ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

Πρόγραμμα Σπουδών

Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα

Διπλωματική Εργασία

Σχεδιασμός και ανάπτυξη λογισμικού ηλεκτρονικής περιήγησης με τεχνολογίες φορητών συσκευών για τουριστικούς χώρους ενδιαφέροντος

Στυλιανός Κιούσης

Επιβλέπων καθηγητής: Ιωάννης Καρύδης

Ερμούπολη, Ιούλιος 2018

Σχεδιασμός και ανάπτυξη λογισμικού ηλεκτρονικής περιήγησης με τεχνολογίες φορητών συσκευών για τουριστικούς χώρους ενδιαφέροντος

Κιούσης Στυλιανός

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:
Ιωάννης Καρύδης
Μέλος Σ.Ε.Π. του Ε.Α.Π.

Συν-Επιβλέπων Καθηγητής:
Ευάγγελος Σακκόπουλος
Μέλος Σ.Ε.Π. του Ε.Α.Π.

Ερμούπολη, Ιούλιος 2018

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Ιωάννη Καρύδη για την εμπιστοσύνη που μου έδειξε σε όλη τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας και για την υπομονή του στη δική μου ασυνέπεια.

Επίσης θα ήθελα να αφιερώσω την εργασία σε όσους επέμειναν για την ολοκλήρωσή της έως και την τελευταία στιγμή.

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως αντικείμενο τη δημιουργία μιας ολοκληρωμένης και πλήρως λειτουργικής εφαρμογή τουριστικού οδηγού για φορητές συσκευές καθώς και τη διαδικτυακή υπηρεσία (Web Service) με το οποίο η εφαρμογή θα αλληλεπιδρά με τα δεδομένα. Παράλληλα μελετώνται προσεγγίσεις υλοποίησης εφαρμογών για φορητές συσκευές και η επικοινωνία αυτών με διαδικτυακές υπηρεσίες και RESTful APIs.

Η υλοποίηση αφορά μια διαπλατφορμική εφαρμογή (cross-platform), εκτελέσιμη στα δύο επικρατέστερα λειτουργικά συστήματα φορητών συσκευών, Android και Apple iOS, ενώ το αντίστοιχο web API έχει δημιουργηθεί με βάση τις προδιαγραφές openAPI v3.

Εκτός από την κύρια λειτουργικότητα της εφαρμογής, που είναι η εύχρηστη και καλαίσθητη προβολή σημείων τουριστικού ενδιαφέροντος (πληροφορίες, οικισμοί, αξιοθέατα, παραλίες) για το νησί της Σύρου, κατά την ανάπτυξη έχουν ληφθεί υπόψη δύο επιπλέον σημεία.

α) Η αρχιτεκτονική της είναι τέτοια ώστε με αλλαγή των δεδομένων στη βάση δεδομένων να μπορεί να επαναχρησιμοποιηθεί με ελάχιστες ρυθμίσεις σε αρχείο ρυθμίσεων για την προβολή οποιασδήποτε τουριστικής περιοχής χωρίς αλλαγές στον κώδικα.

β) Η εφαρμογή θα αποστέλλει ανωνυμοποιημένα γεωχωρικά δεδομένα των περιηγήσεων για περαιτέρω στατιστική μελέτη ή πιθανή επέκταση.

Λέξεις – Κλειδιά

Ταξιδιωτικός οδηγός, RESTful API, διαδικτυακή υπηρεσία, διαπλατφορμική εφαρμογή, φορητές συσκευές, Android, iOS

Abstract

This thesis aims at creating an integrated and fully functional application of a tourist guide for mobile devices as well as the Web Service with which the application will interact with the data. At the same time, application implementation approaches for mobile devices and their communication with online services and RESTful APIs are being studied.

The implementation is a cross-platform application, executable on the two most popular operating systems for handheld devices, Android and Apple iOS. The corresponding web API is built based on the openAPI v3 specifications.

In addition to the main functionality of the application, which is the easy to use and tasteful sightseeing of tourist sites (information, settlements, sights, beaches) for the island of Syros, two additional points have been taken into account during the development.

a) Project architecture is such that with a change of data in the database it can be reused with minimum settings in a configuration file for viewing any tourist area without changes in the code.

b) The application will send anonymized geospatial data of the tours for further statistical study or possible extension.

Keywords

Travel guide, RESTful API, Web Service, cross-platform, mobile app, Android, iOS

Περιεχόμενα

Κατάλογος Εικόνων

Συντομογραφίες και Ακρωνύμια

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
PHP	PHP: Hypertext Preprocessor
PWA	Progressive Web Apps
REST	Representational State Transfer
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
UI	User Interface
UX	User Experience
URI	Uniform Resource Identifier
WSDL	Web Services Description Language
XML	eXtensible Markup Language

Κεφάλαιο 1ο - Εισαγωγή

1.1 Γενικά

Η χρήση φορητών συσκευών έχει γίνει αναπόσπαστο κομμάτι της καθημερινότητας μας. Εκατομμύρια εφαρμογές όλων των ειδών χρησιμοποιούνται από δισεκατομμύρια ανθρώπους καθημερινά [1]. Ειδικά στην κατηγορία των τουριστικών οδηγών, χιλιάδες εφαρμογές για όλους τους τουριστικούς προορισμούς είναι διαθέσιμες στα app stores [2][3] καθώς τα πλεονεκτήματα ενός τουριστικού οδηγού σε φορητή συσκευή είναι αυτονόητα. Ο ταξιδιώτης μπορεί να βρει την πληροφορία άμεσα, γρήγορα και εύκολα με τη χρήση της κατάλληλης εφαρμογής.

Η ανάπτυξη μιας τέτοιας εφαρμογής, σε ένα τόσο ανταγωνιστικό περιβάλλον, για τις δύο επικρατούσες πλατφόρμες δεν είναι εύκολη. Τα λειτουργικά συστήματα Android και iOS είναι ασύμβατα μεταξύ τους. Η ανάπτυξη εγγενών εφαρμογών σε αυτά, απαιτεί διαφορετικές γλώσσες προγραμματισμού και διαφορετικά εργαλεία προγραμματισμού (IDE). Κάθε πλατφόρμα παρέχει στους προγραμματιστές και υποστηρίζει το δικό της SDK και διαφορετικά APIs επικοινωνίας με τα υποσυστήματα και τους αισθητήρες των συσκευών και επιπλέον έχουν διαφορετικά γραφικά στοιχεία ελέγχου (GUI) και διαφορετικές οδηγίες (Quality guidelines) εμφάνισης [4][5]. Το Android με γλώσσες προγραμματισμού Java και Kotlin, προτεινόμενο IDE το Android Studio και διεπαφές χρήστη στηριζόμενες στο Material Design [6] ενώ το Apple iOS γλώσσες προγραμματισμού Objective C και Swift, IDE το Xcode και GUI βασισμένο στο UIKit [7].

Το αποτέλεσμα είναι το αυξημένο κόστος υλοποίησης, καθώς η ανάπτυξη απαιτεί διαφορετικές ομάδες εξειδικευμένων προγραμματιστών και αυξημένο χρόνο ακόμα και αν μια ομάδα μπορεί να αναλάβει την υλοποίηση και στις δύο πλατφόρμες.

Η δυσκολίες στην ανάπτυξη εφαρμογών και για τις δύο πλατφόρμες ταυτόχρονα έχει οδηγήσει ακόμα και τις μεγάλες εταιρείες του χώρου στην προσπάθεια ανάπτυξης εργαλείων που επιτρέπουν την επαναχρησιμοποίηση κώδικα στις υλοποιήσεις. Η Google με το Flutter, το Facebook με τη React Native και η Microsoft με το Xamarin προτείνουν τις δικές τους λύσεις στο πρόβλημα.

Μια άλλη προσέγγιση είναι η ανάπτυξη υβριδικών εφαρμογών, εφαρμογών δηλαδή που συνδυάζουν τις εγγενείς (native) με τις διαδικτυακές (web) εφαρμογές και την ανάπτυξη με τεχνολογίες web.

Η χρήση τέτοιων προγραμματιστικών εργαλείων και frameworks εκτός του ότι επιτρέπουν στον προγραμματιστή την επαναχρησιμοποίηση ολόκληρου ή μέρους του κώδικα και στις δύο

πλατφόρμες χωρίς αλλαγές, έχει το επιπλέον πλεονέκτημα της χρήση διαφορετικών γλωσσών προγραμματισμού από αυτές που εγγενώς υποστηρίζουν οι δύο πλατφόρμες. Με αυτό το τρόπο προγραμματιστές χωρίς εξειδίκευση στην κατασκευή εγγενών εφαρμογών μπορούν να χρησιμοποιήσουν τις προϋπάρχουσες γνώσεις τους από άλλες γλώσσες προγραμματισμού.

Παρόμοια προβλήματα συμβατότητας συστημάτων είναι και ένας από τους λόγους που διαδικτυακές υπηρεσίες (Web Services) έχουν γίνει ιδιαίτερα δημοφιλείς στη ανάπτυξη σύγχρονων εφαρμογών καθώς ο σημαντικότερος λόγος ανάπτυξης τους ήταν η διαλειτουργική επικοινωνία συστημάτων ανεξαρτήτως πλατφόρμας υλικού ή λογισμικού.

1.2 Συνεισφορά της εργασίας

Εφαρμογές τουριστικοί οδηγοί υπάρχουν κατά χιλιάδες στα app stores. Τα σημαντικότερα στοιχεία που χαρακτηρίζουν έναν καλό τουριστικό οδηγό είναι η πληρότητα του περιεχομένου, οι δυνατότητες που δίνει ο οδηγός στους χρήστες και η ευχρηστία του σαν εφαρμογή.

Η συγκεκριμένη εφαρμογή που υλοποιήθηκε σε αυτή τη διπλωματική εργασία δεν διεκδικεί δάφνες πρωτοτυπίας και ποιότητας σε αυτά τα στοιχεία.

Η ιδιαιτερότητα της έγκειται στην προσπάθεια υλοποίησης της σαν πρότυπο ή γεννήτρια παρόμοιων εφαρμογών, με επαναχρησιμοποίηση του κώδικα χωρίς αλλαγές. Η εφαρμογή αυτή δηλαδή, αν και περιλαμβάνει δεδομένα ταξιδιωτικού ενδιαφέροντος για το νησί της Σύρου, μπορεί να χρησιμοποιηθεί, χωρίς αλλαγές του κώδικα της εφαρμογής ή του web API που την τροφοδοτεί με τα δεδομένα, για οποιονδήποτε άλλο προορισμό.

Επιπρόσθετα η εφαρμογή συλλέγει ανωνυμοποιημένα γεωχωρικά δεδομένα σαν πρώτο βήμα για μια επέκταση της, σχετικά με προτάσεις σημείων ενδιαφέροντος που βρίσκονται κοντά στον χρήστη αλλά δεν έχει επισκεφθεί ή προτάσεις τουριστικών διαδρομών ή ίσως για περαιτέρω στατιστική μελέτη γενικού ενδιαφέροντος για κάποιον τουριστικό προορισμό.

1.3 Διάρθρωση εργασίας

Μετά την εισαγωγή και των επισημάνσεων των ιδιαιτεροτήτων της εφαρμογής, στο δεύτερο κεφάλαιο αναλύεται το τεχνολογικό υπόβαθρο που αφορά τις διαφορετικές τεχνικές δημιουργίας εφαρμογών για φορητές συσκευές, ενώ παράλληλα εξετάζονται οι διαδικτυακές υπηρεσίες και οι δύο σημαντικότερες κατηγορίες τους.

Στο τρίτο κεφάλαιο περιγράφεται η τεχνολογική στοίβα που επιλέχθηκε για την υλοποίηση του έργου και αναλύονται οι λόγοι αυτής της επιλογής.

Στο τέταρτο κεφάλαιο γίνεται επεξήγηση των κυριότερων σημείων της προγραμματιστικής υλοποίησης όλων των τμημάτων του έργου συμπεριλαμβανομένου της βάσης δεδομένων, της εφαρμογής και του API και επιπλέον τεκμηριώνεται το API.

Η λειτουργία της εφαρμογής από την πλευρά του χρήστη περιγράφεται στο πέμπτο κεφάλαιο με παράθεση στιγμιοτύπων της εφαρμογής για τις δύο πλατφόρμες υλοποίησης.

Τα συμπεράσματα από την εκπόνηση της εργασίας καταγράφονται στο έκτο κεφάλαιο και επιπλέον προτείνονται θέματα βελτίωσης και επέκτασης του έργου.

Στο παραρτήματα Α της εργασίας δίνονται οι υπερσυνδέσεις των αποθετηρίων στο οποίο φιλοξενείται ο κώδικας της εφαρμογής, του web API, και του αρχείου προδιαγραφών του web API, καθώς και υπερσυνδέσεις των οδηγιών εγκατάστασης της διαδικτυακής υπηρεσίες σε διακομιστή και της εφαρμογής στις δύο υποστηριζόμενες πλατφόρμες.

Κεφάλαιο 2ο - Θεωρητικό Τεχνολογικό υπόβαθρο

2.1 Πλατφόρμες φορητών συσκευών

Το 2007 παρουσιάστηκε το λειτουργικό σύστημα Apple iOS ακολουθούμενο ένα χρόνο αργότερα από την απάντηση της Google με την παρουσίαση του Android. Τα δύο αυτά λειτουργικά συστήματα μαζί, υποστηρίζουν την πλειονότητα των φορητών συσκευών, κάνοντας τα, τα πιο δημοφιλή λειτουργικά συστήματα φορητών συσκευών [8].

Η επιτυχία των δύο αυτών λειτουργικών συστημάτων βασίστηκε κατά πολύ στη δυνατότητα τους να εκτελούν εφαρμογές τρίτων κατασκευαστών δημιουργώντας ένα ολόκληρο οικοσύστημα ανάπτυξης εφαρμογών. Οι χρήστες των συσκευών μπορούν να εγκαταστήσουν και να χρησιμοποιήσουν χιλιάδες εφαρμογές όλων των ειδών [9].

2.1.1 Android

Το Android είναι μια πλατφόρμα για φορητές συσκευές η οποία περιλαμβάνει το λειτουργικό σύστημα, το ενδιάμεσο λογισμικό και βασικές εφαρμογές. Στηρίχθηκε στον πυρήνα Linux και το μεγαλύτερο μέρος του κώδικα δημοσιεύεται υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού [10].

Το Android για τη διευκόλυνση ανάπτυξης εφαρμογών παρέχει την δική του Εργαλειοθήκη Ανάπτυξης Λογισμικού (SDK – Software Development Kit) με τη βοήθεια της οποίας προγραμματιστές μπορούν να δημιουργήσουν εφαρμογές εκμεταλλευόμενοι στο έπακρο τις δυνατότητες του λειτουργικού συστήματος χρησιμοποιώντας τις γλώσσες προγραμματισμού Java ή/και Kotlin. Σαν περιβάλλον ανάπτυξης εφαρμογών προτείνεται το Android Studio [11] χωρίς να αποκλείονται όμως διαφορετικές επιλογές .

Οι εφαρμογές μπορούν να “ανακαλυφθούν” από τους χρήστες μέσω των καταστημάτων εφαρμογών (App Stores) όπως το Play Store της Google [12], να εγκατασταθούν στις συσκευές και να χρησιμοποιηθούν είτε δωρεάν είτε με κάποιο μικρό κόστος .

2.1.2 Apple iOS

Το Apple iOS πρωτοεμφανίστηκε σαν το λειτουργικό σύστημα του iPhone από την Apple Inc. Σήμερα όμως υποστηρίζει κι άλλες συσκευές της συγκεκριμένης εταιρείας όπως iPads, iPods, Apple TV και iWatch [13].

Ο προγραμματισμός των εφαρμογών απαιτεί χρήση προϊόντων της Apple όπως έναν υπολογιστή Mac με λειτουργικό σύστημα OS X, την εργαλειοθήκη iOS SDK και το περιβάλλον ανάπτυξης εφαρμογών Xcode [14]. Οι γλώσσες προγραμματισμού που μπορούν να χρησιμοποιηθούν για την ανάπτυξη είναι η Objective C και η Swift [15] και οι εφαρμογές μπορούν να διανεμηθούν στους χρήστες μέσω του Apple App Store [16].

2.2 Τύποι εφαρμογών για κινητές συσκευές

Αν και οι δύο πλατφόρμες, Android και iOS, παρέχουν τα δικά τους προγραμματιστικά εργαλεία, οι προγραμματιστές μπορούν να χρησιμοποιήσουν διαφορετικές τεχνικές και εργαλεία τρίτων για τη δημιουργία εφαρμογών. Ανάλογα με τον τρόπο υλοποίησης τους. οι εφαρμογές αυτές μπορούν να ταξινομηθούν σε διάφορες κατηγορίες [17][18].

Μια κατηγοριοποίση ανάλογα με την τεχνολογία υλοποίησης φαίνεται παρακάτω.

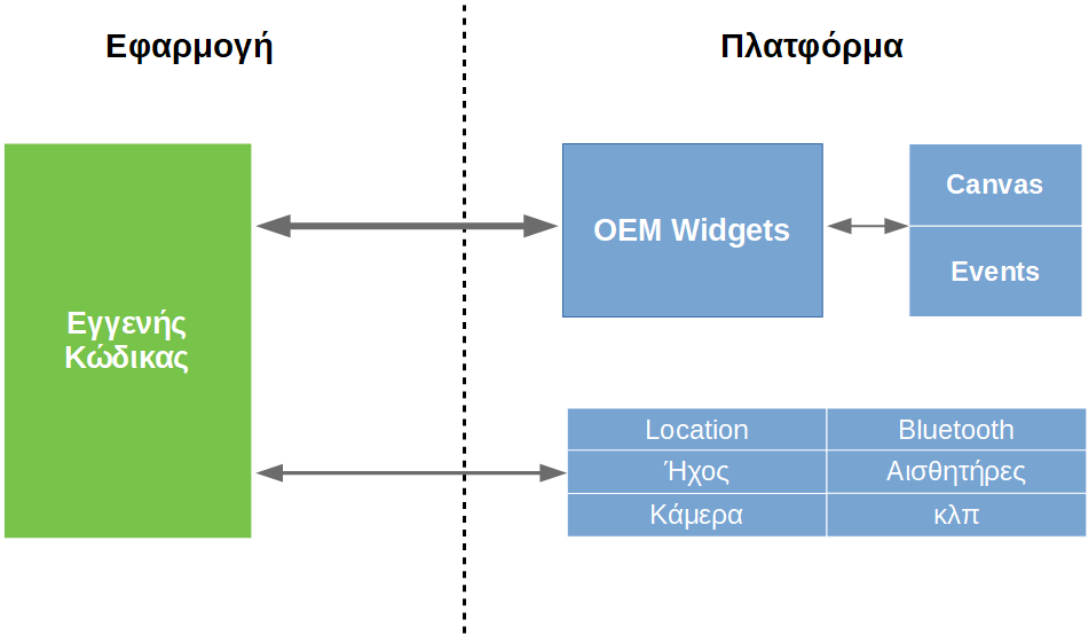
2.2.1 Εγγενείς εφαρμογές (Native apps)

Εγγενείς εφαρμογές είναι οι εφαρμογές που έχουν αναπτυχθεί για ένα συγκεκριμένο λειτουργικό σύστημα με τη χρήση των προτεινόμενων εργαλείων και γλωσσών προγραμματισμού του συστήματος αυτού.

Ο τρόπος υλοποίησης με αυτόν το τρόπο, επιτρέπει την πλήρη εκμετάλλευση του υλικού, των αισθητήρων, του λογισμικού και γενικά των πόρων των συσκευών. Η πρόσβαση σε αυτούς τους πόρους γίνεται μέσω των API που διατίθενται και τεκμηριώνονται από τους δημιουργούς του κάθε συστήματος μέσω του αντίστοιχου SDK κάθε πλατφόρμας.

Επιπλέον οι προγραμματιστές μπορούν να χρησιμοποιήσουν έτοιμες, τυποποιημένες γραφικές διεπαφές, γραφικά στοιχεία δηλαδή, όπως κουμπιά, μενού, μπάρες κλπ τα οποία κληρονομούν την εμφάνιση του λειτουργικού συστήματος. Παράλληλα, οδηγίες και καλές πρακτικές για τη δομή και

την εμφάνιση των εφαρμογών, καθοδηγούν τους προγραμματιστές στη δημιουργία εφαρμογών σχεδιασμένες έτσι ώστε να είναι οικείες και διαισθητικά χρηστικές από τους χρήστες του λειτουργικού συστήματος λόγω της εμπειρίας τους στα στοιχεία αυτά.



Εικόνα 1: Εγγενείς Εφαρμογές - Native Apps

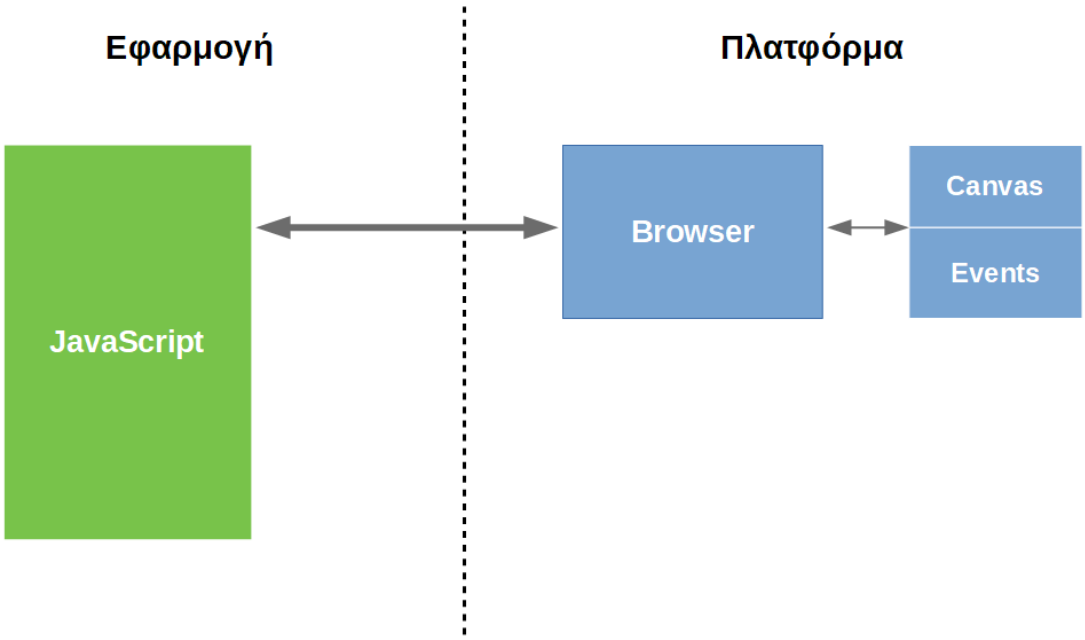
2.2.2 Διαδικτυακές εφαρμογές (Web Apps)

Διαδικτυακές εφαρμογές για φορητές συσκευές είναι οι εφαρμογές του ιστού κατασκευασμένες με κύριο γνώμονα τη χρήση τους από φορητές συσκευές (mobile first) με τέτοιο τρόπο ώστε να είναι λειτουργικές και εύχρηστες στις οθόνες αυτών των συσκευών.

Τα σύγχρονα κινητά τηλέφωνα και ταμπλέτες υποστηρίζουν φυλλομετρητές ιστού (web browsers) παρόμοιων δυνατοτήτων με αυτούς των σταθερών υπολογιστών, υποστηρίζοντας σύγχρονα πρότυπα (standards), επιτρέποντας την εκτέλεση απαιτητικών εφαρμογών ιστού.

Το κύριο μειονέκτημα αυτών των υλοποιήσεων είναι ότι δεν εγκαθίστανται μέσω των app stores και δεν υποστηρίζουν όλες τις δυνατότητες των φορητών συσκευών αφού φιλοξενούνται σαν ιστοσελίδες στον φυλλομετρητή ιστού, με αποτέλεσμα να μην έχουν πρόσβαση σε όλα τα υποσυστήματα, τους αισθητήρες και γενικά στις δυνατότητες των σύγχρονων συσκευών παρά μόνο σε αυτά που υποστηρίζονται από τον φυλλομετρητή.

Επιπρόσθετα η χρήση τους απαιτεί συνεχή σύνδεση με το διαδίκτυο για να είναι λειτουργική και η σχεδίαση τους καθώς κληρονομεί την ελευθερία υλοποίησης των ιστοσελίδων δεν ακολουθεί κάποια συγκεκριμένα πρότυπα ομοιομορφίας και χρηστικότητας.



Εικόνα 2: Διαδικτυακές Εφαρμογές - Web Apps

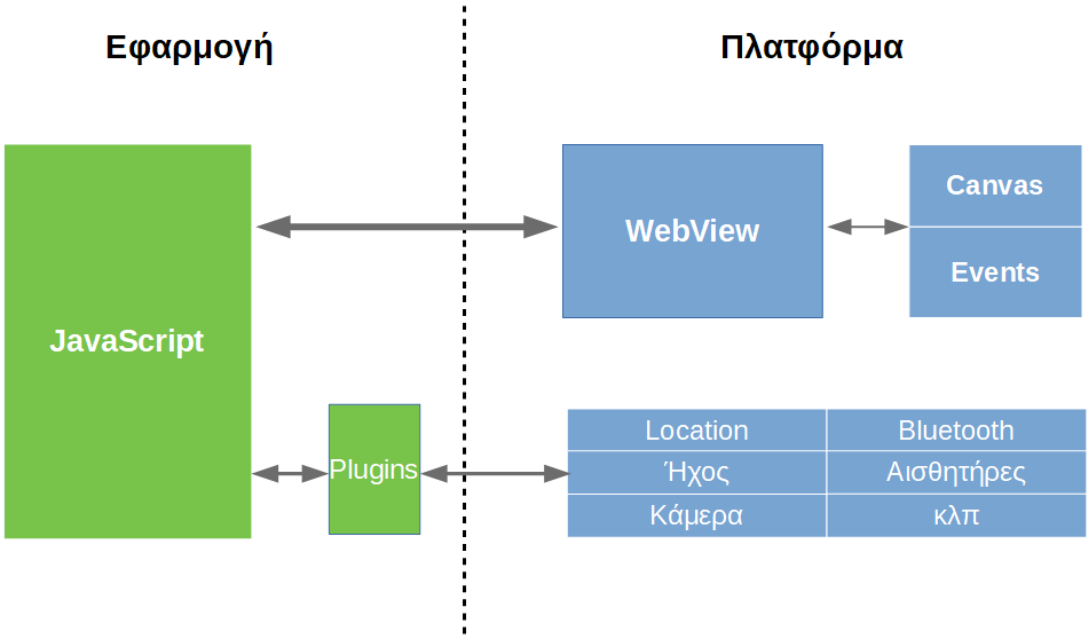
2.2.3 Υβριδικές εφαρμογές (Hybrid Apps)

Οι υβριδικές εφαρμογές είναι μια προσπάθεια να συνδυαστούν οι δύο προηγούμενες τεχνικές. Στην ουσία πρόκειται για εγγενείς εφαρμογές που ενσωματώνουν μια προβολή ιστού (webview) και μέσα σε αυτή φιλοξενούν μια web εφαρμογή, συνήθως αποθηκευμένη τοπικά στη συσκευή.

Η τεχνική αυτή επιτρέπει την ανάπτυξη της λειτουργικότητας της εφαρμογής όπως ακριβώς αναπτύσσονται οι web εφαρμογές, με τεχνολογίες ιστού. Οι εφαρμογές αυτές μπορούν να εγκατασταθούν όπως οι εγγενείς εφαρμογές μέσω των app stores και να εκτελεστούν στη συσκευή χωρίς την ανάγκη ύπαρξης σύνδεσης δικτύου [19].

Για την πρόσβαση της εφαρμογής στα υποσυστήματα της συσκευής ο προγραμματιστής θα πρέπει να υλοποιήσει εγγενώς γέφυρες διασύνδεσης προσβάσιμες από τον κώδικα JavaScript με τον οποίο

θα επικοινωνεί η φιλοξενούμενη web εφαρμογή ή να χρησιμοποιήσει κάποιο framework στο οποίο αυτές οι γέφυρες διασύνδεσης παρέχονται προς χρήση σαν πρόσθετα [20]. Με αυτόν τον τρόπο κάμερες, γυροσκόπια, αισθητήρες, ειδοποιήσεις κ.α γίνονται διαθέσιμα προς χρήση από τον κώδικα της φιλοξενούμενης εφαρμογής.



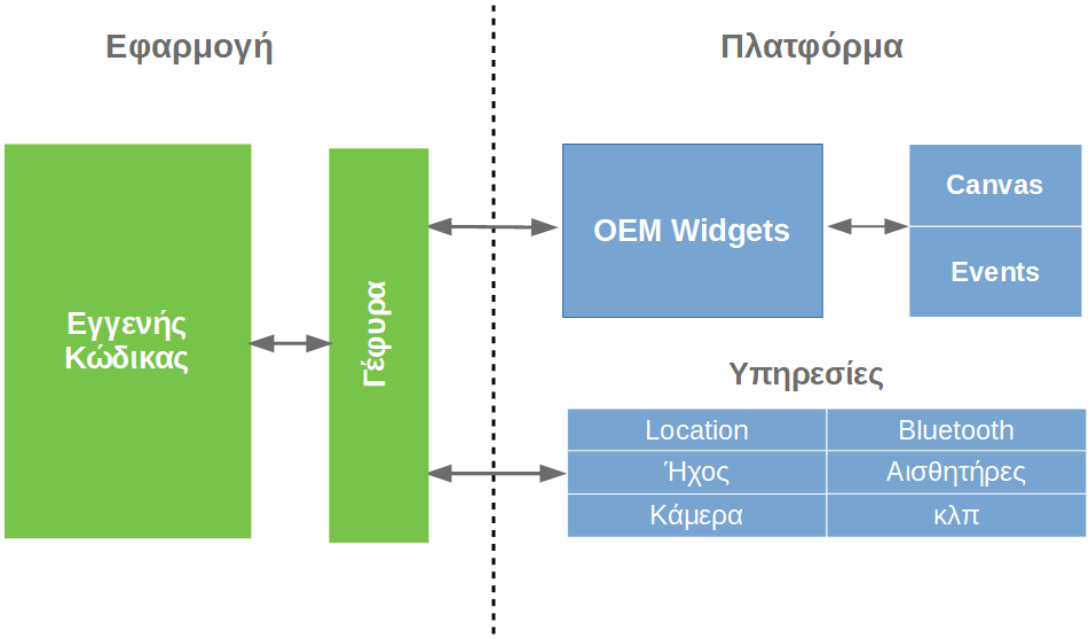
Εικόνα 3: Υβριδικές Εφαρμογές - Hybrid Apps

2.2.4 Εφαρμογές μεταγλωτισμένες ανά λειτουργικό σύστημα (cross compiled application)

Στην προσπάθεια υλοποίησης εφαρμογών για διαφορετικές πλατφόρμες από τον ίδιο κώδικα, έχουν αναπτυχθεί εργαλεία με τα οποία κώδικας γραμμένος σε κάποια γλώσσα προγραμματισμού μεταγλωττίζεται σε εγγενή κώδικα της κάθε πλατφόρμας.

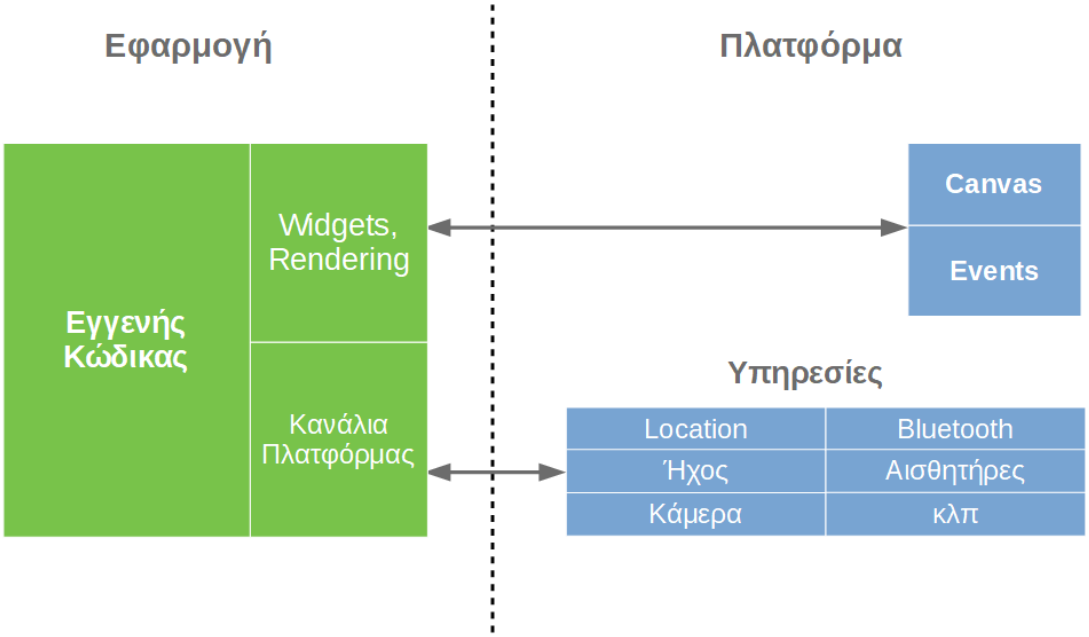
Η δημιουργία εκτελέσιμων εφαρμογών σε διαφορετικές πλατφόρμες με τον ίδιο κώδικα είναι τόσο σημαντική ώστε ακόμα και οι μεγαλύτερες εταιρείες του χώρου να ασχολούνται ενεργά και να αναπτύσουν τα δικά τους εργαλεία. Η Google με το Flutter, το Facebook με τη React Native και η Microsoft με το Xamarin προτείνουν τις δικές τους λύσεις στο πρόβλημα.

Για παράδειγμα εφαρμογή γραμμένη με JavaScript και React Native [21] ή σε C# και Xamarin [22], μεταγλωττίζεται εγγενώς και χρησιμοποιεί τα γραφικά στοιχεία διεπαφών της κάθε πλατφόρμας.



Εικόνα 4: Εφαρμογή React Native

Σε άλλες περιπτώσεις όπως το Flutter της Google γραφικά στοιχεία διεπαφών για την κάθε πλατφόρμα έχουν δημιουργηθεί εγγενώς ώστε να προσομοιώνουν τα στοιχεία της εκάστοτε πλατφόρμας και ενσωματώνονται στο εκτελέσιμο της εφαρμογής που δημιουργείται από κώδικα σε Dart [23].



Εικόνα 5: Εφαρμογή Flutter

Σε κάθε περίπτωση η εφαρμογή που εκτελείται είναι εγγενής εφαρμογή αλλά προκύπτει από κοινό κώδικα γραμμένο σε διαφορετική γλώσσα από την προτεινόμενη κάθε πλατφόρμας.

2.2.5 Σύγκριση μεταξύ Εγγενών και Υβριδικών εφαρμογών

Η άμεση πρόσβαση στα APIs των λειτουργικών συστημάτων επιτρέπουν την ανάπτυξη εγγενών εφαρμογών που εκμεταλλεύονται στο έπακρο τις δυνατότητες των συσκευών. Αυτό σημαίνει ότι, εκτός από τη δυνατότητα χρήσης κάθε υποσυστήματος και λειτουργικότητας της πλατφόρμας, οι εγγενείς εφαρμογές έχουν τη βέλτιστη απόκριση και ταχύτητα, καθώς δεν διαμεσολαβεί κάποιο επιπλέον αφαιρετικό επίπεδο για την υλοποίηση. Επιπλέον η χρήση των έτοιμων γραφικών διεπαφών που παρέχονται από κάθε πλατφόρμα, σε συνδυασμό με τις σχεδιαστικές οδηγίες που προτείνονται σε κάθε νέα έκδοση των λειτουργικών συστημάτων, προσφέρουν βελτιωμένη χρηστικότητα και εμπειρία χρήσης (UX User Experience) [24][25].

Η δυσκολία υλοποίησης είναι το μεγαλύτερο μειονέκτημα των εγγενών εφαρμογών. Το πρόβλημα δεν είναι μόνο σχετικό με το κόστος και το χρόνο [26]. Το θέμα είναι ότι συνήθως πρέπει να αναπτυχθεί ακριβώς η ίδια εφαρμογή, με την ίδια λειτουργικότητα και τις ίδιες δυνατότητες από διαφορετικές ομάδες. Το πρόβλημα δηλαδή είναι η επικοινωνία των ομάδων στην ανάπτυξη κοινής

εφαρμογής σε διαφορετικές αρχιτεκτονικές, με διαφορετικές γλώσσες προγραμματισμού και διαφορετικά εργαλεία. Γι' αυτό άλλωστε και οργανισμοί με πλούσιο ανθρώπινο δυναμικό και τεράστιο χρηματικό αποθεματικό προσπαθούν να βρουν λύσεις στην ανάπτυξη εφαρμογών για τις δύο πλατφόρμες με κοινό κώδικα.

Αυτό ακριβώς είναι και το πλεονέκτημα των μεταγλωττισμένων ανά πλατφόρμα εφαρμογών. Με αυτή την τεχνική στην ουσία αναπτύσσεται μία εφαρμογή που εκτελείται και στα δύο συστήματα. Ο τρόπος που αυτό γίνεται διαφέρει σε κάθε προγραμματιστικό εργαλείο, όπως και το ποσοστό του κώδικα που μπορεί να επαναχρησιμοποιηθεί ή το πόσο κοντά στην "εγγενή" όψη είναι το τελικό αποτέλεσμα. Σίγουρο όμως είναι ότι αυτές οι τεχνικές έχουν μεγάλη ανάπτυξη τα τελευταία χρόνια, ωριμάζουν, βελτιώνονται, τεκμηριώνονται καλύτερα και χρησιμοποιούνται όλο και περισσότερο.

Μεγάλη ανάπτυξη επίσης έχουν και οι διαδικτυακές εφαρμογές. Αυτό οφείλεται κυρίως στην αλματώδη ανάπτυξη της πλατφόρμας στην οποία στηρίζονται, δηλαδή στους φυλλομετρητές ιστού. Επιπλέον νέες τεχνολογίες όπως οι εφαρμογές PWA (Progressive Web Apps) [27] προσπαθούν να μετριάσουν τα μειονεκτήματα των web εφαρμογών προσομοιώνοντας εν μέρη την εμφάνιση των εφαρμογών και βελτιώνοντας την ταχύτητα και την απόκριση τους ακόμα και με μικρές ταχύτητες διαδικτύου και χρησιμοποιώντας δυνατότητες που παλαιότερα δεν ήταν δυνατές όπως ειδοποιήσεις και λειτουργία εκτός σύνδεσης.

Σε κάθε περίπτωση όμως οι web εφαρμογές στερούνται δύο σημαντικών παραγόντων. Το πρώτο είναι η δυνατότητα ανακαλυψιμότητας (Discoverability) που προσφέρουν τα app stores και δεύτερο ότι δεν έχουν πρόσβαση σε όλα τα χαρακτηριστικά των συσκευών.

Η Υβριδικές εφαρμογές από την πλευρά τους εγκαθίστανται μέσω των app stores, έχουν πρόσβαση στα υποσυστήματα και στους αισθητήρες των συσκευών αλλά στερούνται την ομαλότητα, την ομοιομορφία και γενικά την εμπειρία χρήσης που προσφέρουν οι καλογραμμένες εγγενείς εφαρμογές [28][29].

Βλέπουμε λοιπόν ότι όλες οι τεχνικές έχουν τα πλεονεκτήματα και τα μειονεκτήματα τους. Στη δική μας περίπτωση, που είναι η ανάπτυξη μιας απλής εφαρμογής χωρίς περίπλοκο περιβάλλον χρήστη και χωρίς ιδιαίτερες απαιτήσεις από τα υποσυστήματα των συσκευών, προτιμήθηκε η ανάπτυξη με βάση το υβριδικό μοντέλο που προσφέρει εύκολη και γρήγορη υλοποίηση με τα επιπλέον θετικά της εγκατάστασης μέσω των app stores και τη χρήση ορισμένων πρόσθετων δυνατοτήτων της συσκευής όπως το GPS μέσω πρόσθετου (plugin).

2.3 Υπηρεσίες Ιστού (Web Services)

Ένα Web Service είναι ένα σύστημα λογισμικού σχεδιασμένο να υποστηρίζει διαλειτουργική, μηχανή-προς-μηχανή αλληλεπίδραση μέσω δικτύου ή διαφορετικά μια συλλογή πρωτοκόλλων και προδιαγραφών για την ανταλλαγή δεδομένων μεταξύ συστημάτων και εφαρμογών.

2.3.1 SOAP web services

Σε ορισμένες υπηρεσίες ιστού οι προδιαγραφές επικοινωνίας των συστημάτων προσδιορίζονται με την Web Service Description Language (WSDL). Η WSDL είναι μια γλώσσα βασισμένη σε XML που περιγράφει πως η εισερχόμενη και η εξερχόμενη πληροφορία θα πρέπει να δομηθεί, ώστε η υπηρεσία και η εφαρμογές που την χρησιμοποιούν να μπορούν να επικοινωνήσουν, για παράδειγμα πως γίνονται οι αιτήσεις στην υπηρεσία, τι παράμετροι αναμένονται ή τι δομές επιστρέφονται.

Αφού ο τρόπος επικοινωνίας καθοριστεί μέσω της WSDL τα δεδομένα ανταλλάσσονται μέσω του SOAP το οποίο είναι ένα πρωτόκολλο ανταλλαγής δομημένης πληροφορίας σε μορφή μηνυμάτων σε δίκτυα, συνήθως βασιζόμενο στο πρωτόκολλο HTTP για τη αποστολή αυτών των μηνυμάτων.

Το SOAP βασίζεται κι αυτό σε XML και αποτελείται από τρία μέρη. Έναν φάκελο (envelope), ο οποίος ορίζει τη μορφή του μηνύματος και τον τρόπο επεξεργασίας του, ένα σύνολο κανόνων κωδικοποίησης για την περιγραφή καθορισμένων τύπων δεδομένων και μια σύμβαση για την αναπαράσταση των αιτημάτων και των απαντήσεων.

Τα κυριότερα χαρακτηριστικά του πρωτοκόλλου είναι η επεκτασιμότητα, η ουδετερότητα που το κάνει να μπορεί να λειτουργήσει πάνω από άλλα πρωτόκολλα μεταφοράς όπως HTTP, SMTP, TCP και η ανεξαρτησία καθώς επιτρέπει οποιοδήποτε μοντέλο προγραμματισμού. [30]

2.3.2 RESTful Web Services

Μια διαφορετική αρχιτεκτονική προσέγγιση δημιουργίας web services είναι η REST (Representational State Transfer).

Πρόκειται για ένα σύνολο αρχιτεκτονικών αρχών με τις οποίες μπορούν να σχεδιαστούν διαδικτυακές υπηρεσίες που εστιάζουν στους πόρους ενός συστήματος καθώς και του τρόπου που οι καταστάσεις των πόρων αντιμετωπίζονται και μεταφέρονται μέσω HTTP.

Βασικό στοιχείο είναι ο πόρος (resource) που στην ουσία είναι ένα αντικείμενο και η αναπαράσταση του (representation) που είναι το περιεχόμενο του πόρου με τα μεταδεδομένα που τον χαρακτηρίζουν.

Η αναγνώριση αυτών των πόρων γίνεται με χρήση ενιαίων προσδιοριστών πόρων URI (Uniform Resource Identifiers) που μπορούν να είναι διευθύνσεις διαδικτύου και η αλλαγή της κατάστασης τους γίνεται με χρήση μεθόδων HTTP αποκλειστικά όπως GET, POST, PUT και DELETE.

Ένα RESTful σύστημα πρέπει να στηρίζεται σε τέσσερις βασικές αρχές.

Αναγνώριση πόρων μέσω URI

Μια υπηρεσία παρέχει ένα σύνολο πόρων προς αλληλεπίδραση με του πελάτες. Οι αναπαραστάσεις αυτών των πόρων προσδιορίζονται από τα URI.

Ομοιόμορφη διεπαφή

Οι πόροι διαχειρίζονται χρησιμοποιώντας ένα σταθερό σύνολο τεσσάρων HTTP μεθόδων δημιουργίας, ανάγνωσης, ενημέρωσης, διαγραφής (POST, GET, PUT, DELETE).

Αυτοπεριγραφικά μηνύματα

Οι πόροι αποσυνδέονται από την αναπαράσταση τους έτσι ώστε το περιεχόμενο τους να είναι προσβάσιμο σε διαφορετικές μορφές όπως HTML, text, JSON, PDF, JPEG και άλλες μορφές εγγράφων. Επιπλέον μετά-δεδομένα (metadata) σχετικά με τον πόρο είναι διαθέσιμα και χρησιμοποιούνται. Παραδείγματα χρήσης αυτών των μετά-δεδομένων είναι η εκτέλεση ελέγχου ταυτότητας, η ανίχνευση σφαλμάτων μετάδοσης, ο έλεγχος της προσωρινής αποθήκευσης (cache) και η μορφή της αναπαράστασης.

Stateless - Ανεξάρτητο κατάστασης

Τα αιτήματα είναι ανεξάρτητα το ένα από το άλλο. Ο πελάτης στέλνει τα αίτημα με όλη την απαραίτητη πληροφορία που απαιτείται ώστε ο διακομιστής να μη χρειάζεται γνώση προηγούμενης κατάστασης η ενδιάμεσης πληροφορίας για την διεκπεραίωση τους.

2.3.3 Σύγκριση REST και SOAP

Το REST και το SOAP δεν μπορούν να συγκριθούν απευθείας καθώς το πρώτο είναι ένα αρχιτεκτονικό στυλ υλοποιήσεων υπηρεσιών ενώ το δεύτερο ένα πρωτόκολλο. Παρόλα αυτά μπορούμε να εντοπίσουμε ορισμένες σημαντικές διαφορές.

Τα RESTful APIs βασίζονται στο HTTP και χρησιμοποιούν τις μεθόδους του, με αποτέλεσμα να είναι ιδιαίτερα απλή η υλοποίηση της υπηρεσίας, του πελάτη που τα χρησιμοποιεί αλλά και της τεκμηρίωσης καθώς αυτές οι μέθοδοι είναι καλά καθορισμένες και γνώριμες στους προγραμματιστές ιστού.

Το REST επιτρέπει διάφορα φορμά δεδομένων, ενώ το SOAP μόνο XML. Αν και αυτό εκ πρώτης άποψης φαίνεται να προσθέτει περιπλοκότητα καθώς χρειάζεται χειρισμός πολλαπλών μορφών αρχείων, στην πραγματικότητα είναι ιδιαίτερα σημαντικό και βολικό καθώς αυτός ο χειρισμός γίνεται εύκολα και επιπλέον, μερικές μορφές αρχείων, όπως για παράδειγμα το JSON, έχουν συγκριτικά πλεονεκτήματα απέναντι στο XML. Τα JSON επεξεργάζονται πιο γρήγορα, είναι μικρότερα σε μέγεθος και άρα μεταφέρονται μέσω δικτύου ταχύτερα, αλλά και υποστηρίζονται εγγενώς από τη γλώσσα προγραμματισμού JavaScript.

Το SOAP από την άλλη είναι αυστηρά ορισμένο με ενσωματωμένες δυνατότητες προστασίας, παράδοσης και ασφάλειας των μηνυμάτων και επιπλέον υποστηρίζει πρωτόκολλα μεταφοράς επιπλέον του HTTP, όπως SMTP και sockets. [31]

Το REST τα τελευταία χρόνια κερδίζει όλο και περισσότερο έδαφος σε σχέση με το SOAP. Αυτό οφείλεται κυρίως στο ότι έχει καλύτερες επιδόσεις, ευκολότερη επεκτασιμότητα και καθώς πρόκειται για ένα αρχιτεκτονικό στυλ και όχι ένα αυστηρό πρωτόκολλο, επιτρέπει μεγαλύτερη ευελιξία και ελευθερία στις υλοποιήσεις [32][33][34][35].

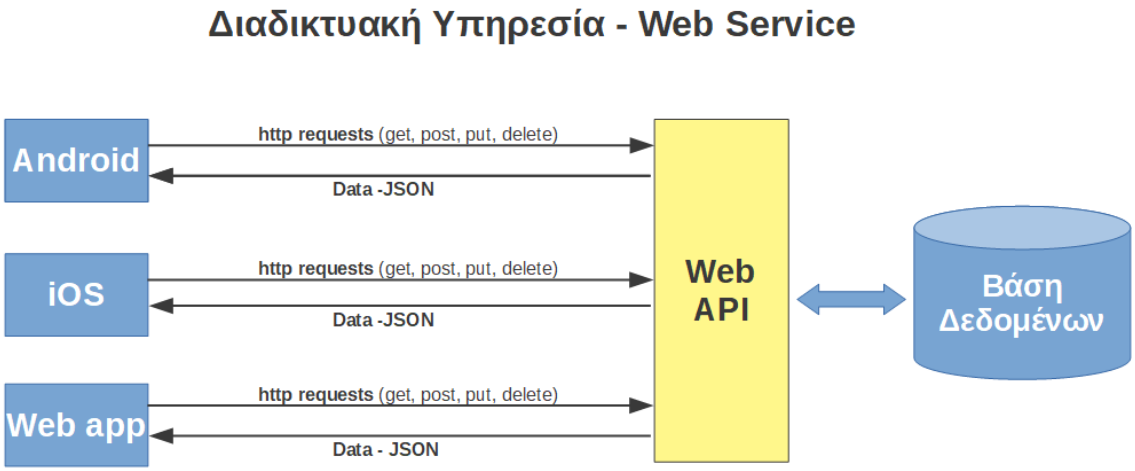
Ένας εξίσου σημαντικός λόγος της επικράτησης του είναι η ευκολία με την οποία οι πελάτες επικοινωνούν με την υπηρεσία, με αποτέλεσμα τα RESTful APIs να είναι ιδιαίτερα εύκολα προσβάσιμα από web και mobile εφαρμογές ή από άλλες διαδικτυακές υπηρεσίες.

2.3.4 Web Apis

Η Υπηρεσία Ιστού παρέχει μια διεπαφή, η οποία αποκρύπτει τις λεπτομέρειες υλοποίησης της υπηρεσίας έτσι ώστε να μπορεί να χρησιμοποιηθεί από εφαρμογές ανεξάρτητα από την πλατφόρμα υλικού ή λογισμικού και ανεξάρτητα από τη γλώσσα προγραμματισμού στην οποία η εφαρμογή είναι γραμμένη.

Το Web Api δηλαδή είναι ένα σύνολο σημείων διασύνδεσης (endpoints) μέσω των οποίων οι πελάτες στέλνουν HTTP αιτήματα. Ο διακομιστής μέσω του API επιστρέφει τις απαντήσεις στην κατάλληλη μορφή.

Όταν αυτό το API σχεδιαστεί ακολουθώντας την αρχιτεκτονική REST τότε πρόκειται για ένα RESTful Web Api.



Εικόνα 6: REST Web Service

2.3.5 Προδιαγραφές openApi

Ένα Web API, καθώς αναπτύσσεται ανεξάρτητα από τις εφαρμογές που το χρησιμοποιούν, για να μπορέσει να χρησιμοποιηθεί από τους πελάτες, χρειάζεται τεκμηρίωση (documentation). Η τεκμηρίωση αυτή περιγράφει τη λειτουργία του API, τα σημεία επαφών και τί μεθόδους δέχεται κάθε σημείο, τις παραμέτρους που μπορούν να γίνουν δεκτές, καθώς και την αναμενόμενη απάντηση. Επιπλέον, καθώς τα APIs αλλάζουν, τροποποιούνται ή προσθέτουν επιπλέον λειτουργίες, η αντίστοιχη τεκμηρίωση πρέπει να ενημερωθεί, ώστε τα νέα χαρακτηριστικά να μπορούν να ενσωματωθούν στις εφαρμογές.

Η τεκμηρίωση αυτή γίνεται με απλό, κατανοητό και φιλικό προς τον άνθρωπο τρόπο, πολλές φορές ακόμα και με τη χρήση παραδειγμάτων, αφού δεν αφορά τεχνικές υλοποίησης ή πρωτόκολλα διασύνδεσης παρά μόνο τη χρήση.

Η ανάγκη σωστής και πλήρης τεκμηρίωσης των API έχει οδηγήσει στην εμφάνιση τυποποιήσεων (standards) όπου περιγράφονται επακριβώς η δομή της τεκμηρίωσης ενός API ώστε να υπάρχει πληρότητα και συνέπεια. Οι τυποποιήσεις αυτές παρέχουν επιπλέον τη δυνατότητα αυτόματης παραγωγής της τεκμηρίωσης αλλά και αυτόματη παραγωγή τμημάτων κώδικα της υπηρεσίας που παρέχει το API και της εφαρμογής που το χρησιμοποιεί, σε διάφορες γλώσσες προγραμματισμού.

Η πιο δημοφιλής τυποποίηση σήμερα είναι το OpenAPI Specification (ή Swagger) [36] που είναι μια κοινή πρωτοβουλία διαφόρων εταιρειών δημιουργίας ενός κοινού σημείου αναφοράς για την τεκμηρίωση APIs.

Κεφάλαιο 3ο - Τεχνολογία Εφαρμογής

3.1 Κριτήρια επιλογής τεχνολογικής στοίβας

Όλο το έργο σχεδιάστηκε με γνώμονα την εύκολη επαναχρησιμοποίηση του για την δημιουργία κλωνοποιημένων, παρόμοιων εφαρμογών χωρίς αλλαγή στον κώδικα. Γι' αυτό το λόγο ένα βασικό κριτήριο στην επιλογή της τεχνολογίας και των εργαλείων υλοποίησης ήταν η ευκολία χρήσης και εγκατάστασης, η δημοτικότητα και η απλότητα τους. Δεν επιλέχθηκαν εξεζητημένες και περίπλοκες λύσεις ώστε η εγκατάσταση και η κλωνοποίηση μιας εφαρμογής και του αντίστοιχου web service να μπορεί να γίνει όσο το δυνατόν πιο εύκολα.

3.2 Τεχνολογία που επιλέχθηκε για την εφαρμογή

Για τη δημιουργία της εφαρμογής επιλέχθηκε το Υβριδικό μοντέλο ανάπτυξης, στηριζόμενο στο framework Apache Cordova [37], με τη χρήση της γλώσσας προγραμματισμού JavaScript και τις τεχνολογίες διαδικτύου HTML5 και CSS3.

Το Apache Cordova είναι ένα ανοικτού κώδικα framework ανάπτυξης, το οποίο επιτρέπει τη χρήση τεχνολογιών ιστού (όπως HTML5, CSS3 και JavaScript) για τη δημιουργία cross-platform εφαρμογών. Οι εφαρμογές αυτές εκτελούνται μέσα σε υποδοχείς (wrappers) υλοποιημένους εγγενώς για κάθε πλατφόρμα που υποστηρίζεται από το framework. Επιπρόσθετα παρέχεται ένα κοινό προγραμματιστικό API για την πρόσβαση στις δυνατότητες κάθε συσκευής όπως τους αισθητήρες, το σύστημα αρχείων, τα δεδομένα, το δίκτυο κ.α.

Οι υποδοχείς αυτοί λειτουργούν σαν προβολείς ιστού (web views) και μπορούν να φιλοξενήσουν web εφαρμογές αποθηκευμένες τοπικά. Οι εφαρμογές υλοποιημένες με τη χρήση του Apache Cordova μπορούν να εγκατασταθούν ακριβώς όπως και οι εγγενείς εφαρμογές μέσα από τα app stores και λειτουργούν χωρίς την ανάγκη ύπαρξης δικτύου.

Η πρόσβαση στις δυνατότητες της κάθε συσκευής γίνεται με τη βοήθεια πρόσθετων (Cordova plugins). Αυτά τα πρόσθετα, υλοποιημένα εγγενώς, παρέχουν τις γέφυρες διασύνδεσης της εφαρμογής με το API του λειτουργικού συστήματος και επιτρέπουν τον έλεγχο των αισθητήρων και των υπηρεσιών του λειτουργικού συστήματος μέσω κώδικα γραμμένο σε JavaScript.

Το Apache Cordova είναι ένα από τα πιο δημοφιλή πλαίσια ανάπτυξης υβριδικών εφαρμογών και υποστηρίζεται από πλήθος προγραμματιστών και εταιρειών με αποτέλεσμα να υπάρχει πληθώρα πρόσθετων. Σε περίπτωση που κάποια εφαρμογή απαιτεί πρόσβαση σε χαρακτηριστικά του λειτουργικού συστήματος ή της συσκευής για τα οποία δεν υπάρχουν αντίστοιχα πρόσθετα, τότε θα πρέπει η δυνατότητα αυτή να υλοποιηθεί εγγενώς και να γεφυρωθεί με τη δημιουργία νέου plugin.

Αν και το Apache Cordova δρα σαν υποδοχέας μιας εφαρμογής, δεν παρέχει τα εργαλεία για την ανάπτυξη του γραφικού περιβάλλοντος χρήστη (GUI). Τόσο το Android όσο και το iOS έχουν συγκεκριμένες προδιαγραφές εμφάνισης (guidelines) των εφαρμογών [38][39].

Η συνέπεια σε αυτές τις οδηγίες είναι ιδιαίτερη σημαντική καθώς πολλά στοιχεία είναι γνώριμα και οικεία στο χρήστη και επιπλέον βελτιώνουν το αισθητικό και χρηστικό αποτέλεσμα της εφαρμογής.

Για τη δημιουργία του UI έχει χρησιμοποιηθεί το Framework7 [40] το οποίο είναι ένα ανοιχτού κώδικα HTML framework για την ανάπτυξη υβριδικών εφαρμογών ή εφαρμογών ιστού για φορητές συσκευές με εμφάνιση που προσομοιώνει τα εγγενή στοιχεία του Android και του iOS.

Η επιλογή του Framework7 έγινε κυρίως γιατί προσομοιώνει πολύ καλά τα στοιχεία εμφάνισης κάθε πλατφόρμας αλλά και για την ευκολία χρήσης, την ευελιξία και την ταχύτητα του. Η ενσωμάτωση βιβλιοθηκών Template Engine [41], DOM manipulation και Touch Slider [42] επιτρέπει τη γρήγορη δημιουργία εφαρμογών χωρίς απαιτήσεις τρίτων frameworks και βιβλιοθηκών.

Για τις αιτήσεις (http requests) προς το API χρησιμοποιήθηκε η βοηθητική βιβλιοθήκη ανοιχτού λογισμικού axios [43], ενώ για την καταγραφή πληροφοριών (logging) η βιβλιοθήκη loglevel [44].

3.3 Τεχνολογία που επιλέχθηκε για το web service

Για την υλοποίηση του web service και την ανάπτυξη του API που τροφοδοτεί και αλληλεπιδρά με την εφαρμογή χρησιμοποιήθηκε η γλώσσα προγραμματισμού PHP [45] με χρήση του Slim micro framework [46] ενώ τα δεδομένα αποθηκεύονται σε βάση δεδομένων MySQL [47].

Ο σημαντικότερος λόγος επιλογής της PHP είναι η εγκατεστημένη βάση της στους παρόχους φιλοξενίας ιστοσελίδων που κάνει την εγκατάσταση του web service διαδικασία εύκολη και προσιτή. Παράλληλα η ταχύτητα της, ειδικά μετά την έκδοση 7 και η ευκολία χρήσης της είναι δύο ακόμα παράγοντες επιλογής για την υλοποίηση του web API.

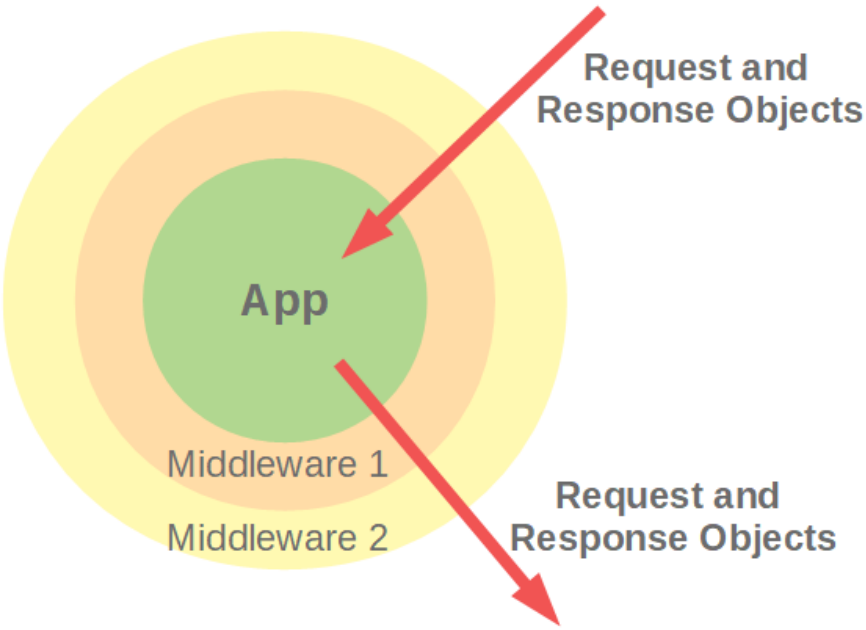
Η σχεσιακή βάση δεδομένων MySQL χρησιμοποιήθηκε γιατί πέρα από την ταχύτητα, την σταθερότητα και την πληθώρα βοηθητικών εργαλείων που μπορούν να χρησιμοποιηθούν για την εισαγωγή και τροποποίηση των δεδομένων, παρέχει έτοιμους τύπους χωρικών δεδομενων (spatial data types) όπως το POINT και συναρτήσεις σφαιρικών συντεταγμένων [48], δίνοντας δυνατότητες αποθήκευσης και ταξινόμησης γεωχωρικών δεδομένων.

Το Slim micro framework είναι ένα απλό, βασικό framework δημιουργίας web εφαρμογών και web APIs. Είναι ανοιχτό λογισμικό, ελαφρύ, ευέλικτο, εύκολο στη χρήση και ιδιαίτερα γρήγορο καθώς δεν ενσωματώνει εξειδικευμένες δυνατότητες που άλλωστε δεν απαιτούνται στη δημιουργία APIs περιλαμβάνει όμως τρία βασικά χαρακτηριστικά.

α) Dependency Injection, την τεχνική δηλαδή με την οποία οι εξαρτήσεις του κώδικα γίνονται διαθέσιμες με ενιαίο τρόπο στην εφαρμογή χωρίς να χρειάζεται να ζητηθούν κάθε φορά που χρειάζονται.

β) Μηχανισμό Middleware, με τον οποίο στρώματα (layers) προστίθενται πριν και μετά το κύριο μέρος της web εφαρμογής παρέχοντας τη δυνατότητα φιλτραρίσματος, ελέγχου και τροποποίησης των αιτήσεων (requests) και των απαντήσεων (responses) ανεξάρτητα από το τμήμα της εφαρμογής που τελικά είναι υπεύθυνο γι’ αυτές τις αιτήσεις.

Στο Slim η διαστρωμάτωση των middlewares φαίνεται στην εικόνα όπου το πρώτο middleware μέσω του οποίου φιλτράρονται οι αιτήσεις είναι και το τελευταίο πριν την έξοδο.



Εικόνα 7: Slim Middleware

γ) HTTP Router με το οποίο υλοποιούνται τα endpoints του web API.

Τέλος για την καταγραφή δεδομένων εκτέλεσης του API χρησιμοποιήθηκε η ανοιχτού λογισμικού βιβλιοθήκη monolog [49].

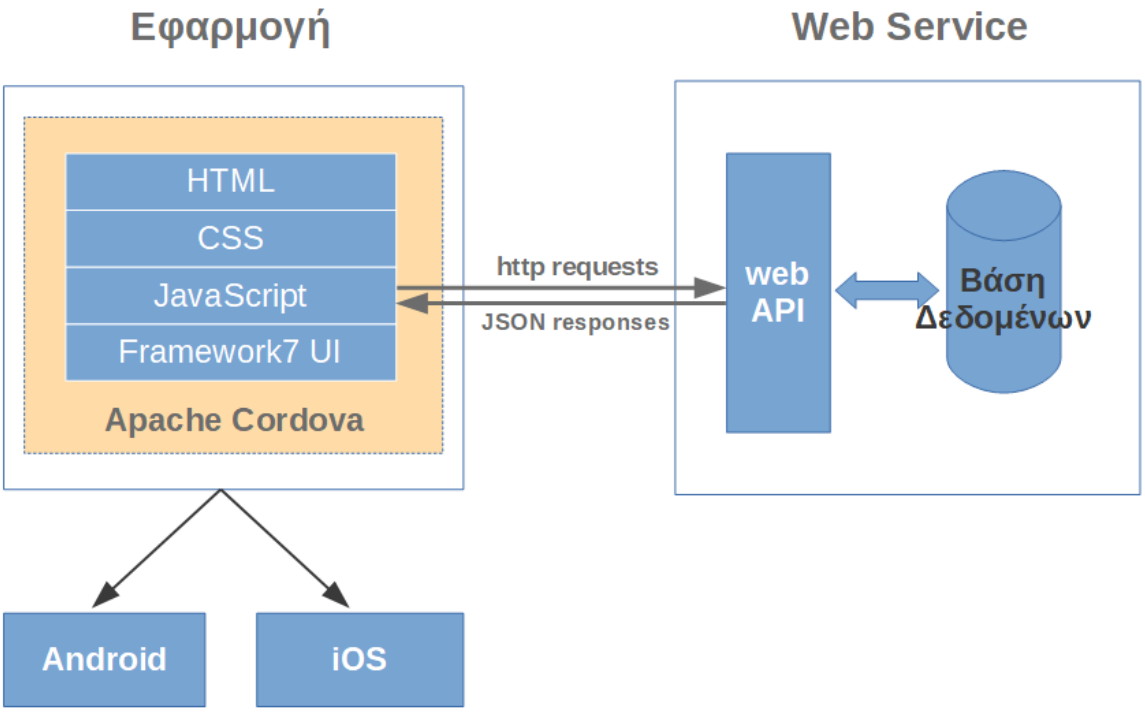
Η στοίβα λογισμικού που έχει επιλεγεί είναι ιδιαίτερα κοινή, καλά τεκμηριωμένη και σχετικά γρήγορη ενώ έχει χαμηλές απαιτήσεις πόρων συστήματος και φιλοξενίας.

Επιπλέον όλο το λογισμικό που έχει χρησιμοποιηθεί είναι ανοιχτού κώδικα.

Κεφάλαιο 4ο - Υλοποίηση Εφαρμογής

4.1 Αρχιτεκτονική της Εφαρμογής

Το έργο αποτελείται από δύο ξεχωριστά τμήματα. Την εφαρμογή που εγκαθίσταται στη συσκευή και το web service που την τροφοδοτεί με τα δεδομένα. Το web service αποτελείται από ένα RESTful API και από τη βάση δεδομένων στην οποία αποθηκεύονται τα δεδομένα.



Εικόνα 8: Αρχιτεκτονική του έργου

Η εφαρμογή είναι σχεδιασμένη να προσομοιώνει τα γραφικά στοιχεία και γενικά την εμφάνιση της εκάστοτε πλατφόρμας στην οποία εκτελείται. Δεδομένα που εμφανίζονται στις οθόνες ή δεδομένα που συλλέγονται από τη χρήση της εφαρμογής προέρχονται η επιστρέφονται και αποθηκεύονται, μέσω του web API, στη βάση δεδομένων.

4.2 Βάση Δεδομένων

Η βάση δεδομένων του συστήματος είναι απλή στη σχεδίαση αλλά εύκολα τροποποιήσιμη και επεκτάσιμη.

Αποτελείται από τέσσερις πίνακες.

items

Ο βασικός πίνακας που περιέχει τις πληροφορίες του ταξιδιωτικού προορισμού. Είναι ο πίνακας που με αλλαγή των πληροφοριών μπορεί να επαναχρησιμοποιηθεί για τη δημιουργία κλωνοποιημένων πανομοιότυπων εφαρμογών διαφορετικού προορισμού.

counts

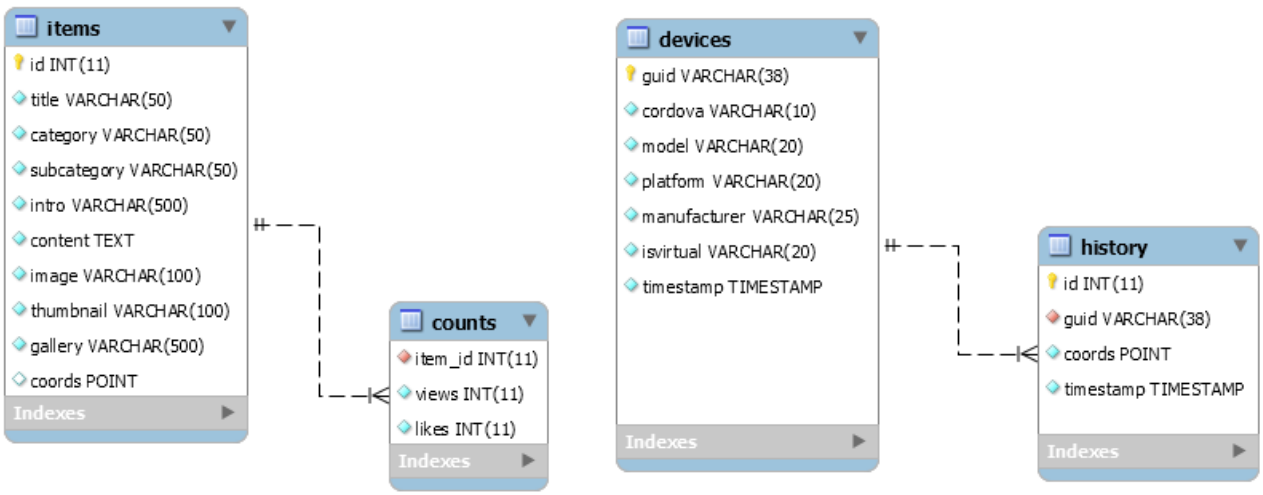
Ο πίνακας που κρατά πληροφορίες σχετικά με το πόσο συχνά ζητήθηκε κάποιο στοιχείο του πίνακα items καθώς και ο αριθμός των αγαπημένων από τους χρήστες περιοχών.

devices

Ο πίνακας που καταγράφονται πληροφορίες για τη συσκευή κατά την πρώτη χρήση της εφαρμογής. Τα στοιχεία αυτά αφορούν το είδος της συσκευής, του λειτουργικού συστήματος και της έκδοσης της εφαρμογής και είναι μη προσωποποιημένα.

location

Ο πίνακας που καταγράφονται οι τοποθεσίες στις οποίες χρησιμοποιήθηκε η εφαρμογή.



Εικόνα 9: Βάση Δεδομένων

Η απλή σχεδίαση των πινάκων και ειδικά η μη κανονικοποιημένη μορφή του πίνακα items προτιμηθήκαν ώστε να είναι εύκολη η επαναχρησιμοποίηση και η εισαγωγή δεδομένων είτε με ένα

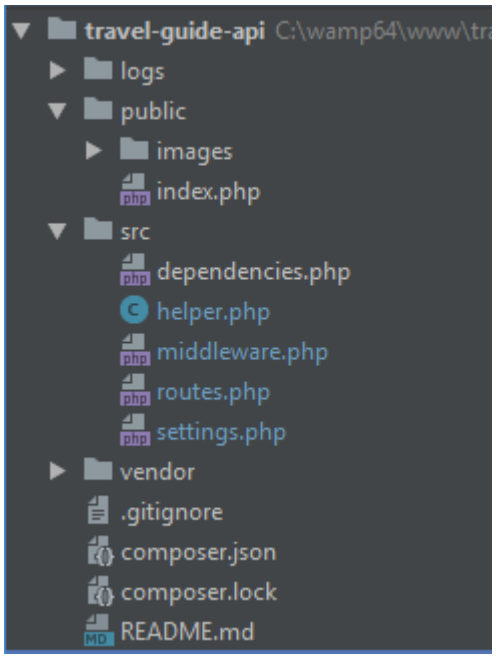
απλό εργαλείο διαχείρησης βάσων δεδομένων ή με εισαγωγή δεδομένων μέσω αρχείου csv χωρίς τη χρήση κώδικα.

4.3 Το web API

Το web API έχει υλοποιηθεί με PHP και το Slim micro framework.

4.3.1 Η δομή των αρχείων του web API

Στην εικόνα φαίνεται η ιεραρχική δομή των αρχείων που το αποτελούν.



Εικόνα 10: Δομή αρχείων του web API

Στον φάκελο logs αποθηκεύονται τα αρχεία καταγραφής logs που δημιουργούνται από το web API. Το API διαθέτει δύο επίπεδα καταγραφής logs. Development mode όπου καταγράφονται λεπτομέρειες χρήσιμες για την αποσφαλμάτωση και Production mode όπου καταγράφονται μόνο σφάλματα εκτέλεσης.

Ο φάκελος public είναι ο μόνος φάκελος που πρέπει να εκτίθεται στο διαδίκτυο και περιέχει το αρχείο index.php που είναι και το τμήμα του κώδικα που ξεκινάει το web API και τον υποφάκελο images που περιέχει τις εικόνες των αξιοθέατων.

Ο φάκελος src είναι η καρδία του συστήματος. Περιέχει το αρχείο dependencies.php στο οποίο γίνονται οι αρχικοποιήσεις της βάσης δεδομένων και του μηχανισμού καταγραφής (logger) σε containers κατάλληλα για χρήση από τα διάφορα τμήματα της εφαρμογής (Dependency Injections). Το αρχείο middleware.php υλοποιεί ένα middleware στο οποίο ελέγχεται η εγκυρότητα των παραμέτρων που δόθηκαν από την αίτηση (request) του πελάτη ή προστίθενται οι προκαθορισμένες τιμές σε παραμέτρους που δεν απαιτούνταν κατά την αίτηση. Αυτοί οι έλεγχοι γίνονται με τη βοήθεια της κλάσης helper του αρχείου helper.php. Στο αρχείο routes υλοποιούνται τα endpoints του web API. Στο αρχείο settings.php ορίζονται γενικές ρυθμίσεις και εισάγονται τα στοιχεία σύνδεσης με τη βάση δεδομένων.

4.3.2 Υλοποίηση των endpoints

Στο αρχείο **`src/routes.php`** υλοποιούνται τα εξής endpoints του web API.

`GET/items/`

Επιστρέφει όλα τα αξιοθέατα

`GET /items/{id}`

Επιστρέφει ένα μόνο αξιοθέατο με βάση το id

`GET /nearby/`

Επιστρέφει αξιοθέατα ταξινομημένα ανά απόσταση σε σχέση με τη θέση που δίνεται σαν παράμετρος

`GET /sights/`

Επιστρέφει μόνο τα αξιοθέατα

`GET /beaches/`

Επιστρέφει μόνο τις παραλίες

`GET /places/`

Επιστρέφει μόνο τους οικισμούς

`GET /info/`

Επιστρέφει μόνο τις πληροφορίες

`PUT /favorites/{action}/{id}`

Ανεβάζει ή κατεβάζει τον αριθμό των αγαπημένων κατά ένα για κάποιο σημείο.

`POST /register/`

Δημιουργεί μια νέα εγγραφή με τα στοιχεία τη συσκευής και το ανωνυμοποιημένο αλφαριθμητικό

`POST /history/`

Καταγράφει τη νέα θέση του χρήστη με βάση το ανωνυμοποιημένο αλφαριθμητικό

Τα endpoints που εμφανίζουν λίστες όπως το items, sights, beaches, places δέχονται τις παρακάτω παραμέτρους.

`limit=10` Πόσα αποτελέσματα να εμφανίσει

`offset=10` Από ποια εγγραφή να ξεκινήσει

`orderby=popularity/likes/name/id` Ταξινόμηση ανά

`order=asc/desc ή ascending/descending` Αύξουσα ή φθίνουσα ταξινόμηση

`full=true/false` Αν εμφανίσει όλη την πληροφορία ή ορισμένα μόνο πεδία

4.3.3 Περιγραφή κύκλου εκτέλεσης του web API

Σε μια εφαρμογή βασισμένη στο Slim συνήθως εργαζόμαστε απευθείας με αντικείμενα αιτήματος (Request Objects) και απόκρισης (Response objects). Αυτά τα αντικείμενα αντιπροσωπεύουν το πραγματικό HTTP αίτημα που λαμβάνει ο διακομιστής ιστού και την τελική απόκριση που επιστρέφεται στον πελάτη.

Αυτά τα αντικείμενα αιτήματος κι απόκρισης μέσω του δρομολογητή (Router) περνούν στο κατάλληλο τμήμα της εφαρμογής όπου μπορούν να τροποποιηθούν και να επιστραφούν.

Αν υπάρχουν Middlewares τα αντικείμενα διέρχονται μέσω αυτών και μπορούν να τροποποιηθούν.

Για να κατανοήσουμε καλύτερα πως δουλεύει το web API θα περιγράψουμε ένα παραδείγματα.

- Η εφαρμογή από τη φορητή συσκευή πρέπει να εμφανίσει μια λίστα με δέκα παραλίες ταξινομημένες αλφαβητικά.
- Από την εφαρμογή γίνεται ένα GET request στο endpoint **`/beach/`** με τις παραμέτρους **`/beach?limit=10&order=alphabetically`**.
- Το web API δέχεται το αίτημα και δημιουργεί το Request και το Response Object.
- Αν υπάρχουν middlewares τα αντικείμενα διέρχονται πρώτα από αυτά.

- Στο middleware γίνεται ο έλεγχος των παραμέτρων. Στο συγκεκριμένο παράδειγμα έχουν δοθεί δύο παράμετροι σωστά ενώ οι υπόλοιπες παίρνουν τις προκαθορισμένες τιμές.
- Στη συνέχεια το Request και Response object με τις συμπληρωμένες παραμέτρους, μέσω του Router περνούν στην κατάλληλη συνάρτηση προς εκτέλεση γι' αυτό το endpoint.
- Η συνάρτηση αυτή εκτελεί μια SQL εντολή αιτώντας τις δέκα παραλίες ταξινομημένες αλφαβητικά από τη βάση δεδομένων.
- Τα δεδομένα από τη βάση δεδομένων μετατρέπονται σε μορφή JSON και προστίθενται στο Response object.
- Το Request και Response object διέρχονται από το middleware που στη συγκεκριμένη περίπτωση δεν τροποποιεί κάτι σε αυτά.
- Το Response object μετατρέπεται από το framework στην http απάντηση και δίνεται στον πελάτη με σώμα το JSON αρχείο με τα αποτελέσματα του ερωτήματος.

4.4 Η Εφαρμογή της φορητής συσκευής

Η εφαρμογή έχει δημιουργηθεί με το Framework7 και μόνη εξωτερική εξάρτηση τη βιβλιοθήκη axios για τα http αιτήματα ενώ για την εμφάνιση των δεδομένων χρησιμοποιείται η ενσωματωμένη Template Engine Template7.

4.4.1 Apache Cordova plugins

Έχουν χρησιμοποιηθεί τα παρακάτω plugins από το αποθετήριο του Apache Cordova.

Device (cordova-plugin-device)

Επιστρέφει ένα αντικείμενο με τα χαρακτηριστικά της συσκευής στην οποία εκτελείται η εφαρμογή. Τα χαρακτηριστικά αυτά αφορούν την πλατφόρμα εκτέλεσης, την έκδοση του λειτουργικού συστήματος, το μοντέλο τη συσκευής, και την έκδοση του Cordova που χρησιμοποιήθηκε.

Geolocation (cordova-plugin-geolocation)

Επιτρέπει τη χρήση του GPS της εφαρμογής επιστρέφοντας δεδομένα του γεωγραφικού πλάτους και μήκους.

Splashscreen (cordova–plugin-splashscreen)

Το plugin αυτό εμφανίζει και αποκρύπτει τη εισαγωγική οθόνη (splash screen) κατά την έναρξη της εφαρμογής.

4.4.2 Templates

Η χρήση Templates είναι ένας τρόπος να διαχωριστεί η δομή της εμφάνισης (view) από τα δεδομένα (content). Αυτός ο διαχωρισμός επιτρέπει την επαναχρησιμοποίηση κώδικα HTML, κάνει τον κώδικα πιο ευανάγνωστο και επιτρέπει να γίνονται πιο εύκολα τροποποιήσεις αλλάζοντας απλά τα δεδομένα.

Για παράδειγμα για να εμφανιστεί η οθόνη με τις λεπτομέρειες ενός αξιοθέατου χρησιμοποιείται το παρακάτω template.

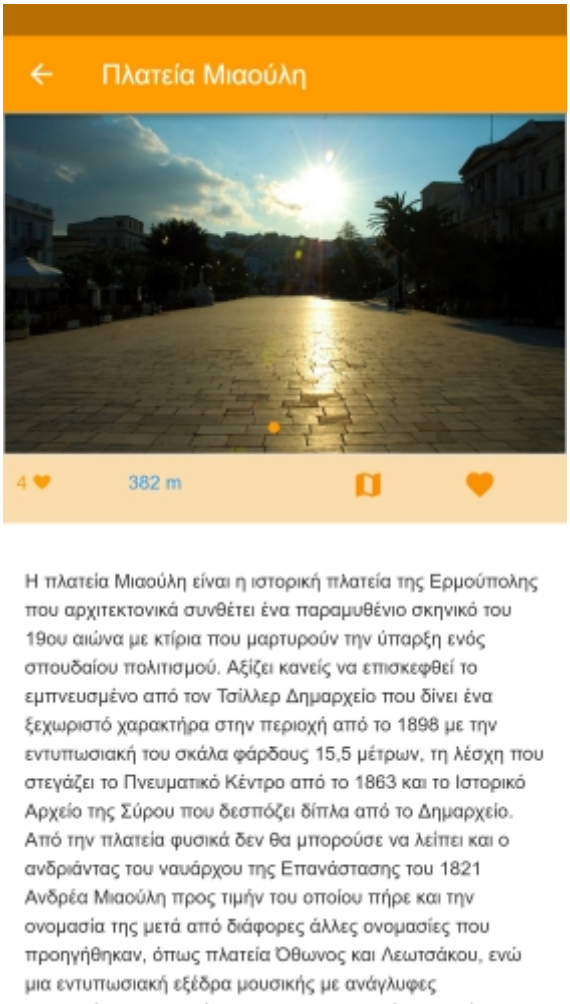
```
<script id="details-template" type="text/template7">
    <div class="row row-pall-orange">
        <div class="col-20 text-color-orange">{{likes}} &#10084;</div>
        <div class="col-40 text-color-blue">{{#if lat}} {{distance}}
{{/if}}</div>
        <div class="col-20">
            <a href="#" class="link"><i class="material-icons md-dark
orange600">map</i></a>
        </div>
        <div class="col-20">
            <a href="#" class="link" id="add-to-fav"><i class="fav-icon
material-icons md-dark orange600">favorite_border</i></a>
        </div>
    </div>
    <div class="block">
        {{content}}
    </div>
</script>
```

Το template περιλαμβάνει κάποιες μεταβλητές μέσα σε διπλά άγκιστρα **{{ }}** οι οποίες σχετίζονται με τα αντίστοιχα δεδομένα. Τα δεδομένα είναι ένα αντικείμενο JavaScript με την παρακάτω μορφή.

```
{
      "id": "12",
      "title": "Πλατεία Μιαούλη",
      "category": "Αξιοθέατα",
      "intro": "",
      "image": "Syros_emoupolis_rathaus_240707.jpg",
      "thumbnail": "Syros_emoupolis_rathaus_240707.jpg",
      "gallery": "miaouli-
109190932_eedb894d79_z.jpg,Syros_emoupolis_rathaus_240707.jpg",
      "content": "Η πλατεία Μιαούλη είναι η ιστορική πλατεία της Ερμούπολης
που ... ",
      "likes": "4",
```

```
      "lng": "24.942867",
      "lat": "37.444644",
      "distance": "382 m"
}
```

Το template με τα αντίστοιχα δεδομένα παράγουν την παρακάτω οθόνη.



Εικόνα 11: Template δεδομένα οθόνη

Με αλλαγή των δεδομένων και χρησιμοποιώντας το ίδιο template δημιουργείται σελίδων κάθε αξιοθέατου.

Αντίστοιχα templates μπορούν να χρησιμοποιηθούν για μια λίστα αντικειμένων.

Για παράδειγμα το παρακάτω template χρησιμοποιείται για τις εμφανίσεις λιστών με αξιοθέατα στην εφαρμογή.

```
<script id="list-template" type="text/template7">
    {{#each data}}
```

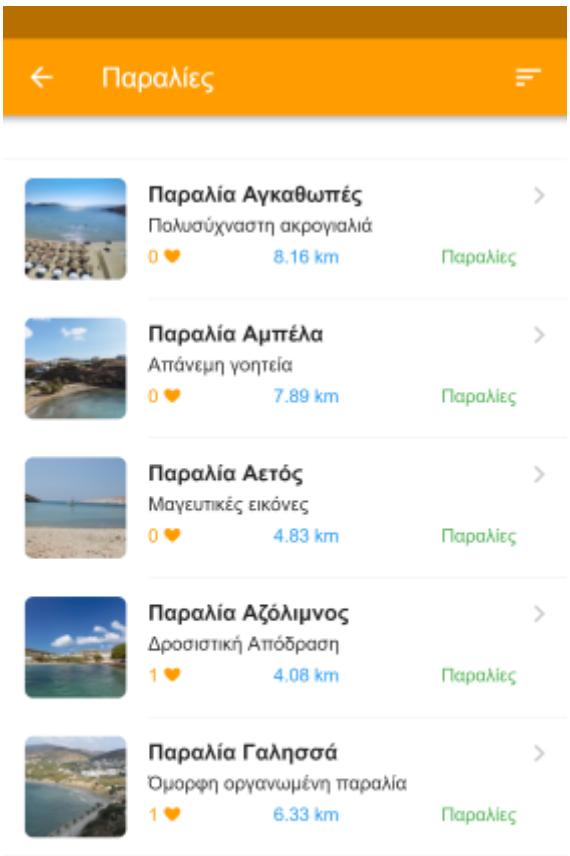
```
    <li>
        <a href="/details-template/{{id}}/" class="item-link item-content">
            <div class="item-media"><img src="{{thumbnail}}" width="74"/></div>
            <div class="item-inner">
                <div class="item-title-row">
                    <div class="item-title">{{title}}</div>
                </div>
                <div class="item-subtitle">{{intro}}  </div>
                <div class="row">
                    <div class="col-20">
                        <small><span class="text-color-orange">{{likes}}
&#10084;</span></small>
                    </div>
                    <div class="col-30">
                        <small><span
class="text-color-blue">{{distance}}</span></small>
                    </div>
                    <div class="col-30">
                        <small><span
class="text-color-green">{{category}}</span></small>
                    </div>
                </div>
            </div>
        </a>
    </li>
    {{/each}}
</script>
```

Τα αντίστοιχα δεδομένα είναι ένα αντικείμενο όπως το παρακάτω.

```
{
      "data": [{
                  "id": "6",
                  "title": "Παραλία Αγκαθωπές",
                  "category": "Παραλίες",
                  "intro": "Πολυσύχναστη ακρογιαλιά",
                  "image":
"http://travel-guide.lrn.gr/api/public/images/paralia-agkathopes-01-600.jpg",
                  "thumbnail":
"http://travel-guide.lrn.gr/api/public/images/thumbs/paralia-agkathopes-02-
600.jpg",
                  "likes": "0",
                  "lng": "24.88097",
                  "lat": "37.38566",
                  "distance": "8.16 km"
            }, {
                  "id": "7",
                  "title": "Παραλία Αμπέλα",
                  "category": "Παραλίες",
                  "intro": "Απάνεμη γοητεία",
                  "image":
"http://travel-guide.lrn.gr/api/public/images/paralia-ampela-02-600.jpg",
                  "thumbnail":
"http://travel-guide.lrn.gr/api/public/images/thumbs/paralia-ampela-02-600.jpg",
                  "likes": 0,
                  "lng": "24.905829",
                  "lat": "37.376402",
                  "distance": "7.89 km"
            },
```

```
        ....
        ]
}
```

Η αντίστοιχη οθόνη που παράγεται από το συνδυασμό του template με τα δεδομένα:



Εικόνα 12: Template λίστας αντικειμένων

4.4.3 HTTP Requests

Τα αντικείμενα που χρησιμοποιούνται προς εμφάνιση στα templates δημιουργούνται απευθείας από τα δεδομένα που προέρχονται από το API.

Για το πρώτο παράδειγμα γίνεται μια αίτηση στο αντίστοιχο endpoint του API **`GET /items/12`** .

Η αίτηση αυτή γίνεται με κώδικα ως εξής:

```
var url = settings.baseUrl + 'items/' + page.route.params.id;

axios.get(url)
```

```
    .then(function (response) {
            log.debug(response.data);
            var context = response.data;
            var html = compiledDetailsTemplate(context);
            $$('#details-div').html(html);
        })
        .catch(function (error) {
            log.error(error);
        });
```

Η εφαρμογή πέρα από τα GET Request εκτελεί PUT και POST.

Σαν παράδειγμα POST request εκτελείται κατά την πρώτη εκτέλεση της εφαρμογής στο endpoint **`POST /register/`** του API όπου δηλώνεται το τυχαίο αλφαριθμητικό που δημιουργείται από την εφαρμογή και μια σειρά από χαρακτηριστικά της συσκευής.

```
function registerDevice() {
    var url = settings.baseUrl + 'register';
    var data = {
        guid: appData.guid,
        cordova: deviceData.cordova,
        model: deviceData.model,
        platform: deviceData.platform,
        manufacturer: deviceData.manufacturer,
        isvirtual: deviceData.isVirtual
    };
    axios({
        method: 'post',
        url: url,
        data: data
    })
        .then(function (response) {
            log.debug(response);
        })
        .catch(function (error) {
            log.error(error);
        });
}
```

4.4.4 Μόνιμη αποθήκευση δεδομένων

Η εφαρμογή σε διάφορα τμήματα της λειτουργίας της χρειάζεται μόνιμη αποθήκευση δεδομένων. Για παράδειγμα η λίστα με τα αγαπημένα ή οι ρυθμίσεις του χρήστη σχετικά με την αποστολή και αποθήκευση του στίγματος πρέπει να αποθηκευθούν και να διατηρηθούν μόνιμα.

Το cordova αν και παρέχει plugin για μόνιμη αποθήκευση στον αποθηκευτικό χώρο της συσκευής προτιμήθηκε η απλούστερη λύση του αντικειμένου localStorage το οποίο αποθηκεύει ζευγάρια key/value στο web view καθώς τα δεδομένα αποθήκευσης δεν είναι μεγάλου όγκου.

4.5 Ανωνυμοποίση δεδομένων

Η εφαρμογή, σε διάφορα σημεία της λειτουργίας της, συγκεντρώνει στοιχεία της συσκευής και συντεταγμένες της θέσης. Αυτά τα στοιχεία αποθηκεύονται τοπικά αλλά και στη βάση δεδομένων.

Στα στοιχεία της συσκευής που αποθηκεύονται για στατιστικούς λόγους (έκδοση λειτουργικού συστήματος, μοντέλο συσκευής, έκδοση εφαρμογής κ.α) δεν περιλαμβάνονται σειριακοί αριθμοί συσκευών ή στοιχεία του χρήστη ενώ τα στοιχεία της θέσης, αν ο χρήστης επιτρέπει την αποστολή και την αποθήκευσή τους, συνοδεύονται από ένα μοναδικό αναγνωριστικό δημιουργημένο τυχαία.

4.6 Γεννήτρια εφαρμογών – τουριστικών οδηγών

Μια από τις βασικές προδιαγραφές του συστήματος είναι η δυνατότητα επαναχρησιμοποίησης του σαν βάση για άλλους τουριστικούς οδηγούς διαφορετικών περιοχών χωρίς αλλαγές στον κώδικα.

Οι αλλαγές που πρέπει να γίνουν σε αυτή τη περίπτωση είναι οι εξής:

α) Αλλαγή στο αρχείο settings.php του API με τα στοιχεία σύνδεσης της νέα βάση δεδομένων και τον ορισμό ενός κεντρικού σημείου συντεταγμένων του νέου τουριστικού προορισμού (center).

β) Αλλαγή στο αντικείμενο app του αρχείου app.js της εφαρμογής με ενημέρωση των πεδίων name, id, version και αλλαγή στο αντικείμενο settings με τη διεύθυνση url του νέου api.

γ) Προσθήκη των νέων δεδομένων στον πίνακα items με εισαγωγή των στοιχείων με τη βοήθεια εργαλείου διαχείρισης βάσεων δεδομένων ή με χρήση ενός κατάλληλα προετοιμασμένου αρχείου csv με τα δεδομένα.

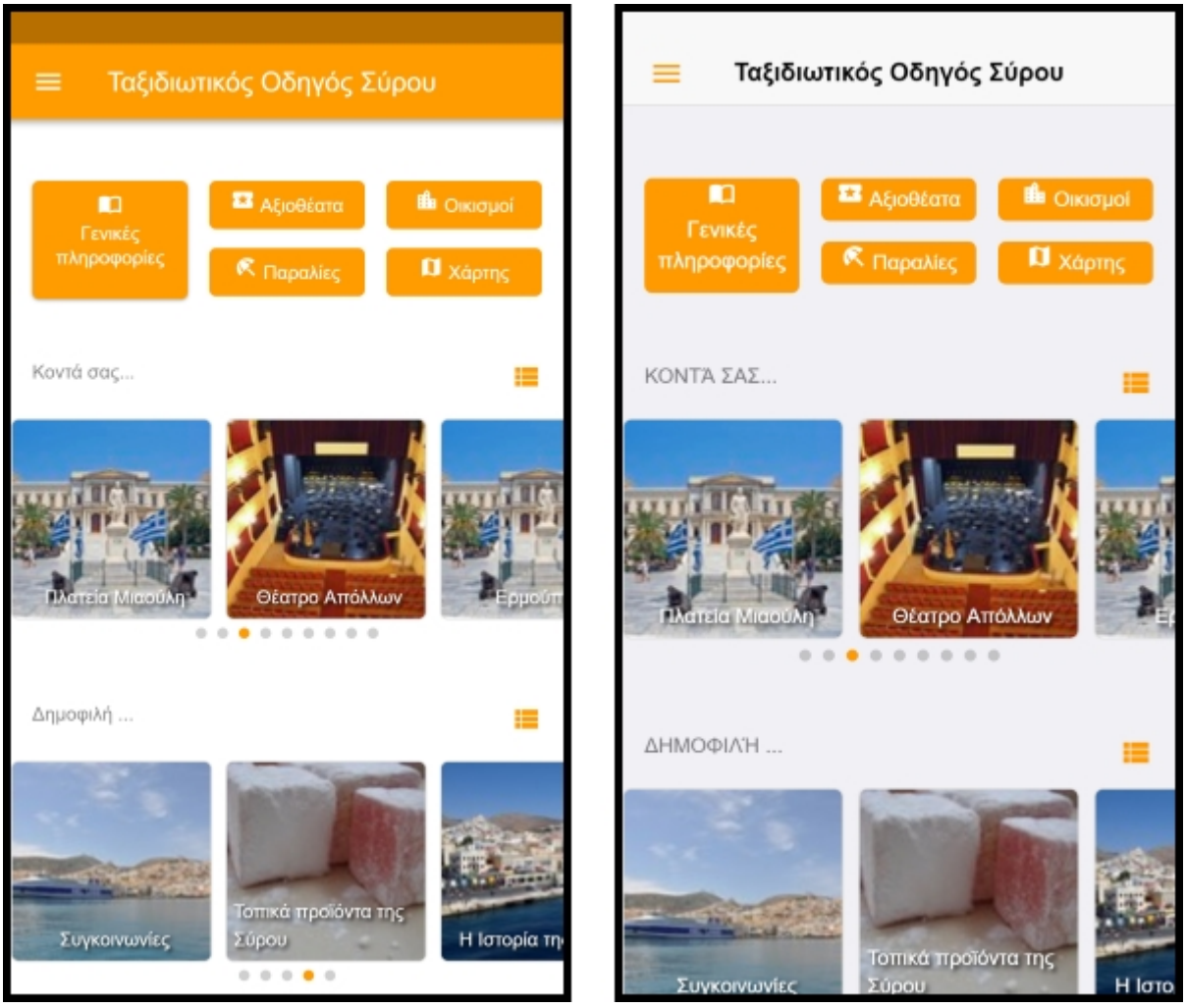
δ) Προσθήκη των φωτογραφιών και των μικρογραφιών στον φάκελο images και thumbs.

Επιπρόσθετα η εφαρμογή μπορεί να εμπλουτίζεται με νέα δεδομένα ή να τροποποιούνται και να διορθώνονται τα παλαιότερα απλά με αλλαγές στον πίνακα items και την προσθήκη των αντίστοιχων φωτογραφιών στον φάκελο images και thumbs του api χωρίς την ανάγκη αναβάθμισης και νέας έκδοσης της εφαρμογής στα app stores.

Κεφάλαιο 5ο – Λειτουργία Εφαρμογής

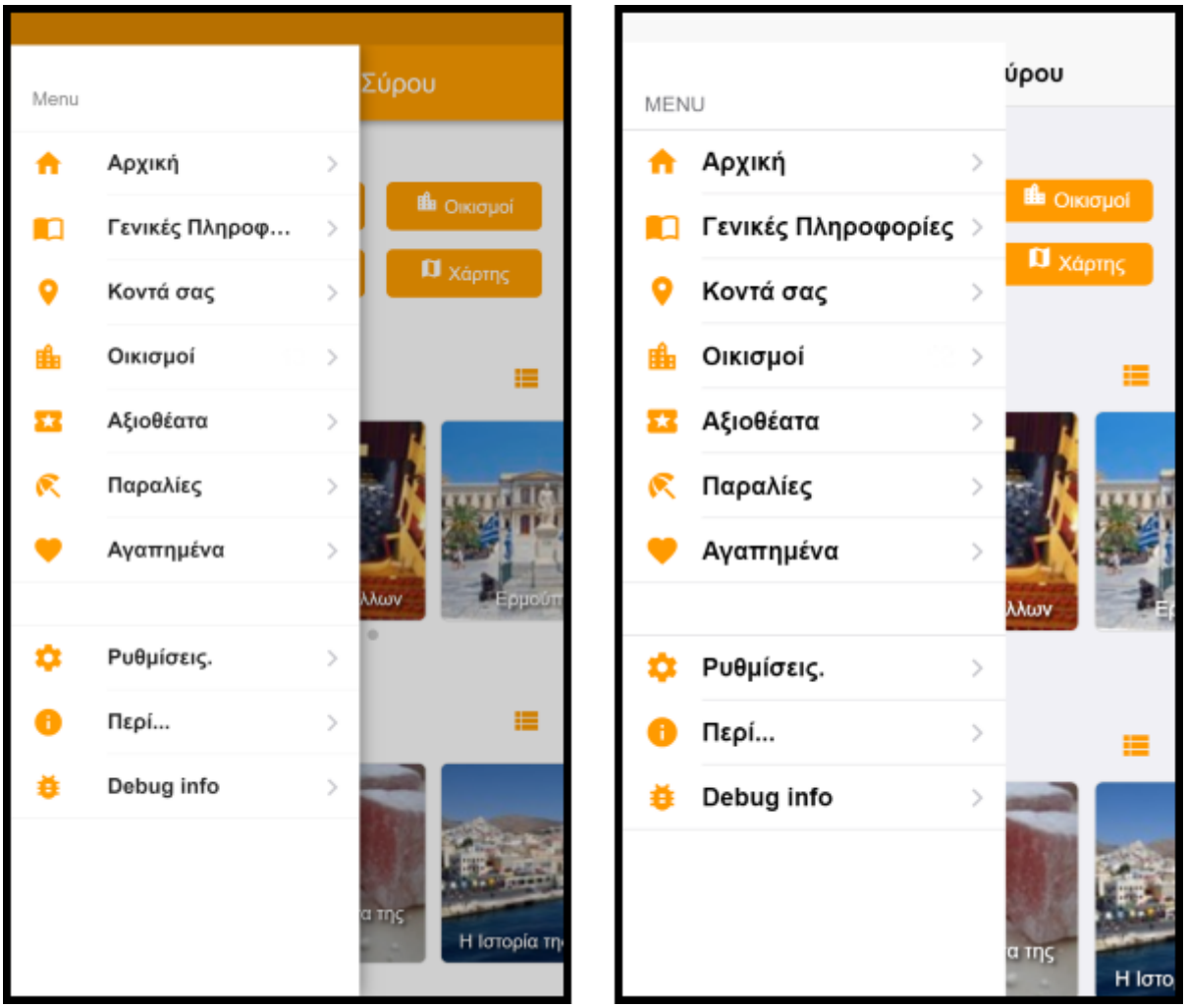
Κύρια λειτουργία της εφαρμογής είναι η προβολή σημείων τουριστικού ενδιαφέροντος ανά κατηγορία ή με βάση την απόσταση από τη θέση του χρήστη. Για τα σημεία αυτά περιλαμβάνεται κείμενο και φωτογραφίες καθώς και η απόσταση τους από τη θέση του χρήστη. Επιπρόσθετα δίνεται η δυνατότητα να προστεθεί το σημείο στα "αγαπημένα" ώστε αργότερα ο χρήστης να έχει εύκολη πρόσβαση στις πληροφορίες.

Στην αρχική οθόνη ο χρήστης έχει τη δυνατότητα να επιλέξει κάποια από τις κατηγορίες των αξιοθέατων ή να περιηγηθεί στα sliders που εμφανίζουν τα κοντινά σε αυτόν σημεία, τα πιο δημοφιλή με βάση πόσες φορές έχουν γίνει αγαπημένα από τους άλλους χρήστες ή στα δικά του αγαπημένα σημεία.

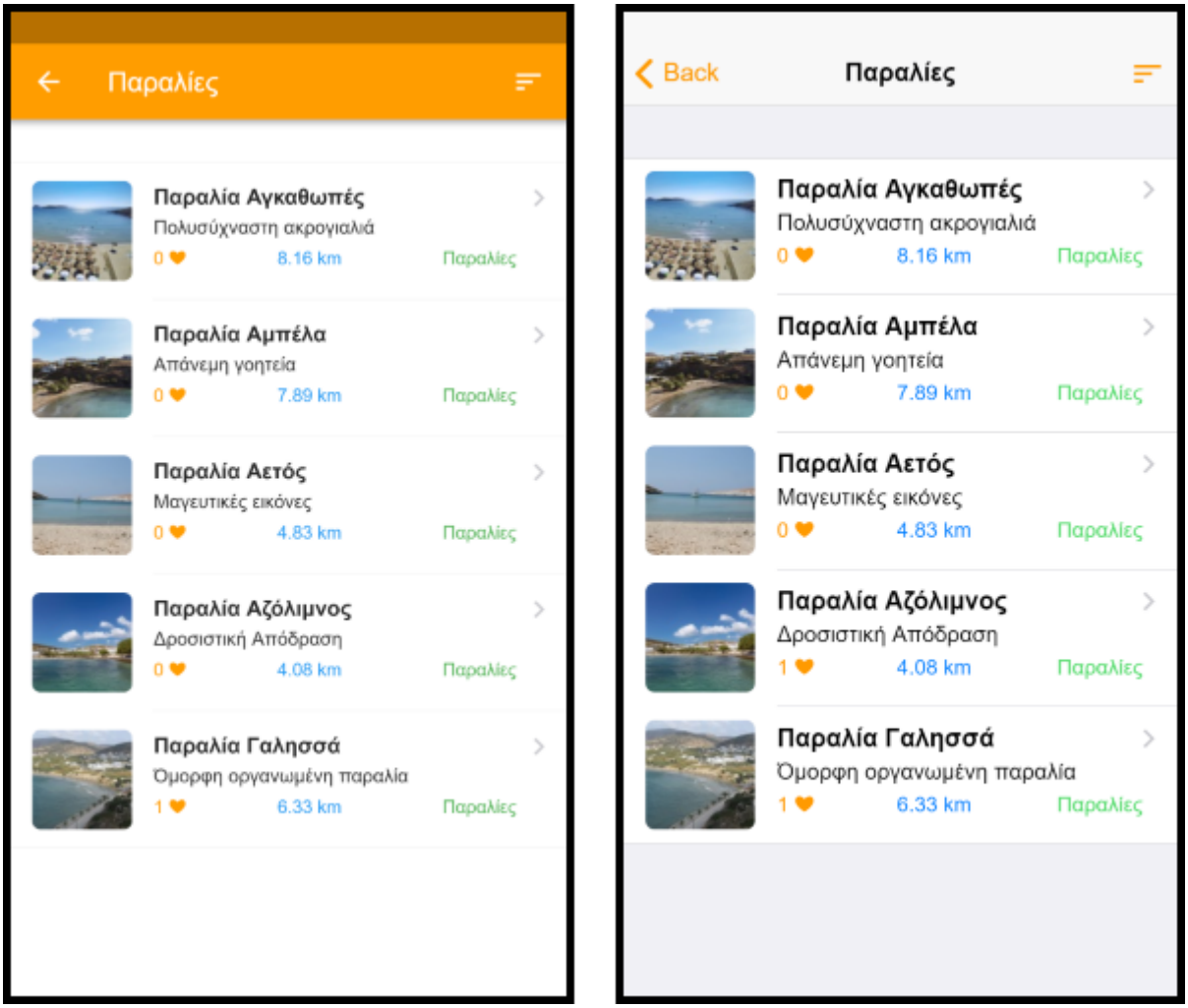


Εικόνα 13: Αρχική οθόνη εφαρμογής

Σε κάθε οθόνη ο χρήστης μπορεί να εμφανίσει το μενού της εφαρμογής για εύκολη πρόσβαση στις οθόνες.

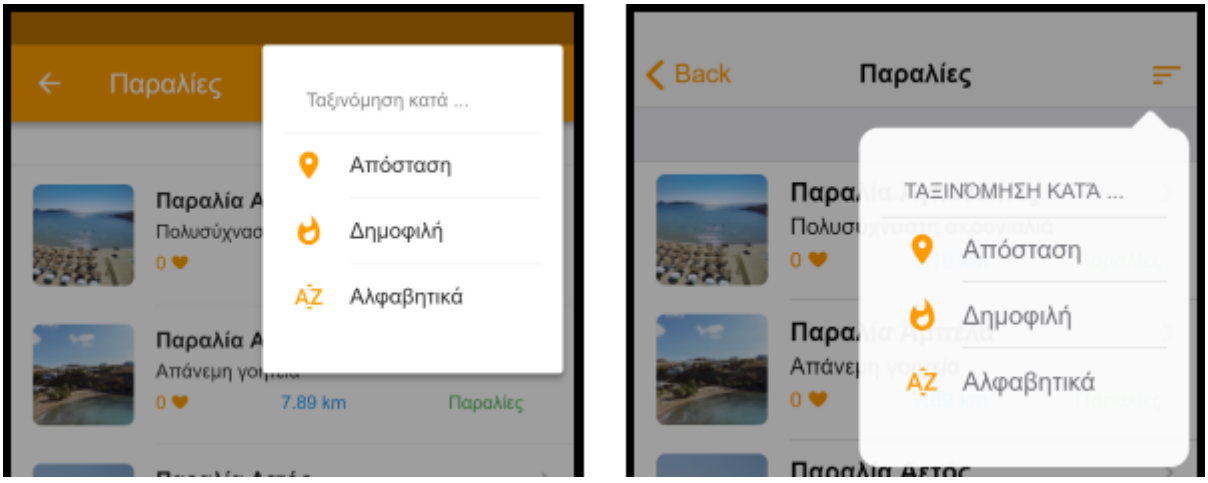


Εικόνα 14: Μενού Επιλογών



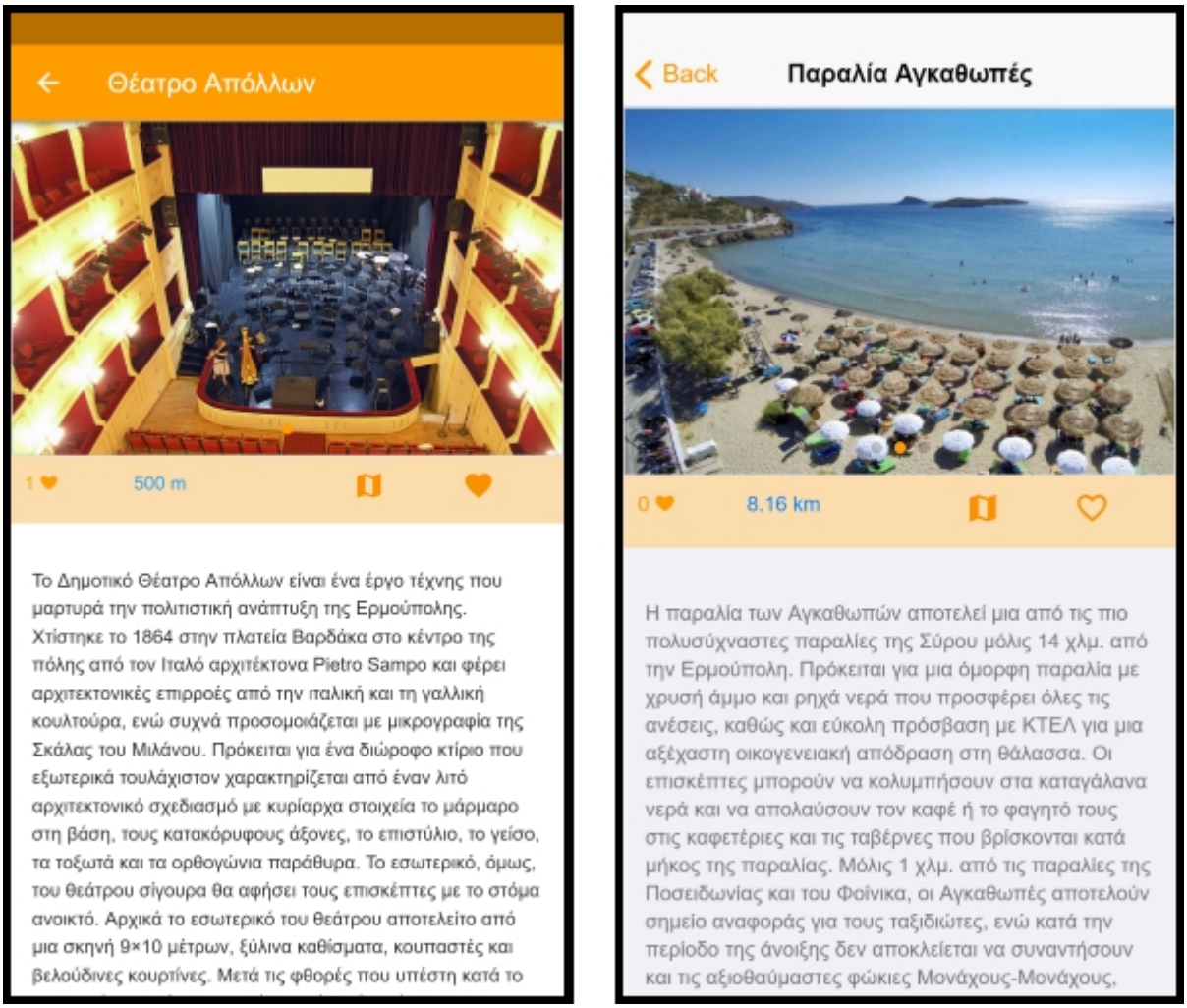
Εικόνα 15: Προβολή Λίστας

Όταν ο χρήστης επιλέγει μια κατηγορία εμφανίζεται μια λίστα με σημεία ενδιαφέροντος αυτής της κατηγορίας. Η λίστα αυτή μπορεί να ταξινομηθεί με βάση την απόσταση, τη δημοφιλία ή αλφαβητικά.



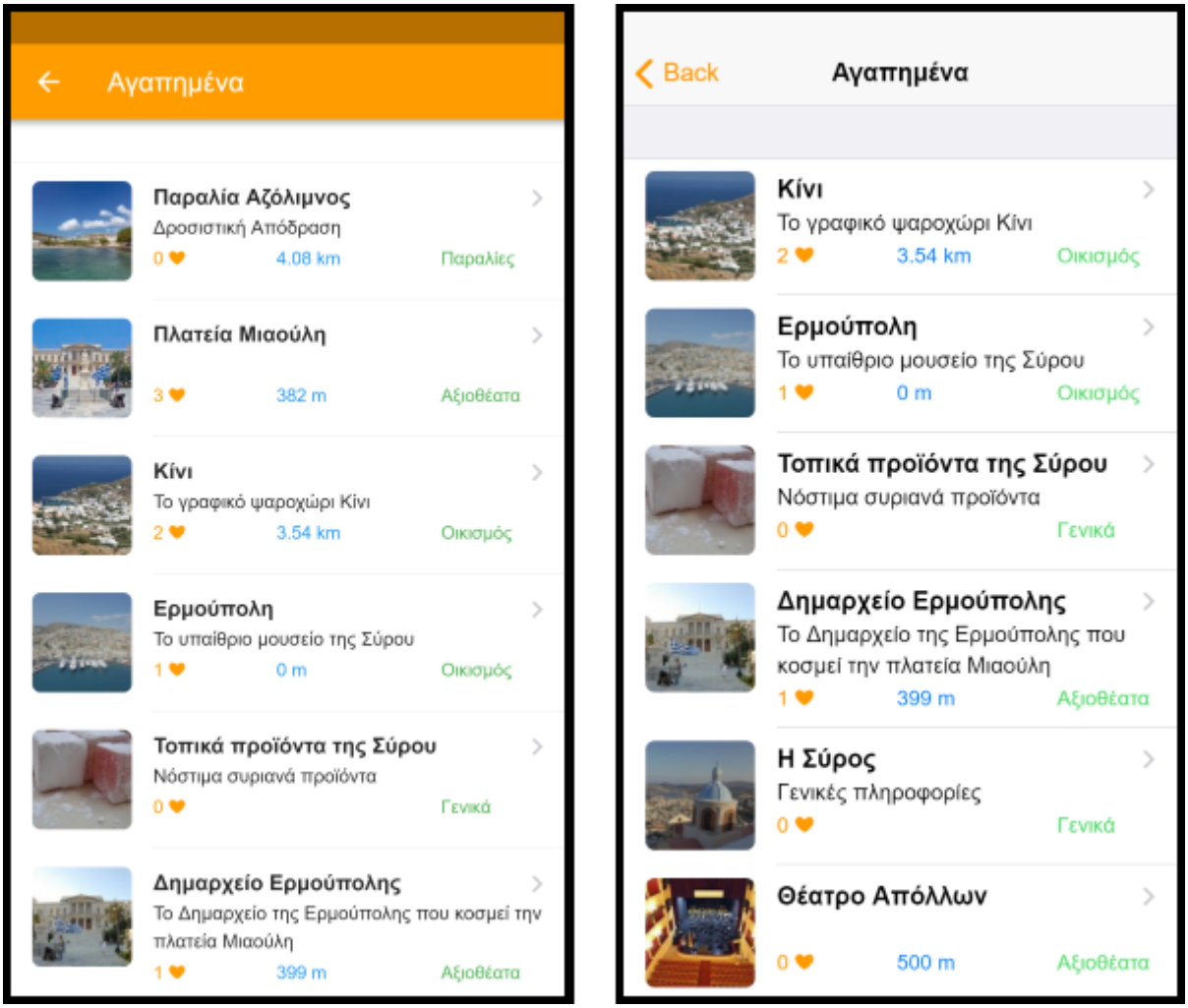
Εικόνα 16: Ταξινόμηση λίστας

Με την επιλογή ενός αξιοθέατου εμφανίζεται η οθόνη λεπτομερειών, με το κείμενο καθώς και με κυλιόμενη προβολή (slider) φωτογραφιών που είναι διαθέσιμες .



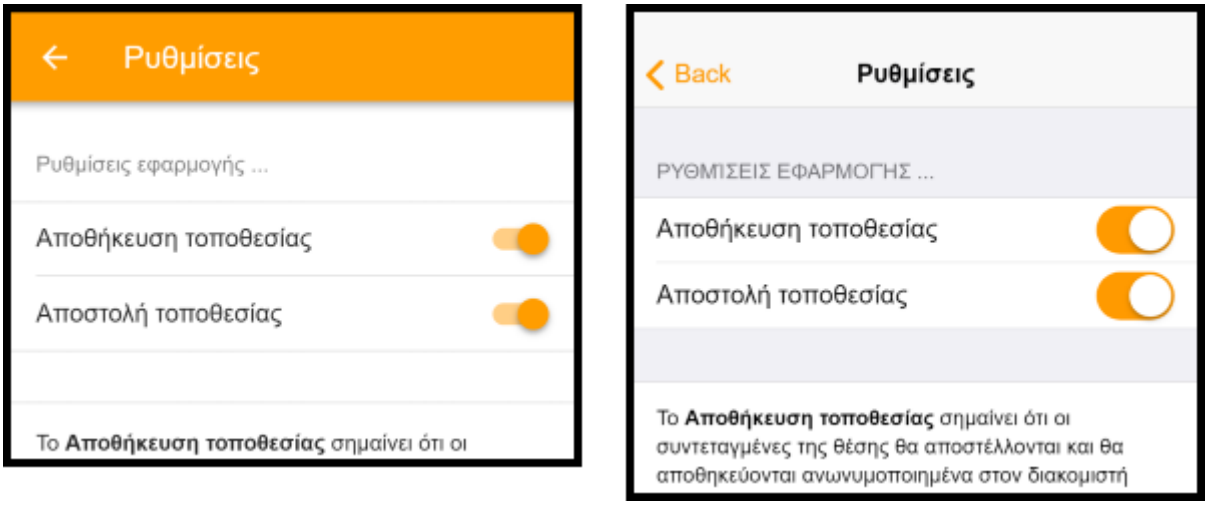
Εικόνα 17: Προβολή Λεπτομερειών

Στην προβολή λεπτομερειών ο χρήστης μπορεί πατώντας το εικονίδιο “χάρτης” να ανοίξει την εφαρμογή των χαρτών της συσκευής στο σημείο που βρίσκεται το συγκεκριμένο αξιοθέατο ενώ με πάτημα στο κουμπί “καρδιά” να προσθέσει (ή να αφαιρέσει αν το κουμπί είναι ήδη πατημένο) κάποιο στοιχείο στα αγαπημένα.



Εικόνα 18: Λίστα 'Αγαπημένων' του Χρήστη

Στην οθόνη ρυθμίσεις ο χρήστης μπορεί να επιλέξει αν τα γεωχωρικά δεδομένα θα αποστέλλονται και θα αποθηκεύονται στο διακομιστή.



Εικόνα *19: Ρυθμίσεις εφαρμογής*

Εκτός από τη διαδραστικότητα με το χρήστη όπως περιγράφηκε παραπάνω η εφαρμογή εκτελεί μια σειρά διαδικασιών στο παρασκήνιο.

Κατά την έναρξη ελέγχει αν είναι πρώτη φορά που εκτελείται στην συγκεκριμένη συσκευή. Σε αυτή τη περίπτωση δημιουργεί ένα τυχαίο αλφαριθμητικό, το αποθηκεύει τοπικά και το στέλνει στον διακομιστή μαζί με επιπλέον στοιχεία όπως την έκδοση του λειτουργικού συστήματος και της εφαρμογής.

Πριν από κάθε αίτημα προς το API, η εφαρμογή ελέγχει τις συντεταγμένες θέσης μέσω το GPS της συσκευής. Αν οι ρυθμίσεις του χρήστη επιτρέπουν την αποστολή αυτής της πληροφορίας, τότε οι συντεταγμένες προστίθενται στο αίτημα που αποστέλλεται στο API. Με τη βοήθεια αυτών των συντεταγμένων το API επιστρέφει πέρα από τα δεδομένα και την απόσταση του χρήστη από το αξιοθέατο ή την λίστα των τοποθεσιών που αιτήθηκαν. Παράλληλα, αν οι ρυθμίσεις του χρήστη επιτρέπουν και την αποθήκευση των συντεταγμένων θέσης του, το στίγμα αποθηκεύεται με το αλφαριθμητικό του χρήστη στη βάση δεδομένων.

Σε περίπτωση που ο χρήστης δεν επιτρέπει την χρήση των συντεταγμένων, το API χρησιμοποιεί τις συντεταγμένες κεντρικού σημείου της περιοχής που έχουν ορισθεί γι' αυτή τη περίπτωση στο αρχείο ρυθμίσεων.

Με αυτό το τρόπο συγκεντρώνονται διαδοχικά δεδομένα θέσης κάθε χρήστη κατά τη διάρκεια της χρήσης της εφαρμογής τα οποία μπορούν να χρησιμοποιηθούν σε επεκτάσεις τις εφαρμογής ή στατιστική μελέτη.

Κεφάλαιο 6ο - Συμπεράσματα & μελλοντική εργασία

6.1 Συμπεράσματα

Στην παρούσα διπλωματική εργασία υλοποιήθηκε μια εφαρμογή για τις δύο επικρατούσες πλατφόρμες φορητών συσκευών και η αντίστοιχη διαδικτυακή υπηρεσία με την οποία οι εφαρμογές αλληλεπιδρούν και αντλούν τα δεδομένα. Το θέμα της εφαρμογής είναι τα τουριστικά αξιοθέατα της Σύρου.

Παράλληλα όμως η αρχιτεκτονική του έργου έγινε με τέτοιο τρόπο ώστε να επιτρέπεται, με μικρές αλλαγές, η επαναχρησιμοποίηση του συστήματος για δημιουργία πανομοιότυπων κλωνοποιημένων εφαρμογών άλλων περιοχών.

Για την επιλογή της τεχνολογικής στοίβας που χρησιμοποιήθηκε έγινε μια μελέτη στις τεχνικές δημιουργίας εφαρμογών για φορητές συσκευές όπου επιλέχθηκε το υβριδικό μοντέλο εφαρμογών και στην τεχνολογία των διαδικτυακών υπηρεσιών και των web APIs με την επιλογή δημιουργίας ενός RESTful API.

Το λογισμικό για την υλοποίηση επιλέχθηκε με βάση την ευκολία χρήσης και εγκατάστασης παράλληλα με την ταχύτητα και την αποδοτικότητα του σε τέτοια είδους έργα.

Το αποτέλεσμα δείχνει ότι για απλές εφαρμογές εμφάνισης δεδομένων το υβριδικό μοντέλα λειτουργεί πολύ καλά και επιπλέον η “διαδικτυακή” του φύση κάνει την επικοινωνία με RESTful APIs ιδιαίτερα εύκολη.

6.2 Μελλοντική εργασία

Το σημαντικότερο στοιχείο ενός τουριστικού οδηγού είναι η πληρότητα της πληροφορίας που περιλαμβάνει. Αυτονόητα λοιπόν το πρώτο σημείο βελτίωσης θα μπορούσε να είναι η προσθήκη δυνατοτήτων προβολής περισσότερης πληροφορίας όπως ξενοδοχεία, ενοικιαζόμενα δωμάτια, εστιατόρια και γενικά καταστήματα τουριστικού ενδιαφέροντος της περιοχής.

Επιπλέον μια τέτοια εφαρμογή αφορά επισκέπτες από διαφορετικές χώρες που μιλούν διαφορετικές γλώσσες. Οπότε μια ιδιαίτερα χρήσιμη προέκταση θα ήταν η δυνατότητα πολλών γλωσσών (multi language) όπου η εφαρμογή αυτόματα ή ο χρήστης με επιλογή θα μπορεί να αλλάξει τη γλώσσα του περιβάλλοντος και το web API με την κατάλληλη παράμετρο να επιστρέφει τα περιεχόμενα στην κάθε γλώσσα επιλογής.

Η εφαρμογή είναι δημιουργημένη με γνώμονα την ευκολία μετατροπής και χρήσης για διαφορετικές περιοχές. Αυτή τη στιγμή όμως δεν παρέχει τα εργαλεία που θα διευκόλυναν την προσθήκη της κατάλληλης πληροφορίας. Ο δημιουργός της κλωνοποιημένης εφαρμογής πρέπει να κάνει τις αλλαγές σε αρχείο ρυθμίσεων και να προσθέσει τα νέα δεδομένα στη βάση δεδομένων με χρήση τρίτων εργαλείων όπως το phpMyAdmin και το Adminer.

Μια προέκταση της εφαρμογής λοιπόν θα ήταν η δημιουργία ενός γραφικού διαδικτυακού περιβάλλοντος διαχείρισης της εφαρμογής όπου ο διαχειριστής θα μπορεί να προσθέσει και να τροποποιήσει τα δεδομένα και τις φωτογραφίες με εύκολο τρόπο. Επιπρόσθετα αυτό το διαχειριστικό θα μπορούσε να περιλαμβάνει τη διαδικασία εγκατάστασης του web API με απλά βήματα χωρίς να απαιτείται η τροποποίηση αρχείων.

Για την υλοποίηση της εφαρμογής για τις φορητές συσκευές έχει χρησιμοποιηθεί καθαρή JavaScript σε συνδυασμό με το Framework7 για το περιβάλλον χρήστη. Η υλοποίηση αυτή είναι εύκολη και αποδοτική σε απλές εφαρμογές αλλά σε πιο περίπλοκες περιπτώσεις εφαρμογών η χρήση κάποιου framework επιβάλλεται. Το Framework7 επιτρέπει τη χρήση σύγχρονων frameworks όπως το Vuejs και Angular και βιβλιοθηκών όπως το React. Μια προέκταση λοιπόν θα ήταν η συγγραφή του κώδικα της εφαρμογής με χρήση ενός από τα σύγχρονα αυτά frameworks.

Ένα τμήμα της εφαρμογής επιτρέπει την αποστολή ανωνυμοποιημένων δεδομένων της θέσης του χρήστη. Αυτά τα δεδομένα θα μπορούσαν να χρησιμοποιηθούν παράγοντας προτάσεις επισκέψεων η διαδρομών προς τους χρήστες. Επιπλέον αυτά τα δεδομένα θα μπορούσαν να ιδιαίτερα χρήσιμα για εξόρυξη πληροφορίας για έναν τουριστικό προορισμό ανεξάρτητα από την χρήση τους στην εφαρμογή.

Βιβλιογραφία

[1] Statista (2018) Annual number of global mobile app downloads 2017-2022. https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/

[2] Statista (2018) Leading Android app categories worldwide 2018. https://www.statista.com/statistics/200855/favourite-smartphone-app-categories-by-share-of-smartphone-users/

[3] Statista (2018) Most popular Apple App Store categories 2018. https://www.statista.com/statistics/270291/popular-categories-in-the-app-store/

[4] Győrödi R, Adrian V, Zmaranda D, Győrödi C (2017). A Comparative Study between Applications Developed for Android and iOS. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No. 11, 2017. https://pdfs.semanticscholar.org/71e9/75dd9dedaf434b4e18b2eb6612524443ef25.pdf

[5] Smt. Annapurna, K.V.S. Pavan Teja, Dr. Y. Satyanarayana Murty (2016). A Comparative Study on Mobile Platforms (Android vs. IOS). International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 5 Issue 3, March 2016. http://ijarcet.org/wp-content/uploads/IJARCET-VOL-5-ISSUE-3-547-553.pdf

[6] Reto Meier, Ian Lake (2018). Professional Android. John Wiley & Sons, Inc

[7] Matt Neuburg (2017). Programming iOS 11. O'Reilly

[8] Statista (2018) Global mobile OS market share 2009-2018, by quarter. https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[9] Statista (2018) Number of available apps in the Apple App Store 2008-2017. https://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/

[10] Android Platform Architecture. https://developer.android.com/guide/platform/

[11] Android Studio and SDK tools. https://developer.android.com/studio/

[12] Google Play. https://developer.android.com/distribute/

[13] iOS 11 Release Notes (2017). https://developer.apple.com/library/archive/releasenotes/General/RN-iOSSDK-11/index.html

[14] Xcode. https://developer.apple.com/xcode/

[15] Swift. https://developer.apple.com/swift/

[16] Disribute – App Store. https://developer.apple.com/distribute/

[17] Wafaa El-Kassas, Bassem Abdullah, Ahmed Yousef, Ayman Wahba. (2015). Taxonomy of Cross-Platform Mobile Applications Development Approaches. https://www.sciencedirect.com/science/article/pii/S2090447915001276#b0065

[18] Thakare B, Shirodkar D, Parween N, Parween S (2014). State of Art Approaches to Build Cross Platform Mobile Application. International Journal of Computer Applications (0975 – 8887) Volume 107 – No. 20, December 2014. https://pdfs.semanticscholar.org/834d/4b5ad7b32d0e065ed726ee9cd25e42bd14df.pdf

[19] Camden R (2015). Apache Cordova in Action. Manning

[20] Shotts K (2016). Mastering PhoneGap obile Application Development. Packt

[21] React Native – A framework for building native apps using React https://facebook.github.io/react-native/

[22] Xamarin App Development with Visual Studio https://visualstudio.microsoft.com/xamarin/

[23] Flutter – Technical Overview https://flutter.io/technical-overview/

[24] Lachgar M, Abdalia A (2017). Decision Framework for Mobile Development Methods. International Journal of Advanced Computer Science and Applications, Vol. 8, No. 2, 2017. https://pdfs.semanticscholar.org/6dfa/5e4ab2140892638f0d67a8db918adde93013.pdf

[25] Xanthopoulos S, Xinogalos S (2013). A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications. https://www.researchgate.net/publication/258010031_A_Comparative_Analysis_of_Cross-platform_Development_Approaches_for_Mobile_Applications

[26] Comentum (2015). Native vs Hybrid / PhoneGap App Development Comparison https://www.comentum.com/phonegap-vs-native-app-development.html

[27] Progressive Web Apps https://developers.google.com/web/progressive-web-apps/

[28] Huynh M, Ghimire P, Truong D (2017). Hybrid App Approach: Could It Mark the End of Native App Domination? Issues in Informing Science and Information Technology Volume 14, 2017. http://iisit.org/Vol14/IISITv14p049-065Huynh3472.pdf

[29] Charland A, LeRoux B (2011). Mobile Application Development: Web vs. Native https://dl.acm.org/citation.cfm?id=1968203

[30] Ethan Cerami (2002) Web Services Essentials. O'Reilly

[31] Vibha Kumari, (2015). Web Services Protocol: SOAP vs REST. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 5, May 2015 http://ijarcet.org/wp-content/uploads/IJARCET-VOL-4-ISSUE-5-2467-2469.pdf

[32] Mumbaikar S, Padiys P (2013). Web Services Based On SOAP and REST Principle. International Journal of Scientific and Research Publications, Volume 3, Issue 5, May 2013. https://pdfs.semanticscholar.org/1f1e/56119acd79c101e7f43f4d7b19f931a6e315.pdf

[33] P.A. Castillo, J.L. Bernierm M.G. Arenasm J.J. Merelom P. Garcia-Sanchez (2011). SOAP vs REST: Comparing a master-slave GA implementation https://arxiv.org/pdf/1105.4978.pdf

[34] Potti P, Ahuja S, Umapathy K, Prodanoff Z (2012). Comparing Performance of Web Service Interaction Styles: SOAP vs. REST. Proceedings of the Conference on Information Systems Applied Research. http://proc.conisar.org/2012/pdf/2208.pdf

[35] Fahad Aijaz, Muzzamil Aziz Chaudhary, Bernhard Walke. Performance Comparison of a SOAP and REST Mobile Web Server. http://www.uet.edu.pk/Conferences/icosst2009/presentations_2009/Research_Papers/Performance_Comparison_of_a_SOAP_and_REST_Mobile_Web_Server.pdf

[36] OpenAPI Specifications https://swagger.io/specification/

[37] Apache Cordova https://cordova.apache.org/

[38] Guidlines – Materia Design https://material.io/design/guidelines-overview/

[39] iOS Human Interface Guidlines https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/

[40] Framework7 http://framework7.io/

[41] Template7 http://idangero.us/template7/

[42] Swiper Touch Slider http://idangero.us/swiper/

[43] axios https://github.com/axios/axios

[44] loglevel https://github.com/pimterry/loglevel

[45] PHP: Hypertext Preprocessor http://php.net/

[46] Slim Framework https://www.slimframework.com/

[47] MySQL https://www.mysql.com/

[48] MySQL Spatial Function References https://dev.mysql.com/doc/refman/5.6/en/spatial-function-reference.html

[49] Monolog – Logging for PHP https://seldaek.github.io/monolog/

Παράρτημα Α

Κώδικας έργου

Ο κώδικας του έργου βρίσκεται στο GitHub στις παρακάτω διευθύνσεις:

Ο κώδικας της διαδικτυακής υπηρεσίας (web API)

https://github.com/std130962/travel-guide-api

Ο κώδικας της εφαρμογής για φορητές συσκευές

https://github.com/std130962/travel-guide-app

Οδηγίες εγκατάστασης

Αναλυτικές οδηγίες εγκατάστασης της διαδικτυακής υπηρεσίας, της βάσης δεδομένων, της εφαρμογής για τις δύο πλατφόρες καθώς της τροποποίησης για δημιουργία εφαρμογών διαφορετικής περιοχής.

https://github.com/std130962/travel-guide-info/wiki