ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογής GIS σε Swift για iOS συσκευές.

ΣΠΥΡΙΔΩΝ ΓΕΩΡΓΙΟΣ ΣΚΟΡΔΟΣ

(AM: it21544)

Αθήνα, Σεπτέμβριος 2020

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Μιχαλακέλης Χρήστος,
Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Τσερπές Κωνσταντίνος,
Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Κουσιουρής Γεώργιος,
Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Εγώ, ο Σπυρίδων Γεώργιος Σκόρδος δηλώνω υπεύθυνα ότι:

1. Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.
2. Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

Ευχαριστίες

Με αφορμή την παρούσα πτυχιακή εργασία εμβάθυνα στις σύγχρονες τεχνολογίες που χρησιμοποιούνται στον κλάδο του mobile development, απέκτησα γενικές βοηθητικές γνώσεις και ήρθα σε επαφή με ένα σύνολο από ανθρώπους τους οποίους θα ήθελα να ευχαριστήσω για την υποστήριξη που μου παρείχαν σε όλη τη διάρκεια αυτής της προσπάθειας.

Αρχικά θα ήθελα να ευχαριστήσω την ίδια μου την σχολή που μου έδωσε την ευκαιρία μέσω της πρακτικής μου άσκησης να ασχοληθώ σε επαγγελματικό επίπεδο με το mobile Developing.Στην συνέχεια ένα μεγάλο ευχαριστώ οφείλω να πω στον κ. Βασίλη Αλεξάντρωφ ο όποιος με τις γνώσεις του και με πολύ υπομονή με έβαλε στον κόσμο του iOS Developing.

Τέλος θα ήθελα να ευχαριστήσω τον κ. Γιώργο Χατζηθανάση για την άριστη συνεργασία μας και την εμπιστοσύνη που έδειξε στο πρόσωπο μου όπως επίσης και τον κ. Χρήστο Μιχαλακέλη για την πολύτιμη βοήθεια του.

Πίνακας περιεχομένων

Περίληψη

Σκοπός της παρακάτω πτυχιακής εργασίας είναι η δημιουργία μιας εφαρμογής τύπου Geographic Information System για την αποτύπωση του χάρτη του Kitchener,η οποία θα υποστηρίζεται από iOS συσκευές.

Η εφαρμογή χρησιμοποιεί τους χάρτες της Apple και το Kitchener's Map API για την εμφάνιση όλων των απαραίτητων δεδομένων. Ο χρήστης μπορεί να εφαρμόσει πληθώρα γεωγραφικών φίλτρων και επιπέδων πάνω στον χάρτη προκειμένου να δει και να συγκρίνει μοτίβα μιας περιοχής σε σχέση με την σύγχρονη μορφή της Κύπρου. Παράλληλα μπορεί να αποθηκεύσει μια περιοχή προκειμένου να έχει πρόσβαση σε αυτήν όταν βρεθεί εκτός δικτύου.

Η εφαρμογή υλοποιήθηκε με τη χρήση της γλώσσας αντικειμενοστραφούς προγραμματισμού Swift και το iOS SDK στο ολοκληρωμένο περιβάλλον ανάπτυξης Xcode.

Λέξεις κλειδιά: iOS, Swift, Χάρτες Apple, Σύστημα Γεωγραφικών πληροφοριών, Λειτουργία Εκτός Σύνδεσης.

Abstract ή Περίληψη στα Αγγλικά

The objective of the project is to create a Geographic Information System in order to visual the map of Kitchener.

For the purpose of displaying the required data, we use Apple maps and Kitchener's Map API. The user can apply a variety of map layers and filter on the map so that he can reveal and compare the geographic patterns of an area with the current form of Cyprus. Furthermore, he can download an area as a mean to use it when being offline.

The application was implemented with the usage of objected oriented language called Swift and iOS SDK. For the development process the platform used was Xcode IDE.

Keywords: iOS, Swift, Apple Maps, G.I.S, Offline Usage

Κατάλογος Εικόνων

1.Εισαγωγή στο προγραμματισμό για iOS

1.1. Ιστορία του iOS

Ένα από τα πιο διαδεδομένα λειτουργικά συστήματα σήμερα είναι το iOS. Το iOS είναι είναι ένα λογισμικό για κινητά και tablet το οποίο αναπτύχθηκε και διανέμεται από την Apple.

Η πρώτη παρουσίαση της πλατφόρμας iOS έγινε τον Ιανουάριο του 2007 με αφορμή την κυκλοφορία το πρώτου iPhone.Η αρχική του ονομασία ήταν iPhone OS μέχρι και την κυκλοφορία του iPhone Software Development Kit(2008) όπου και μετονομάστηκε σε iOS. Πριν την καθιέρωση του ως iOS η Apple το προωθούσε στην αγορά απλά ως μια έκδοση του macOS που ήταν συμβατή με iPhone. Η τελευταία μέχρι στιγμής έκδοση είναι το iOS 13 ενώ τον Σεπτέμβρη του 2020 αναμένεται να κυκλοφορήσει το iOS 14.

Εικόνα 1. Λογότυπο iOS

Το iOS είναι ουσιαστικά λειτουργικό σύστημα και όπως όλα τα λειτουργικά συστήματα ο ρόλος του είναι να διαχειριστεί τους πόρους ενός υπολογιστικού συστήματος και να παράσχει υπηρεσίες προς τους χρήστες και τις εφαρμογές του (Τσερπές, 2016).Αποτελεί λογισμικό κλειστού κώδικα, πιο συγκεκριμένα η χρήση του λογισμικού είναι εντός περιορισμένου περιβάλλοντος. Αυτό έχει ως αποτέλεσμα η ανάκτηση του πηγαίου κώδικα, η τροποποίηση και αναδημιουργία του λογισμικού να θεωρούνται αδύνατες. Συνήθως απαγορεύονται επίσης η αντιγραφή και διανομή του λογισμικού (είτε δωρεάν είτε επί πληρωμή). Η Apple έχει κατασκευάσει το λογισμικό αποκλειστικά για τις συσκευές της, με αποτέλεσμα να μπορεί να αξιοποιήσει σε μέγιστο βαθμό τους πόρους κάθε συσκευής της αλλά και να διασφαλίσει ένα ασφαλές περιβάλλον απέναντι σε κακόβουλες επιθέσεις προγραμματιστών περιορίζοντας τα κενά ασφαλείας.

1.2. Η γλώσσα προγραμματισμού Objective-C

Το iOS υποστηρίζει 2 βασικές γλώσσες προγραμματισμού, την Objective-C και την Swift.

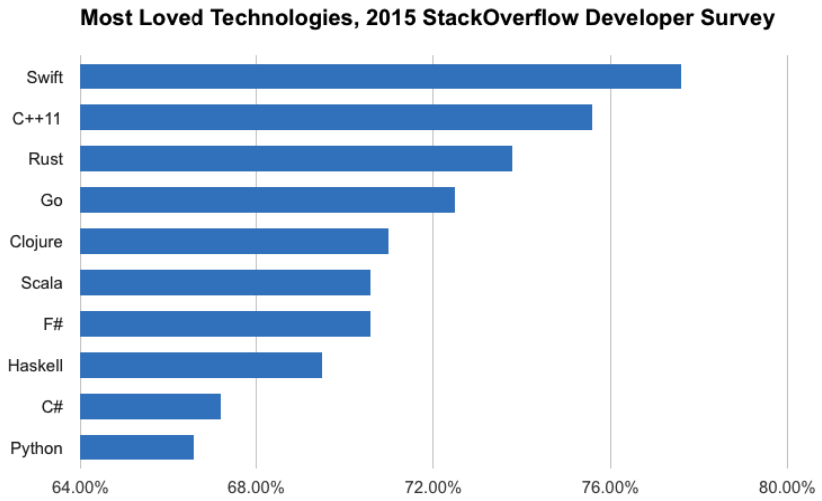
Η Objective-C αναπτύχθηκε το 1984. Χρησιμοποιήθηκε από την εταιρεία NeXT για το λειτουργικό σύστημα NeXTSTEP από το οποίο προέχονται τα λειτουργικά συστήματα MacOS και iOS. Είναι μια γλώσσα προγραμματισμού γενικής χρήσης. Αν και έχει αναπτυχθεί πριν από 30 χρόνια είναι ακόμα ευρέως συμβατή για iOS και Mac OS Development. Λόγω της μακροχρόνια κυκλοφορία της στην αγορά, θεωρείται από πολλούς πιο αξιόπιστη – σταθερή από την Swift.

Είναι η βασική γλώσσα προγραμματισμού που χρησιμοποιήθηκε από την Apple για την υλοποίηση του MacOS, iOS και των αντίστοιχων APIs τους (Cocoa και Cocoa Touch αντίστοιχα) πριν την κυλοφορία της γλώσσας προγραμματισμού Swift. Προγράμματα γραμμένα στην γλώσσα Objective-C που δεν χρησιμοποιούν τις βιβλιοθήκες των Cocoa ή Cocoa Touch, ή που χρησιμοποιούν άλλα μέρη ,είναι συμβατά με άλλα συστήματα και μπορούν να μεταγλωττιστούν από κάθε σύστημα που υποστηρίζει των γνωστό μεταγλωττιστή GCC (GNU Compiler Collection) ή τον Clang. Τα αρχεία πηγαίου κώδικα "υλοποίησης" (implementation files) των προγραμμάτων που είναι γραμμένα σε Objective-C συνήθως έχουν την κατάληξη .m ενώ τα αρχεία "κεφαλίδας/διεπαφής" (header/interface files) έχουν την κατάληξη .h . Πρόκειται για την ίδια κατάληξη που έχουν και τα αρχεία κεφαλίδας/διεπαφής στην γλώσσα προγραμματισμού C.

1.3. Η γλώσσα προγραμματισμού Swift και η ιστορία της

Η Swift είναι μία ανοιχτή γλώσσα αντικειμενοστραφούς προγραμματισμού. Βρίσκεται στις πρώτες θέσεις στην κατηγορία με τις πιο γρήγορα αναπτυσσόμενες γλώσσες προγραμματισμού. Το 2010 άρχισε να αναπτύσσεται από τον Chris Lattner σε συνεργασία με άλλους προγραμματιστές της Apple. Ο σκοπός τους ήταν η δημιουργία μιας μοντέρνας γλώσσας η οποία θα υποστήριζε πολλές βασικές έννοιες που σχετίζονται με την Objective-c αλλά με έναν ασφαλέστερο τρόπο και με βασικό παράγοντα τον περιορισμό των σφαλμάτων και την απλοποίηση της διαδικασία του debug. Το 2014 κυκλοφόρησε η πρώτη επίσημη έκδοση της Swift, η έκδοση 1.1 , πλέον η τελευταία επίσημη κυκλοφορία της είναι η 5.3.

Ένα από τα προβλήματα που είχε να αντιμετωπίσει η Apple πριν την επίσημη κυκλοφορία της Swift ήταν το πώς θα μπορέσει η νέα αυτή γλώσσα να εισχωρήσει στο υπάρχον οικοσύστημα της Objective-c χωρίς να διαταράξει την κοινότητα των mobile developers.Η λύση ήταν να συνεχίσει να συντηρεί την Objective-c κάνοντας παράλληλα μικρά σταθερά βήματα κυκλοφορόντας beta της Swift ώστε να προετοιμάσει το έδαφος για την επίσημη κυκλοφορία. Αξίζει να σημειωθεί ότι η πρώτη beta της Swift είχε πάνω από 11 εκατομμύρια downloads τον πρώτο μήνα κυκλοφορίας. Ύστερα από παραμετροποίηση της γλώσσας σύμφωνα με το feedback των developers η Swift μετά από ένα χρόνο της επίσημης κυκλοφορία της κατάφερε να πάει στην πρώτη θέση όσον αφορά τις πιο αγαπητές γλώσσες προγραμματισμού.



Εικόνα 2. Αποτελέσματα από έρευνα του Stack Overflow to 2015

1.4. Σημαντικά Project γραμμένα σε Swift

Πολλά σημαντικά projects ανοιχτού κώδικα όπως επίσης και εφαρμογών (Applications - apps) που έχουν κατακτήσει τις πρώτες θέσεις στο App Store, είναι γραμμένα σε Swift. Μερικά από αυτά είναι:

- **Flappy iOS App (Github Stars: 9,200):** Η συγκεκριμένη εφαρμογή είναι ένα side controller game που κυκλοφόρησε το 2013 και τον Ιανουάριο του 2014 έκανε τα περισσότερα downloads στο app store. Τον Φεβρουάριο του 2014 αποσύρθηκε από το App Store από τον developer, ισχυριζόμενος ότι είχε ενοχές για τα φαινόμενα εθισμού που προκαλούσε το παιχνίδι. Πλέον διατηρείται στο GitHub από την Fullstackio η υλοποίησή του project σε Swift.

Εικόνα 3. Στιγμιότυπο της εφαρμογής Flappy bird

- **Firefox for iOS (GitHub stars: 9,600):** Το Firefox για iOS είναι ένας φυλλομετρητής ανοιχτού κώδικα για τις φορητές συσκευές της Apple. Είναι ο πρώτος φυλλομετρητής της Firefox που δεν χρησιμοποιεί το Gecko layout engine που υπάρχει στις αντίστοιχες εκδόσεις των mobiles και των desktops. Η πολιτική της Apple απαιτεί στις εφαρμογές που θέλουν να έχουν περιήγηση στο web να χρησιμοποιούν το Webkit framework και Webkit Javascript, έτσι η υλοποίηση της της εφαρμογής με gecko είναι αδύνατη.

Εικόνα 4. Στιγμιότυπο της εφαρμογής Firefox για iOS

- **Facebook:** Η πιο δημοφιλή πλατφόρμα κοινωνικής δικτύωσης στον κόσμο με 42 εκατομμύρια downloads στο app store που ήταν γραμμένη σε Objective-C,μεταφέρθηκε σε Swift.

- **Άλλες διάσημες εφαρμογές που χρησιμοποιούν Swift:**

Εικόνα 5. Εφαρμογές που χρησιμοποιούν την γλώσσα Swift

1.5. Χαρακτηριστικά και πλεονεκτήματα της Swift

Τα χαρακτηριστικά – πλεονεκτήματα για τα οποία έδειξαν και δείχνουν τόσο μεγάλη προτίμηση οι developers στην Swift είναι τα εξής:

- **Ασφάλεια:** Η Swift περιορίζει πολλές κατηγορίες σφαλμάτων ειδικά κατά την διάρκεια εκτέλεσης της εφαρμογής (run time crashes).Ένα από τα πιο συχνά προβλήματα που μπορούν να προκύψουν κατά τον χρόνο εκτέλεσης της εφαρμογής είναι τα σφάλματα μηδενικού δείκτη (null pointer exception).Τα συγκεκριμένα προκαλούν τερματισμό (crash) της εφαρμογής όταν αντικείμενα που περιμένουμε να μην είναι μηδέν τελικά είναι. Με τον όρο optional η Swift μας ενημερώνει εκ των προτέρων εάν ένα αντικείμενο μπορεί να είναι μηδενικό η όχι και άμα είναι αναγκάζει τον προγραμματιστή να χειριστεί επαρκώς το μηδέν. Η ασφαλής αρχικοποίηση ενός αντικειμένου μας εμποδίζει στο να αρχικοποιούμε ένα αντικείμενο με τρόπο που μπορεί να το καταλήξει μηδενικό. Επιπλέον η Swift είναι type-safe που σημαίνει ότι αν καλέσουμε μια μέθοδο σε ένα αντικείμενο που δεν ανταποκρίνεται σε αυτή τότε το σφάλμα θα εντοπιστεί από τον μεταγλωττιστή και δεν θα μας αφήσει να εκτελέσουμε το πρόγραμμα.

- **Ταχύτερος χρόνος ανάπτυξης:** Η Swift είναι σχετικά εύκολή γλώσσα στο γράψιμο και στην ανάγνωση αφού αποτελείται από απλοποιημένη σύνταξη και γραμματική. Είναι αρκετή περιεκτική που σημαίνει ότι λιγότερος κώδικας απαιτείται για την υλοποίηση της ιδίας εργασίας σε σύγκριση με την Objective-C.Χαρακτηριστικό παράδειγμα είναι ότι η εταιρία που έχει υλοποιήσει το Lyft app, αναδημιούργησε σε Swift το project 75 χιλιάδων που ήταν γραμμένο σε Objective-C και η έκταση του καινούργιου ήταν το ένα τρίτο.

- **Κατανοητή σύνταξη της γλώσσας**: Πλησιάζει την φυσική αγγλική με αποτέλεσμα να κάνει ένα project γραμμένο σε Swift πιο προσιτό σε κλιμάκωση και τα μέλη μιας ομάδας προγραμματιστών να μπορούν ενταχθούν πιο εύκολα σε αυτό. Έτσι νέες δυνατότητες – λειτουργίες μπορούν να προστεθούν σ αυτό και η συντήρηση του μακροχρόνια θα είναι πιο εύκολη αν θεωρήσουμε ότι η Apple είναι πιο πιθανό να υποστηρίζει την Swift παρά την Objective-C μετά από κάποια χρόνια.

- **Βελτιωμένη απόδοση:** Με σκοπό να ξεπεράσει την Objective-c η Swift εστιάζει στη ταχύτητα και την απόδοση. Είναι χαρακτηριστικό ότι στην πρώτή κυκλοφορία η Apple ισχυρίστηκε 40% αύξηση της απόδοση σε σχέση με τον προκάτοχο της. Με το πέρασμα του χρόνου, ύστερα από πολλούς δοκιμές και benchmarks από μεμονωμένους προγραμματιστές η παραπάνω θεωρία επιβεβαιώθηκε. Είναι σημαντικό να αναφέρουμε ότι υπάρχουν πολλοί τρόποι παραμετροποίησης του κώδικα που είναι γραμμένος σε Swift για βελτιστοποίηση της απόδοσης.

- **Μειωμένο memory footprint:** Όταν δημιουργούμε μια εφαρμογή είναι πιθανό να χρησιμοποιήσουμε κώδικα τρίτου, frameworks ανοιχτού πηγαίου κώδικα ή βιβλιοθήκες. Αυτές οι βιβλιοθήκες είναι ή στατικές ή δυναμικές. Η Swift όταν κυκλοφόρησέ εισήγαγε πρώτη φορά τις δυναμικές βιβλιοθήκες στο κόσμο του iOS.Οι στατικές βιβλιοθήκες είναι κλειδωμένες στον κώδικα την στιγμή του compile,γίνονται κομμάτι του εκτελέσιμου αρχείου και έτσι αυξάνουν το μέγεθος και τον χρόνο φόρτισης του. Ένα ακόμη μειονέκτημα των στατικών είναι ότι δεν ενημερώνονται αυτόματα αφού παραμένουν στην έκδοση που είχαν την στιγμή του compile.Αντίθετα οι δυναμικές βιβλιοθήκες υπάρχουν εκτός του κώδικα που γράφουμε και μεταφορτώνονται μόνο όταν είναι απαραίτητο. Ακόμη οι στατικές πρέπει να έχουν αντίγραφα σε όλα τα αρχεία του project ενώ η δυναμικές μόνο ένα.

- **Συμβατή με την Objective-C:** Η Swift είναι απόλυτα συμβατή με την Objective-C και μπορεί να χρησιμοποιηθεί ακόμη και ως εναλλακτική στο ίδιο έργο. Για παράδειγμα αν θέλουμε να επεκτείνουμε ή να ενημερώσουμε ένα project που είναι γραμμένο σε Objective-C μπορούμε τα νέα κομμάτια κώδικα να τα γράψουμε σε Swift.Το συγκεκριμένο γεγονός κάνει ακόμη πιο εύκολη την μετάβαση και περιορίζει τον κίνδυνο δημιουργίας bugs.

- **Αυτόματη διαχείριση μνήμης:** Ένα συχνό πρόβλημα που αντιμετωπίζουν οι προγραμματιστές είναι η διαχείριση μνήμης σ ένα project.Η Swift χρησιμοποιεί αυτόματο μετρητή μνήμης (ARC- Automatic Reference) ο οποίος στοχεύει σε μια λειτουργία η οποία δεν είχε εισαχθεί στο iOS πριν, την συλλογή απορριμμάτων – άχρηστων δεδομένων. Γλώσσες όπως η Java και η C# λειτουργούν με τους

λεγομένους garbage collectors προκειμένου να διαγράφουν αναφορές (instances) κλάσεων που δεν χρησιμοποιούνται. Με αυτό το τρόπο επιτυγχάνεται μειωμένη χρήση της μνήμης αλλά και η αύξηση της απόδοσης του επεξεργαστή. Πριν την χρήση του ARC στην Swift, οι iOS προγραμματιστές καλούνταν να διαχειριστούν με χειροκίνητο τρόπο την μνήμη όπως επίσης και να συγκρατούν τον αριθμό κάθε κλάσης που υπήρχε. Πλέον με τον ARC εντοπίζονται αυτόματα ποια instances κλάσεων δεν χρησιμοποιούνται και σβήνονται από την μνήμη. Με αυτό τον τρόπο έχουμε καλύτερη απόδοση χωρίς να αφήνουμε πίσω την μνήμη και τον επεξεργαστή.

- **Προοπτική για full stack development και υποστήριξη cross-device**: Επηρεασμένη από την IBM η τοποθέτηση της γλώσσας σε Cloud περιβάλλον είναι αρκετά επιτυχημένη. Ο διακομιστής Swift ενσωματώνεται στις περισσότερες δημοφιλείς τεχνολογίες backend. Έτσι χρησιμοποιώντας την Swift και για το frontend και για το backend της εφαρμογής δίνεται η δυνατότητα στους developers για εκτεταμένη κοινή χρήση – επαναχρησιμοποίηση του κώδικα που έχει ως αποτέλεσμα τον γρηγορότερο χρόνο ανάπτυξης της. Ακόμη η Apple παρέχει υποστήριξη της γλώσσας προς όλες τις συσκευές της (iPhone,iPad,Mac υπολογιστές).Τα όρια της υποστήριξης δεν σταματάνε εκεί, υπάρχει ήδη συμβατότητα με το λειτουργικό σύστημα Linux όπως επίσης και για Windows πλατφόρμες σε project ανοικτού κώδικα.

- **Ενεργή κοινότητα ανοιχτού κώδικα**:Η Apple με σκοπό να καθιερώσει την Swift ως κύρια γλώσσα προγραμματισμού τα επόμενα 20 χρόνια, έχει εστιάσει στην εισαγωγή της Swift στην open source code κοινότητα. Έτσι με την ισχυρή εταιρική υποστήριξη της IBM η Swift απέκτησε ισχυρή θέση στην κοινότητα.

- **Τεχνολογία σε μεγάλη ζήτηση:**Σύμφωνα με την εταιρεία τοποθέτησης Toptal, η Swift είναι η γλώσσα προγραμματισμού με τη μεγαλύτερη ζήτηση. Πιο συγκεκριμένα υπήρξε αύξηση κατά 600% για ανεξάρτητους προγραμματιστές που γνωρίζουν Swift.

Οι παραπάνω λόγοι αποτέλεσαν σημαντικό παράγοντα στο να επιλέξω την Swift για την υλοποίηση του συγκεκριμένου project.

1.6. Εργαλεία ανάπτυξης και Xcode

Προκειμένου να υλοποιηθεί η εφαρμογή για τις iOS συσκευές είναι απαραίτητο να έχουμε στην διάθεση μας ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Development Environment - IDE).Το λεγόμενο IDE είναι μια εφαρμογή λογισμικού η οποία παρέχει στο προγραμματιστή ένα ολοκληρωμένο πακέτο εργαλείων ώστε να μπορεί να θεωρηθεί δυνατή η ανάπτυξη λογισμικού. Πιο συγκεκριμένα τέτοια εργαλεία μπορούν να θεωρηθούν:

- Κατάλληλα frameworks (δομές στις οποίες μπορεί να βασιστεί ο πηγαίος κώδικας ώστε να επιτύχει το επιθυμητό αποτέλεσμα χωρίς boilerplate και low-level κώδικα.
- Επεξεργαστής πηγαίου κώδικα (Source Code Editor).
- Εργαλείο εντοπισμού σφαλμάτων (Debugger).
- Λογισμικό το οποίο μεταφράζει τον πηγαίο κώδικα που γράφει ο προγραμματιστής, ο οποίος θεωρείται γλώσσα υψηλού επιπέδου, σε γλώσσα μηχανής η οποία μπορεί να αξιοποιηθεί από το hardware ενός υπολογιστή(Compiler)
- Εξομοιωτές (Emulators) οι οποίοι ειδικά στην περίπτωση του mobile development είναι σημαντικοί αφού μέσω αυτών ο προγραμματιστής μπορεί να τρέξει το λογισμικό που γραφεί υπό διάφορες συνθήκες(π.χ. διαφορετικές εκδόσεις συσκευών και λογισμικού, διαφορετικές συνθήκες δικτύου, προσομοίωση τοποθεσίας, προσομοίωση μπαταρίας)
- Εργαλείο δημιουργίας διεπαφών (Interface Builder) , το οποίο είναι ιδιαίτερα σημαντικό στην περίπτωση του mobile development αφού μέσω αυτού γίνεται η σύνδεση του κώδικα με την User Interface Components που βλέπει ο χρήστης.
- Διαγράμματα ιεραρχίας των κλάσεων προκειμένου ο χρήστης να μπορεί να απεικονίσει την δομή του αντικειμενοστραφούς κώδικα προγραμματισμού.
- Περιηγητής δεδομένων προκειμένου να ελέγχονται τα αντικείμενα που δημιουργούνται σ ένα εκτελούμενο πρόγραμμα.

Η πρώτη προσπάθεια για κατασκευή ενός IDE έγινε το 1983 όταν η Borland Ltd απέκτησε έναν μεταγλωττιστή pascal και τον κυκλοφόρησε ως turboPascal ο οποίος διέθετε για πρώτη φορά μεταγλωττιστή και επεξεργαστή κώδικα. Σύμφωνα με πολλές απόψεις το πρώτο πραγματικό IDE ήταν το Visual basic που κυκλοφόρησε το 1991 η Microsoft.Το Visual Basic βασίστηκε στην παλαιότερη γλώσσα Basic, η οποία ήταν δημοφιλής το την δεκαετία του 80, και με την κυκλοφορία του άνοιξε τον δρόμο στον προγραμματισμό για την παραγωγικότητα αλλά και τους γραφικούς ορούς. Κάποιοι από τους λογούς που προτιμούμε το IDE από έναν απλό source code editor είναι:

- Γρήγορη εγκατάσταση και παραμετροποίηση του project
- Γρηγορότερες διαδικασίες προγραμματισμού
- Συνεχής ενημέρωση των τελευταίων υποστηριζόμενων frameworks
- Τυποποίηση των διαδικασιών που βοηθάει στην συνεργασία μεταξύ των developers

Εικόνα 6. Το λογότυπο του Xcode

Το επίσημο IDE για την υλοποίηση iOS εφαρμογών είναι το Xcode. Η πρώτη κυκλοφορία του Xcode ήταν το 2003 και η τελευταία εγκεκριμένη (Stable) έκδοση είναι το Xcode 11.7. Διατίθεται δωρεάν μέσω το Mac App Store και είναι συμβατό μόνο με Mac Operating System.Προκειμένου ένας developer να εκδώσει την εφαρμογή του στο App Store είναι απαραίτητη να έχει Apple developer account(επί πληρωμή).Το Xcode είναι ένα IDE το οποίο συμπεριλαμβάνει τα Cocoa και Cocoa Touch Frameworks (Framework τα οποία περιέχουν κλάσεις υπεύθυνες για την παραγωγή των γραφικών διεπαφών χρήσης(GUI). Επιπλέον διαθέτει intercace builder προκειμένου να γίνεται η σχεδίαση του UI, επεξεργαστή πηγαίου κώδικα,emulators ώστε να μπορεί να προσομοιωθεί η εφαρμογή σε όλες τις iOS συσκευές και σε όλα τα iOS Versions χωρίς την παρουσία του αντίστοιχου hardware και αλλά απαραίτητα εργαλεία όπως compiler και debugger.

Ακόμη μέσω του Xcode γίνεται να προσθέσουμε βιβλιοθήκες στο project μας. Πιο συγκεκριμένα μέσω του Cocoa Pods ο οποίος είναι ένας διαχειριστής εξαρτήσεων (dependency manager) ,συμβατός με Objective – C, Swift και οποιαδήποτε άλλη γλώσσα η οποία εκτελείται σε runtime Objective-C,μπορούμε να εισάγουμε έτοιμα εργαλεία στο project μας τα οποία έχουν κατασκευάσει άλλοι developers για το επιθυμητό αποτέλεσμα πχ network manager.Αλλά γνωστά IDE τα οποία χρησιμοποιούνται για iOS development είναι: το Atom, το Appcode, το CodeRunner 2 και το SublimeText .

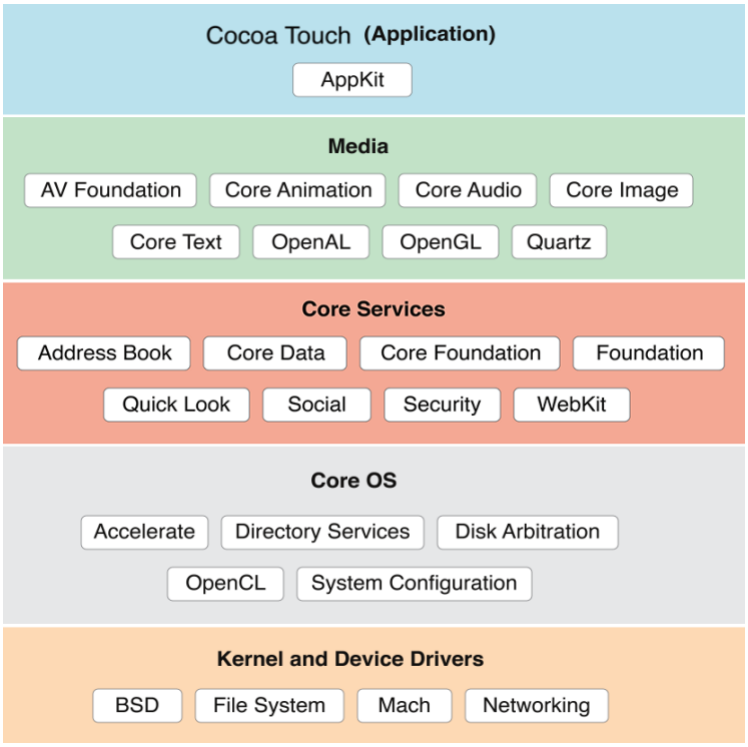
Για την υλοποίηση του συγκεκριμένου project χρησιμοποίησα το Xcode λόγω εμπειρίας πάνω σ αυτό αλλά και γιατί θεωρείται το πιο αξιόπιστο.

1.7. Η Δομή μιας iOS εφαρμογής και η αλληλεπίδραση της με το λειτουργικό σύστημα

1.7.1 Αρχιτεκτονική iOS

Η αρχιτεκτονική του iOS είναι πολυεπίπεδη. Στο ανώτερο επίπεδο το iOS λειτουργεί ως διαμεσολαβητής αναμεσά στο hardware και στην εφαρμογή που έχουμε προγραμματίσει. Οι εφαρμογές δηλαδή δεν επικοινωνούν απευθείας με το hardware αλλά έρχονται σε επαφή με αυτό μέσω μιας συλλογής καλά καθορισμένων διεπαφών συστήματος. Τα κατώτερα επίπεδα παρέχουν τις βασικές υπηρεσίες στις οποίες στηρίζεται μια εφαρμογή και τα ανωτέρα εξελιγμένες υπηρεσίες γραφικών και διεπαφών.

Το λειτουργικό σύστημα iOS είναι μια στοίβα τύπου Unix από δομικά στοιχεία λογισμικού, που προέρχεται απευθείας από την διαδρομή ανάπτυξης του OS X.Μπορεί να διαιρεθεί σε 4 ενότητες όπως φαίνεται στο παρακάτω διάγραμμα.



Εικόνα 7. Διάγραμμα επίπεδων αρχιτεκτονικής στο iOS

Πιο συγκεκριμένα τα επίπεδα είναι:

- **Kernel and Device Drivers:** Το χαμηλότερο επίπεδο του iOS που περιλαμβάνει κυρίως τους drivers,τον πυρήνα και βασίζεται κυρίως σε τεχνολογίες ανοιχτού κώδικα. Το περιβάλλον του πυρήνα είναι χτισμένο πάνω από το Mach 3.0 το οποίο είναι ένας μικροπυρήνας που αντικαθιστά τον πυρήνα στην έκδοση BSD του UNIX.Το περιβάλλον αυτό παρέχει υποδομές και υποστήριξη δικτύων υψηλής απόδοσης για πολλαπλά ολοκληρωμένα συστήματα αρχείων. Το OS X επεκτείνει αυτό το περιβάλλον χαμηλού επιπέδου με βασικές τεχνολογίες υποδομής που διευκολύνει την διαδικασία του προγραμματισμού.

- **Core OS Layer:** Το επίπεδο Core OS αποτελείται από τεχνολογίες και frameworks που παρέχουν υπηρεσίες χαμηλού επιπέδου που σχετίζονται με το hardware και τα δίκτυα. Αυτές οι υπηρεσίες βασίζονται σε υποδομές και σε εγκαταστάσεις στο επίπεδο πυρήνα και σε drivers συσκευών.

- **Core Services Layer:** Οι τεχνολογίες που περιέχει το συγκεκριμένο επίπεδο ονομάζονται βασικές υπηρεσίες γιατί παρέχουν βασικές υπηρεσίες σε εφαρμογές αλλά δεν έχουν άμεσο αντίκτυπο στην διεπαφή του χρήστη με την εφαρμογή. Τέτοιες υπηρεσίες είναι το βιβλίο διευθύνσεων, η ασφάλεια, το Social framework και Foundation framework.Γενικά αυτές οι τεχνολογίες εξαρτώνται από frameworks και τεχνολογίες που βρίσκονται στα δυο χαμηλότερα επίπεδα του iOS.

- **Media Layer:** Το επίπεδο πολυμέσων επιτρέπει στον προγραμματιστή να ενσωματώσει γραφικά 2d και 3d, κινούμενα σχέδια, εφέ και λειτουργίες ήχου και βίντεο επαγγελματικής ποιότητας στην εφαρμογή.

- **Cocoa Touch Layer:** Το συγκεκριμένο επίπεδο είναι το υψηλότερο αφού είναι κυρίως υπεύθυνο για την εμφάνιση των εφαρμογών και την ανταπόκριση τους στις ενέργειες του χρήστη. Παρέχει πρόσβαση σε κυρίες λειτουργίες του συστήματος όπως επαφές, κάμερα, είσοδο αφής, κοινή χρήση με άλλες εφαρμογές , ειδοποιήσεις, λειτουργία πλήρους οθόνης και αυτόματη αποθήκευση δεδομένων.

1.7.2 Σημαντικά iOS Frameworks

Η Apple παρέχει τις διεπαφες του συστήματος σε ειδικά πακέτα τα λεγόμενα frameworks. Το framework είναι ένας κατάλογος ο οποίος περιέχει μια δυναμική κοινόχρηστη βιβλιοθήκη με αρχεία τύπου .a ,κατάλληλους πόρους όπως header files,εικόνες, και βοηθητικές εφαρμογές προκειμένου να υποστηριχθεί αυτή η βιβλιοθήκη. Κάθε επίπεδο έχει ένα σύνολο από frameworks τα οποία μπορεί να χρησιμοποιήσει ο προγραμματιστής για την υλοποίηση της εφαρμογής .Μερικά από αυτά είναι :

- **UIKit:** Το UIKit framework παρέχει την απαραίτητη υποδομή για τις εφαρμογές iOS ή tvOS. Παρέχει την αρχιτεκτονική παραθύρου (window architecture) και προβολής για την υλοποίηση της εφαρμογής, την υποδομή διαχείρισης γεγονότων Multi-Touch και άλλων τύπων εισόδων όπως επίσης και τον κύριο βρόχο εκτέλεσης που απαιτείται για την διαχείριση των αλληλεπιδράσεων μεταξύ του χρήστη, του συστήματος και της εφαρμογής.

  Επιπλέον δυνατότητες που προσφέρει το framework είναι υποστήριξη animation, υποστήριξη εγγράφων, υποστήριξη σχεδίασης και εκτύπωσης ,πληροφορίες σχετικά με την συσκευή , διαχείριση και προβολή κειμένου, υποστήριξη αναζήτησης, υποστήριξη προσβασιμότητας, υποστήριξη επέκτασης εφαρμογών και διαχείρισης πόρων.

- **Foundation:** Το foundation framework είναι παρόμοιο με το UIKit όσον αφορά τον καθορισμό τον κλάσεων γενικής χρήσης. Η βασική διαφορά είναι ότι ενώ το UIKit περιορίζεται σε κλάσεις που εφαρμόζονται στο περιβάλλον διεπαφής με τον χρήστη, το foundation εστιάζει σε όλα τα άλλα πράγματα που δεν αφορούν την άμεση επαφή με τον χρήστη και τα οποία χρειάζεται η εφαρμογή.

  Πιο συγκεκριμένα καθορίζει την συμπεριφορά των αντικειμένων, την διαχείριση της μνήμης, τις ειδοποιήσεις, την τοπικοποίηση και την διεθνοποίηση. Παρέχει βασικές λειτουργίες όπως αποθήκευση και διατήρηση δεδομένων, επεξεργασία κειμένου, υπολογισμούς ημερομηνίας και ώρας, ταξινόμηση, φιλτράρισμα και δικτύωση. Οι κλάσεις, τα πρωτόκολλα και οι τύποι δεδομένων, που ορίζονται από το foundation χρησιμοποιούνται σε όλα τα SDK macOS, watcOS , iOS και tvOS .

- **CoreGraphics:** Το CoreGraphics framework περιέχει τις διεπαφές για το Quartz 2D API σχεδίασης που είναι η ίδια προηγμένη μηχανή σχεδίασης η οποία χρησιμοποιείται με βάση το διάνυσμα που χρησιμοποιείται στο macOS. Παρέχει υποστήριξη για path-based σχεδίαση,anti-aliased rendering,εικόνες, χρώματα, συντεταγμένες μετασχηματισμού χώρου, δημιουργία αρχείου pdf, προβολή και ανάλυση. Παρόλο που το API βασίζεται σε C, χρησιμοποιεί αφαιρέσεις για αφαιρέσεις βάσει αντικειμένων για να είναι πιο προσιτό προς τον προγραμματιστή. Αν και αποτελεί την

βάση για πολλά πράγματα που βλέπει στην οθόνη ο χρήστης δεν χρησιμοποιείται απευθείας από τους προγραμματιστές για πολλά project εφαρμογών.

- **MapKit:** Επιτρέπει στον προγραμματιστή να ενσωματώσει μια πλήρως λειτουργική διεπαφή χάρτη στην εφαρμογή. Το συγκεκριμένο framework παρέχει πολλές από τις λειτουργίες που μπορεί να βρει κάποιος σε μια εφαρμογή χάρτη. Έτσι μπορούμε να χρησιμοποιήσουμε το MapKit για να πάρουμε πληροφορίες για μια τοποθεσία, να ενσωματώσουμε χάρτες απευθείας σε ένα παράθυρο της εφαρμογής, να προσθέσουμε overlays και σχολιασμούς για να επισημάνουμε σημεία ενδιαφέροντος, να παρέχουμε αυτόματη ολοκλήρωση κειμένου προκειμένου να διευκολυνθεί ο χρήστης στην αναζήτηση ενός προορισμού ή ενός σημείου ενδιαφέροντος.

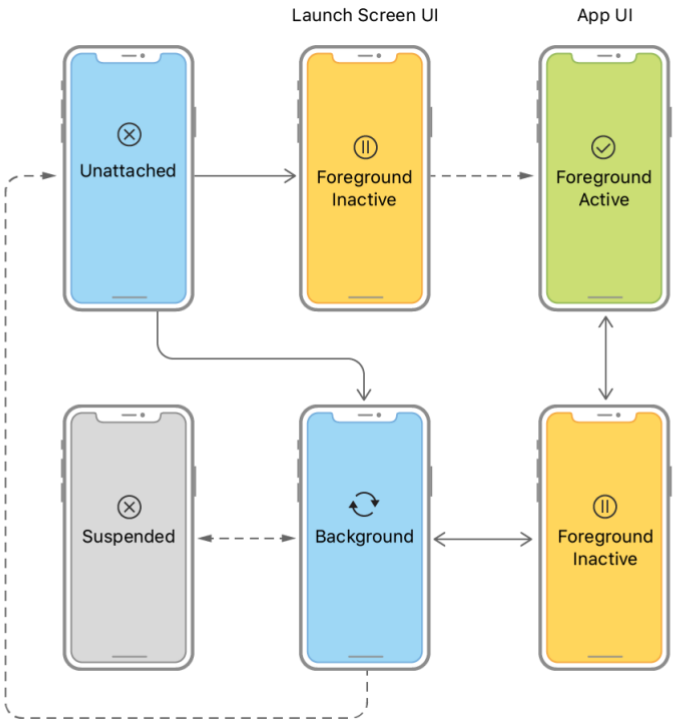
1.7.3 iOS App Lifecycle & AppDelegate

Όλες οι iOS εφαρμογές είναι περιπτώσεις UIApplication. Κάθε εφαρμογή έχει έναν αριθμό γεγονότων στον κύκλο ζωής της. Όπως και με άλλες κλάσεις του UIKit framework, ο προγραμματιστής έχει την δυνατότητα να ελέγχει τον τρόπο με τον οποίο θα ανταποκριθεί η εφαρμογή σε αυτά γεγονότα μέσω του εξουσιοδώτη(delegate).Το UIApplicationDelegate εφαρμόζει το πρωτόκολλο UIApplicationDelegate για αυτό το σκοπό.

Πως ξεκινάει μια iOS εφαρμογή:

Όταν ο χρήστης ενεργοποιήσει την συσκευή, δεν εκτελούνται εφαρμογές εκτός απ' αυτές του λειτουργικού συστήματος. Αφού ο χρήστης πατήσει το εικονίδιο της εφαρμογής, το springboard το οποίο είναι κομμάτι του λειτουργικού συστήματος υπεύθυνο για την διαχείριση της αρχικής οθόνης, εκκινεί την εφαρμογή. Η εφαρμογή μαζί με τις κοινόχρηστες βιβλιοθήκες που χρειάζεται για να εκτελεστεί, θα φορτωθεί στην μνήμη ενώ το springboard δείχνει το launchScreen της εφαρμογής. Όταν ολοκληρωθεί η παραπάνω διαδικασία και αρχίσει η εκκίνηση της εφαρμογής, θα ειδοποιηθεί το AppDelegate.

Το λειτουργικό σύστημα iOS διαχειρίζεται τις καταστάσεις της εφαρμογής, αλλά η εφαρμογή είναι υπεύθυνη για τον χειρισμό της εμπειρίας χρήσης μέσω αυτών των μεταβάσεων. Πιο συγκεκριμένα οι καταστάσεις είναι οι εξής:



Εικόνα 8. Διάγραμμα για τον κύκλο ζωής μιας εφαρμογής στο iOS

- **Not Running - Unattached:** Είτε η εφαρμογή δεν έχει ξεκινήσει ακόμη είτε ήταν υπό εκτέλεση και έχει τερματιστεί από το σύστημα.

- **Inactive:** Η εφαρμογή εκτελείται στο προσκήνιο (Foreground) αλλά δεν λαμβάνει συμβάντα συνεπώς δεν μπορούμε και να αλληλοεπιδράσουμε με το UI της. Αυτό θα μπορούσε να συμβεί σε περίπτωση λήψης μηνύματος ή κλήσης. Σε αυτή την κατάσταση θα μπορούσε να βρεθεί και κατά την διάρκεια της μετάβασης σε άλλη κατάσταση.

- **Active:** Η εφαρμογή εκτελείται στο προσκήνιο και λαμβάνει τα συμβάντα. Αυτή είναι η κανονική λειτουργία για τις εφαρμογές προσκηνίου. Ό μόνος τρόπος για να πάμε στην ενεργή κατάσταση είναι μέσω της ανενεργής. Ο χρήστης συνήθως αλληλοεπιδρά με το UI και σε αυτή την κατάσταση μπορεί να δει την απόκριση – αποτέλεσμα για τις ενέργειες του.

- **Background:** Η εφαρμογή τρέχει στο παρασκήνιο και εκτελεί τον κώδικα. Οι εφαρμογές που ξεκινούν πρόσφατα εισέρχονται απευθείας στην ανενεργή κατάσταση και μετά στην ενεργή. Οι εφαρμογές που έχουν τεθεί σε αναστολή θα

μεταβούν στην κατάσταση παρασκηνίου πριν εισέλθουν στα στάδια ανενεργού -> ενεργού.

- **Suspend:** Η εφαρμογή βρίσκεται στο παρασκήνιο αλλά δεν εκτελείται ο κώδικας. Το σύστημα αναστέλλει μια εφαρμογή αυτόματα χωρίς ειδοποίηση. Σε περιπτώσεις χαμηλής μνήμης το λειτουργικό σύστημα μπορεί να εκκαθαρίσει μια εφαρμογή που βρίσκεται σε αναστολή χωρίς προειδοποίηση προκειμένου να απελευθερώσει χώρο στην μνήμη για τις εφαρμογές που βρίσκονται στο προσκήνιο. Συνήθως μετά από 5 δευτερόλεπτα που βρίσκεται μια εφαρμογή στο προσκήνιο μεταβαίνει σε αναστολή, αλλά αυτός χρόνος μπορεί να επεκταθεί από τον προγραμματιστή αν το χρειάζεται η εφαρμογή.

Το αντικείμενο UIApplication ορίζει μερικές μεθόδους που καλούνται ή θα ανταποκριθούν σε ορισμένες από τις παραπάνω καταστάσεις που είναι πιο σημαντικές , για να μας επιτρέψει να χειριστούμε τις μεταβάσεις αυτές και να προσαρμόσουμε τις λειτουργίες της εφαρμογής μας. Οι μέθοδοι παρουσιάζονται στην παρακάτω εικόνα:

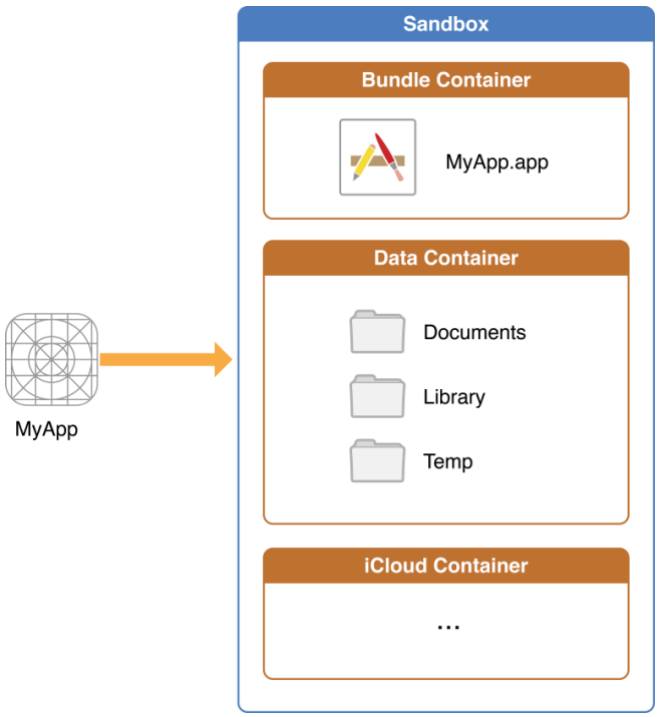
App Termination Condition	App Execution State	Delegate Called
Device power off	suspended	none
Device power off	background	`applicationWillTerminate`
Device power off	foreground	`applicationDidEnterBackground`, `applicationWillTerminate`
Force quit from multitasking UI	suspended	none
Force quit from multitasking UI	background	`applicationWillTerminate`
Force quit from multitasking UI	foreground	`applicationDidEnterBackground`, `applicationWillTerminate`
Low memory/dropped by iOS	suspended	none
Low memory/dropped by iOS	background	`applicationWillTerminate`

Εικόνα 9. Μέθοδοι της κλάση UIApplication για τις πιθανές καταστασεις μιας εφαρμογής

1.7.4 Δομή αρχείου .ipa

Ένα αρχείο .ipa (iOS App Store Package) είναι ουσιαστικά ένα αρχείο αρχειοθέτησης εφαρμογών iOS που αποθηκεύει μια εφαρμογή iOS. Κρυπτογραφείται με την τεχνολογία FairPlay DRM της Apple. Κάθε αρχείο .ipa συμπιέζεται με δυαδικό αρχείο για την αρχιτεκτονική ARM και μπορεί να εγκατασταθεί μόνο σε συσκευές που τρέχουν το λειτουργικό σύστημα iOS.

Το αρχείο .ipa αποτελείται από τα εξής επίπεδα:



Εικόνα 10. Διάγραμμα για την δομή ενός αρχείου .ipa

- **Sandbox:** Το sandbox είναι μια τεχνολογία ελέγχου πρόσβασης που εφαρμόζεται στο επίπεδο πυρήνα. Έχει σχεδιαστεί για να παρέχει ζημίες στο σύστημα και στα δεδομένα του χρήστη σε περίπτωση που μια εφαρμογή παραβιαστεί. Οι εφαρμογές που κυκλοφορούν στο App Store πρέπει να χρησιμοποιούν το App Sandbox.

- **Bundle Container:** Ο κατάλογος Bundle αποτελείται απ' όλα τ αρχεία που συνοδεύουν την εφαρμογή όταν εγκαθίσταται από το App store. Έτσι τα αρχεία σε αυτόν τον κατάλογο θα παραμείνουν ίδια σε μια συγκεκριμένη έκδοση της εφαρμογής. Πολλά σημαντικά αρχεία που διαθέτουν απαραίτητες πληροφορίες για την λειτουργία της εφαρμογής, τοποθετούνται εκεί. Μερικά από αυτά παρουσιάζονται στην παρακάτω φωτογραφία:

Resource Name	Resource Type	Description
iTunesMetadata.plist	File	This file is used to provide information to iTunes about an iOS application using Ad Hoc distribution for either testing or Enterprise deployment.
iTunesArtWork	File	This file contains the image that is used to represent the application in iTunes
META-INF	Directory	It contains two files which hold the metadata about the IPA file.
.app Directory	Directory	This directory holds all the components of the IPA file.
Application Binary	File	This file contains the application's executable code. It's name is same as that of the name of .app Directory excluding the extension ".app".
Info.plist	File	This file is the manifest of the iOS application. It contains information about - Supported Devices, Bundle ID, Display Name, Application Transport Security etc.
Application Icon	File	These are the icon files of the application. There are multiple icon files (Icon.png, Icon@2x.png) for representation of the application on devices with different resolution like iPhone, iPad etc.
Launch Images	File	These files (Default.png, Default-portrait.png) are used as launch screen images before the application launches. They are removed as soon as the application is ready to display the UI.
Storyboard/nib files	File / Directory	These files contain the encrypted information about the storyboard of an application.

Εικόνα 11. Σημαντικά αρχεία που βρίσκονται στον bundle container

- **Data Container:** Ο κατάλογος δεδομένων ή αλλιώς κοντέινερ τοπικής αποθήκευσης δεδομένων αποτελείται από τα αρχεία που ο προγραμματιστής επιθυμεί να αποθηκεύσει για την εφαρμογή την διαδικασία της εγκατάστασης της στην συσκευή. Τα αρχεία αυτά μπορούν να χρησιμοποιηθούν για την αποθήκευση πληροφοριών για γρήγορη πρόσβαση ή για να εξασφαλίσει (back up) σε περίπτωση που η συσκευή βρεθεί εκτός σύνδεσης ότι θα συνεχιστεί η ροή της από εκεί που σκόπευε ο προγραμματιστής. Έτσι τα αρχεία αυτά θα συνεχίσουν να αλλάζουν, όσο η εφαρμογή εκτελείται, όπως τα έχει προγραμματίσει ο προγραμματιστής.

- **iCloud Container:** Ο κατάλογος αυτός περιέχει δεδομένα που χρησιμοποιούν όσες εφαρμογές υποστηρίζουν δυνατότητες iCloud.Τα αρχεία σε αυτό τον κατάλογο προορίζονται να αποθηκεύονται και να ενημερώνονται από τις πηγές τις οποίες αποφασίζει ο χρήστης να ενημερώσει το αρχείο.

Αποτελείται συνήθως από δύο μέρη τα έγγραφα και τα δεδομένα. Τα έγγραφα είναι τα αρχεία που προορίζονται να αναγνωστούν και να ενημερωθούν απευθείας από τον χρήστη. Για να διατηρούνται συγχρονισμένα δημιουργούνται τακτικά αντίγραφα ασφάλειας στο iCloud.Τα δεδομένα είναι τα αρχεία που δεν προορίζονται για άμεση αλληλεπίδραση με τον χρήστη. Τα δεδομένα μπορούν να αποθηκεύονται σε διαφορετικούς καταλόγους ανάλογα με τις ανάγκες του προγραμματιστή.

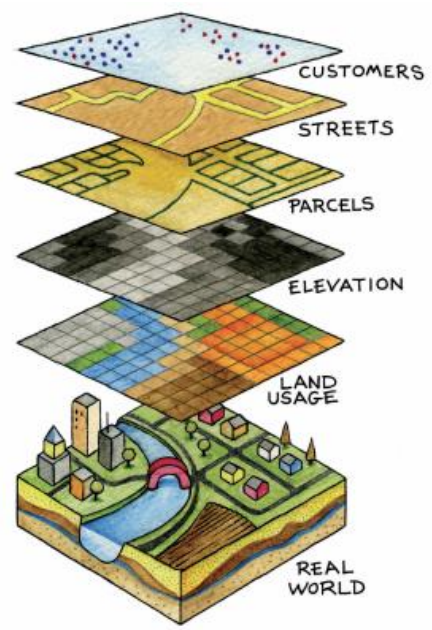
2. G.I.S

2.1. Τι είναι το G.I.S από την μεριά του Software και του Hardware.

Οι περισσότερες από τις πληροφορίες που χρησιμοποιούμε σήμερα είναι γεωαναφορές. Ειδικότερα είναι πληροφορίες οι οποίες μπορούν να συσχετιστούν με μια γεωγραφική θέση και έτσι να αποτελέσουν βοηθητικό υλικό με την τοποθεσία που αντιστοιχεί.

- **Από την μεριά του λογισμικού:** Ένα σύστημα γεωγραφικών πληροφοριών αποτελείται από ένα ειδικό τύπο προγράμματος, ικανό για αποθήκευση, επεξεργασία, ανάλυση και παρουσίαση γεωγραφικών δεδομένων και πληροφοριών ως χάρτες. Υπάρχουν αρκετοί πάροχοι λογισμικού G.I.S όπως Environmental Systems Research Institute Inc, το οποίο διανέμει το ArcGIS και το PitneyBowes το οποίο διανέμει MapInfo GIS. Αν και οι διαδικτυακές υπηρεσίες χαρτογράφησης παρέχονται από εταιρείες όπως η Google, το Yahoo , και η Microsoft, τέτοιες υπηρεσίες δεν θεωρούνται ακόμη πλήρως ολοκληρωμένες πλατφόρμες GIS. Υπάρχουν επίσης επιλογές GIS ανοιχτού κώδικα, όπως το GRASS , το οποίο διανέμεται ελεύθερα και συντηρείται από την κοινότητα ανοιχτού κώδικα. Όλο το λογισμικό GIS, ανεξάρτητα από τον πάροχο, αποτελείται από ένα σύστημα διαχείρισης βάσεων δεδομένων που είναι ικανό να χειρίζεται και να ενσωματώνει δύο τύπους δεδομένων: τα χωρικά δεδομένα και τα δεδομένα χαρακτηριστικών.

  Πιο αναλυτικά τα χωρικά δεδομένα(spatial data) αναφέρονται σε πραγματικά γεωγραφικά αντικείμενα ενδιαφέροντος, όπως δρόμοι, κτίρια, λίμνες και χώρες, καθώς και στις αντίστοιχες τοποθεσίες τους. Επιπρόσθετα για την τοποθεσία, κάθε ένα από αυτά τα αντικείμενα διαθέτει επίσης ορισμένα στοιχεία ενδιαφέροντος ή χαρακτηριστικά όπως όνομα, αριθμός ιστοριών, βάθος ή πληθυσμός. Το λογισμικό GIS παρακολουθεί τόσο τα χωρικά όσο και τα δεδομένα χαρακτηριστικών και μας επιτρέπει να τα συνδέσουμε μαζί για τη δημιουργία πληροφοριών και τη διευκόλυνση της ανάλυσης. Ένας δημοφιλής τρόπος για να περιγράψουμε και να απεικονίσουμε ένα GIS είναι με πολλά στρώματα όπως φαίνεται στην παρακάτω φωτογραφία.



Εικόνα 12. Παράδειγμα απεικόνισης G.I.S

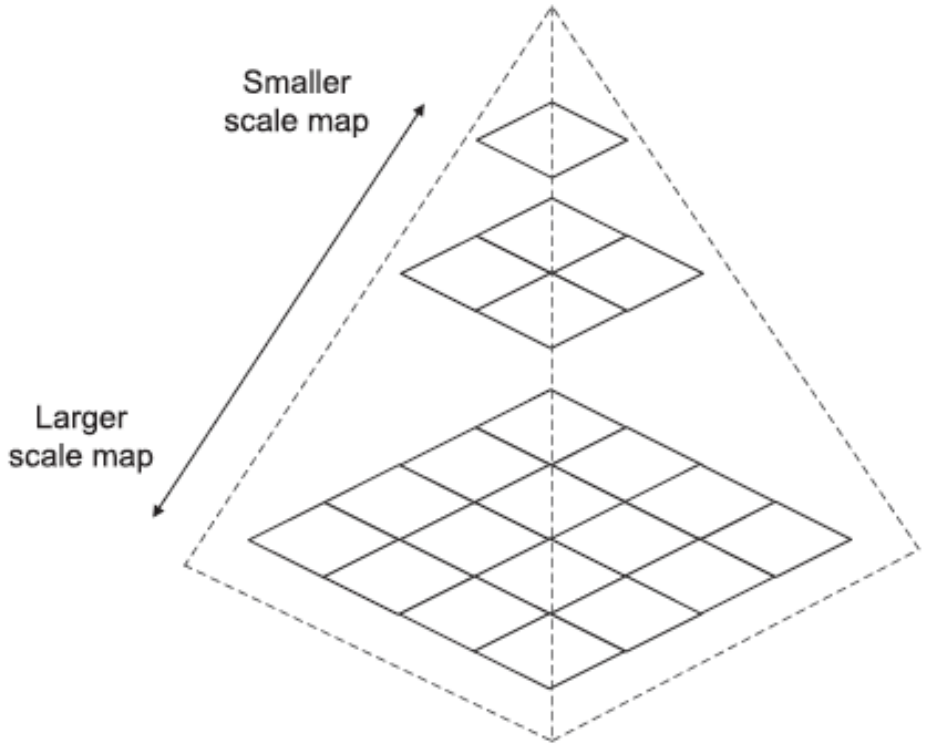
Κάθε στρώμα αντιπροσωπεύει ένα διαφορετικό γεωγραφικό θέμα, όπως το νερό, τα κτίρια και τους δρόμους, και κάθε στρώμα τοποθετείται το ένα πάνω από το άλλο.

- **Από την μεριά του υλικού:** Ένα GIS αποτελείται από έναν υπολογιστή, μνήμη, συσκευές αποθήκευσης, σαρωτές, εκτυπωτές, μονάδες παγκόσμιου συστήματος εντοπισμού θέσης (GPS) και άλλα φυσικά στοιχεία. Αν ο υπολογιστής βρίσκεται σε δίκτυο, το δίκτυο μπορεί επίσης να θεωρηθεί ως αναπόσπαστο στοιχείο του GIS επειδή μας επιτρέπει να μοιραζόμαστε δεδομένα και πληροφορίες που χρησιμοποιεί το GIS ως εισόδους και δημιουργεί ως έξοδοι. Ως εργαλείο, ένα GIS μας επιτρέπει να διατηρούμε, να αναλύουμε και να μοιραζόμαστε πολλά δεδομένα και πληροφορίες. Από το σχετικά απλό έργο της χαρτογράφησης του μονοπατιού ενός τυφώνα έως το πιο σύνθετο έργο του καθορισμού των πιο αποτελεσματικών διαδρομών συλλογής απορριμμάτων σε μια πόλη, ένα GIS χρησιμοποιείται στον δημόσιο αλλά και στον ιδιωτικό τομέα. Κινητές υπηρεσίες χαρτογράφησης, πλοήγησης και βάσει τοποθεσίας εξατομικεύουν το GIS με σκοπό την πρόσβαση των χαρτών και τη χαρτογράφηση στις μάζες.

2.2. Τι είναι και πως προέκυψε ο Tiled Web Map

Στα μέσα της δεκαετίας του 2000 μετά τους χάρτες Google, το Microsoft visual Earth και άλλες δημοφιλείς εφαρμογές χαρτογράφησης που έπληξαν το ιστό, οι άνθρωποι άρχισαν να συνειδητοποιούν ότι ίσως δε χρειάζονται την δυνατότητα να παραμετροποιούν τις ιδιότητες κάθε επίπεδου. Αυτοί οι πάροχοι είχαν αρχίσει να συγχωνεύουν διανυσματικά το στρώματα τους σε μία μόνο ραστεροποιημένη εικόνα χωρίστηκε σε εικόνες 256 x 256 pixel ή αλλιώς τα λεγόμενα πλακίδια (tiles). Αυτά τα πλακίδια προ γεννήθηκαν και αποθηκεύτηκαν σε δίσκο για γρήγορη διανομή στους clients. Αυτό έγινε λόγω της αναγκαιότητας για την υποστήριξη εκατοντάδων ή χιλιάδων ταυτόχρονων χρηστών, ένα πολύ μεγάλο βάρος για την σχεδίαση των χαρτών.

Το παρακάτω σχήμα δείχνει πώς ένας χάρτης με tiles αποτελείται από μια "πυραμίδα" εικόνων που καλύπτουν την έκταση του χάρτη σε διάφορες κλίμακες. Οι tiled maps συνήθως συνοδεύονται από ένα σχήμα αρίθμησης επιπέδων, σειρών και στηλών που μπορεί να κοινοποιηθεί σε κρυφές μνήμες για να εξασφαλίζεται ότι τα όρια των πλακιδίων ταιριάζουν εάν επικαλύπτονται από δύο σετ πλακιδίων.



Εικόνα 13. Απεικόνιση των tiles ανά επίπεδο

Μέσα δύο χρόνια από την κυκλοφορία των Χαρτών Google, το εμπορικό λογισμικό GIS άρχισε να προσφέρει τη δυνατότητα κατασκευής tiled maps. Για πολλούς, ο διακομιστής ArcGIS ήταν επιθυμητός επειδή ο χάρτης θα μπορούσε να δημιουργηθεί χρησιμοποιώντας

τα ώριμα εργαλεία σύνταξης χαρτών στο ArcMap. Ωστόσο, το κόστος ήταν ανησυχητικό για ορισμένους.

Οι tiled maps ήταν το μόνο μοντέλο που θα μπορούσε εύλογα να λειτουργήσει για την εξυπηρέτηση σύνθετων χαρτών Ιστού σε χιλιάδες ταυτόχρονους χρήστες. Ωστόσο, εξάλειψαν τη δυνατότητα των χρηστών να αλλάζουν σειρά επιπέδων ή σύμβολα. Οι άνθρωποι άρχισαν να δουλεύουν γύρω από αυτό, εξυπηρετώντας τα στρώματα γενικού σκοπού ως πλακίδια και έπειτα επικαλύπτοντας ένα ξεχωριστό επίπεδο με θεματικές πληροφορίες. Τα πλακίδια γενικού σκοπού θα μπορούσαν να επαναχρησιμοποιηθούν σε πολλές εφαρμογές. Τα θεματικά επίπεδα μπορούν επίσης να μετατραπούν σε tiles εάν τα δεδομένα δεν αλλάξουν πολύ γρήγορα ή καλύπτουν πολύ ευρεία περιοχή σε μεγάλες κλίμακες.

3.Ο χάρτης του Kitchener

Ο νεαρός υπολοχαγός του Βασιλικού Σώματος Μηχανικών Herbert Horatio Kitchener στάλθηκε στην Κύπρο το 1878, τη χρονιά που η Μεγάλη Βρετανία είχε αναλάβει τη διοίκηση της Κύπρου από τους Οθωμανούς. Σύμφωνα με τον πρώτο Ύπατο Αρμοστή Sir Garnet Wolseley, σκοπός της αποστολής του Kitchener ήταν να πραγματοποιήσει μια συνοπτική τοπογραφική τεκμηρίωση, απαραίτητη για τη διαχείριση των εσόδων. Όμως, ενάντια στις οδηγίες και επιθυμίες του Υπάτου Αρμοστή, ο Kitchener χαρτογράφησε με τη μέγιστη δυνατή λεπτομέρεια ολόκληρο το νησί. Ο Herbert Kitchener κατάφερε να μετρήσει 1.208 πηγάδια, 88 γέφυρες και 53 μοναστήρια, 261 ποιμνιοστάσια, 104 πηγές και 11 σπηλιές, αλλά και να καταγράψει όλα τα τοπωνύμια, όπως και τη σύνθεση του πληθυσμού στην Κύπρο του 1885 με σχεδόν απόλυτη ακρίβεια, και να τα χωρέσει σε έναν χάρτη 16 φύλλων. Πολλές από αυτές τις πληροφορίες χρήζουν ακόμα και στις μέρες μας προσεκτικής διερεύνησης από την επιστημονική κοινότητα.

Εικόνα 14. Herbert Kitchener

Ο χάρτης του Herbert Kitchener που προέκυψε από την πρώτη επιστημονική χαρτογράφηση της Κύπρου εκδόθηκε το 1885 στο Λονδίνο από τον Edward Stanford. Θεωρείται κομβικής σημασίας για την χαρτογραφία της Κύπρου, περιέχει πληθώρα αξιόλογων πληροφοριών γεωγραφικού, ιστορικού αλλά και πολιτισμικού ενδιαφέροντος για το νησί και αποτέλεσε χαρτογραφικό πρότυπο έως τα μέσα του 20ού αιώνα.

Εικόνα 15. Απόσπασμα από τον χάρτη του Kitchener

4.Υλοποίηση της εφαρμογής

Κατά τη διάρκεια της υλοποίησης του Project δόθηκε έμφαση στην εμπειρία χρήσης, στην αποδοτικότητα της εφαρμογής και στην διατήρηση της γενικής φιλοσοφίας του Project ανάμεσα σε όλες πλατφόρμες(Android, iOS, Web). Για να επιτευχθεί αυτό αναλύθηκαν όλα τα απαιτούμενα της εφαρμογής και αναγνωρίστηκαν τα πιθανά σενάρια χρήσης της (Use Cases) προκειμένου να προγραμματιστούν οι κατάλληλες εργασίες. Στο παρακάτω κεφάλαιο αναλύονται τα βήματα που ακολουθήθηκαν για να υλοποιηθούν τα απαιτούμενα για κάθε οθόνη της εφαρμογής.

4.1. Σημαντικές εξωτερικές πηγές που χρησιμοποιήθηκαν

- **Kitchener's Map API:** Το automatic programming interface είναι ένα μεσάζων εργαλείο πληροφόρησης που επιτρέπει την επικοινωνία μεταξύ δύο εφαρμογών. Οι πληροφορίες παρέχονται μέσω των API Endpoints.Το συγκεκριμένο API υλοποιήθηκε από το τμήμα γεωγραφίας του Χαροκοπείου Πανεπιστημίου το οποίο κάνει χρήση GIS εφαρμογών. Είναι υπεύθυνο για την ανατροφοδότηση της εφαρμογής με όλα τα απαραίτητα δεδομένα όπως μενού γεωγραφικών επιπέδων και φίλτρων, tiles και πολλές άλλες εργασίες που θα αναλυθούν στην συνέχεια.

- **Realm Database:** Αποτελεί εναλλακτική της SQLite & Core Data. Ουσιαστικά είναι ένα σύστημα διαχείρισης της βάσης δεδομένων που αρχικά εστίαζε στα λειτουργικά συστήματα κινητών Android – iOS.Βασίζεται σε zero-copy design.Πιο συγκεκριμένα σε τέτοιες περιπτώσεις ο επεξεργαστής για να υλοποιεί τις εργασίες δεν χρειάζεται να αντιγράφει τα δεδομένα από την μια περιοχή μνήμης στην άλλη. Αυτό έχεις ως αποτέλεσμα η φόρτωση και η αποθήκευση δεδομένων τοπικά να είναι αρκετά πιο γρήγορη σε σχέση με άλλες μεθόδους. Για τον παραπάνω λόγο χρησιμοποιήθηκε το συγκεκριμένο component για την αποθήκευση των tiles και την εξυπηρέτηση της offline λειτουργικότητας.

- **Firebase Crashlytics:** Η εφαρμογή θα εξυπηρετεί ένα εύρος χρηστών, αυτό απαιτεί την παρουσία ενός συστήματος εντοπισμού πιθανών crashes. Το firebase είναι ένα ελαφρύς αναφορέας σφαλμάτων πραγματικού χρόνου. Έτσι με τις κατάλληλες ρυθμίσεις που μας υποδεικνύει το documentation του firebase, για όλα τα σφάλματα που θα παρουσιαστούν , η πλατφόρμα θα τα εντοπίζει, θα τα ομαδοποιεί έξυπνα και θα μας επισημαίνει τις περιστάσεις που οδηγούν σε αυτά προκειμένου να τα επιλύσουμε. Με αυτό τον τρόπο θα μπορέσει να γίνει πιο ομαλή η συντήρηση του project και η διατήρηση μιας ποιοτικής εμπειρίας χρήσης.

4.2. Κεντρική οθόνη Kitchener Map (MapViewController)

Η συγκεκριμένη οθόνη είναι η κεντρική οθόνη της εφαρμογής στην οποία προβάλλεται ο χάρτης του Kitchener.Με τον όρο UIViewController αναφερόμαστε στο αντικείμενο που διαχειρίζεται μία ιεραρχία προβολής για την εφαρμογή UIKit. Ο MapViewController αποτελείται από τα παρακάτω components:

- **MKMapView:** Το βασικό component που περιέχει και λαμβάνει το μεγαλύτερο μέρος της οθόνης είναι το MKMapView.Το MKMapView είναι ουσιαστικά μια ενσωματωμένη διεπαφή χάρτη, παρόμοια με αυτήν που παρέχεται από την εφαρμογή Χάρτες. Προκειμένου να τον παραμετροποιήσουμε σύμφωνα με τα απαιτούμενα του project έγινα τα εξής βήματα – προσθήκες:

1. **User Location Button**: Το user interface του MKMapView δεν παρέχει την δυνατότητα στον προγραμματιστή να προσθέσει κουμπί εντοπισμού θέσης, έτσι έπρεπε να φτιαχτεί από την αρχή. Για να γίνει αυτό προστέθηκε στο storyboard(αντικείμενο το οποίο διαχειρίζεται όλους τους controllers), πάνω από το χάρτη, ένα κουμπί (UIButton) το όποιο συνδέθηκε με την κλάση του Controller προκειμένου να παίρνουμε τα touch events.Έτσι στην μέθοδο που καλείται όταν ο χρήστης πατάει το κουμπί έγινε η υλοποίηση που φαίνεται στην παρακάτω εικόνα :

```
@IBAction func getMylocationClicked(_ sender: UIButton) {
    if let coordinates = mapView.userLocation.location?.coordinate {
        mapView.setCenter(coordinates, animated: true)
    }
}
```

Εικόνα 16. Μέθοδος για τον εντοπισμό θέσης του χρήστη

Πιο συγκεκριμένα από το object του MKMapView που χούμε συνδεδεμένο στον controller παίρνουμε τις συντεταγμένες της τοποθεσίας του χρήστη και στην συνέχεια κεντράρουμε την κάμερα του χάρτη πάνω σε αυτές.

Για να μπορέσουμε να πάρουμε την τοποθεσία του χρήστη θα πρέπει να δηλώσουμε στο info.plist (αρχείο για τις ρυθμίσεις του project) ότι η εφαρμογή κάνει χρήση της τοποθεσίας του χρήστη. Αφού έχει γίνει η δήλωση, κατά την πρώτη φορά που εκτελείται η εφαρμογή μετά την εγκατάσταση της, ζητάμε από τον χρήστη τα κατάλληλα δικαιώματα. Από την στιγμή που κάνει την επιλογή των δικαιωμάτων, ο μόνος τρόπος για να αλλάξουν είναι από την αντίστοιχη σελίδα στις ρυθμίσεις του iOS.

Privacy - Location Always and When In Use Usage De...	String	This application requires location servic
Privacy - Location When In Use Usage Description	String	This application requires location servic

Εικόνα 17. Στιγμιότυπο από το info.plist στο όποιο δηλώνεται η χρήση της τοποθεσίας του χρήστη

2. **Zoom Level:** Για τη διευκόλυνση της πλοήγησης του χρήστη προστέθηκε η ένδειξη για το επίπεδο zoom που βρίσκεται ο χάρτης. Για να υλοποιηθεί αυτό προστέθηκε στο κάτω μέρος του χάρτη ένα UILabel(αντικείμενο για την προβολή κειμένου) το όποιο συνδέσαμε με την κλάση του Controller.Στην viewDidLoad (μέθοδος η οποία εκτελείται όταν ενεργοποιείται ο Controller) του Controller δηλώσαμε ότι ο Controller εφαρμόζει το πρωτόκολλο MKMapViewDelegate του χάρτη. Με αυτό τον τρόπο μας έρχονται ειδοποιήσεις διάφορων γεγονότων σχετικά με τον χάρτη που θα χρειαστούμε στην συνέχεια. Έτσι όταν ο χρήστης κουνήσει την κάμερα του χάρτη το αντικείμενο mapview θα καλέσει την μέθοδο "regionDidChangeAnimated" του πρωτόκολλου. Αφού ο controller εφαρμόζει το πρωτόκολλο, θα ενημερωθεί και μείς με την σειρά μας στην αντίστοιχη μέθοδο θα πάρουμε το τρέχων zoom της κάμερας προκειμένου να το τυπώσουμε στο UILabel.

3. **Κουμπί επαναφοράς:** Στο δεξί πάνω μέρος του controller έχει τοποθετηθεί ένα UIButton με τίτλο επαναφορά. Όταν το πατήσει ο χρήστης δίνεται η εντολή στον Controller να γίνει η εκκαθάριση του χαρτη. Με αυτή την διαδικασία αφαιρούνται όλα τα overlays και τα σχόλια που υπήρχαν σε αυτόν και επαναφέρεται στις προεπιλεγμένες ρυθμίσεις του.

4. **UIGestureRecognizer**: Στα απαιτούμενα της εφαρμογής ήταν ο χρήστης να έχει την δυνατότητα ένα επιλέξει ένα σημείο του χάρτη και να μπορεί να στείλει ένα σχόλιο για αυτό στον server με σκοπό την βελτίωση της λειτουργία της εφαρμογής ή να δει περιγραφές για το συγκεκριμένο σημείο. Προκείμενου να γίνει το πρώτο βήμα της επιλογής προστέθηκε στο mapView ένας αναγνωριστής χειρονομίας που αναγνωρίζει χειρονομίες παρατεταμένου τύπου ο λεγόμενος UILongPressGestureRecognizer. Έτσι με το πού ειδοποιηθεί το mapView γία ένα τέτοιο γεγονός(touch event) καλείται η αντίστοιχη μέθοδος η οποία προσθέτει στον χάρτη object τύπου MKPoint Annotation (αντικείμενο για σχολιασμό που εφαρμόζεται σε ένα συγκεκριμένο σημείο του χάρτη).

- **Offline Map Cache:** Το project απαιτούσε την αποθήκευση των tiles τοπικά προκειμένου να μπορεί η εφαρμογή να υποστηρίζει την offline λειτουργεία. Με σκοπό να γίνει αυτό δημιουργήθηκε η κλάση OfflineMapCache.

  Η κλάση αυτή εφαρμόζει το πρωτόκολλο MapCacheProtocol ώστε να πει σε κάθε επιφάνεια πλακιδίων(tile) που καλύπτει μια περιοχή (MKTileOverlay) πως θα πάρει τα δεδομένα για το κάθε tile. Με τον όρο KTileOverlay εννοούμε μια επιφάνεια με πλακίδια bitmap πάνω από μια περιοχή του MKMapView.

Έτσι στην κλάση OfflineMapCache δηλώνονται το URL- API Endpoint το όποιο θα καλεί η εφαρμογή για να πάρει το tile,το format του κάθε URL προκείμενου να αντικαθιστά σε κάθε κλήση τις αντίστοιχους παραμέτρους με τις συντεταγμένες των tiles και άλλα απαραίτητα δεδομένα.

Αφού έχουν γίνει οι απαραίτητες αρχικοποιήσεις μέσω του initializer της κλάσης, μπορούμε μέσω της μεθόδου loadTile να καθορίσουμε ποια θα είναι η πηγή της εικόνας που θα προβάλει το MKTileOverlay για κάθε tile.Η συγκεκριμένη μέθοδος έχει ως παράμετρο το path που είναι αντικειμένου της κλάσης MKTileOverPath, το οποίο το χρησιμοποιήσουμε προκειμένου να καθορίσουμε τις τιμές εύρεσης ενός πλακιδίου. Με το path καλούμε την μέθοδο url προκειμένου να πάρουμε το URL που θα καλέσουμε με τις παραμέτρους x , y , z που έχουν προκύψει από το path,όπου x και y οι συντεταγμένες του πλακιδίου και z το zoom.

Έχοντας στην κατοχή μας το URL ψάχνουμε στην realmDatabase εάν υπάρχει ήδη αποθηκευμένο το συγκεκριμένο tile.Για να γίνει αυτή η διαδικασία, έχουμε δημιουργήσει την κλάση OfflineTile η οποία είναι υπό κλάση της Object.Οι μεταβλητές που έχει αυτή η κλάση είναι το URL το οποίο αποτελεί το πρωτεύων κλειδί (primary key) και το data που είναι η εικόνα του πλακιδίου σε μορφή Data.
Συνεπώς αφού αποκτήσουμε το URL μπορούμε να προσπελάσουμε την RealmDatabase και να βρούμε αν υπάρχει αντικείμενο OfflineTile με πρωτεύων κλειδί το συγκεκριμένο URL. Σε περίπτωση που βρεθεί τέτοιο αντικείμενο τότε μέσω της escaping closure τερματίζεται η συνάρτηση και το MKTileOverlay παίρνει το data του αντικειμένου για να το προβάλει στο πλακίδιο.

Εάν δεν βρεθεί αντικείμενο στην βάση δεδομένων η διαδικασία είναι διαφορετική. Η κλάση URLSession παρέχει ένα API για την λήψη και μεταφόρτωση δεδομένων από τα endpoints που υποδεικνύον τα URLs.Έτσι χρησιμοποιώντας την παραπάνω κλάση γίνεται κλήση στον αντίστοιχο server μέσω του URL προκειμένου γίνει λήψη του πλακιδίου. Αφού ολοκληρωθεί η λήψη και δεν υπάρχει κάποιο σφάλμα (error) τότε δημιουργείται αντικείμενο OfflineTile με το data και το URL και αποθηκεύεται στην RealmDatabase,ενώ παράλληλα μέσω του escaping closure το MKTileOverlay έχει πάρει το πλακίδιο που έχει ζητήσει.

```swift
func loadTile(at path: MKTileOverlayPath, result: @escaping (Data?, Error?) -> Void) {

    let url = self.url(forTilePath: path)
    let db = RealmDaoHelper<OfflineTile>()
    guard db.findFirst(key: url.absoluteString) == nil else {
        let foundTile = db.findFirst(key: url.absoluteString)
        result(foundTile?.data, nil)
        return
    }
    var request = URLRequest(url: url)
    request.timeoutInterval = 90
    request.httpMethod = "GET"
    if self.hasHeader {
        request.addValue("private-user=mobileAPIuser1&private-pw=OesomEtaT",
            forHTTPHeaderField: "X-Credentials")
    }

    let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
        guard error == nil else {
            result(nil, error)
            return
        }
        guard let data = data else { result(nil, error); return }
        let db = RealmDaoHelper<OfflineTile>()
        let tile = OfflineTile(data: data, url: url.absoluteString)
        db.update(object: tile)
        result(data, error)

    }
    task.resume()
}
```

Εικόνα 18. Στιγμιότυπο από την μέθοδο loadTile

Έτσι για παράδειγμα προκειμένου να αρχικοποιήσουμε το Overlay το όποιο είναι υπεύθυνο για την προβολή του επιπέδου “Μωσαϊκό πρωτογενών πινακίδων Kitchener” στον χάρτη έγιναν τα βήματα που αναφέρονται στην συνέχεια. Αρχικά δημιουργήθηκε η optional μεταβλητή kitchenerLayer η οποία είναι τύπου CachedTileOverlay (υπο κλάση της MKTileOverlay).Στην συνέχεια στην viewDidLoad καλείται η setup Tile RendererKitchener η οποία φτιάχνει το configuration για τον constructor του Offline Map Cache με το URL του kitchener map.Ύστερα περνάμε το αντικείμενο στο mapview μέσω του extension που έχουμε φτιάξει και πιο συγκεκριμένα της μεθόδου useCache.Με την παραπάνω διαδικασία κάθε φορά που ο χρήστης θα επιλέγει συγκεκριμένο χαρτογραφικό επίπεδο από το μενού θα ελέγχεται αν το αντίστοιχο overlay είναι κενό, εάν είναι σημαίνει ότι δεν εμφανίζεται στον χάρτη οπότε θα πρέπει να καλέσουμε την setUpTileRendererKitchener προκειμένου να δημιουργηθεί, αντίθετα εάν υπάρχει ήδη θα πρέπει να την ορίσουμε την τιμή του overlay ως nil και να την σβήσουμε από τον χάρτη.

```swift
private func setupTileRendererKitchener() {
    let config = MapCacheConfig(withUrlTemplate: "https://gaia.hua.gr/tms/kitchener_review/{z}/{x}/{y}.jpg")
    let mapCache = OfflineMapCache(config: config)
    mapCache.hasHeader = true
    kitchenerLayer = mapView.useCache(mapCache, isGeometryFlipped: true)
}
```

Εικόνα 19. Στιγμιότυπο από την αρχικοποίηση του Kitchener Layer

4.3. Πλαϊνό Μενού (MenuViewController)

Η παραπάνω οθόνη είναι υπεύθυνη για την πλοήγηση του χρήστη στα βασικά σημεία της εφαρμογής. Η υλοποίηση του θα μπορούσε να γίνει είτε με UITableView είτε με ένα UIStackView εμφωλευμένο σε ένα UIScrollView. Από την στιγμή που το περιεχόμενο του μενού είναι στατικό προτιμήθηκε ο δεύτερος τρόπος. Πιο συγκεκριμένα δημιουργήθηκε στο storyboard ένα scrollView το οποίο περιέχει μέσα ένα UIStackView.Το UIStackView είναι ένα UI Component υπεύθυνο για την προβολή μιας συλλογής UI Components είτε οριζόντια είτε κάθετα. Έτσι χρησιμοποιώντας την κάθετη επιλογή για κάθε κουμπί του μενού προστέθηκε στο UIStackView ένα UIButton που θα ειδοποιεί τον MenuViewContoller για όταν το πατάει ο χρήστης. Ακόμη προστέθηκε ένα UILabel για να εμφανίζει το όνομα της επιλογής και όπου χρειάστηκε UIImage για τα εικονίδια της εφαρμογής.

Με τον πάτημα ενός κουμπιού από την χρήστη ο MenuViewController θα ειδοποιηθεί και θα δείξει την ανάλογη σελίδα. Για να επιτευχθεί αυτό έχουν προστεθεί στο stoyrboard UIContainerViews τα οποία μέσω των embed segues συνδέονται με τους αντίστοιχους controllers.Ένα UIContainerView είναι αντικείμενο της κλάσης UIVIew και ορίζει μια περιοχή στον UIViewController στην οποία θα φιλοξενείτε το παιδί ViewController.Με αυτήν διαδικασία μπορούμε να ενσωματώνουμε πολλές οθόνες μέσα σ έναν ViewController και να τις εμφανίζουμε ανάλογα με τις ενέργειες του χρήστη. Για παράδειγμα όταν ο χρήστης πατήσει πάνω στο κουμπί των όρων χρήσης τότε θα ειδοποιηθεί ο controller και μέσα από την αντίστοιχη μέθοδο που έχουμε φτιάξει θα εμφανίσουμε τον termsContainerView ο οποίος συνδέεται με το TermsViewController. Σε αυτή την περίπτωση ο termsViewController είναι childViewController για τον MapViewController.

4.4. Αναζήτηση σημείου – SearchViewController

Στην συγκεκριμένη οθόνη ο χρήστης έχει την δυνατότητα να αναζητήσει ένα σημείο του χάρτη γράφοντας τουλάχιστον 3 γράμματα από το όνομα του. Η υλοποίηση για αυτό το χαρακτηριστικό έγινε με τα παρακάτω components – resources :

- **UISearchBar στο πάνω μέρος της οθόνης:** Το UISearchBar είναι ένα εξειδικευμένο UIComponent για την λήψη πληροφοριών που σχετίζονται με την αναζήτηση από τον χρήστη. Ουσιαστικά αποτελείται από ένα πεδίο υποβολής στο οποίο ο χρήστης μπορεί να γράψει ότι θέλει να αναζητήσει.

- **UITableView:** Το UITableView είναι UIComponent το οποίο προβάλει δεδομένα χρησιμοποιώντας σειρές διατεταγμένες σε στήλη. Προκειμένου να γεμίσουμε τα κελιά του table με δεδομένα χρησιμοποιούμε την κλάση UITableViewDataSource. Η συγκεκριμένη κλάση περιέχει όλες τις μεθόδους οι οποίες υιοθετούνται από το αντικείμενο που χρησιμοποιείτε για τη διαχείριση δεδομένων και την παροχή κελιών με σκοπό την προβολή του table. Για να ειδοποιείται ο SearchViewController κάθε φορά που ο χρήστης επιλέγει ένα κελί (όταν έχει δεδομένα το UITableView) θα πρέπει να εφαρμοστεί το πρωτόκολλο UITableViewDelagate.Το συγκεκριμένο πρωτόκολλο περιέχει μεθόδους για τη διαχείριση επιλογών, τη διαμόρφωση κεφαλίδων και υποσέλιδων ενότητας, τη διαγραφή και αναδιάταξη κελιών και την εκτέλεση άλλων ενεργειών σε προβολή πίνακα. Χρησιμοποιώντας την μέθοδο didSelectRowAt παίρνουμε την θέση που αντιστοιχεί στο κελί που επιλέχθηκε και κάνουμε τις αντίστοιχες ενέργειες.

- **API Call:** Η αναζήτηση του σημείου που πληκτρολογεί ο χρήστης γίνεται στην βάση δεδομένων του server.Για αυτό τον σκοπό το κείμενο που πληκτρολογείτε στο UISearchBar στέλνεται στον Interactor ο οποίος το τοποθετεί στις παραμέτρους του URL που θα γίνει το call.Η κλήση στον server γίνεται ασύγχρονα και όταν ολοκληρωθεί ενημερώνεται ο Search ViewController.Τα δεδομένα που λαμβάνουμε τα μετατρέπουμε σε μορφή NSDictionary το οποίο είναι ένα αντικείμενο που αντιπροσωπεύει μια στατική συλλογή ζευγών κλειδιού-τιμής, για χρήση αντί μιας σταθεράς λεξικού σε περιπτώσεις που απαιτούν σημασιολογία αναφοράς. Έτσι ορίζεται η κλάση SearchResults η οποία θα περιέχει όλες τις τιμές που θα προβληθούν στο table σε μορφή πίνακα.

  Συνεπώς η ροή της παραπάνω σελίδας είναι η εξής: Ο χρήστης πληκτρολογεί στο UISearchBar το σημείο που επιθυμεί ,ο SearchViewController ειδοποιείται και με την σειρά του καλεί την fetchSearchResults μέθοδο του interactor με παράμετρο την τιμή που έχει πληκτρολογήσει ο χρήστης. Στην συνέχεια ο interactor κάνει το API call το οποίο όταν ολοκληρωθεί ενημερώνει το ViewController ώστε να περάσει τα δεδομένα στο UITableView μέσω του UITableViewDataSource. Τέλος ο controller

ενημερώνει το UITableView ότι εφαρμόζει το πρωτόκολλο UITableViewDelegate ώστε τον ειδοποιεί για κάθε γεγονός αλληλεπίδρασης του χρήστη με αυτό.

4.5. Χαρτογραφικά επίπεδα (MapLayersViewController)

Η συγκεκριμένη οθόνη επιτρέπει στον χρήστη την επιλογή διάφορων χαρτογραφικών επιπέδων και φίλτρων τα όποια μπορεί να εφαρμόσει στον χάρτη. Για να γίνει αυτό είναι απαραίτητη η παρουσία UITableView ώστε να προβληθούν όλες οι επιλογές. Για την συγκεκριμένη λειτουργία απαιτούνται 4 βασικά στάδια τα όποια είναι τα παρακάτω:

1. **Σχεδιασμός και αρχικοποίηση UITableView και UITableViewCell:** Ο σκοπός του συγκεκριμένου table είναι να προβάλλει όλες τις κατηγορίες χαρτογραφικών επιπέδων και όταν ο χρήστης επιλέγει μια να βλέπει τις επιλογές που έχει. Για να γίνει αυτό εφικτό πρώτα προστέθηκε στο storyboard του controller ένα UITableView στο όποιο σχεδιάστηκε το κελί για την επιλογή του χρήστη μ όλα τα απαραίτητα UIComponents. Πιο συγκεκριμένα αποτελείται από ένα UILabel για την προβολή του ονόματος της επιλογής και από ένα CheckBox για την ένδειξη εάν ένα φίλτρο είναι επιλεγμένο ή όχι. Στην συνέχεια δηλώθηκε στο table ότι ο MapLayersViewController εφαρμόζει και υλοποιεί τα πρωτοκολλά UITableView και UITableViewDataSource (ίδια διαδικασία που εφαρμόστηκε στο Search ViewController).

2. **Σχεδιασμός και αρχικοποίηση UITableHeaderView:**Θεωρούμε κάθε κατηγορία των χαρτογραφικών επιπέδων - φίλτρων ως μια ενότητα(Section) του UITableView.Έτσι λοιπόν προκειμένου να προβάλουμε όλες τις κατηγορίες όταν ο χρήστης ανοίγει την συγκεκριμένη οθόνη αρχικά δείχνουμε μόνο όλους τους UITableViewHeaderViews. Το συγκεκριμένο UIComponent είναι ένα επαναχρησιμοποιήσιμο View το οποίο τοποθετείτε στο επάνω μέρος μιας ενότητα table για να εμφανίζει πρόσθετες πληροφορίες για αυτήν. Η σχεδίαση του έγινε σε ειδικά διαμορφωμένο τύπο αρχείου το οποίο είναι υπεύθυνο για την σχεδίαση Custom Views , το λεγόμενο .xib .Αποτελείται από το UILabel για την προβολή του ονόματος της ενότητας και ένα UIButton για να ειδοποιείται ο MapLayers ViewController όταν πατιέται η ενότητα. Αφού έγινε η σχεδίαση του και φτιάχτηκε η αντίστοιχη κλάση, δηλώσαμε στο table ότι θα χρησιμοποιεί το παραπάνω view για κάθε ενότητα του.

```
tableView.dataSource = self
tableView.delegate = self
tableView.sectionHeaderHeight = 54
tableView.estimatedSectionHeaderHeight = 54
tableView.estimatedRowHeight = 50
tableView.rowHeight = 50
let headerNib = UINib.init(nibName: "MapLayersHeader", bundle: Bundle.main)
tableView.register(nib: headerNib,
                   withHeaderFooterViewClass: MapLayersHeader.self)
```

Εικόνα 20. Στιγμιότυπο από την δήλωση του tableView για την προβολή χαρτογραφικό επιπέδων

Για να δείχνουμε στην αρχή μόνο τα συγκεκριμένα Views και όχι τα κελιά των ενοτήτων θα πρέπει να δηλώνουμε στο table ότι κάθε ενότητα έχει 0 κελιά. Αυτό επιτυγχάνεται με την μέθοδο numberOfRowsInSection του πρωτοκόλλου UITableViewDelegate.Όταν ο χρήστης πατήσει μια ενότητα τότε ενημερώνεται ο controller και σε ένα πίνακα που κρατάει τις ανοιγμένες ενότητες την αποθηκεύει. Στην συνέχεια γίνεται ανανέωση του πίνακα σε αυτή την ενότητα και φαίνονται κανονικά τα κελιά. Αντίστοιχα αν ξαναπατηθεί το συγκεκριμένο HeaderView σβήνεται από τον πίνακα με τις ανοιγμένες ενότητες και τα κελιά για την συγκεκριμένη ενότητα γίνονται πάλι 0.

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    guard openedSections.contains(section) else {
        return 0
    }
    if section < data?.baseMapGroups.count ?? 0 {
        return data?.baseMapGroups[section].layers.count ?? 0
    } else {
        return data?.layerGroups[section - (data?.baseMapGroups.count ?? 0)].layers.count ?? 0
    }
}
```

Εικόνα 21.Στιγμιότυπο της μεθόδου numberOfRowsInSection για το tableView χαρτογραφικών επιπέδων

3. **Τροφοδότηση του UITableView με δεδομένα:** Τα χαρτογραφικά επίπεδα και τα φίλτρα τα οποία περιέχει ο πίνακας προέρχονται από τον server.Προκειμένου να γίνει η ανάκτηση τους γίνεται ασύγχρονο call στον server στην κλάση Layerhelper.Όταν ολοκληρωθεί το call ελέγχεται εάν υπάρχει κάποιο σφάλμα. Σε περίπτωση που δεν υπάρχει τότε τα δεδομένα αποθηκεύονται με την μορφή NSDictionary στα UserDefaults. Αυτό γίνεται με σκοπό την επόμενη φορά που θα γίνει το call και προκύψει κάποιο σφάλμα (πχ ο χρήστης να ναι εκτός σύνδεσης) να μπορούν να εμφανιστούν οι επιλογές που έχουν αποθηκευτεί τοπικά. Με αυτή την υλοποίηση εξυπηρετείται και το απαιτούμενο για την offline λειτουργεία. Στο σημείο αυτό θα πρέπει να αναφερθεί ότι τα UserDefaults είναι μια διεπαφή με την βάση δεδομένων για τις προεπιλογές του χρήστη, όπου αποθηκεύονται ζεύγη κλειδιών-τιμών.

```
if let resonseValue = response.result.value {
    self?.data = HuaSettings(with: resonseValue as? NSDictionary)
    UserDefaultsUtilities.shared.set(self?.data.dictionary as NSDictionary?, forKey: .huaSettings)
} else {
    self?.data = HuaSettings(with: UserDefaultsUtilities.shared.getNSDictionary(key: .huaSettings))
}
guard let kitchenerMapLayer = self?.data?.baseMapGroups.first?.layers.first else { return }
self?.layers.append(kitchenerMapLayer)
self?.layersToDownload.append(kitchenerMapLayer)
```

Εικόνα 22.Στιγμιότυπο από την μέθοδο τροφοδότησης των χαρτογραφικών επιπέδων

Αφού έχει ολοκληρωθεί η παραπάνω διαδικασία, ο MapLayersViewController μπορεί να πάρει τα δεδομένα και να τα περάσει στον πίνακα (ίδια διαδικασία με τον SearchViewController)

4. **Ειδοποίηση MenuViewController για αλλαγή χαρτογραφικού επιπέδου – φίλτρου:** Μέσω της μεθόδου didSelectRowAt του UITableViewDelegate ειδοποιούμαστε για την επιλογή ενός κελιού. Οπότε με το πάτημα της επιλογής λαμβάνουμε την θέση του κελίου και παίρνουμε το φίλτρο – επίπεδο που περιέχει. Στην συνέχεια μέσω closure το περνάμε στον MenuViewController ο οποίος με την σειρά του θα ειδοποιήσει τον MapViewController για τις ανάλογες ενέργειες.

4.6. Εναλλαγή γλώσσας

Η εφαρμογή υποστηρίζει την Ελληνική και την Αγγλική γλώσσα. Όταν ο χρήστης πατήσει τα αντίστοιχα εικονίδια στο πλαϊνό μενού τότε το περιεχόμενο της εφαρμογής θα ανανεωθεί σύμφωνα με την επιλογή του. Για να επιτευχθεί αυτό έχει δημιουργηθεί η κλάση LocaleHelper. Η συγκεκριμένη κλάση περιέχει την μεταβλητή language η οποία είναι προσβάσιμη σε όλες τις άλλες κλάσεις – UIViewControllers προκειμένου να γνωρίζουν όποτε χρειαστεί την επιλεγμένη γλώσσα. Η αρχικοποίηση της γίνεται στον initializer της κλάσης παίρνοντας από τα userDefaults το String για το κλειδί KITCHENERLOCALE.Έτσι η language ως enum μπορεί να έχει δυο επιλογές: Greek για Ελληνικά και other για τα Αγγλικά. Με σκοπό να γίνεται η αποθήκευση της εναλλαγής όταν πατάει το κουμπί ο χρήστης, έχει υλοποιηθεί η μέθοδος saveLanguage η οποία αποθηκεύει την γλώσσα της παραμέτρου της στα UserDefaults. Έχοντας στην διάθεση μας αυτή την κλάση μπορούμε οποιαδήποτε στιγμή σε κάθε UIViewController και Manager να γνωρίζουμε την επιλεγμένη γλώσσα προκειμένου να εμφανίσουμε τα δεδομένα στην σωστή μορφή.

```
enum Language: String {
    case greek
    case other
}

class LocaleHelper {
    static let shared = LocaleHelper()
    private let key = "KITCHENERLOCALE"
    var language: Language

    private init() {
        language = Language(rawValue: UserDefaults().object(forKey: key) as? String ?? "greek") ?? .greek
    }

    func saveLanguage(_ language: Language) {
        self.language = language
        UserDefaults().set(language.rawValue, forKey: key)
    }
}
```

Εικόνα 23. Κλάση LocaleHelper

4.7. Χαρτογραφικό υπόβαθρο.

Με την συγκεκριμένη επιλογή ο χρήστης μπορεί να παραμετροποιήσει την ορατότητα των φίλτρων – γεωγραφικών επιπέδων με σκοπό να συγκρίνει αλλαγές – μοτίβα με την τωρινή μορφή της Κύπρου. Με το πάτημα του κουμπιού ειδοποιείται ο MapViewController ο οποίος εμφανίζει στο κάτω μέρος της οθόνης ένα UISlider. Ο UISlider είναι ένα UIView αντικείμενο ελέγχου για την επιλογή μιας μεμονωμένης τιμής από ένα συνεχές εύρος τιμών. Στην συγκεκριμένη περίπτωση το εύρος τιμών είναι 0-100.Κάθε φορά που ο χρηστής αλλάζει την τιμή καλείται η μέθοδος reloadRenders η οποία αφαιρεί όλα τα overlays από τον χάρτη και τα ξανά τοποθετεί με καινούργιο alpha. Alpha ορίζεται η ποσότητα του transparency που θέλουμε να εφαρμόσουμε σε ένα overlay.Ένα overlay με 0 alpha σημαίνει ότι δεν είναι καθόλου ορατό στον χρήστη και αντίστοιχα με 1 ότι είναι πλήρες ορατό.

4.8. Ανατροφοδότηση (TakeCommentViewController)

Ο χρήστης της εφαρμογής προκειμένου να προσφέρει και αυτός από την πλευρά του επιπλέον γνώση και πληροφορίες με σκοπό την βελτίωση της εφαρμογής, μπορεί με λίγα βήματα να στείλει μία φωτογραφία, ένα σχόλιο για ένα σημείο στο οποίο θέλει να αναφερθεί.Αυτή είναι και η λειτουργικότητα που πραγματοποιεί η συγκεκριμένη σελίδα. Η υλοποίηση έγινε ως εξής:

- **Δημιουργία πεδίου εισαγωγής σχόλιου:** Προκειμένου να μπορεί ο χρήστης να εισάγει το σχόλιο του προστέθηκε στο storyboard του controller UITextView.Το συγκεκριμένο UIComponent ορίζει μια περιοχή πάνω στο ViewController η οποία περιέχει UIScrollView και υποστηρίζει την προβολή και επεξεργασία κειμένου πολλαπλών γραμμών. Μέσω των κατάλληλων πρωτόκολλων παίρνουμε ειδοποιήσεις για τα γεγονότα εισόδου του χρήστη και αποθηκεύουμε την τιμή που έχει πάρει το UITextView από τον χρήστη.

- **Επισύναψη φωτογραφίας:** Όταν ο χρήστης πατήσει το κουμπί της επισύναψης, ειδοποιείται ο controller προκείμενου να γίνουν οι κατάλληλες ενέργειες. Αρχικά ο UIAlertController είναι υπεύθυνος για να εμφανίσει στον χρήστη τις επιλογές που έχει σε μορφή alert.Πιο συγκεκριμένα ο χρήστης μπορεί να επιλέξει ανάμεσα σε επιλογή φωτογραφίας από την κάμερα και σε επιλογή φωτογραφίας από την βιβλιοθήκη του κινητού. Ο UIImage PickerViewController είναι ένας UIViewController που διαχειρίζεται τις διεπαφές συστήματος για λήψη φωτογραφιών, εγγραφή βίντεο και επιλογή αντικειμένων από την βιβλιοθήκη πολυμέσων του χρήστη. Όταν ο χρήστης επιλέξει την επισύναψη από κάμερα θα θέσουμε στον UIImagePickerView Controller ως τύπο πηγής την κάμερα του κινητού. Αντίστοιχα για την δεύτερη επιλογή θα τεθεί πηγή η βιβλιοθήκη πολυμέσων. Στην συνέχεια ορίζουμε ότι ο TakeCommentViewController θα εφαρμόζει το πρωτόκολλο UIImagePicker ControllerDelegate ώστε να ειδοποιηθούμε όταν ο χρήστης διαλέξει ή τραβήξει φωτογραφία.

- **Αποστολή:** Η αποστολή της ανατροφοδότησης του χρήστη γίνεται μέσω mail.Ο MFMailComposeViewController είναι ένας τυπικός UIViewController του οποίου η διεπαφή επιτρέπει στον χρήστη να διαχειρίζεται να επεξεργάζεται και να στέλνει μηνύματα mail.Έτσι όταν ο χρήστης πατήσει το κουμπί αποστολή θα πάρουμε αρχικά το περιεχόμενο της ανατροφοδότησης που έχει δώσει ο χρήστης και το χούμε αποθηκεύσει στην cache της συσκευής. Στην συνέχεια τα δεδομένα θα μεταβιβαστούν στον MFMail ComposeViewController μέσω της κατάλληλης μεθόδου. Επιπρόσθετα θα ενσωματωθούν οι συντεταγμένες της τοποθεσίας που αναφέρεται ο χρήστης όπως επίσης και ο παραλήπτης ο οποίος έχει συμφωνηθεί από τα προαπαιτούμενα του project.Έτσι ο χρήστης θα έχει την δυνατότητα εφόσον είναι συνδεδεμένος σε λογαριασμό mail, να στείλει τα παραπάνω δεδομένα. Τέλος θα πρέπει να σημειώσουμε ότι η αποστολή του mail είναι ευθύνη του λειτουργικού συστήματος iOS και της προεπιλεγμένης εφαρμογής για mail.

4.9. Πολιτική απορρήτου και όροι χρήσης

Σε κάθε εφαρμογή προτείνεται να προβάλλεται σε μια οθόνη οι όροι χρήσης και η πολιτική απορρήτου. Έτσι και στην συγκεκριμένη περίπτωση έχει δοθεί η δυνατότητα στον χρήστη να δεις τις συγκεκριμένες πληροφορίες όταν πατήσει τα ανάλογα κουμπιά από το πλαϊνό μενού. Από την στιγμή που μιλάμε για όρους χρήσης και πολιτική απορρήτου η μορφή των δεδομένων είναι δυναμική. Έτσι με σκοπό να μην βγαίνει update της εφαρμογής κάθε φορά που τροποποιείται κάθε όρος, επιλέχθηκε να λαμβάνουμε τα συγκεκριμένα δεδομένα μέσω URL.Πιο συγκεκριμένα το WKWebView είναι ένα αντικείμενο που εμφανίζει διαδραστικό περιεχόμενο web, όπως ένα πρόγραμμα περιήγησης εντός εφαρμογής. Για να χρησιμοποιήσουμε αυτό το αντικείμενο αρχικά του περνάμε του αρχικοποιούμε το URL στο οποίο θα περιηγηθούμε. Επιπρόσθετα βάζουμε ως παράμετρο στο URL την γλώσσα που χρησιμοποιεί ο χρήστης. Στην συνεχεία υλοποιώντας το πρωτόκολλο WKNavigation Delegate, μπορούμε να λαμβάνουμε ειδοποιήσεις για το πότε φορτώθηκε επιτυχώς το web περιεχόμενο και το πότε προέκυψε σφάλμα προκειμένου να εμφανίσουμε το κατάλληλο μήνυμα στον χρήστη.

5. Επιλογή Αποθήκευση περιοχής

5.1. DownloaderViewController

Ένα από τα βασικά απαιτούμενα του project είναι να μπορεί ο χρήστης να πλοηγείτε στον χάρτη ενώ δεν έχει πρόσβαση στο δίκτυο. Για το σκοπό υπήρχαν 3 επιλογές υλοποίησης:

- **Τοποθέτηση των δεδομένων μέσα στην εφαρμογή:** Εάν και αυτή η μέθοδος θα ήταν η πιο γρήγορη ως υλοποίηση, δεν προτιμήθηκε αφού το μέγεθος της εφαρμογής θα ήταν πολύ μεγάλο. Επιπλέον δεν θα δινόταν στον χρήστη η δυνατότητα να παραμετροποιήσει την περιοχή που θέλει να έχει αποθηκευμένη, όπως επίσης και να επιλέξει τα γεωγραφικά επίπεδα που επιθυμεί.
- **Λήψη δεδομένων από τον server κατά την πρώτη εκτέλεση της εφαρμογής:** Η επιλογή αυτή θα απαιτούσε από τον server την δημιουργία ενός API call ώστε να μας φέρνει όλα μαζί τα δεδομένα - tiles όταν του τα ζητήσουμε. Αυτό θα είχε ως αποτέλεσμα το παραπάνω request στον server να ήταν πολύ χρονοβόρο συνεπώς θα έπρεπε να διασπαστεί σε επιμέρους request.Αυτή η τακτική δεν προτιμήθηκε γιατί και σε αυτή την περίπτωση δεν θα υπήρχαν πολλές επιλογές παραμετροποίησης αλλά και θα απαιτούσε πιο πολύπλοκη δουλειά για το τμήμα του back end χωρίς πολλά οφέλη.
- **Χειροκίνητη αποθήκευση του κάθε tile στην μνήμη της συσκευής:** Με την υλοποίηση αυτή η ευθύνη για την αποθήκευση των tiles είναι αποκλειστικά της εφαρμογής. Πιο συγκεκριμένα κάθε φορά που λαμβάνουμε επιτυχημένα ένα tile από τον server, το αποθηκεύουμε τοπικά στην μνήμη ώστε να μην χρειαστεί να το καλέσουμε ξανά μετά. Η διαδικασία αποθήκευσης έχει και αναλυθεί σε προηγούμενο κεφάλαιο με την κλάση OfflineMapCache

Η τρίτη επιλογή είναι αυτή που επιλέχθηκε και μας έδωσε την δυνατότητα ώστε να δημιουργήσουμε την οθόνη αποθήκευση περιοχής. Ο παραπάνω UIViewController αποτελείται από:

- **MKMapview:** Το συγκεκριμένο αντικείμενο του χάρτη είναι υπεύθυνο ώστε να δείχνει στον χρήστη την περιοχή που πρόκειται να αποθηκευτεί. Αρχικά έχουμε προσθέσει το γεωγραφικό επίπεδο του Kitchener προκειμένου ο χρήστης να προσανατολίζεται πιο εύκολα. Στην συνέχεια έχουμε δηλώσει στον OfflineViewController να εφαρμόζει το πρωτόκολλο MKMapViewDelegate. Μέσω της μεθόδου regionDidChangeAnimated ειδοποιείται ο controller κάθε φορά που αλλάζει η ορατή περιοχή του χάρτη. Έτσι όταν ο χρήστης εφαρμώσει ένα gesture στον χάρτη και αλλάξει περιοχή, το πρωτόκολλο θα καλέσει την συγκεκριμένη μέθοδο και εμείς με την σειρά μας θα μπορούμε να πάρουμε τις συντεταγμένες του ορατού πλαισίου. Στη συνέχεια με την updateRegion που έχουμε φτιάξει παίρνουμε τα 4 άκρα της περιοχής με σκοπό να βρούμε όλα τα tiles σύμφωνα με τις παραπάνω συντεταγμένες λαμβάνοντας υπόψιν το zoom που έχει επιλέξει ο χρήστης.

```swift
func mapView(_ mapView: MKMapView, regionDidChangeAnimated animated: Bool) {
    updateRegion()
    messageView.text = "Σύνολο \(region?.count ?? 0) tiles | zoom: \(region?.zoomRange.min ?? 0) -> \(region?.zoomRange.max ?? 0)"
    limitMessage.isHidden = (region?.count ?? 500) < 500
    downloadBtn.isEnabled = (region?.count ?? 500) < 500
}
```

Εικόνα 24. Μέθοδο regionDidChangeAnimated

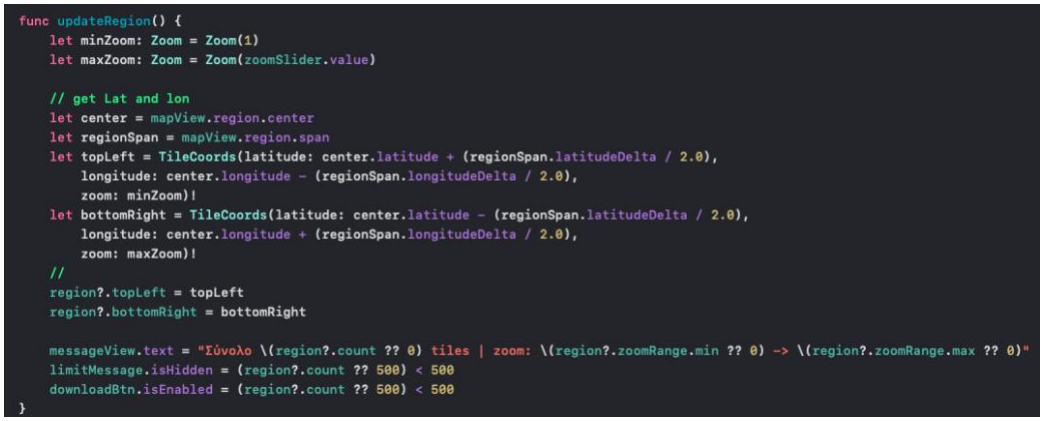
- **Επιλογή χαρτογραφικών επιπέδων:** Ο χρήστης έχει την επιλογή να παραμετροποιήσει τα γεωγραφικά επίπεδα που επιθυμεί να αποθηκεύσει για την συγκεκριμένη περιοχή. Για αυτό τον σκοπό έχει τοποθετηθεί ένα UIButton για την ενέργεια αυτή. Όταν το πατήσει ο χρήστης ενημερώνεται ο controller. Στην συνέχεια ο χρήστης μεταβιβάζεται στον SelectLayers ViewController.Ο συγκεκριμένος controller περιέχει το ίδιο UITableView με τα ίδια data που υπάρχει όταν ο χρήστης πατάει από το πλαϊνό μενού τα χαρτογραφικά επίπεδα (Map Layers ViewController). Στην συγκεκριμένη περίπτωση όμως για τα events αλληλεπίδρασης του χρήστη με το UItableView ενημερώνεται ο DownloaderViewController μέσω του Swift closure.Έτσι όταν επιλεχθεί ή αφαιρεθεί ένα επίπεδο καλείται η μέθοδος layers changed η οποία με την σειρά της αποθηκεύει σ ένα array όλα τα ids των χαρτογραφικών επιπέδων που έχει επιλέξει ο χρήστης για αποθήκευση.

- **ZoomSlider – UISlider:**Ο UISlider είναι ένα UIView – ένα αντικείμενο ελέγχου για την επιλογή μιας μεμονωμένης τιμής από ένα συνεχές εύρος τιμών. Για την συγκεκριμένη περίπτωση το εύρος τιμών είναι 1-19 και απεικονίζει το ελάχιστο και το μέγιστο zoom που μπορεί να επιλέξει ο χρήστης για αποθηκεύσει. Οι παραπάνω τιμές προκύπτουν από την παράμετρο z - zoom που βάζουμε κάθε φορά στο URL όταν θέλουμε να πάρουμε ένα tile και το 19 είναι το μέγιστο zoom που υποστηρίζει ο server.Έτσι κάθε φορά που ο χρήστης αλλάζει την τιμή του, καλείται η updateRegion προκειμένου να εντοπίσουμε τα tiles που πρέπει να κατεβούν σύμφωνα με την τιμή του zoom.Πιο συγκεκριμένα ο εντοπισμός των tiles γίνεται παίρνοντας τις γωνίες της ορατής περιοχής του χάρτη και υπολογίζοντας τις τιμές των tiles για κάθε zoom level. Ανεβάζοντας το επίπεδο του zoom level αυξάνεται εκθετικά και ο αριθμός των tiles όπως φαίνεται και στην παρακάτω φωτογραφία.

Level	# Tiles	Tile width (° of longitudes)	m / pixel (on Equator)	~ Scale (on screen)	Examples of areas to represent
0	1	360	156 412	1:500 million	whole world
1	4	180	78 206	1:250 million	
2	16	90	39 103	1:150 million	subcontinental area
3	64	45	19 551	1:70 million	largest country
4	256	22.5	9 776	1:35 million	
5	1 024	11.25	4 888	1:15 million	large African country
6	4 096	5.625	2 444	1:10 million	large European country
7	16 384	2.813	1 222	1:4 million	small country, US state
8	65 536	1.406	610.984	1:2 million	
9	262 144	0.703	305.492	1:1 million	wide area, large metropolitan area
10	1 048 576	0.352	152.746	1:500 thousand	metropolitan area
11	4 194 304	0.176	76.373	1:250 thousand	city
12	16 777 216	0.088	38.187	1:150 thousand	town, or city district
13	67 108 864	0.044	19.093	1:70 thousand	village, or suburb
14	268 435 456	0.022	9.547	1:35 thousand	
15	1 073 741 824	0.011	4.773	1:15 thousand	small road
16	4 294 967 296	0.005	2.387	1:8 thousand	street
17	17 179 869 184	0.003	1.193	1:4 thousand	block, park, addresses
18	68 719 476 736	0.001	0.596	1:2 thousand	some buildings, trees
19	274 877 906 944	0.0005	0.298	1:1 thousand	local highway and crossing details
20	1 099 511 627 776	0.00025	0.149	1:5 hundred	A mid-sized building

Εικόνα 25. Πίνακας αναπαράστασης της εκθετικής αύξησης των tiles ανά επίπεδο

Η στήλη tiles ορίζει τα πόσα tiles χρειάζονται στο συγκεκριμένο zoom προκειμένου να απεικονιστεί ολόκληρος ο κόσμος. Είναι αντιληπτό ότι εάν ο χρήστης έχει το χάρτη σε χαμηλό zoom και επιλέξει από τον slider μια μεγάλη τιμή για αποθήκευση τότε το εύρος του zoom θα μεγαλώσει όπως επίσης και η ποσότητα των tiles. Από την στιγμή που μιλάμε για API calls στον server η ποσότητα αυτή θα πρέπει ναι είναι περιορισμένη προκειμένου να μπορεί να ανταπεξέλθει ο server.Λαμβάνοντας υπόψιν ότι κάθε tile για κάθε ένα γεωγραφικό επίπεδο είναι ένα call αποφασίστηκε να υπάρχει το όριο των 500 tiles σε κάθε διαδικασία αποθήκευσης περιοχής.



```
func updateRegion() {
    let minZoom: Zoom = Zoom(1)
    let maxZoom: Zoom = Zoom(zoomSlider.value)

    // get Lat and lon
    let center = mapView.region.center
    let regionSpan = mapView.region.span
    let topLeft = TileCoords(latitude: center.latitude + (regionSpan.latitudeDelta / 2.0),
        longitude: center.longitude - (regionSpan.longitudeDelta / 2.0),
        zoom: minZoom)!
    let bottomRight = TileCoords(latitude: center.latitude - (regionSpan.latitudeDelta / 2.0),
        longitude: center.longitude + (regionSpan.longitudeDelta / 2.0),
        zoom: maxZoom)!
    //
    region?.topLeft = topLeft
    region?.bottomRight = bottomRight

    messageView.text = "Σύνολο \(region?.count ?? 0) tiles | zoom: \(region?.zoomRange.min ?? 0) -> \(region?.zoomRange.max ?? 0)"
    limitMessage.isHidden = (region?.count ?? 500) < 500
    downloadBtn.isEnabled = (region?.count ?? 500) < 500
}
```

Εικόνα 26. Μέθοδος updateRegion

- **Ένδειξη των tiles:** Όπως αναφέρθηκε και πριν είναι απαραίτητο η ποσότητα των tiles που κατεβάζει ο χρήστης να έχει ένα μέγιστο όριο. Κρίνεται λοιπόν αναγκαίο ο χρήστης να ενημερώνεται κάθε φορά που αλλάζει η τιμή τους. Έτσι κάτω από τον zoomSlider έχει προστεθεί ένα UILabel που η τιμή του ανανεώνεται κάθε φορά που αλλάζει η ορατή περιοχή ή το zoom.Σε περίπτωση που ο χρήστης με τις επιλογές του ξεπερνάει το επιτρεπτό όριο των 500 tiles, ένα ακόμη UILabel εμφανίζεται όπου με κόκκινα γράμματα προειδοποιεί τον χρήστη για το σφάλμα και απενεργοποιεί το κουμπί της αποθήκευσης.

5.2 Διαδικασία λήψης δεδομένων περιοχής

Όταν ο χρήστης πατήσει το κουμπί της αποθήκευσης θα ξεκινήσει η διαδικασία λήψης των tiles.Όσον αφορά το λειτουργικό της κομμάτι αποτελείται από τα εξής μέρη:

- **RegionDownloader**: Προκειμένου να κατεβάσουμε όλα τα tiles της περιοχής χρησιμοποιούμε την κλάση RegionDownloader. Η συγκεκριμένη κλάση παίρνει ως παράμετρο στον constructor της την περιοχή που θέλουμε να κατεβάσουμε και ένα αντικείμενο που εφαρμόζει το MapCaheProtocol με σκοπό να πει στην κλάση το τρόπο απόκτησης των tiles. Έτσι τοποθετείτε το region που είχε αποθηκευτεί από την μέθοδο updateRegion και δημιουργείται ένα αντικείμενο της κλάσης OfflineMapCache με την ίδια ακριβώς διαδικασία πού έχουμε αναλύσει σε προηγούμενο κεφάλαιο. Στην συνέχεια δηλώνουμε στον downloader ότι εφαρμόζουμε το πρωτόκολλο RegionDownloader Delegate έτσι ώστε να γνωρίζουμε την πρόοδο της διαδικασίας μέσω των αντίστοιχων μεθόδων. Τέλος με την μέθοδο start ξεκινάει η διαδικασία της λήψης
- **Λήψη χαρτογραφικών φίλτρων:** Για να γίνει λήψη των δεδομένων που δεν είναι τύπου tiles όπως ορίζει ο server η διαχείριση είναι διαφορετική. Μέσα σε μια for loop γίνεται προσπέλαση των συγκεκριμένων layers και για κάθε να από αυτά γίνεται λήψη για όλες τις συντεταγμένες των tiles της ορατής περιοχής. Επειδή τα requests στον server γίνονται ασύγχρονα δεν μπορούμε να γνωρίζουμε πότε θα τελειώσουν όλα τα requests που προέκυψαν από την for loop. Για το συγκεκριμένο σκοπό χρησιμοποιείται το DispatchGroup το οποίο ορίζεται ως μια ομάδα εργασιών που

παρακολουθείτε ως μια μονάδα. Στην συγκεκριμένη περίπτωση η ομάδα εργασιών αποτελείται από τα requests στον server.Για να εισέλθει μια τέτοια εργασία στο DispatchGroup χρησιμοποιούμε την μέθοδο enter, ενώ όταν ειδοποιούμαστε από το Swift closure ότι ολοκληρώθηκε το συγκεκριμένο call, καλούμε την μέθοδο leave προκειμένου να γνωρίζει το dispatchGroup ότι τελείωσε η εκτέλεση της συγκεκριμένης εργασίας. Με την ολοκλήρωση όλων των εργασιών καλείται το Swift closure της μεθόδου notify.Ύστερα από αυτό το βήμα το μόνο που απομένει είναι να ολοκληρωθεί η λήψη των δεδομένων τύπου tiles.

```
func downloadSublayers() {
    DispatchQueue.global(qos: .background).async {
        let dispatchGroup = DispatchGroup()

        for layer in self.sublayersToDownload {
            let url = WMSHelper.shared.mapLayersUrl.replacingOccurrences(of: "%s", with: layer)
            let overlay = WMSMKTileOverlay(urlArg: url, useMercatorArg: true, reversed: false)
            for range: TileRange in self.region?.tileRanges() ?? [] {
                for tileCoords: TileCoords in range {

                    let mktileOverlayPath = MKTileOverlayPath(tileCoords: tileCoords)
                    dispatchGroup.enter()
                    overlay.loadTile(at: mktileOverlayPath) { (data, _) in
                        dispatchGroup.leave()
                    }

                }
            }
        }
        dispatchGroup.notify(queue: .main) { [weak self] in
            self?.hasDownloadedSublayers = true
            self?.handleCompletedDownload()
        }
    }
}
```

Εικόνα 27. Μέθοδος downloadSublayers

- **Κατέβασμα των Tiles που έχουν αποτύχει:** Όταν ολοκληρωθεί η διαδικασία της λήψης δεδομένων μέσω του πρωτοκόλλου RegionDownloaderDelegate καλείται η didFinishDownload.Μέσω αυτής της μεθόδου μπορούμε να δούμε ποια calls στον server έχουν αποτύχει. Με αφορμή αυτό δημιουργήθηκε η downloadFailedTiles η οποία δέχεται ως παράμετρο αυτόν τον πίνακα και μέσα σε μια for loop γίνεται η λήψη των συγκεκριμένων tiles χειροκίνητα.

```swift
func downloadFailedTiles(tiles: [MKTileOverlayPath]) {
    let dispatchgroup = DispatchGroup()
    var successCalls = 0
    tiles.forEach {
        dispatchgroup.enter()
        mapCache?.loadTile(at: $0, result: { (data, error) in
            if error != nil {
                successCalls += 1
            }
            dispatchgroup.leave()
        })
    }

    dispatchgroup.notify(queue: .main) {
        debugPrint("completed \(successCalls)")
    }
}
```

Εικόνα 28. Μέθοδος downloadFailedTiles

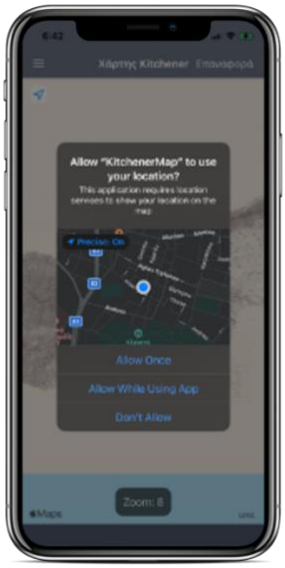
Προκειμένου ο χρήστης να ενημερώνεται για την πορεία της διαδικασίας η συγκεκριμένη οθόνη περιέχει τα εξής :

- **UILabel** για την γραπτή ένδειξη της πορείας της λήψης. Οι τιμές που μπορεί να πάρει είναι: ”Λήψη δεδομένων" όταν είναι ακόμα σε εξέλιξη και “Επιτυχής λήψη περιοχής” όταν ολοκληρωθεί.
- **UIProgressView**: για την γραφική ένδειξη της πορείας. Το UIProgressView είναι ένα UIView αντικείμενο το όποιο είναι υπεύθυνο για την απεικόνιση της προόδου μιας εργασίας με την πάροδο του χρόνου. Έτσι εφαρμόζοντας το πρωτόκολλο Region DownloaderDelegate, μέσω της μεθόδου didDownloadPercentage ενημερωνόμαστε κάθε φορά που αλλάζει το ποσοστό ολοκλήρωσης της διαδικασίας και περνάμε την τιμή στο progressview.
- **UIButton για την επιστροφή του χρήστη στην αρχική σελίδα:** Το συγκεκριμένο κουμπί εμφανίζεται μόνο όταν ολοκληρωθεί η λήψη προκειμένου ο χρήστης να μπορεί να ανακατευθυνθεί στην αρχική οθόνη. Κατά την διάρκεια της λήψης ο μόνος τρόπος για αυτό είναι πατώντας το κουμπί επιστροφή το οποίο ανακατευθύνει στην προηγούμενη σελίδα και ακυρώνει την λήψη.

6. Παράδειγμα χρήσης της εφαρμογής

6.1. Πρώτη εκτέλεση της εφαρμογής.

Όταν ο χρήστης εκτελέσει για πρώτη φορά την εφαρμογή ύστερα από την εγκατάσταση της, καλείται να επιλέξει για το εάν θα δώσει δικαιώματα τοποθεσίας στην εφαρμογή. Αυτή η διαδικασία απαιτείται προκειμένου να ξέρει η εφαρμογή εάν μπορεί να εμφανίζει την τοποθεσία του χρήστη στο χάρτη.



Εικόνα 29. Οθόνη για τα δικαιώματα τοποθεσίας

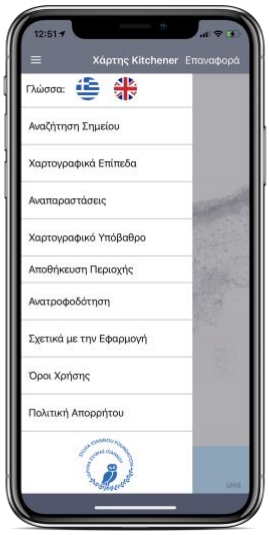
6.2 Κεντρική οθόνη με τον χάρτη του Kitchener

Ύστερα από την παραπάνω διαδικασία ο χρήστης ανακατευθύνετε στην κεντρική οθόνη της εφαρμογής. Πιο συγκεκριμένα εμφανίζεται ο χάρτης του Kitchener με εφαρμοσμένο από προεπιλογή το χαρτογραφικό επίπεδο «Μωσαϊκό πρωτογενών πινακίδων Kitchener». Εάν ο χρήστης έχει δώσει δικαιώματα τοποθεσίας στην εφαρμογή και βρίσκεται στην Κύπρο τότε η κάμερα του χάρτη θα εστιάσει πάνω στην θέση του.

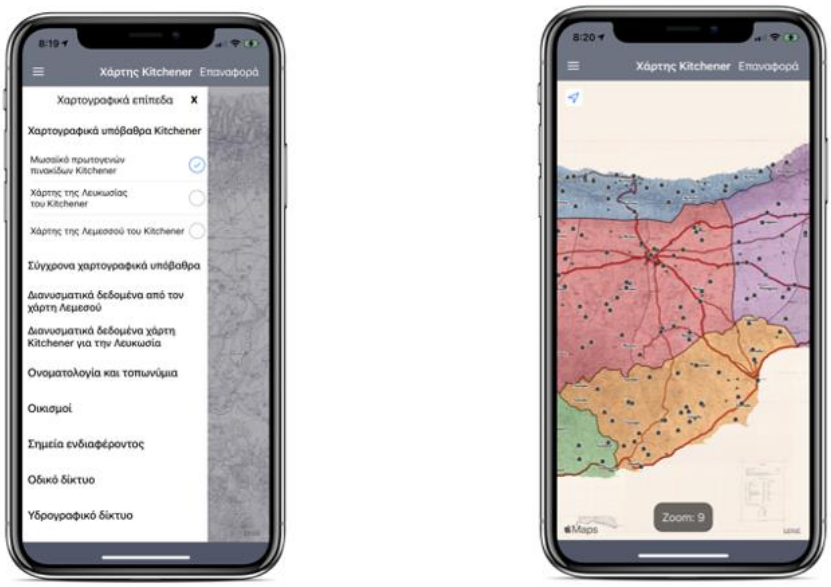
Εικόνα 30. Οθόνη χάρτη

6.3 Πλαϊνό Μενού

Ο χρήστης πατώντας το πάνω αριστερά κουμπί μεταφέρεται στο πλαϊνό μενού όπου μπορεί να κάνει διάφορες ενέργειες όπως έχει αναλυθεί και σε προηγούμενα κεφάλαια. Εφαρμόζοντας τα χαρτογραφικά φίλτρα και επίπεδα της επιλογής του το αποτέλεσμα θα είναι το εικονιζόμενο.



Εικόνα 31. Οθόνη με το πλαϊνό μενού



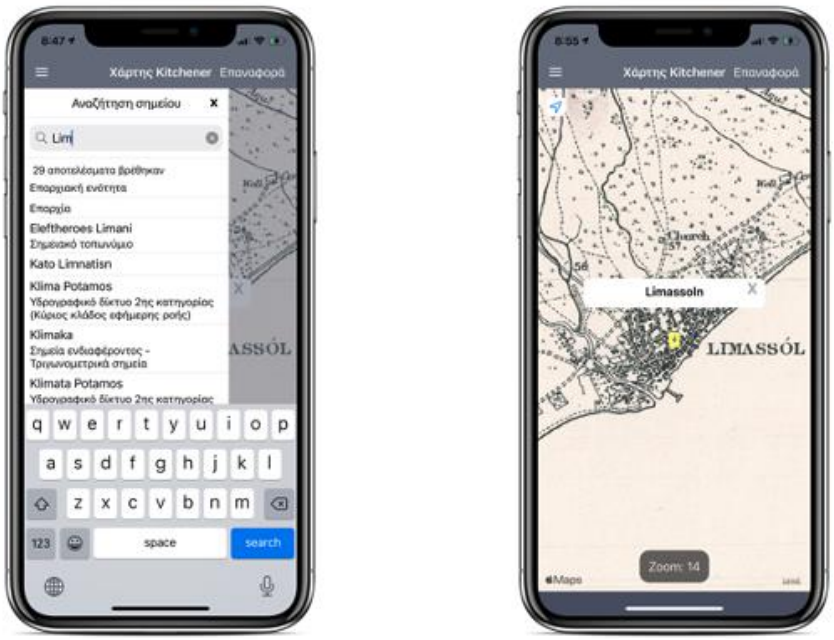
Εικόνα 32. Εφαρμογή χαρτογραφικών επιπέδων στον χάρτη

Με την επιλογή χαρτογραφικό υπόβαθρο ο χρήστης μπορεί να αυξομείωσει την ορατότητα των χαρτογραφικών επιπέδων μέσω του slider και να συγκρίνει τον χάρτη ανά επίπεδο.

Εικόνα 33. Εφαρμογή χαρτογραφικού υπόβαθρου

6.4 Αναζήτηση σημείου

Ο χρήστης πληκτρολογώντας το όνομα ενός σημείου μπορεί μέσω της αναζήτησης να βρει την τοποθεσία του πάνω στον χάρτη. Έτσι αν γράψει τα αρχικά της Λεμεσού, θα του εμφανιστούν όλα τα ανάλογα αποτελέσματα που ταιριάζουν. Ύστερα πατώντας την επιλογή που επιθυμεί θα κατευθυνθεί στην περιοχή που ορίζουν οι συντεταγμένες του σημείου.



Εικόνα 34. Αναζήτηση σημείου στον χάρτη

6.5 Feedback χρήστη.

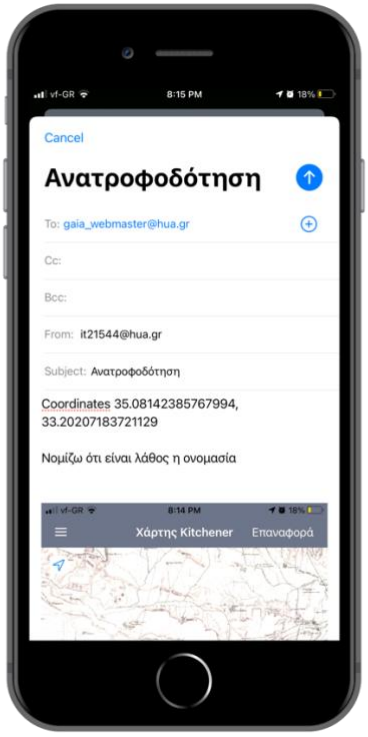
Για την βελτίωση της εφαρμογής και την ομαλή εμπειρία χρήσης, όπως αναφέρθηκε υπάρχει δυνατότητα feedback από τον χρήστη. Σαν πρώτο βήμα για την επιλογή του σημείου, πατώντας παρατεταμένα πάνω στον χάρτη ανεξαρτήτως χαρτογραφικού επίπεδου εμφανίζεται το παρακάτω annotation, το οποίο προτρέπει τον χρήστη να πατήσει πάνω του εάν θέλει να αφήσει κάποιο σχόλιο.

Εικόνα 35. Επιλογή σημείου στον χάρτη για feedback

Αφού ο χρήστης επιλέξει και πατήσει πάνω στο σημείο ανακατευθύνεται στην οθόνη αναφοράς. Εκεί μπορεί να γράψει το σχόλιο του και να επισυνάψει την φωτογραφία σχετικά με το πρόβλημα επιλέγοντας είτε εικόνα από κάμερα βιβλιοθήκη της συσκευής.

Εικόνα 36. Οθόνη feedback

Πατώντας το κουμπί αποστολή δημιουργείται αυτόματα ένα email το οποίο συμπεριλαμβάνει όλες τις πληροφορίες που έχει εισάγει όπως επίσης και τις συντεταγμένες του σημείου.Το συγκεκριμένο e-mail μπορεί να το στείλει στον παραλήπτη που έχει προεπιλεχθεί σύμφωνα με τις απαιτήσεις του Project.



Εικόνα 37. Αποστολή mail ανατροφοδότηση

6.6 Αποθήκευση περιοχής.

Πατώντας την παραπάνω επιλογή από το πλαϊνό μενού ο χρήστης έχει τη δυνατότητα να επιλέξει μία περιοχή την οποία θέλει να αποθηκεύσει. Ως πρώτο βήμα καλείται να εστιάσει στην περιοχή που επιθυμεί πάνω στο χάρτη και να προσδιορίσει μέσω του slider μέχρι ποιο zoom θέλει να γίνει η αποθήκευση.

Εικόνα 38. Επιλογή περιοχής για αποθήκευση

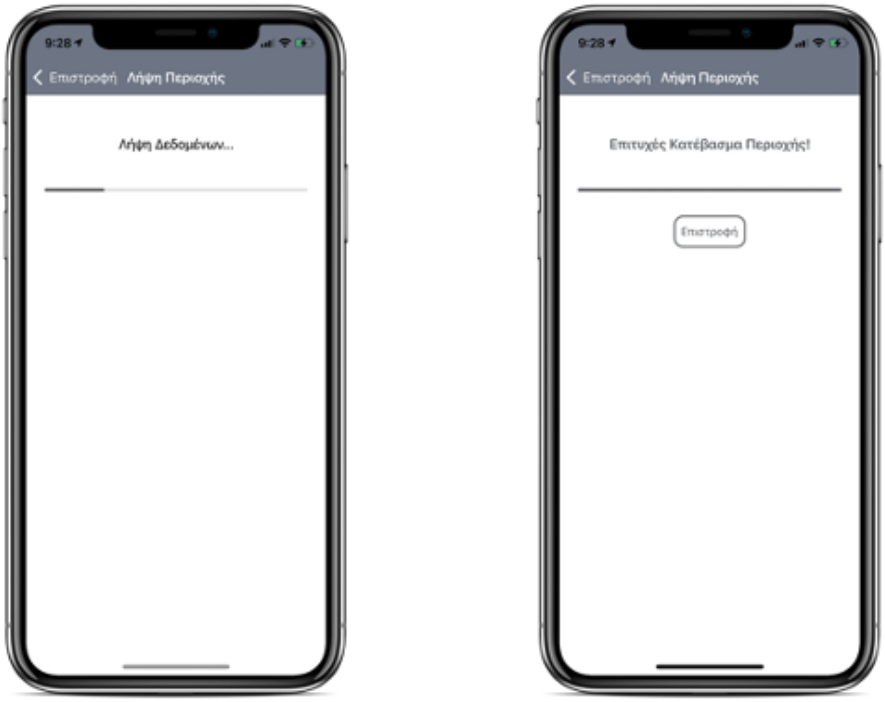
Εάν ο χρήστης με τις συγκεκριμένες επιλογές υπερβεί το μέγιστο όριο των tiles θα εμφανιστεί το ανάλογο μήνυμα.

Εικόνα 39. Εμφάνιση μέγιστου ορίου των tiles

Επιπρόσθετα πατώντας το κουμπί επιλογή χαρτογραφικών επιπέδων ο χρήστης έχει τη δυνατότητα να επιλέξει σε ποια χαρτογραφικά επίπεδα θα έχει πρόσβαση όταν βρεθεί εκτός σύνδεσης

Εικόνα 40. Επιλογή χαρτογραφικών επιπέδων για αποθήκευση

Όταν ο χρήστης εισάγει όλες τις απαραίτητες πληροφορίες, πατώντας το κουμπί λήψη περιοχής θα ξεκινήσει η διαδικασία της αποθήκευσης. Στην παρακάτω οθόνη μπορεί να δει την πρόοδο της και να ενημερωθεί όταν ολοκληρωθεί με επιτυχία.



Εικόνα 41. Οθόνη λήψης περιοχής

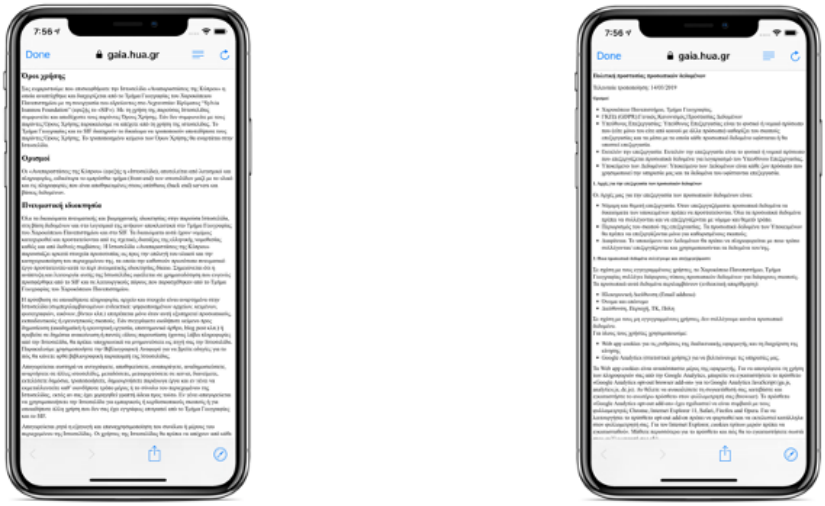
6.7 Πληροφορίες σχετικά με την εφαρμογή

Ο χρήστης έχει τη δυνατότητα να δει πληροφορίες σχετικά με την εφαρμογή όπως επίσης και με τον χάρτη του Kitchener. Πατώντας την αντίστοιχη επιλογή προκύπτει η παρακάτω οθόνη.

Εικόνα 42. Οθόνη με πληροφορίες της εφαρμογής

6.8 Όροι χρήσης και πολιτική απορρήτου

Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο ο χρήστης πρέπει να έχει πρόσβαση στους όρους χρήσης και στην πολιτική απορρήτου της εφαρμογής. Πατώντας τις αντίστοιχες επιλογές ο χρήστης ανακατευθύνετε σε οθόνες τύπου Safari προκειμένου να δει το ανάλογο περιεχόμενο.



Εικόνα 43. Οθόνες με τους όρους χρήσης και την πολιτική απορρήτου

7. Συντήρηση και βελτίωση της εφαρμογής

Λαμβάνοντας υπόψιν ότι η συγκεκριμένη εφαρμογή θα κυκλοφορήσει κανονικά στο App Store, με κριτήριο την βέλτιστη εμπειρία χρήσης, κρίνεται αναγκαία η συντήρηση και η βελτίωση της εφαρμογής. Αν εκμεταλλευτούμε την υποδομή της εφαρμογής, μερικά από τα βήματα προκειμένου επιτευχθεί ο παραπάνω στόχος είναι τα παρακάτω:

- **Εντοπισμός και διόρθωση πιθανών σφαλμάτων εφαρμογής**: Δεδομένου ότι χρησιμοποιούμε το Crashlytics framework, με την παρακολούθηση των crash reports στην αντίστοιχη πλατφόρμα μπορούμε να εντοπίσουμε εάν έχει προκύψει κάποιο σφάλμα στην εφαρμογή. Επιπρόσθετα από την στιγμή που θα έχουμε πληροφορίες σχετικά με την έκδοση του λογισμικού και το μοντέλο της συσκευής, θα μας είναι πιο εύκολο να περιορίσουμε και να διορθώσουμε το σφάλμα. Τέλος σε συνδυασμό με την λειτουργία feedback η διαδικασία γίνεται πιο άμεση με τον χρήστη και πιο αποτελεσματική.

- **Προσθήκη χαρτογραφικών επιπέδων:** Οι επιλογές των χαρτογραφικών επιπέδων προέρχονται από το API. Από τη στιγμή που το περιεχόμενο είναι δυναμικό μπορούν να προστεθούν επίπεδα χωρίς να χρειάζεται να γίνει αναβάθμιση της εφαρμογής.

- **Απλοποίηση του πλαϊνού μενού:** Προκειμένου η εφαρμογή να γίνει προσιτή σε όλους τους χρήστες μπορεί να προστεθεί διακόπτης ο οποίος με την ενεργοποίηση του θα απλοποιεί το μενού. Αυτή η διαδικασία θα πρέπει να υλοποιηθεί και από την μεριά του server ώστε να στέλνει τα δεδομένα για το απλοποιημένο μενού.

- **Παρακολούθηση αναβαθμίσεων λειτουργικού συστήματος:** Κρίνεται απαραίτητο σε κάθε σημαντική αναβάθμιση του iOS, να γίνεται έλεγχος για το εάν κάποια αλλαγή επηρεάζει την λειτουργία της εφαρμογής. Ακόμη μέσω των αναβαθμίσεων του λειτουργικού συστήματος μπορούμε να εκμεταλλευτούμε τις νέες τεχνολογίες και να βελτιώσουμε την εφαρμογή.

- **Βελτίωση της υποστήριξης για iPad:** Η συγκεκριμένη έκδοση της εφαρμογής υποστηρίζει την εκτέλεση σε iPad, όμως υπάρχουν περιθώρια βελτίωσης όσον αφορά την λειτουργία αυτή. Λαμβάνοντας υπόψιν το μεγαλύτερο μέγεθος οθόνης των συγκεκριμένων συσκευών μπορεί να προστεθεί η λειτουργία landscape προκειμένου ο χρήστης να μπορεί να βλέπει περισσότερο περιεχόμενο χωρίς να χρειάζεται να κάνει επιπλέον κινήσεις πάνω στον χάρτη.

8. Βιβλιογραφία

8.1. Βιβλία

[1] Tjeerd in 't Veen, "Swift In Depth",2018.

[2] D. Scott, "GIS for Web Developers", 2007.

[3] Victor Olaya,"Introduction to iOS",2018.

[4] Jonathan E. Campbell & Michael Shin,"Geographic Information System Basics",2012.

[5] Rene Cacheaux & Josh Berlin, "Advanced iOS App Architecture (Second Edition)",2019

8.2. Διαδικτυακά άρθρα

1. **Developer-Apple:**
    a. https://developer.apple.com/xcode/ide/
    b. https://developer.apple.com/documentation/foundation
    c. https://developer.apple.com/documentation/uikit
    d. https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemTechnology/SystemTechnology.html

    (Τελευταία επίσκεψη 27 Οκτωβρίου 2020)

2. **Cocoapods:**
    a. https://cocoapods.org

    (Τελευταία επίσκεψη 27 Οκτωβρίου 2020)

3. **Wiki-OpenStreetMap**:
    a. https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames
    b. https://wiki.openstreetmap.org/wiki/Slippy_Map

    (Τελευταία επίσκεψη 27 Οκτωβρίου 2020)

4. **Swift:**
    a. https://swift.org/documentation/

    (Τελευταία επίσκεψη 27 Οκτωβρίου 2020)

5. **Sylvia Ioannou Foundation:**
    a. https://sylviaioannoufoundation.org/en/academic-programmes/research-programmes/kitcheners-survey-cyprus/
    (Τελευταία επίσκεψη 27 Οκτωβρίου 2020)

6. **Dotnettricks:**
    a. https://www.dotnettricks.com/learn/xamarin/understanding-xamarin-ios-build-native-ios-app
    (Τελευταία επίσκεψη 27 Οκτωβρίου 2020)

7. **Insights – Stackoverflow:**
    a. https://insights.stackoverflow.com/survey/2015
    (Τελευταία επίσκεψη 27 Οκτωβρίου 2020)

8. **Hackernoon:**
    a. https://hackernoon.com/application-life-cycle-in-ios-12b6ba6af78b
    (Τελευταία επίσκεψη 27 Οκτωβρίου 2020)

9. **Intellipaat**:
    a. https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/
    (Τελευταία επίσκεψη 27 Οκτωβρίου 2020)