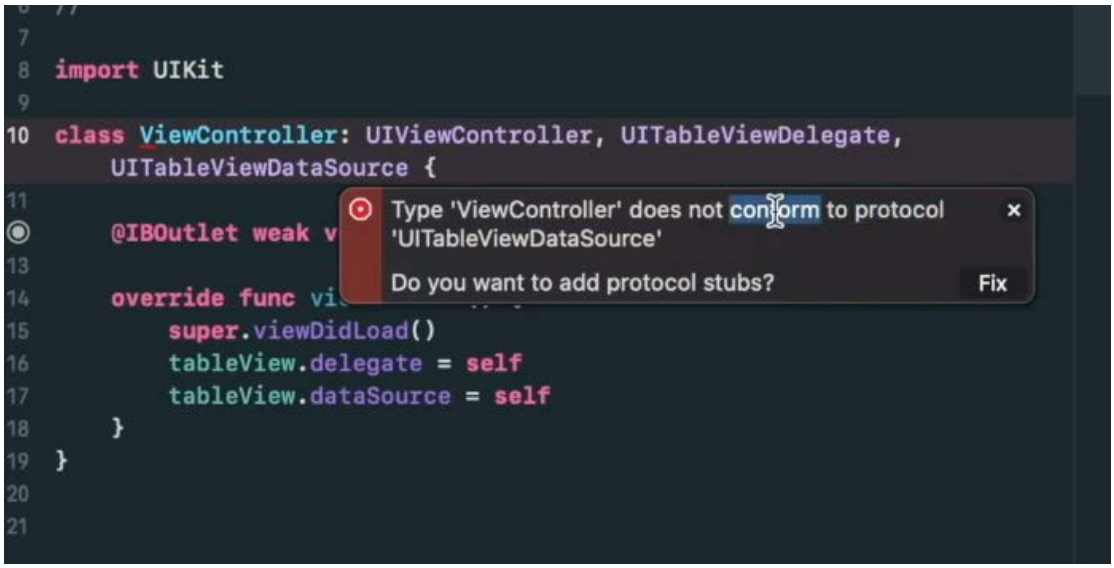


Εικόνα 10



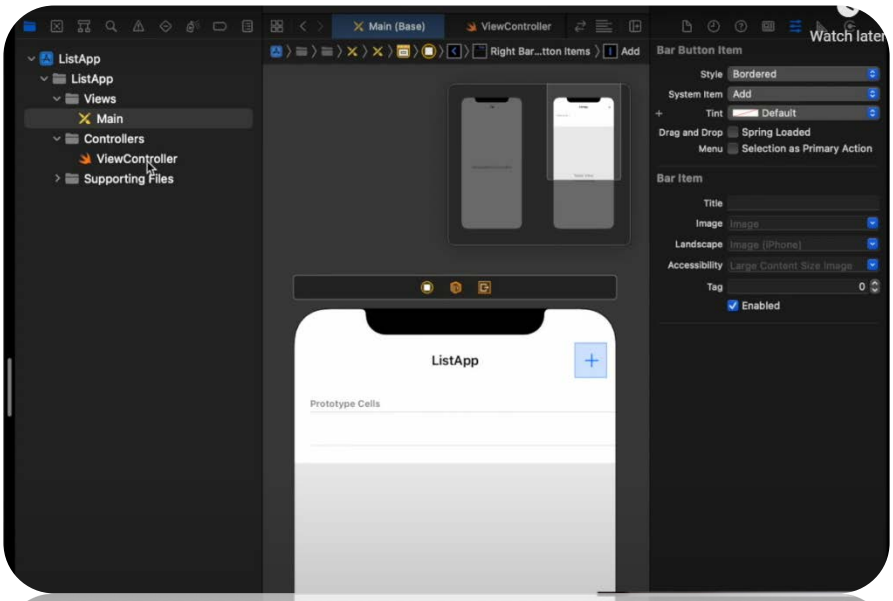
Εικόνα 11

Επιλέγουμε την μεταβλητή που είχαμε ορίσει μέσα στην κλάση ViewController.



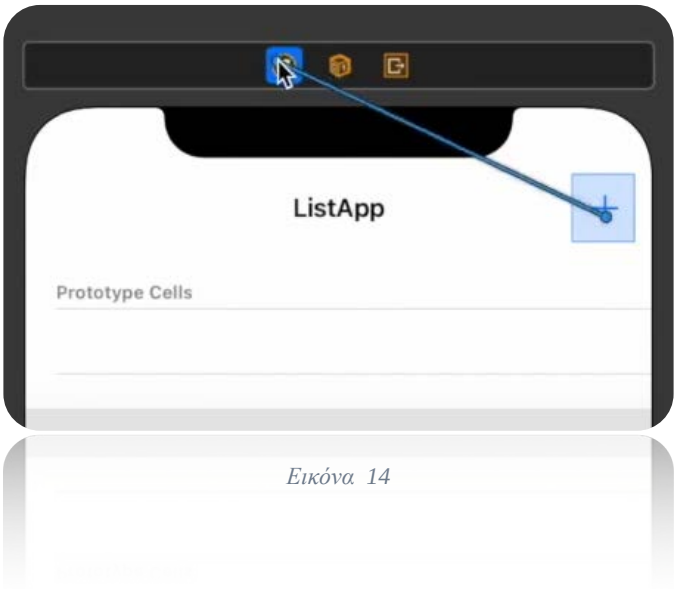
Εικόνα 12

Προσθέτουμε το κουμπί (barbuttonItem) στη γραμμή πλοήγησης(navigation bar) και συσχετίζουμε με την μεταβλητή που είχαμε ορίσει στην κλάση ViewController.



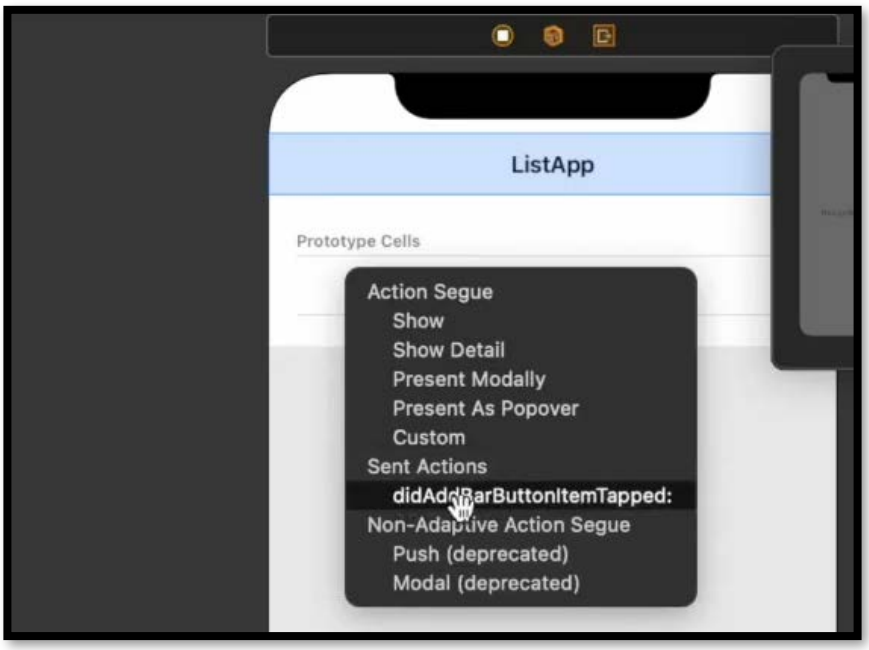
Εικόνα 13

Σέρνοντας το βελάκι που αρχίζει από τον ελεγκτή της γραφικής διεπαφής επάνω στο κουμπί της μπάρας ολοκληρώνουμε την συσχέτιση τους.



Εικόνα 14

Με το που ολοκληρώνω την διαδικασία της συσχέτισης εμφανίζονται οι μέθοδοι που θα δώσει κάποια λειτουργία στο κουμπί μας, επιλέγοντας την κατάλληλη μέθοδο ολοκληρώνουμε την διαδικασία της ανάθεσης.

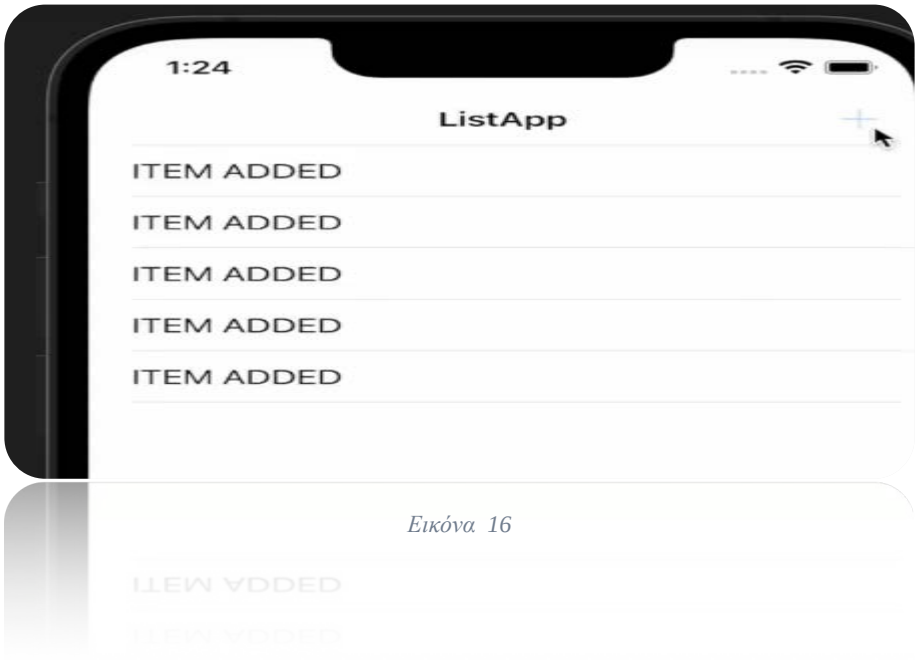


Εικόνα 15

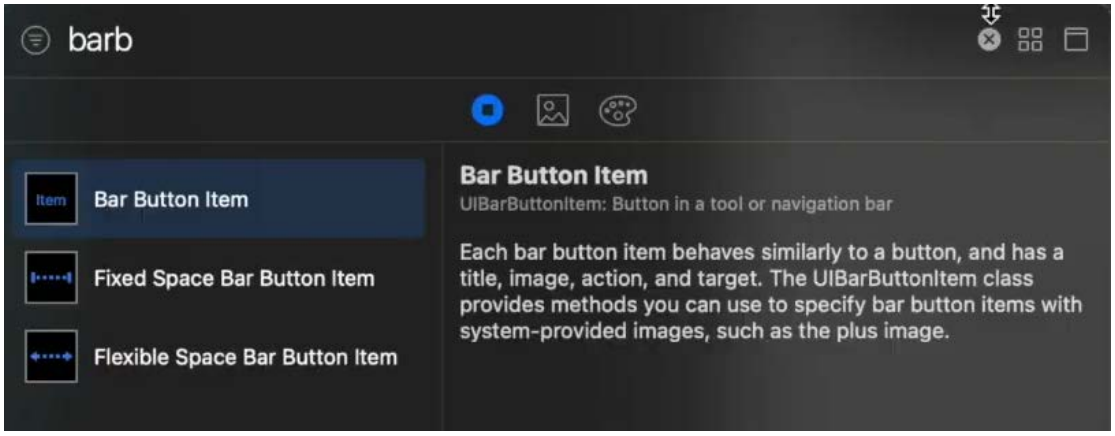
```
1. @IBAction func didAddBarButtonItemTapped(_ sender: UIBarButtonItem)
2.     {
```

```
3.              data.append("ITEM ADDED")
4.              tableView.reloadData()
5.
6.      }
```

Παρακάτω βλέπεται τα αποτελέσματα της υλοποίησης μας.

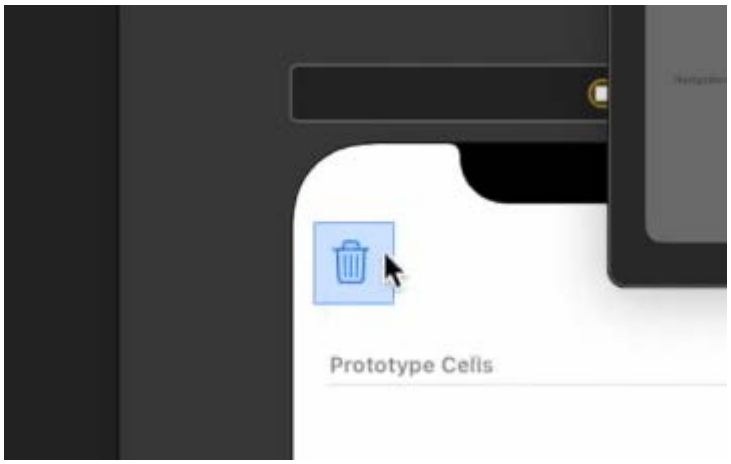


Εικόνα 16

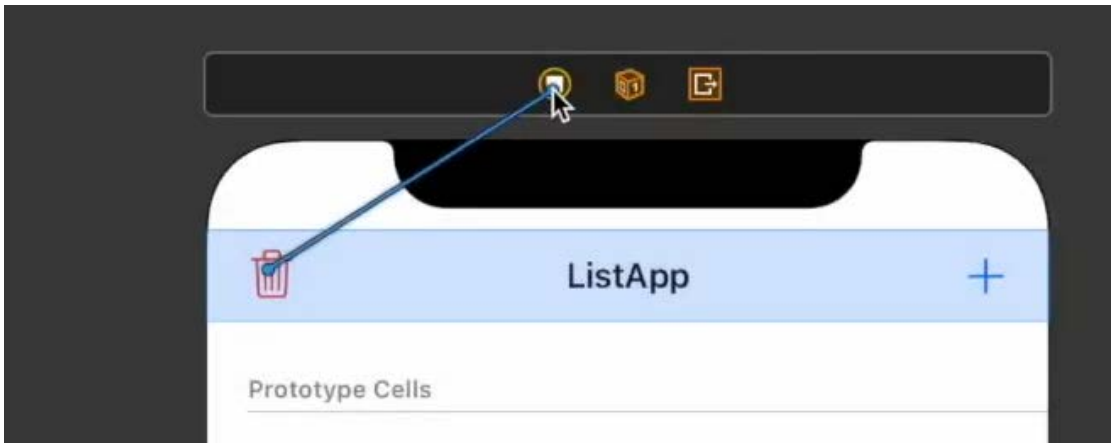


Εικόνα 17

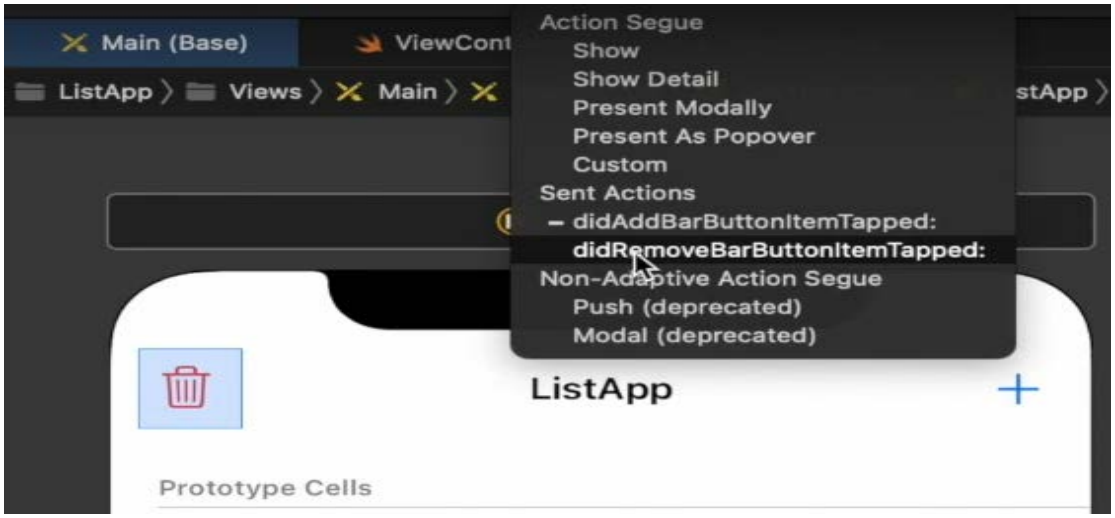
Εικόνα 18



Εικόνα 19



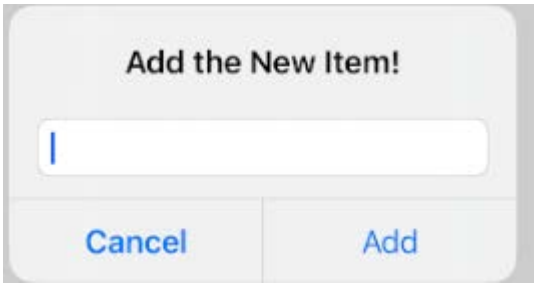
Εικόνα 20



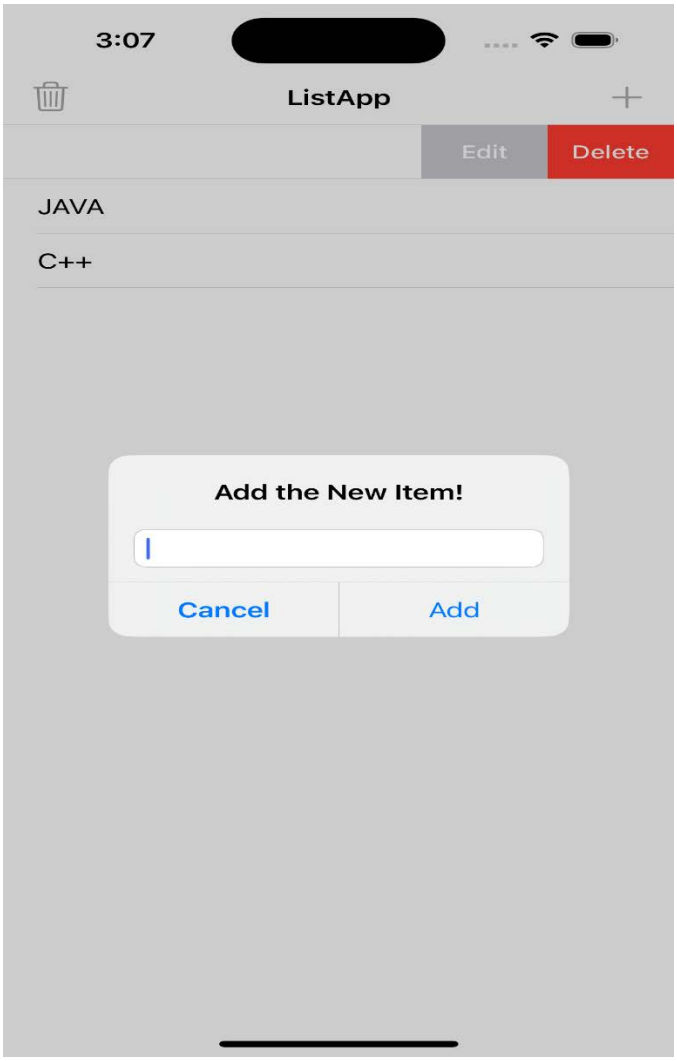
Εικόνα 21

```
1.  @IBAction func didBarButtonRemove(_sender: UIBarButtonItem)
2.  {
3.                  presentAlert(title:"Warning!",
4.                                                  message:"Do you want to delete the
    all items in the list?",
5.                                                  defaultButtonTitle:"Yes",
6.                                                  cancelButtonTitle:"No")
7.                                                  {
8.                                                      //Διαγράφη όλα τα δεδομένα της
    λίστας
9.                                                    self.data.removeAll()
10.                                                     //φορτώνει εκ νέου τον καινούριο
    πίνακα
11.                                                   self.tableView.reloadData()
12.                                                 }
13.
14.
15. }
```

4.3. Προσθήκη των δεδομένων στη λίστα της εφαρμογής μας

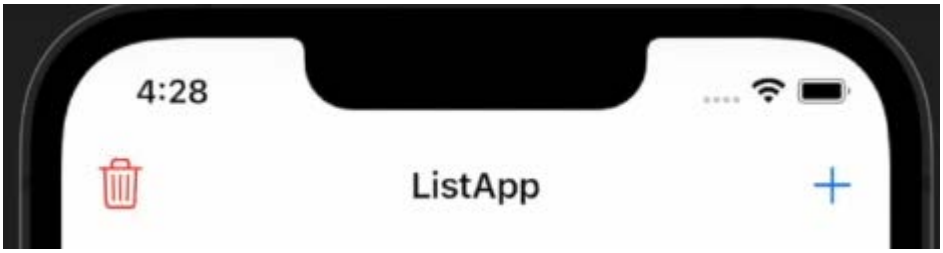


Εικόνα 22



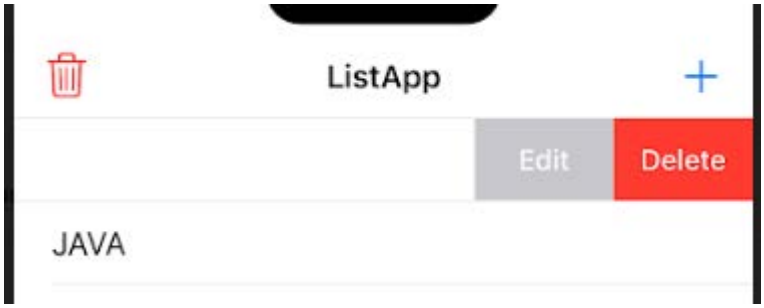
Εικόνα 23

4.4. Διαγραφή και τροποποίηση των δεδομένων μέσα στη λίστα



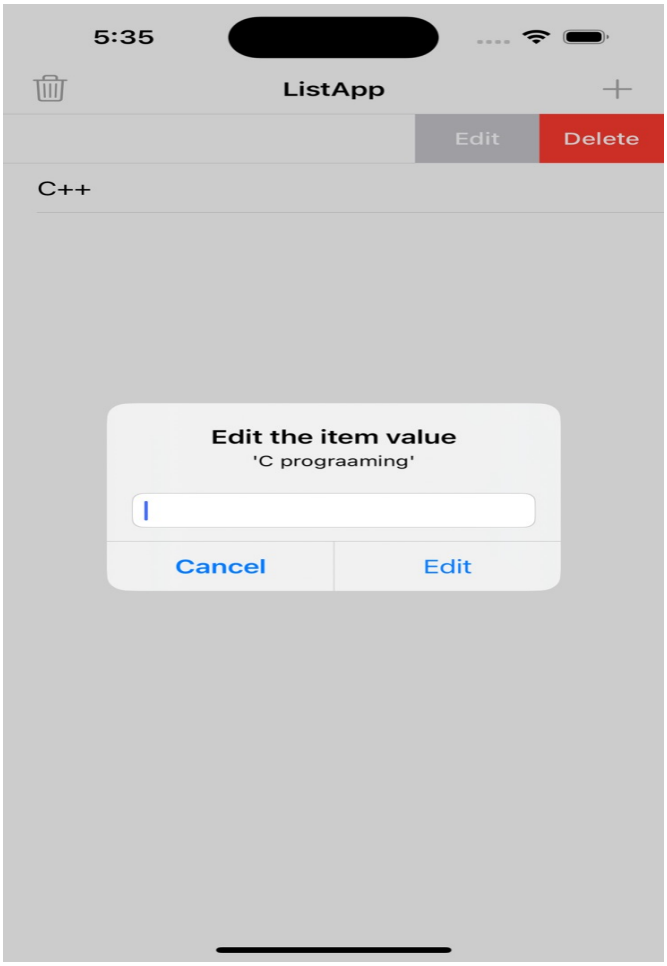
Εικόνα 24

Στο σημείο αυτό σε κάθε πεδίο(cell) της λίστας θέλω να προσθέσω τις λειτουργίες της διαγραφής και τροποποίησης των δεδομένων.



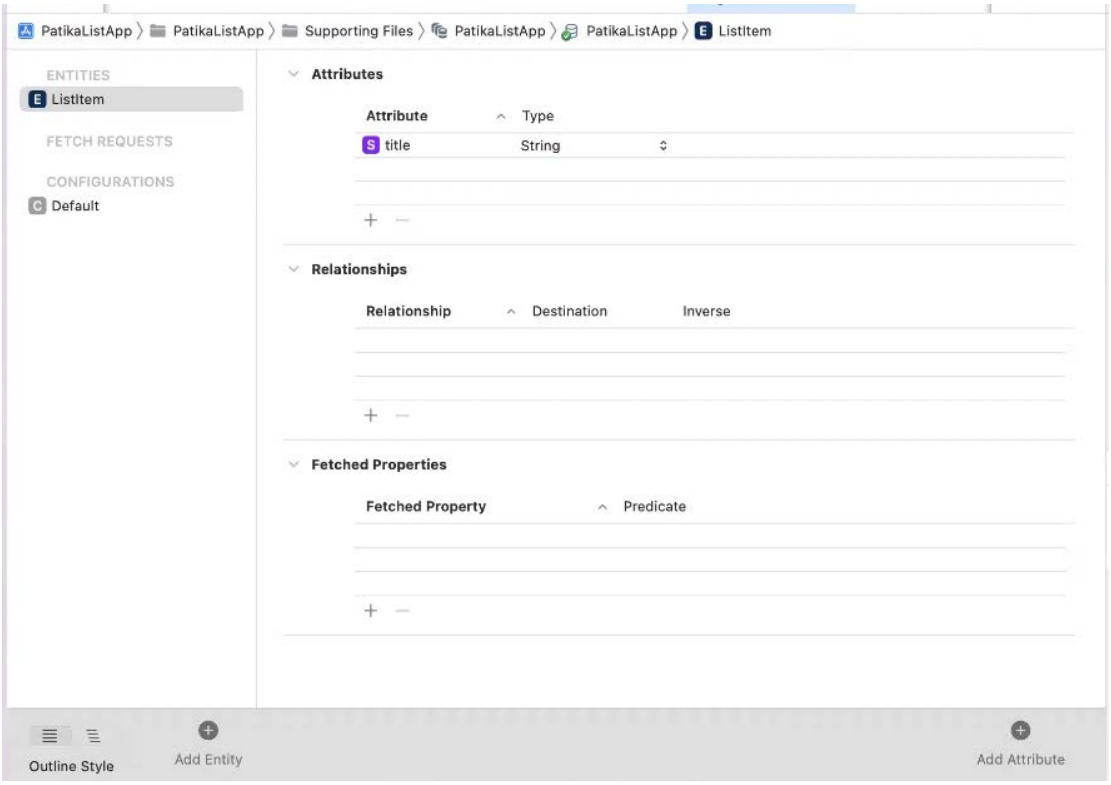
Εικόνα 25

Πατώντας στο κουμπί της τροποποίησης ο χρήστης μπορεί να τροποποιήσει τα ήδη υπάρχοντα δεδομένα στη λίστα και να τα αποθήκευση η να τα διαγράψει..



Εικόνα 26

4.5. Αποθήκευση των δεδομένων στην τοπική βάση (CoreData)



Εικόνα 27

PatikaListApp είναι η ονομασία της τοπικής μας βάσης μας. Η κάθε οντότητα η ο κάθε πίνακας που χρησιμοποιείται για την αποθήκευση των δεδομένων, στις τοπικές βάσεις δεδομένων αντιστοιχούν σε μια Entity (παρόμοια με κλάση στην αντικειμενοστραφή τεχνολογία).

5. **Μελλοντικές βελτιώσεις.**

SwiftUI:

“ Το storyboard είναι μια οπτική αναπαράσταση της διεπαφής χρήστη μιας εφαρμογής iOS, που δείχνει οθόνες περιεχομένου και τις συνδέσεις μεταξύ αυτών των οθονών**”**

Η τεχνολογία SwiftUI μας βοηθά να αναπτύξουμε σύγχρονες οπτικές διεπαφές για τις κινητές εφαρμογές Ios.

- Λιγότερος κώδικας περισσότερο έργο
- Συνδυαζόντας μικρά αντικείμενα κάνουμε πιό ευαναγνωστό κώδικα.
- Εκτέλεση της εφαρμογής σε όλα τα προίοντα της Apple ανεξαρτήτος της συσκευής.
- Βοηθάει στην ανάπτυξη του καθαρού κώδικά αποφέυγοντας την επαναχρησιμοποίηση του κώδικα.
- Λήψη άμεσων αποτελεσμάτων χωρίς μεταγλώττιση του κώδικα στη συσκευή και διευκόλυνση της δοκιμής.
- Εκτελείται πιο γρήγορα από το επίπεδο UIKit και μεταγλωττίζεται πιο γρήγορα στη συσκευή
- Κάνοντας τις λειτουργίες κινούμενων εικόνων και διεπαφής πολύ εύκολες

Firebase:

Είναι η πλατφόρμα της Google που παρέχει πολλά πλεοντεκτήματα στους προγραμματιστές ανάπτυξης εφαρμογών παρέχοντας έτοιμες βιβλιοθήκες διευκολύνει κάποιες λειτουργίες της εφαρμογής σε αρκετό επίπεδο,όπως ταυτοποίηση των χρηστών είτε αποθήκευση των δεδομένων σε μια απομακρυσμένη βάση.

Πάμε να δούμε τώρα με λεπτομέρεια το τι μας παρέχει αυτή η προσθήκη που θα έχουμε μελλοντικά στην εφαρμογή μας:

Δεν έχουμε την ανάγκη για Domain και Hosting για αποθήκευση η για προσπέλαση των δεδομένων για πολλάπλους χρήστες.	To Firebase μας παρέχει την δυνατότητα της αποθήκευσης και προσπέλασης των δεδομένων από ένα διακομιστή(Server) χωρίς να έχουμε κάποια χρηματική επιβάρυνση.
Για την αποθήκευση των δεδομένων χρειάζεται να έχουμε μια από τις τεχνολογίες των σχεσιακών βάσεων δεδομένων.Χρειάζεται να γνωρίζουμε τη γλώσσα προγραμματισμού SQL	Δεν έχουμε την ανάγκη για την ύπαρξη μιας βάσης δεδομένων η να γνωρίζει ο προγραμματιστής τη γλώσσα της βάσεις δεδομένων.
Ο Προγραμματιστής πρέπει να γνωρίζει κάποια γλώσσα προγραμματισμού από τη μεριά του διακομιστή για την αποστολή των δεοδμένων στον τελικό Χρήστη.	Διαθέτει έτοιμες μεθόδους για την αποστολή των δεδομένων οπότε δεν χρειάζεται ο προγραμματισής να γράψει εκστρα κώδικα για την αναπαράσταση δων δεδομένων.
Τα στοιχεία του Χρήστη της εφαρμογής πρέπει να υπάρχει στην τοπική βάση.	Μετά την ενσωμάτωση των βιβλιοθηκών της Firebase αναγνωρίζεται αυτόματα τον χρήστη.
Ο προγραμματιστής πρέπει να κάνει κατάλληλες βελτιώσεςι στον κώδικα για την περίπτωση αλλαγής δεδομένων στην τοπική βάση.	Δεν χρειάζεται κάποια εκστρα εργασία για την ενημέρωση των αλλαγών στα δεδομένα μας.

```
1.  //
2.  //  ViewController.swift
3.  //  PatikaListApp
4.
5.
6.
7.  import UIKit
8.  import CoreData
9.
10.
11. class ViewController: UIViewController {
12.
13.     //Ta dedomema mas  emfanizontai mesa sth grafikh diepafi mesw enos pinaka
14.     //O pinakas TableView sto periballon ios exei duo methodous poy prepei na ylopoihsoyme mesa sto
    controlller mas
15.     //epishs mesa sthn metodo viewDidLoad() prepei na arxikoipoihsoume tous duo metablhtes.
16.     //Delegate:katorizei thn katastash tou pinaka.Orizei an exei ginei kapoia energeia apo th meria tou
    xrhsth
17.     //opws pathma sto pinaka h na kanei kinhsh panw katw.
18.     //Datasource:Ka8orizei thn phgh twn dedomewn gia ton pinaka.Dhlwnei posa cell 8a exoume mesa
    ston pinaka.
19.
20.     @IBOutlet weak var tableView: UITableView!
21.
22.     var data = [NSManagedObject]()
23.     var alertController = UIAlertController()
24.
25.
26.     override func viewDidLoad() {
27.         super.viewDidLoad()
28.
29.         tableView.delegate = self//Dhlwnoume oti oi metablhtes autes anhkoun sthn klash auth
30.         tableView.dataSource = self
31.
32.         fetch()
33.
34.     }
35.
```

```
36.     //Einai h sunarthsh pou diagrafei ola ta stoixeia tou pinaka.
37.     @IBAction func didRemoveBarButtonItemTapped(_ sender: Any) {
38.         //kalw thn synarthsh presentAlert gia na emfanisei ston xrhsth ena parathro an einai sigouros gia thn
39.                     //energeia pou paei  na kanei.
40.                     presentAlert(title:"Warnıng",
41.                 message:"Are you sure do you want to delete the all items on the list?",
42.                 defaultButtonTitle: "Yes",
43.                 cancelButtonTitle: "No") { _ in
44.
45.             let appDelegate = UIApplication.shared.delegate as? AppDelegate
46.             //sto sthmeio auto exoyme thn topikh mas bash
47.             let managedObjectContext = appDelegate?.persistentContainer.viewContext
48.
49.             for item in self.data {
50.                                                 //Diagrafoume to stoixoeio tou pinaka pou epelekse o xrhsths
51.                 managedObjectContext?.delete(item)
52.             }
53.
54.             try? managedObjectContext?.save()
55.
56.             self.fetch()
57.         }
58.
59.     }
60.     //Einai h synarthsh poy ekteleitai otan o xrhshths paei na prosthesei ena
61.     //kainourio stoixeio panw sto pinaka.
62.     @IBAction func didAddBarButtonItemTapped(_ sender: Any) {
63.         presentAddAlert()
64.     }
65.
66.     //Einai o dialogos pou emfanizetai ston xrhshth
67.     // gia na eisagei to keimeno pou paei na pros8esh sth lista.
68.     //Sto parathro emfanizetai epilogh prosthkh  kai akuro.
69.     func presentAddAlert() {
70.         presentAlert(title: "Add the New Item!",
71.                 message: nil,
72.                 defaultButtonTitle: "Add",
73.                 cancelButtonTitle: "Cancel",
74.                 isTextFieldAvailable: true,
75.                 defaultButtonHandler: { _ in
76.             let text = self.alertController.textFields?.first?.text
77.                                 //tesekaroume an o xrhsths exei shmplhrwsei to pedio keimenou swsta h to
    exei afhsei adeio.
78.             //an exei eisagei kati tote ekteleitai o akoulouthos kwdikas,diaforetika emfanizetai mhnhma
    sfalmatos
79.                                 // oti den einai efikto auto pou paei na kanei.
80.                                 if text != "" {
```

```
81.
82.             let appDelegate = UIApplication.shared.delegate as? AppDelegate
83.
84.             let managedObjectContext = appDelegate?.persistentContainer.viewContext
85.
86.             let entity = NSEntityDescription.entity(forEntityName: "ListItem",
87.                                                     in: managedObjectContext!)
88.
89.             let listItem = NSManagedObject(entity: entity!,
90.                                            insertInto: managedObjectContext)
91.
92.             listItem.setValue(text, forKey: "title")
93.
94.             try? managedObjectContext?.save()
95.
96.             self.fetch()
97.           } else {
98.             self.presentWarningAlert()
99.           }
100.      })
101.  }
102.
103.  //Mhnhma sfalmatos pou emfanizetai ston xrhsth otan den eisagei kati sto pedio keimeno.
104.  func presentWarningAlert() {
105.     presentAlert(title: "Warning!",
106.                  message: "The list item cannot  be empty!",
107.                  cancelButtonTitle: "OK")
108.  }
109.
110.
111.  func presentAlert(title: String?,
112.                    message: String?,
113.                    preferredStyle: UIAlertController.Style = .alert,
114.                    defaultButtonTitle: String? = nil,
115.                    cancelButtonTitle: String?,
116.                    isTextFieldAvailable: Bool = false,
117.                    defaultButtonHandler: ((UIAlertAction) -> Void)? = nil) {
118.
119.     alertController = UIAlertController(title: title,
120.                                         message: message,
121.                                         preferredStyle: preferredStyle)
122.
123.     if defaultButtonTitle != nil {
124.       let defaultButton = UIAlertAction(title: defaultButtonTitle,
125.                                         style: .default,
126.                                         handler: defaultButtonHandler)
```

```
127.            alertController.addAction(defaultButton)
128.        }
129.
130.        let cancelButton = UIAlertAction(title: cancelButtonTitle,
131.                                         style: .cancel)
132.
133.        if isTextFieldAvailable {
134.            alertController.addTextField()
135.        }
136.
137.        alertController.addAction(cancelButton)
138.
139.        present(alertController, animated: true)
140.    }
141.
142.
143.    func fetch() {
144.        let appDelegate = UIApplication.shared.delegate as? AppDelegate
145.
146.        let managedObjectContext = appDelegate?.persistentContainer.viewContext
147.
148.        let fetchRequest = NSFetchRequest<NSManagedObject>(entityName: "ListItem")
149.
150.        data = try! managedObjectContext!.fetch(fetchRequest)
151.
152.        tableView.reloadData()
153.    }
154.
155. }
156.
157.
158. extension ViewController: UITableViewDelegate, UITableViewDataSource {
159.
160.    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
161.        return data.count
162.    }
163.
164.
165.    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
    {
166.        let cell = tableView.dequeueReusableCell(withIdentifier: "defaultCell", for: indexPath)
167.        let listItem = data[indexPath.row]
168.        cell.textLabel?.text = listItem.value(forKey: "title") as? String
169.        return cell
170.    }
171.
```

```
172.    func tableView(_ tableView: UITableView, trailingSwipeActionsConfigurationForRowAt indexPath: IndexPath) -> UISwipeActionsConfiguration? {
173.
174.        let deleteAction = UIContextualAction(style: .destructive,
175.                                              title: "Delete") { _, _, _ in
176.
177.            let appDelegate = UIApplication.shared.delegate as? AppDelegate
178.
179.            let managedObjectContext = appDelegate?.persistentContainer.viewContext
180.
181.            managedObjectContext?.delete(self.data[indexPath.row])
182.
183.            try? managedObjectContext?.save()
184.
185.            self.fetch()
186.
187.        }
188.
189.        let editAction = UIContextualAction(style: .normal,
190.                                            title: "Edit") { _, _, _ in
191.            self.presentAlert(title: "Edit the item value",
192.                              message: "'\(self.data[indexPath.row].value(forKey: "title")!)'", //Eleman düzenleme alert ekranında düzenlenecek elemanı gösterir
193.                              defaultButtonTitle: "Edit",
194.                              cancelButtonTitle: "Cancel",
195.                              isTextFieldAvailable: true,
196.                              defaultButtonHandler: { _ in
197.                let text = self.alertController.textFields?.first?.text
198.                if text != "" {
199.                    let appDelegate = UIApplication.shared.delegate as? AppDelegate
200.
201.                    let managedObjectContext = appDelegate?.persistentContainer.viewContext
202.
203.                    self.data[indexPath.row].setValue(text, forKey: "title")
204.
205.                    if managedObjectContext!.hasChanges {
206.                        try? managedObjectContext?.save()
207.                    }
208.
209.                    self.tableView.reloadData()
210.                } else {
211.                    self.presentWarningAlert()
212.                }
213.            })
214.        }
215.
216.        let config = UISwipeActionsConfiguration(actions: [deleteAction, editAction])
```

```
217.
218.        return config
219.    }
220.
221. }
```

Βιβλιογραφικές Αναφορές

1) Anderson, F. (2006). *Step Into Xcode: Mac OS X Development*. Addison-Wesley Professional.

2) Annuzzi, J., Darcey, L., & Conder, S. (2014). *Introduction to Android application development: Android essentials*. Pearson Education.

3) Brunner, S., Blöchlinger, M., Toffetti, G., Spillner, J., & Bohnert, T. M. (2015, December). Experimental evaluation of the cloud-native application design. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)* (pp. 488-493). IEEE.

4) Charland, A., & Leroux, B. (2011). Mobile application development: web vs. native. *Communications of the ACM*, *54*(5), 49-53.

5) Danielsson, W. (2016). React Native application development. *Linköpings universitet, Swedia*, *10*(4), 10.

6) Guo, C. J., Sun, W., Huang, Y., Wang, Z. H., & Gao, B. (2007, July). A framework for native multi-tenancy application development and management. In *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)* (pp. 551-558). IEEE.

7) Holla, S., & Katti, M. M. (2012). Android based mobile application development and its security. *International Journal of Computer Trends and Technology*, *3*(3), 486-490.

8) Javed, M., & Estep, M. (2019, December). Teaching undergraduate software engineering: Xcode mobile App development during dedicated lab periods. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 843-848). IEEE.

9) Kratzke, N., & Peinl, R. (2016, September). Clouns-a cloud-native application reference model for enterprise architects. In *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)* (pp. 1-10). IEEE.

10) Kratzke, N., & Quint, P. C. (2017). Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study. *Journal of Systems and Software*, *126*, 1-16.

11) Malhotra, R., Kumar, D., & Gupta, D. P. (2020). An android application for campus information system. *Procedia Computer Science*, *172*, 863-868.

12) Niveditha, V. R., & Ananthan, T. V. (2019). Improving Acknowledgement in Android Application. *Journal of Computational and Theoretical Nanoscience*, *16*(5-6), 2104-2107.

13) Piper, I. (2010). *Learn Xcode tools for Mac OS X and iPhone development*. Apress.

14) Sarkar, A., Goyal, A., Hicks, D., Sarkar, D., & Hazra, S. (2019, December). Android application development: a brief overview of android platforms and evolution of security systems. In *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)* (pp. 73-79). IEEE.

15) Sihag, V., Vardhan, M., & Singh, P. (2021). A survey of android application and malware hardening. *Computer Science Review*, *39*, 100365.

16) Teng, C. C., & Helps, R. (2010, April). Mobile application development: Essential new directions for IT. In *2010 Seventh International Conference on Information Technology: New Generations* (pp. 471-475). IEEE.