ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Εξατομικευμένη περιήγηση με λειτουργία επαυξημένης πραγματικότητας** **Cross platform personalized guide with Augmented reality features**
Ονοματεπώνυμο Φοιτητή	**Μάριος Πάτσης**
Πατρώνυμο	**Χρήστος**
Αριθμός Μητρώου	**ΜΠΣΠ 18020**
Επιβλέπων	**Ευάγγελος Σακκόπουλος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2022**

Τριμελής Εξεταστική Επιτροπή

Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα
Ευάγγελος Σακκόπουλος	Ευθύμιος Αλέπης	Διονύσιος Σωτηρόπουλος
Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής	Επίκουρος Καθηγητής

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια της ολοκλήρωσης των σπουδών μου στο Μεταπτυχιακό Πρόγραμμα Σπουδών «Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης» του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς.

Θέλω να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, Ευάγγελο Σακκόπουλο, για την πολύτιμη και συνεχή υποστήριξη και καθοδήγησή του καθ' όλη τη διάρκεια της διπλωματικής μου εργασίας. Ακόμη, ευχαριστώ πολύ όλους τους καθηγητές του τμήματος για τις αξιοσημείωτες γνώσεις που μου προσφέραν κατά την διάρκεια της φοίτησης μου.

Περίληψη

Η παρούσα μεταπτυχιακή εργασία αποσκοπεί στην ανάπτυξη μιας εφαρμογής για κινητά τηλεφωνά, η οποία θα χρησιμοποιεί τεχνολογίες και εξαρτήματα της συσκευής όπως είναι το GPS και η επαυξημένη πραγματικότητα για την καλύτερη εξατομικευμένη εμπειρία χρήστη στο πεδίο της ηλεκτρονικής περιήγησης. Η ανάγκη για καλύτερη προβολή διάδρομων με βάση τις δυνατότητες και ανάγκες του κάθε χρήστη αλλά και επίσης στην εμφάνιση πληροφοριών για ένα σημείο ενδιαφέροντος με ένα διαφορετικό τρόπο με οδήγησε στην ανάπτυξη αυτής της εφαρμογής. Το GPS της κινητής συσκευής λοιπόν είναι ένας από τους καλυτέρους τρόπους για την προβολή διάδρομων που ταιριάζουν στον χρήστη. Επίσης ένας διαφορετικός τρόπος από τους συνηθισμένους που επιλέχθηκε για την προβολή πληροφοριών στα σημεία ενδιαφέροντος είναι η τεχνολογία επαυξημένης πραγματικότητας οπού ο χρήστης θα έχει την δυνατότητα να δει μέσω της κάμερας της κινητής συσκευής αντικείμενα οπού θα μπορούν να περιγράψουν και να δώσουν περισσότερες πληροφορίες για τον χώρο.

Η υλοποίηση της εφαρμογής έγινε με χρήση cross compiled applications (Εφαρμογές μεταγλωτισμένες ανά λειτουργικό σύστημα) χρησιμοποιώντας το εργαλείο ανάπτυξής κινητών εφαρμογών React Native. Ενώ ακόμη για την υποστήριξη της κινητής εφαρμογής αναπτύχθηκε το αντίστοιχο RESTful API οπού διανέμει τα καταλληλά δεδομένα στην κινητή εφαρμογή.

Abstract

This master thesis aims to develop a mobile application, which will use technologies and modules of the device, such as GPS and augmented reality, to give the user better and more personalized experience in the field of electronic browsing. The need of having better preview of the routes based on user's possibilities and needs, and the appearance of information about a point of interest in a different way led me to the development of this application. The GPS module of the mobile device is one of the best ways to show personalized routes that are suitable for the user. Also, a different way from the usual chosen for showing information about a point of interest is the augmented reality technology in which the user will be able to see 3D objects through the camera of the mobile device giving with this way more information about one place.

The implementation of the application is done using the React Native tool which is for cross compiled application development. Also, a RESTful API was built to support the mobile application by distributing the proper data.

Πίνακας περιεχομένων

Πίνακας εικόνων

Κεφάλαιο 1 – Εισαγωγή

1.1 Γενικά

Η χρήση φορητών συσκευών έχει γίνει αναπόσπαστο κομμάτι της καθημερινότητας μας. Όλο και περισσότερος κόσμος χρησιμοποιεί κινητές συσκευές στην καθημερινότητα αφού μπορεί με μεγάλη ευκολία να το πραγματοποιήσει οτιδήποτε χρειάζεται μέσα από αυτές χρησιμοποιώντας τις εφαρμογές που είναι διαθέσιμες στα application stores του κινητού τηλεφώνου και μπορεί να τις χρησιμοποιήσει με μεγάλη ευκολία και ακόμα τις περισσότερες φορές δωρεάν [1].

Μια κατηγορία εφαρμογών που υπάρχει σε αυτά τα καταστήματα είναι σχετικά με την ηλεκτρονική πλοήγηση οπού κανείς μπορεί να βρει εφαρμογές τουριστικούς ενδιαφέροντος, εφαρμογές που βοηθούν τον χρήστη να εξερευνήσει καλυτέρα την πόλη του και άλλα πολλά. Έτσι ο χρήστης μπορεί να βρει όποια πληροφορία χρειάζεται για την πλοήγηση του άμεσα, γρήγορα και εύκολα με τη χρήση της κατάλληλης εφαρμογής.

Η ανάπτυξη μιας τέτοιας εφαρμογής και για τις δυο επικρατούσες πλατφόρμες δεν είναι μια εύκολη διαδικασία αφού το κάθε λειτουργικό απαιτεί διαφορετικές γλώσσες προγραμματισμού και διαφορετικά εργαλεία προγραμματισμού. Το Android χρησιμοποιεί τις γλώσσες προγραμματισμού Java και Kotlin, το προτεινόμενο Android Studio και διεπαφές χρήστη στηριζόμενες στο Material Design[2] ενώ από την άλλη το ενώ το iOS τις γλώσσες προγραμματισμού Objective C και Swift, IDE το Xcode και Studio και διεπαφές χρήστη βασισμένο στο Human Interface Guidelines[3][4]. Αυτή η διαφορετικότητα έχει ως αποτέλεσμα αυξημένο κόστος για την ανάπτυξη των εφαρμογών και στις δυο πλατφόρμες αφού χρειάζεται προγραμματιστές και για τις δυο πλατφόρμες.

Το αυξημένο αυτό κόστος οδήγησε στην δημιουργία εργαλείων και framework με την ικανότητα να επιτρέπουν στον χρήστη να επαναχρησιμοποίηση κώδικα στις υλοποιήσεις. 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. Κάποια από τα πιο γνωστά framework είναι το React Native, το Flutter και το Ionic[5].

1.2 Συνεισφορά της εργασίας

Εφαρμογές για την ηλεκτρονική πλοήγηση μπορούν να βρεθούν πολλές στα καταστήματα εφαρμογών των κινητών συσκευών. Από τα σημαντικότερα στοιχεία που χαρακτηρίζουν μια τέτοια εφαρμογή είναι οι δυνατότητες που δίνει η εφαρμογή στους χρήστες, η πληρότητα του περιεχομένου και η ευχρηστία σαν εφαρμογή.

Η εφαρμογή που αναπτύχθηκε στα πρότυπα της διπλωματικής εργασίας δεν έχει σαν κύριο στόχο την πρωτοτυπία και ποιότητα σε αυτά τα στοιχεία. Κύρια ιδιαιτερότητα της είναι η προσπάθεια υλοποίησης της σαν γεννήτρια παρομοίων εφαρμογών, δίνοντας επιπλέον όμως δυνατότητες στον χρήστη όπως είναι τα εξατομικευμένα δρομολόγια αλλά και επίσης χρήση τεχνολογιών όπως η επαυξημένη πραγματικότητα που μπορούν να καλυτερεύσουν την εμπειρία χρήσης.

Στην εφαρμογή επίσης δίνεται έμφαση στο εύχρηστο και εύκολο γραφικό περιβάλλον ώστε να μπορεί να χρησιμοποιηθεί με ευκολία από όλες τις ηλικιακές ομάδες.

1.3 Διάρθρωση εργασίας

Μετά την εισαγωγή και την αναφορά στα χαρακτηριστικά και τις ιδιαιτεροτήτες της εφαρμογής, στο δεύτερο κεφάλαιο αναλύεται το τεχνολογικό υπόβαθρο που αφορά τις διαφορετικές τεχνικές δημιουργίας εφαρμογών για φορητές συσκευές, ενώ επίσης περιγράφονται και οι διαδικτυακές εφαρμογές.

Στο τρίτο κεφάλαιο περιγράφονται οι τεχνολογίες και οι βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής αλλά και τα χαρακτηριστικά τους.

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.

Στο πέμπτο κεφάλαιο με παράθεση στιγμιοτύπων της εφαρμογής περιγράφεται η λειτουργία της εφαρμογής από την πλευρά του χρήστη.

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

Κεφάλαιο 2 – Θεωρητικό τεχνολογικό υπόβαθρο

2.1 Πλατφόρμες φορητών συσκευών και εφαρμογές

Στις μέρες μας δυο είναι τα πιο διαδεδομένα λειτουργικά συστήματα για κινητές συσκευές. Αυτά είναι το iOS που έχει αναπτυχθεί από την Apple το 2007 και χρησιμοποιείται στις φορητές συσκευές τις (iPhone, iPad) και το Android το οποίο είναι λογισμικό ανοιχτού κώδικα και μπορεί να χρησιμοποιηθεί και να γίνει παραμετροποίηση από διάφορους κατασκευαστές κινητών συσκευών. Το τελευταίο έχει αναπτυχθεί από την Google το στα τέλη του 2007 και επίσης είναι και το λειτουργικό σύστημα για κινητές συσκευές που κατέχει το μεγαλύτερο μερίδιο αγοράς στον κόσμο [6][7][8].

Ένα από τα μεγαλύτερα πλεονεκτήματα αυτών των λειτουργικών είναι ότι επιτρέπουν στον χρήστη την εγκατάσταση εφαρμογών που έχουν την δυνατότητα να χρησιμοποιήσουν εξαρτήματά της συσκευής (camera, gps, μικρόφωνο κ.α.). Αυτό δίνει την δυνατότητα στους κατασκευαστές των εφαρμογών να εκμεταλλευτούν πλήρως αυτές τις δυνατότητες και να αναπτύξουν εφαρμογές με ένα μεγάλο εύρος δυνατοτήτων. Μερικές από τις κατηγορίες εφαρμογών που υπάρχουν είναι οι εφαρμογές πλοήγησης, κοινωνικών δικτυών, διαχείρισης κρατήσεων, υγείας, e-commerce και τα παιχνίδια[9].

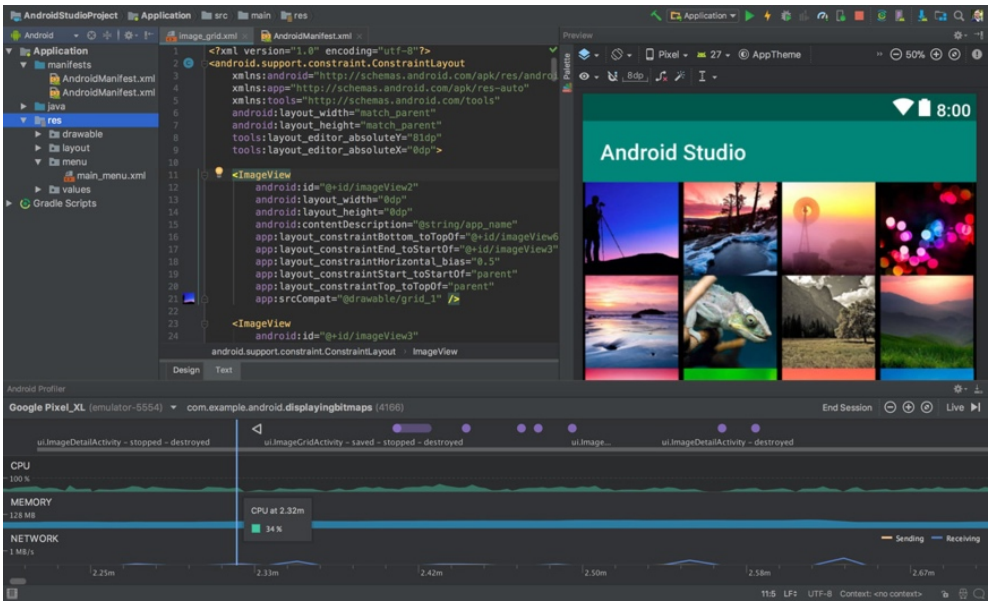
Στα λειτουργικά συστήματα υπάρχουν ηλεκτρονικά καταστήματα από τα οποία ο χρήστης μπορεί να περιηγηθεί στις διαθέσιμες, εφαρμογές και να επιλέξει να εγκαταστήσει όποια εφαρμογή επιθυμεί με εύκολο και γρήγορο τρόπο. Οι περισσότερες εφαρμογές έχουν ως αντίτιμο ένα μικρό ποσό ή πολλές από αυτές διατίθενται δωρεάν, βέβαια υπάρχουν και εφαρμογές που κοστίζουν πολύ περισσότερα χρήματα αλλά αυτό σημαίνει οτί η χρηστικότητα της είναι πολύ μεγαλύτερης σημασίας[9].

2.1.1 Android

ΕΙΚΟΝΑ 2.1: Λογότυπο λειτουργικού android

Το Android 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]. Οι εφαρμογές είναι διαθέσιμες στους χρήστες μέσω των καταστημάτων εφαρμογών (Play store, App Gallery) [11,12].

Για μεγαλύτερη διευκόλυνση των προγραμματιστών, το android παρέχει δική του εργαλειοθήκη ανάπτυξης λογισμικού (SDK – Software Development Kit) χρησιμοποιώντας την Java ή την Kotlin σαν γλώσσα προγραμματισμού. Επίσης υπάρχει και διαθέσιμο περιβάλλον ανάπτυξης εφαρμογών το οποίο προτείνεται και ονομάζεται Android Studio [13]. Ο προγραμματιστής όμως μπορεί να χρησιμοποιήσει οποίο περιβάλλον ανάπτυξης επιθυμεί και δεν περιορίζεται σε αυτό. Τα λειτουργικά που μπορεί να χρησιμοποιήσει για την ανάπτυξη εφαρμογών είναι Windows, Mac OS και Linux.



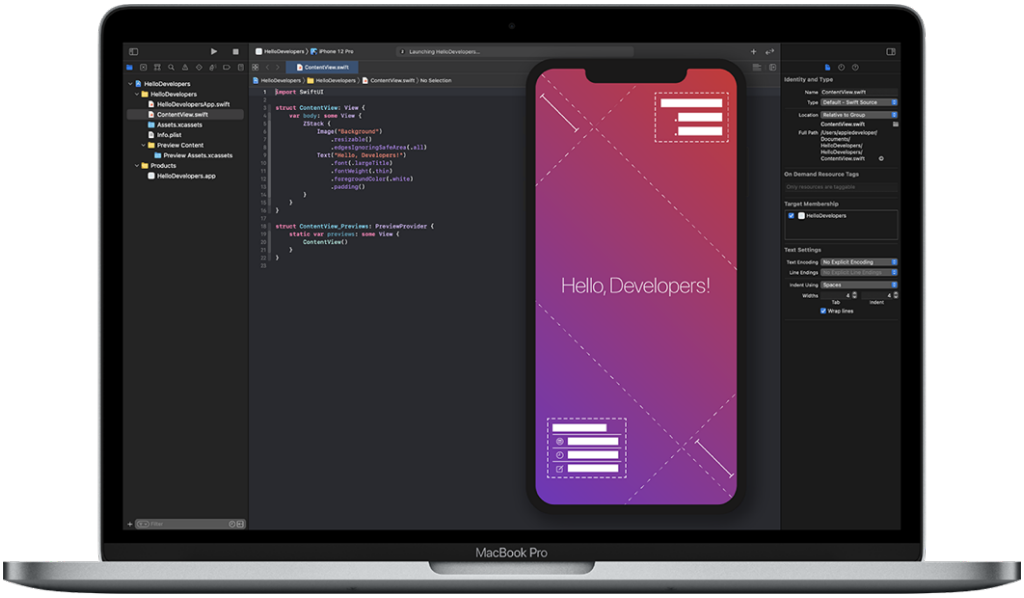
ΕΙΚΟΝΑ 2.2: Android Studio

2.1.2 iOS

ΕΙΚΟΝΑ 2.3: Λογότυπο iOS

Αντίστοιχα όπως και στο Android το iOS περιλαμβάνει το λειτουργικό σύστημα, το ενδιάμεσο λογισμικό και τις βασικές εφαρμογές. Για περισσότερες εφαρμογές ο χρήστης μπορεί να χρησιμοποιήσει το αντίστοιχο καταστήματος εφαρμογών της Apple το App Store [14].

Το iOS παρέχει την εργαλειοθήκη iOS SDK για την ανάπτυξη των εφαρμογών της αλλά και επίσης το περιβάλλον ανάπτυξης εφαρμογών XCode [15]. Οι γλώσσες προγραμματισμού που μπορούν να χρησιμοποιηθούν για την ανάπτυξη είναι η Objective C και η Swift [3]. Ο χρήστης πρέπει να διαθέτει υπολογιστή με λειτουργικό σύστημα Mac OS για την ανάπτυξη εφαρμογών σε iOS και επίσης είναι περιορισμένος στην χρήση μόνο του XCode σαν περιβάλλον ανάπτυξης εφαρμογών.



ΕΙΚΟΝΑ 2.4: XCode

2.2 Τύποι εφαρμογών για κινητές συσκευές

Αν και οι δυο πλατφόρμες προσφέρουν τα δικά τους εργαλεία ανάπτυξης στους προγραμματιστές, μπορούν να χρησιμοποιηθούν διαφορετικές τεχνικές και εργαλεία τρίτων για την δημιουργία εφαρμογών. Ανάλογα τον τρόπο υλοποίησης ταξινομούμε τις εφαρμογές σε διάφορες κατηγορίες.

2.2.1 Εγγενείς εφαρμογές (Native apps)

Μια εγγενής εφαρμογή (native application) είναι αυτή που έχει αναπτυχθεί για ένα συγκεκριμένο λειτουργικό σύστημα χρησιμοποιώντας τα εργαλεία και τις γλώσσες προγραμματισμού αυτού.

Με αυτόν τον τρόπο ανάπτυξης της εφαρμογής δίνεται στο προγραμματιστή η δυνατότητα να χρησιμοποιήσει πλήρως το υλικό της συσκευής (camera, GPS, μικρόφωνο κ.α.). Η πρόσβαση στους πόρους γίνεται μέσω των API που προσφέρει το κάθε SDK της κάθε πλατφόρμας. Επίσης προσφέρονται στον προγραμματιστή έτοιμα γραφικά στοιχεία τα οποία είναι κοντά ή ιδιά με αυτά που χρησιμοποιεί το λειτουργικό σύστημα κάνοντας έτσι υπάρχει μεγαλύτερη οικειότητα στους χρήστες που έχουν συνηθίσει τα ιδιά γραφικά στοιχεία στο λειτουργικό σύστημα που χρησιμοποιούν[15, 16].

Πλεονεκτήματα:

- Συνολικά, έχουν την καλύτερη επίδοση

- Διεπαφή χρήστη για την συγκεκριμένη πλατφόρμα
- Υποστήριξη όλων των λειτουργιών και υλικών της κινητής συσκευής.
- Μεγάλη αξιοπιστία και ασφάλεια

Μειονεκτήματα:

- Κόστος ανάπτυξης σε περίπτωση οπού η εφαρμογή πρέπει να κατασκευαστεί και για τις δυο πλατφόρμες
- Προγραμματιστές με εμπειρία σε διαφορετικές πλατφόρμες και γλώσσες προγραμματισμού

2.2.2 Διαδικτυακές εφαρμογές (Web Apps)

Μια διαδικτυακή εφαρμογή (web app) έχει κατασκευαστεί για να είναι φιλική σε μια οθόνη κινητής συσκευής. Το άνοιγμα αυτής της εφαρμογής γίνεται μέσω του φυλλομετρητη ιστού (web browsers) του λειτουργικού συστήματος. Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι ιδιά με αυτή που χρησιμοποιείται για την ανάπτυξη μια ιστοσελίδας (HTML, JavaScript, CSS). Τα τελευταία χρονιά οι φυλλομετρητές προσφέρουν όλο και περισσότερα API για την χρήση διάφορων πόρων και υλικών της συσκευής (κάμερα, μικρόφωνο, gps κ.α.). Ο χρήστης δεν χρειάζεται να κατεβάσει κάποια εφαρμογή από κάποιο κατάστημα ή να την έχει στην συσκευή του εγκατεστημένη[17].

Πλεονεκτήματα:

- Χαμηλό κόστος ανάπτυξης
- Μια εφαρμογή και για τις δυο πλατφόρμες
- Ευκολότερη συντήρηση της εφαρμογής
- Μόνο μια ομάδα για την ανάπτυξη της εφαρμογής
- Ικανότητα να χρησιμοποιήσει το υλικό της συσκευής
- Δεν χρειάζεται να εγκατασταθεί στο κινητό τηλέφωνο

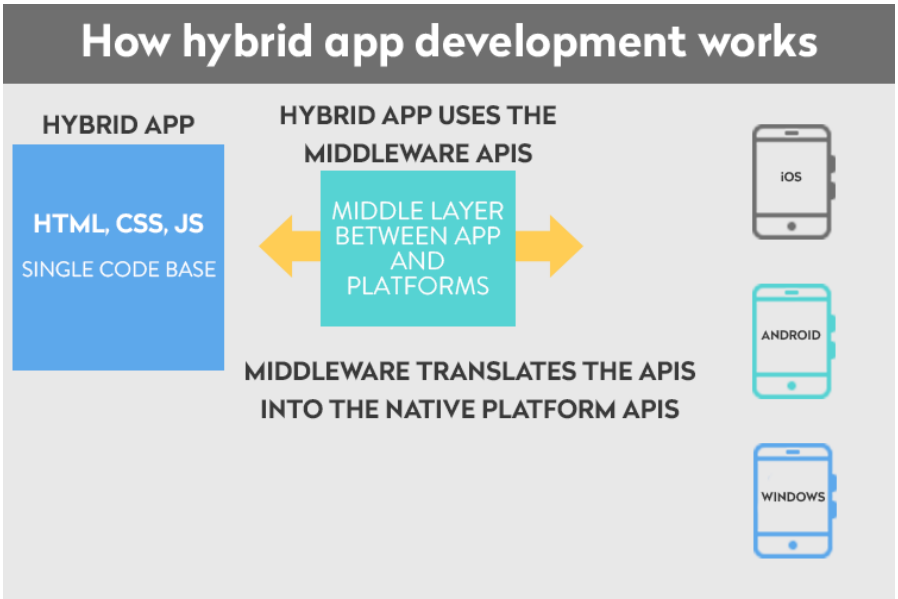
Μειονεκτήματα:

- Δεν έχουν καλή επίδοση
- Η διεσπάρη χρήστη δεν είναι κοντά σε αυτή του λειτουργικού συστήματος
- Δεν υπάρχει πλήρης χρήση όλων των υλικών και πόρων της κινητής συσκευής

2.2.3 Υβριδικές εφαρμογές (Hybrid Apps)

Η υβριδική εφαρμογή είναι ο συνδυασμός της εγγενείς και διαδικτυακής εφαρμογής. Είναι μια εγγενείς εφαρμογή που έχει σαν κύριο στοιχείο μια προβολή ιστού (WebView) οπού μέσα σε αυτό γίνεται προβολή της διαδικτυακής εφαρμογής η οποία είναι επίσης τις περισσότερες φορές αποθηκευμένη τοπικά στην συσκευή. Το

σημαντικό κομμάτι αυτή της τεχνικής είναι ότι η εφαρμογή μπορεί να εγκατασταθεί μέσω των καταστημάτων των λειτουργικών συστημάτων και ο τρόπος ανάπτυξης παραμένει ακριβώς ίδιος με αυτόν των διαδικτυακών εφαρμογών. Επίσης με αυτόν τον τρόπο δίνεται μεγαλύτερη πρόσβαση στο υλικό της συσκευής. Τα πλεονεκτήματα είναι παρόμοια με αυτά των διαδικτυακών εφαρμογών. Κάποια από τα πιο γνωστά εργαλεία ανάπτυξης των υβριδικών εφαρμογών είναι το Ionic και NativeScript [18].



ΕΙΚΟΝΑ 2.5: Hybrid app development

Πλεονεκτήματα:

- Χαμηλό κόστος ανάπτυξης
- Μια εφαρμογή και για τις δυο πλατφόρμες
- Ευκολότερη συντήρηση της εφαρμογής
- Μόνο μια ομάδα για την ανάπτυξη της εφαρμογής
- Ικανότητα να χρησιμοποιήσει μεγαλύτερο εύρος του υλικό της συσκευής
- Γρηγορότερη πρόσβαση αφού μπορεί να εγκατασταθεί στην κινητή συσκευή
- Δεν χρειάζεται διαδίκτυο για να λειτουργήσει

Μειονεκτήματα:

- Δεν έχουν καλή επίδοση αν και είναι λίγο καλύτερη από αυτήν της διαδικτυακής εφαρμογής γιατί βρίσκεται αποθηκευμένη στην συσκευή
- Η διεσπάρη χρήστη δεν είναι κοντά σε αυτή του λειτουργικού συστήματος
- Δεν υπάρχει πλήρης χρήση όλων των υλικών και πόρων της κινητής συσκευής

2.2.4 Εφαρμογές μεταγλωτισμένες ανά λειτουργικό

σύστημα(cross compiled applications)

Η εφαρμογές μεταγλωττισμένες ανά λειτουργικό σύστημα είναι η προσπάθεια στην ανάπτυξη εφαρμογών με κοινό κώδικα. Η διαδικασία είναι ότι χρησιμοποιείται μια γλώσσα προγραμματισμού και στην συνέχεια μπορεί να μεταγλωττιστεί σε εγγενή κώδικα της κάθε πλατφόρμας. Αυτές οι εφαρμογές είναι πιο κοντά με τις εγγενείς εφαρμογές αφού μοιράζουν κοινά στοιχεία και χρησιμοποιείται στην τελική το SDK της κάθε πλατφόρμας. Τα πιο γνωστά εργαλεία ανάπτυξης είναι η React Native, Flutter και Xamarin[19, 20].

Πλεονεκτήματα:

- Χαμηλότερο κόστος ανάπτυξης σε σχέση με τις εγγενείς εφαρμογές
- Ένας κώδικας και για τις δύο πλατφόρμες
- Ευκολότερη συντήρηση της εφαρμογής
- Μόνο μια ομάδα για την ανάπτυξη της εφαρμογής
- Ικανότητα να χρησιμοποιήσει σχεδόν όλα τους διαθέσιμους πόρους και υλικό της συσκευής
- Παρόμοια διεπαφη χρήστη με αυτήν των native apps
- Πολύ καλύτερη επίδοση συγκριτικά με αυτήν των web και hybrid apps

Μειονεκτήματα:

- Για την χρήση κάποιου υλικού της συσκευής θα πρέπει να υπάρχει διαθέσιμο σε αυτά τα εργαλεία και κάποια βιβλιοθήκη που να επιτρέπει την χρήση του υλικού με κάποιο API

2.2.5 Επιλογή και σύγκριση εγγενών και άλλων τεχνικών ανάπτυξης εφαρμογών

Είναι σημαντικό ο προγραμματιστής ανάλογα με τις ανάγκες του να επιλέξει τον πιο σωστό τρόπο ανάπτυξης της εφαρμογής. Πολλές φορές δεν είναι μια εύκολη διαδικασία και άλλες φορές μπορεί στο μέλλον να χρειαστεί να αλλάξει και να επιλέξει διαφορετικό τρόπο ανάπτυξης της εφαρμογής. Σιγουρά τις περισσότερες φορές με την ανάπτυξη εγγενών εφαρμογών δεν θα υπάρχει ποτέ κάποιος περιορισμός στην ανάπτυξη. Αλλά υπάρχουν και περιπτώσεις που ο προγραμματιστής μπορεί να επιλέξει άλλες τεχνικές ανάπτυξης εφαρμογής ανάλογα με τα κριτήρια που έχει επιλέξει. [σε περίπτωση που δεν χρειάζεται να χρησιμοποιήσει σε μεγάλο βαθμό η καθόλου το υλικό της συσκευής [15].

Με τις εγγενής εφαρμογές η άμεση πρόσβαση στα APIs των λειτουργικών συστημάτων έχουν σαν αποτέλεσμα βέλτιστη απόκριση και ταχύτητα, καθώς δεν διαμεσολαβεί κάποιο επιπλέον αφαιρετικό επίπεδο για την υλοποίηση και επίσης μπορούν να εκμεταλλευτούν πλήρως το υλικό της συσκευής. Ακόμα τα έτοιμα

γραφικά στοιχεία και σχεδιαστικές οδηγίες που περιέχονται στην κάθε πλατφόρμα προσφέρουν πολύ καλή εμπειρία χρήσης στον τελικό χρήστη[4, 21].

Σε αντίθεση με τις εγγενείς εφαρμογές οι υπόλοιπες τεχνικές ανάπτυξης προσφέρουν γρήγορη ανάπτυξη κώδικα, λιγότερο κόστος και ένα άλλο σημαντικό στοιχεία ότι δεν χρειάζεται να φτιάχνουν την ιδιά εφαρμογή σε δυο πλατφόρμες. επίσης είναι και πιο εύκολες στην συντήρηση διότι μια αλλαγή γίνεται σε ένα κοινό κώδικα. Τα τελευταία χρονιά όλο και περισσότερο αυτές οι τεχνικές έχουν μεγάλη ανάπτυξη, βελτιώνονται αλλά και χρησιμοποιούνται περισσότερο από πολλές εταιρείες.

Ολες οι τεχνικές έχουν τα πλεονεκτήματα και τα μειονεκτήματα τους. Στην παρούσα διατριβή ο τρόπος ανάπτυξης που επιλέχθηκε είναι αυτός των εφαρμογων μεταγλωτισμένες ανά λειτουργικό σύστημα (cross compiled applications) μέσω του εργαλείου React Native. Έγινε προσπάθεια να γίνει με αυτόν τον τρόπο λόγο των ήδη υπάρχων γνώσεων σε Front-end τεχνολογιών αλλά και επίσης της γνώσης την βιβλιοθήκης της React που μοιράζουν αρκετά κοινά στοιχεία στην ανάπτυξη.

2.3 RESTful web service

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 [22].

Ένα RESTful σύστημα πρέπει να στηρίζεται σε τέσσερις βασικές αρχές [22]:

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

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

3) 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.

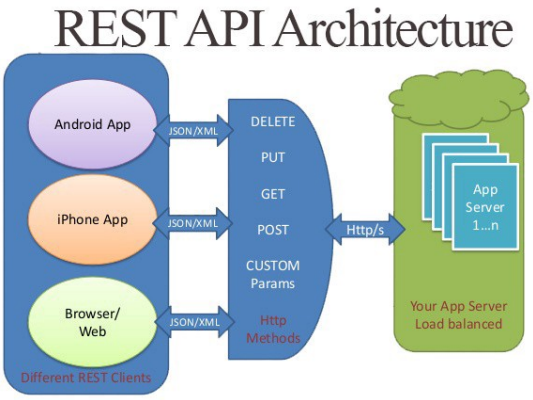
4) Stateless - Ανεξάρτητο κατάστασης: Τα αιτήματα είναι ανεξάρτητα το ένα από το άλλο. Ο πελάτης στέλνει τα αίτημα με όλη την απαραίτητη πληροφορία που απαιτείται ώστε ο διακομιστής να μη χρειάζεται γνώση προηγούμενης κατάστασης η ενδιάμεσης πληροφορίας για την διεκπεραίωση τους.

2.4 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.

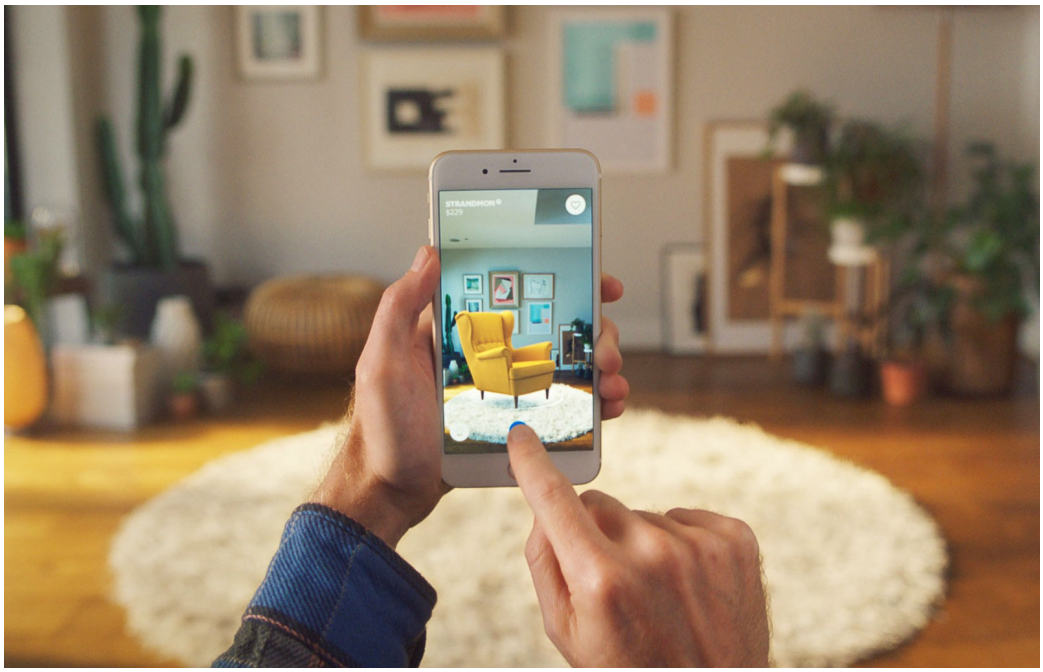


ΕΙΚΟΝΑ 2.6: Rest API Architecture

Κεφάλαιο 3 – Επαυξημένη πραγματικότητα (Augmented Reality)

3.1 Τι είναι η επαυξημένη πραγματικότητα

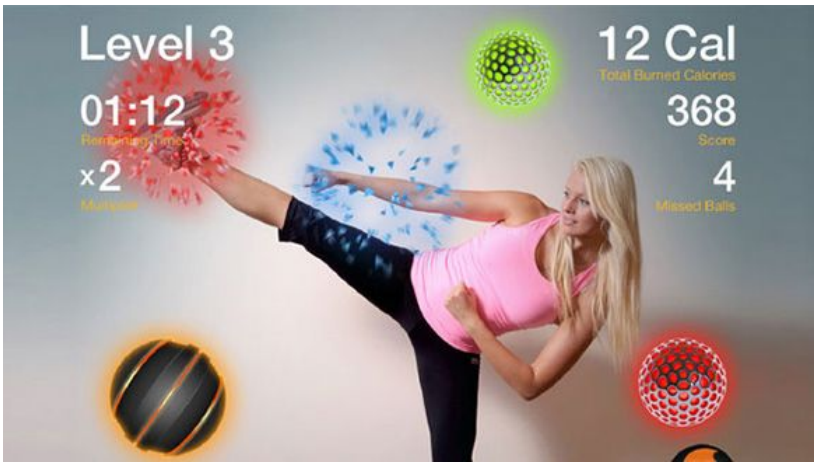
Η επαυξημένη πραγματικότητα (AR) είναι ένα πεδίο στον χώρο της εικονικής (ή ιδεατής) πραγματικότητας (virtual reality). Ο όρος αναφέρεται στην προσθήκη εικονικής πληροφορίας μέσω κατάλληλων συσκευών, στο περιβάλλον (χώρο, αντικείμενα, ανθρώπους κλπ.) το οποίο αντιλαμβάνεται ο άνθρωπος μέσω των αισθητηρίων οργάνων του. Αυτό σημαίνει ότι το πραγματικό περιβάλλον δεν υποκαθίσταται, αλλά αντίθετα ενισχύεται, «επαυξάνεται» από πληροφορίες που παράγονται από υπολογιστές και οι οποίες συνδυάζονται με την εικόνα του κόσμου που δεχόμαστε από τις αισθήσεις μας (όραση, ακοή, αφή κλπ.). Στόχος ενός συστήματος επαυξημένης πραγματικότητας είναι να μην μπορεί ο χρήστης να ξεχωρίσει τον πραγματικό κόσμο από την εικονικά εμπλουτισμένη εικόνα του. [37, 38]



ΕΙΚΟΝΑ 3.1: Επαυξημένη πραγματικότητα μέσω κινητής συσκευής

3.2 Πεδία χρήσης επαυξημένης πραγματικότητας

Τα πεδία και η εφαρμογές που μπορεί να χρησιμοποιηθεί αυτή η τεχνολογία είναι αρκετές. Κάποια από τα πεδία που συναντάμε την επαυξημένη πραγματικότητα είναι στα παιχνίδια, αρχαιολογία, αρχιτεκτονική, εκπαίδευση, εμπόριο, περιήγηση, κοινωνικά δίκτυα, υγεία, γυμναστική αλλά και σε πολλά αλλά πεδία. [38]



ΕΙΚΟΝΑ 3.2: Επαυξημένη πραγματικότητα στην γυμναστική

3.3 Τρόποι απόδοσης επαυξημένης πραγματικότητας

Οι τρόποι απόδοσης επαυξημένης πραγματικότητας μπορούν να διαχωριστούν σε 3 ομάδες με βάσει τον τρόπο που αλληλοεπιδρούν με το φυσικό περιβάλλον και εγγράφουν τα εικονικά αντικείμενα σε αυτό. Είναι απαραίτητο η συσκευή να διαθέτει κάμερα για να είναι εφικτή η επαυξημένη πραγματικότητα. Ανάλογα με την ομάδα που ανήκει μπορεί να χρειάζεται περισσότερους αισθητήρες για την σωστή λειτουργία. [37, 38]

3.3.1 Χρήση φυσικού δείκτη

Στην περίπτωση αυτή τα απαραίτητα εργαλεία που χρειάζονται είναι μια κάμερα και ένας φυσικός εκτυπωμένος οπτικός δείκτης (marker). Αυτό το είδος απαιτεί την υπάρξει δείκτη οπού μπορεί να αναγνωριστεί ευκολά από την κάμερα της συσκευής. Οι δείκτες μπορεί να είναι φυσικά αντικείμενα ή σχέδια σε χαρτί που υπάρχουν στον φυσικό κόσμο. Με την αναγνώριση του φυσικού δείκτη από την κάμερα της συσκευής είναι δυνατό να τοποθετηθεί το εικονικό αντικείμενο στο σημείο εκείνο. [39]

ΕΙΚΟΝΑ 3.3: AR φυσικού δείκτη

3.3.2 Χωρίς χρήση φυσικού δείκτη

Σε αυτήν την περίπτωση δεν χρειάζεται κάποιος φυσικός δείκτης και χρησιμοποιείται κάποιο πραγματικό αντικείμενο, είτε είναι πρόσωπο είτε κάποιο άλλο φυσικό αντικείμενο, στον χώρο. Σε πολλές εφαρμογές συναντάμε την αναγνώρισή του πατώματος ως αντικείμενο και η τοποθέτηση του ψηφιακού εικονικού αντικειμένου πάνω σε αυτό. [39]

ΕΙΚΟΝΑ 3.4: AR χωρίς φυσικό δείκτη

3.3.3 Χρήση γεωγραφικής τοποθεσίας

Μια άλλη περίπτωση είναι η τοποθέτηση των εικονικών αντικειμένων σε συγκεκριμένα μέρη και αυτό μέσω της χρήσης του GPS της συσκευής. Έτσι εικονικά

αντικείμενα εμφανίζονται όταν ο χρήστης έρθει κοντά στις συντεταγμένες που έχουν τοποθετηθεί αυτά. [39]

ΕΙΚΟΝΑ 3.5: AR με χρήση τοποθεσίας

Κεφάλαιο 4 – Τεχνολογίες υλοποίησης εφαρμογής

4.1 Τεχνολογία που επιλέχθηκε για την κινητή εφαρμογή

Όπως προαναφέρθηκε στην παρούσα διπλωματική εργασία για την ανάπτυξη της κινητής εφαρμογής χρησιμοποιήθηκε η τεχνική μεταγλωτισμού ανά λειτουργικό σύστημα (cross compiled application). Πιο συγκεκριμένα το εργαλείο που χρησιμοποιήθηκε είναι η React native.

ΕΙΚΟΝΑ 4.1: React native λογότυπο

Το React Native είναι ένα πλαίσιο λογισμικού UI ανοιχτού κώδικα που δημιουργήθηκε από τη Meta Platforms, Inc. Είναι παρόμοια με την βιβλιοθήκη React με την διαφορά ότι δεν χειρίζεται το DOM αλλά εκτελείται απευθείας από την συσκευή και επικοινωνεί μέσω μιας σειριακής και ασύγχρονης γέφυρας με την εγγενή πλατφόρμα.

Για την καλύτερη κατανόηση μιας εφαρμογής σε React Native πρέπει να γίνουν αντιληπτά τα τέσσερα νήματα(threads) με τα οποία υλοποιείται η εφαρμογή αυτή[23]. Δηλαδή:

(a) UI Thread: Επίσης γνωστό ως κύριο thread έχει χρήση για την απόδοση του εγγενούς χρήστη Android και iOS. Για παράδειγμα, στο android αυτό το thread χρησιμοποιείται για μέτρηση / διάταξη / κλήρωση.

(b) Native Modules Thread: Πρόκειται για το thread με το οποίο η εφαρμογή παίρνει πρόσβαση σε μια API πλατφόρμα.

(c) Render Thread: Προκείται για το thread που χρησιμοποιείται για την σχεδίαση περιβάλλοντος εργασίας του χρήστη μέσω της παραγωγής πραγματικών εντολών OpenGL,μόνο για την περίπτωση λογισμικού Android 5.0.

(d) JS Thread: Πρόκειται για το thread που τρέχει την λογική της εφαρμογής. Με άλλα λόγια, στο τέλος κάθε βρόχου συμβάντων αποστέλλονται οι ενημερώσεις στο thread JS.
Κατά τη λειτουργία μιας React Native εφαρμογής περνάμε από διάφορες διαδικασίες όπως [23]:

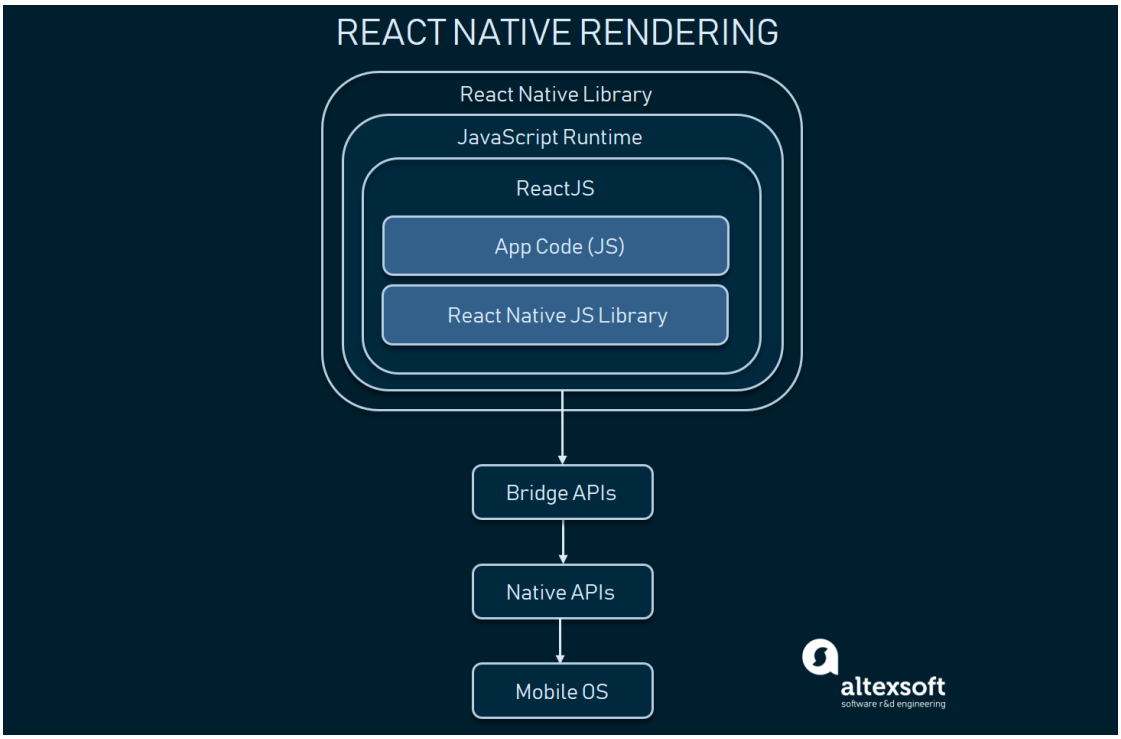
1. Αρχικά υφίσταται η εκκίνηση της εφαρμογής κατά την οποία το κύριο thread(UI Thread) φορτώνει και εκτελεί τις δέσμες JS (bundles).

2. Αφού έχει φορτωθεί ο κώδικας javascript και γίνει μεταφορά διεργασιών στο JS thread εκτελούνται όλες οι διεργασίες για την καλύτερη εμπειρία του χρήστη.

3. Ύστερα, ξεκινάει να κάνει render η React και αρχίζει η διαδικασία diffing από τον συντονιστή (Reconciler Api) οδηγώντας στην δημιουργία ενός καινούργιου εικονικού DOM σε άλλο νήμα (Thread Shadow).

4. Σε αυτό το στάδιο το Thread Shadow στέλνει τις παραμέτρους που έχει αντίστοιχα υπολογίσει η διάταξη (layout) της αντίστοιχης εφαρμογής.

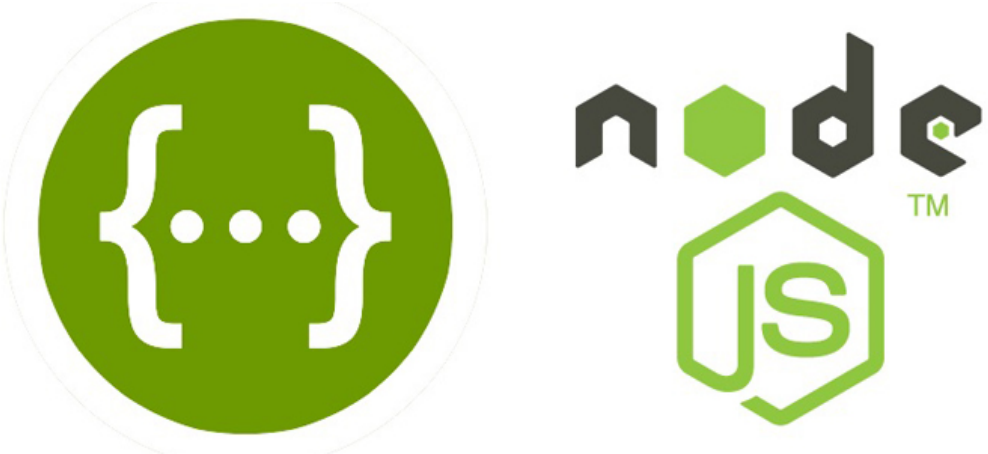
5. Αφού αποστείλει τις παραμέτρους το Thread Shadow, αποτελεί δουλειά του κύριου thread να τα εμφανίσει στην διεπαφή του χρήστη.



ΕΙΚΟΝΑ 4.2 React Native Rendering

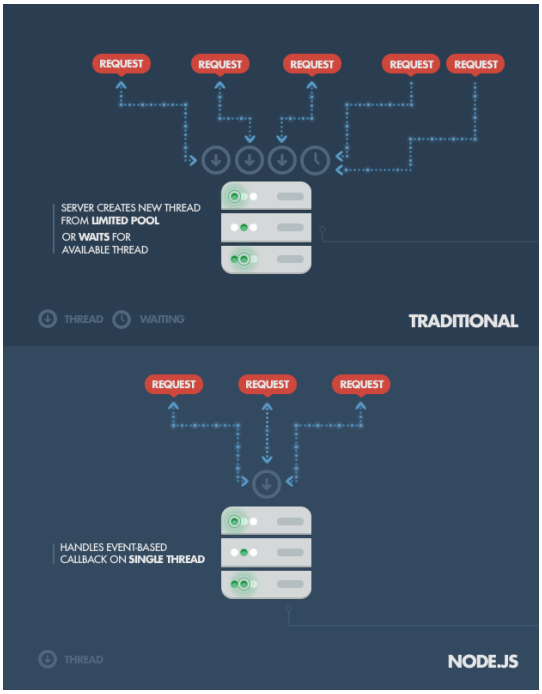
4.2 Τεχνολογία που επιλέχθηκε για το back end σύστημα

Το back end της εφαρμογής αναπτύχθηκε στην γλώσσα προγραμματισμού JavaScript και τρέχει σε Node.js.

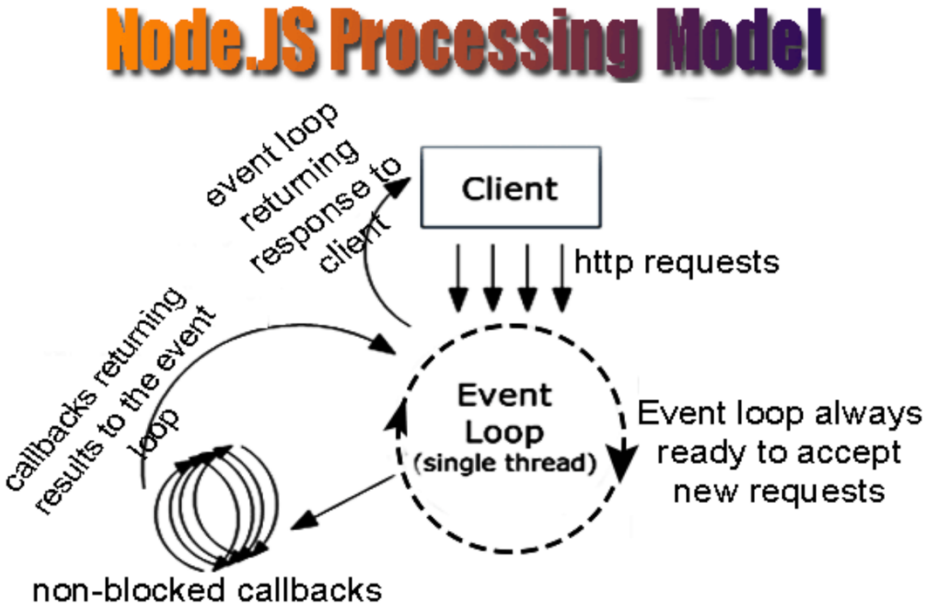


ΕΙΚΟΝΑ 4.3: Λογότυπο Node.js

Η Node.js είναι πολύ δημοφιλής τα τελευταία χρονιά και όλο και πιο πολλές εταιρίες το χρησιμοποιούν για να αναπτύξουν το back end σύστημα τους. Η Node είναι λοιπόν ένας server όπως άλλοι γνωστοί Apache, IIS, TOM ο οποίος δεν τρέχει PHP, .NET ή JAVA αλλά JavaScript και δεν στηρίζεται στην πολυνηματικότητα αλλά υιοθετεί μια διαφορετική προσέγγιση για την υλοποίηση των εισερχόμενων και εξερχόμενων νημάτων από τον διακομιστή. Εκτελεί έναν βρόχο συμβάντων με μια συγκεκριμένη διαδικασία που έχει καταχωρηθεί με το σύστημα για την διαχείριση συνδέσεων και κάθε νέα σύνδεση προκαλεί την πυροδότηση μιας λειτουργίας επανάκλησης JavaScript. Με αυτόν τον τρόπο χειρίζεται τα αιτήματα I/O ενός συστήματος με πολύ αποδοτικό τρόπο αφού μπορεί να εξυπηρετήσει αλλά αιτήματα μέχρι να ολοκληρωθεί ένα άλλο χρησιμοποιώντας επίσης και λιγότερους πόρους συστήματος[25, 26].



ΕΙΚΟΝΑ 4.4: Παραδοσιακό back-end σύστημα σε σύγκριση με την Node.js



ΕΙΚΟΝΑ 4.5: Μοντέλο επεξεργασίας Node.js

Επιπλέον για την ανάπτυξη του Restful API χρησιμοποιήθηκε το framework Nest.js το οποίο κάνει πιο εύκολη την ανάπτυξη εφαρμογών σε microservice αρχιτεκτονική. Είναι ένα framework ελαφρύ, απλό και ανοιχτού κώδικα. Επίσης δίνει την δυνατότητα στον προγραμματιστή να γράψει σε TypeScript οπού είναι ένα αυστηρό συντακτικό υπερσύνολο JavaScript και προσθέτει προαιρετική στατική πληκτρολόγηση στη γλώσσα[27].

ΕΙΚΟΝΑ 4.6: Λογότυπο Nest.js

Μερικά από τα χαρακτηριστικά της Nest.js είναι :

- Είναι εύκολη στην χρήση και γρήγορη στην εκμάθηση
- Υποστηρίζει πλήρως TypeScript
- Κάνει ευκολότερη την ανάπτυξη αφού προσφέρει διεσπάρη γραμμής εντολών
- Εύκολη ενσωμάτωση άλλων τεχνολογιών και βάσεων δεδομένων
- Πολύ καλή τεκμηρίωση

Για την βάση δεδομένων επιλέχθηκε η σχεσιακή βάση δεδομένων MySQL. Η MySQL είναι μια μορφή ελευθέρου συστήματος που διαχειρίζεται βάσεις δεδομένων. Πρόκειται περί μιας πολυνηματικης καθώς και πολλαπλών χρήσεων γλώσσα υποστηρίζοντας τα πρόσφατα standards της SQL και κάποια από τα χαρακτηριστικά της που κάποιος την προτιμάει έναντι άλλων RDBMS (Relational Database Managements Systems) είναι ότι προσφέρει πολύ καλή επίδοση, επεκτασιμότητα, διαθεσιμότητα και εύκολη διαχείριση[28].

ΕΙΚΟΝΑ 4.7: Λογότυπο MySQL

4.3 Βιβλιοθήκες

Για την ανάπτυξη της εφαρμογής, εκτός από τις παραπάνω τεχνολογίες χρειάστηκαν επιπλέον και οι απαραίτητες βιβλιοθήκες που εφαρμόστηκαν με τις τεχνολογίες αυτές. Παρακάτω, θα αναφερθούμε σε κάποιες από τις σημαντικές βιβλιοθήκες των οποίων έγινε η χρήση εκτός από αυτών που έχουν αναφερθεί ήδη (React Native, Nest.js κ.λπ.).

4.3.1 Google Maps API

Είναι μια από τις σημαντικότερες βιβλιοθήκες που χρησιμοποιήθηκε στην κινητή εφαρμογή. Δίνει την δυνατότητα στον χρήστη να έχει πρόσβαση στους χάρτες της

Google αλλά και επίσης σε άλλα χρήσιμα δεδομένα όπως ο χρόνος που χρειάζεται για μια διαδρομή, προβολή της διαδρομής μεταξύ δυο σημείων κ.α[30].

4.3.2 GeoLib

Δίνει την δυνατότητα στον χρήστη να υπολογίσει την γεωγραφική απόσταση μεταξύ δυο σημείων[31].

4.3.3 Viro React

Το Viro react είναι μια βιβλιοθήκη για την React Native οπού προσφέρει στους προγραμματιστές την δυνατότητα να κατασκευάσουν εφαρμογές που περιέχουν επαυξημένη (AR) ή εικονική(VR) πραγματικότητα[32].

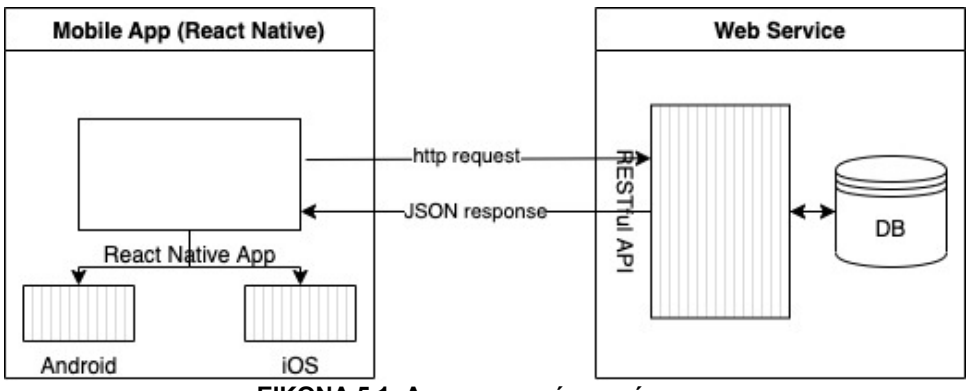
4.3.4 React Vector Icons και React Native Elements

Οι δυο βιβλιοθήκες React Vector Icons και React Native Elements προσφέρουν στον προγραμματιστή εικονίδια αλλά και γραφικά στοιχεία για την κατασκευή του γραφικού περιβάλλοντος της εφαρμογής[33, 34]

Κεφάλαιο 5 – Υλοποίηση εφαρμογής

5.1 Αρχιτεκτονική της εφαρμογής

Το σύστημα αποτελείται από δυο τμήματα. Το πρώτο τμήμα είναι η εφαρμογή οπού ο χρήστης θα χρησιμοποιεί για την πλοήγηση και είναι και αυτή που θα εγκαθίσταται στην κινητή συσκευή και το άλλο τμήμα είναι το web service το οποίο αποτελείται από ένα RESTful API και από την βάση δεδομένων και είναι αυτό που τροφοδοτεί με δεδομένα την κινητή εφαρμογή.



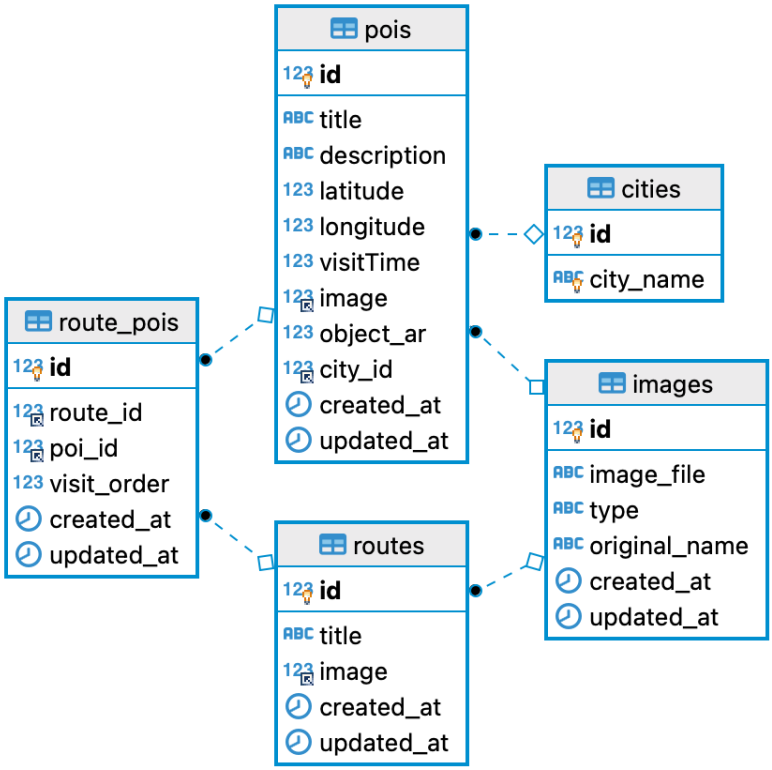
ΕΙΚΟΝΑ 5.1: Αρχιτεκτονική συστήματος

Στην παρούσα εργασία θα γίνει ανάπτυξη και παρουσίαση της εφαρμογής μόνο για την πλατφόρμα iOS ενώ ευκολά μπορεί επίσης να εκτελεστεί και σε Android πλατφόρμα χωρίς πολλές αλλαγές σε επίπεδο κώδικα.

5.2 Βάση δεδομένων συστήματος

Η βάση δεδομένων του συστήματος έχει σχεδιαστεί έτσι ώστε να είναι επεκτάσιμη και ευκολά κατανοητή. Αποτελείται από τους παρακάτω πίνακες:

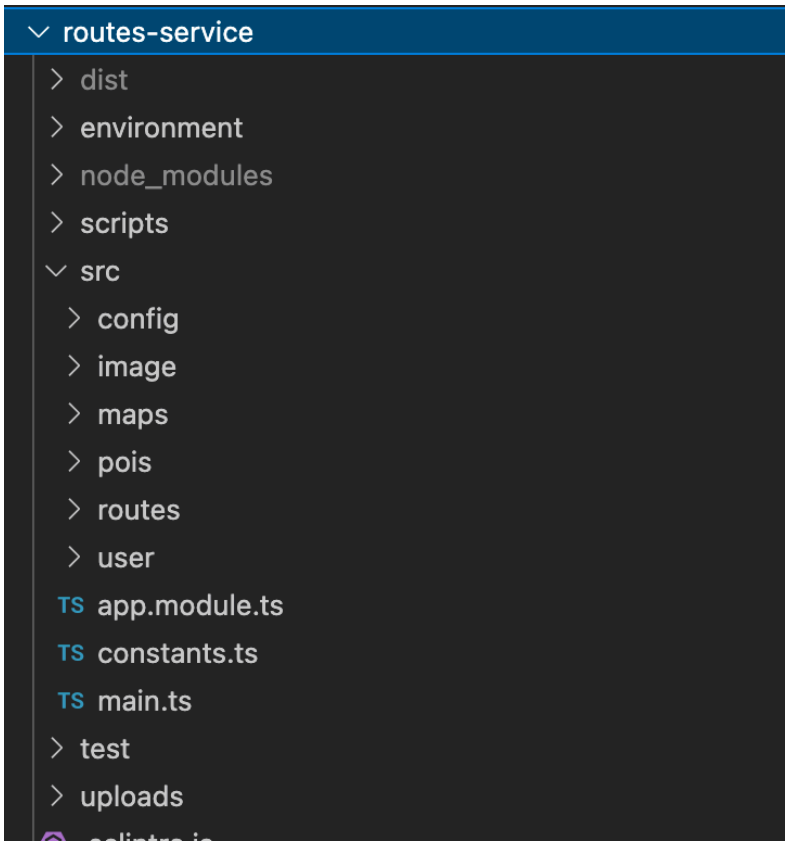
- Cities: περιέχει τα ονόματα πόλεων.
- Images: περιέχει τις εικόνες τις εφαρμογής αλλά ακόμη και τα 3D αντικείμενα που χρησιμοποιούνται από το AR της εφαρμογής.
- Pois: περιέχει πληροφορίες για τα σημεία ενδιαφέροντος (POIS) όπως τίτλο, περιγραφή, συντεταγμένες χάρτη κ.α.
- Routes: περιέχει τα ονόματα των διάδρομων.
- Route_Pois: περιέχει την σύνδεση μεταξύ POI και διαδρομής.



ΕΙΚΟΝΑ 5.2: Βάση δεδομένων συστήματος

5.3 Web API

Παρακάτω στην εικόνα βλέπουμε πως είναι η δομή που έχει το backend σύστημα.



ΕΙΚΟΝΑ 5.3 Δομή του Web API του συστήματος

Ο φάκελος uploads είναι αυτός που περιέχει τα αρχεία εικόνων που χρησιμοποιούνται από την εφαρμογή. Ο φάκελος scripts περιέχει τις αρχικοποιήσεις της βάσης δεδομένων και ο src τον πηγαίο κώδικα της εφαρμογής.

Τα διαθέσιμα API που είναι διαθέσιμα στον χρήστη μέσω της εφαρμογής είναι τα παρακάτω:

/routes

- Δημιουργία διαδρομής
- Αναζήτηση διαδρομής με το id της
- Δημιουργία συνδέσμου μεταξύ διαδρομής και σημείο ενδιαφέροντος
- Αναζήτηση διάδρομων που είναι μικρότερη από 10km από την τοποθεσία του χρήστη
- Αναζήτηση όλων των διαδρομών που στην πόλη στην οποία βρίσκεται ο χρήστης
- Αναζήτηση όλων των διαδρομών σε άλλες πόλεις

/pois

- Δημιουργία σημείου ενδιαφέροντος
- Αναζήτηση σημείου ενδιαφέροντος με το id του
- Διαγραφή σημείου ενδιαφέροντος με το id του

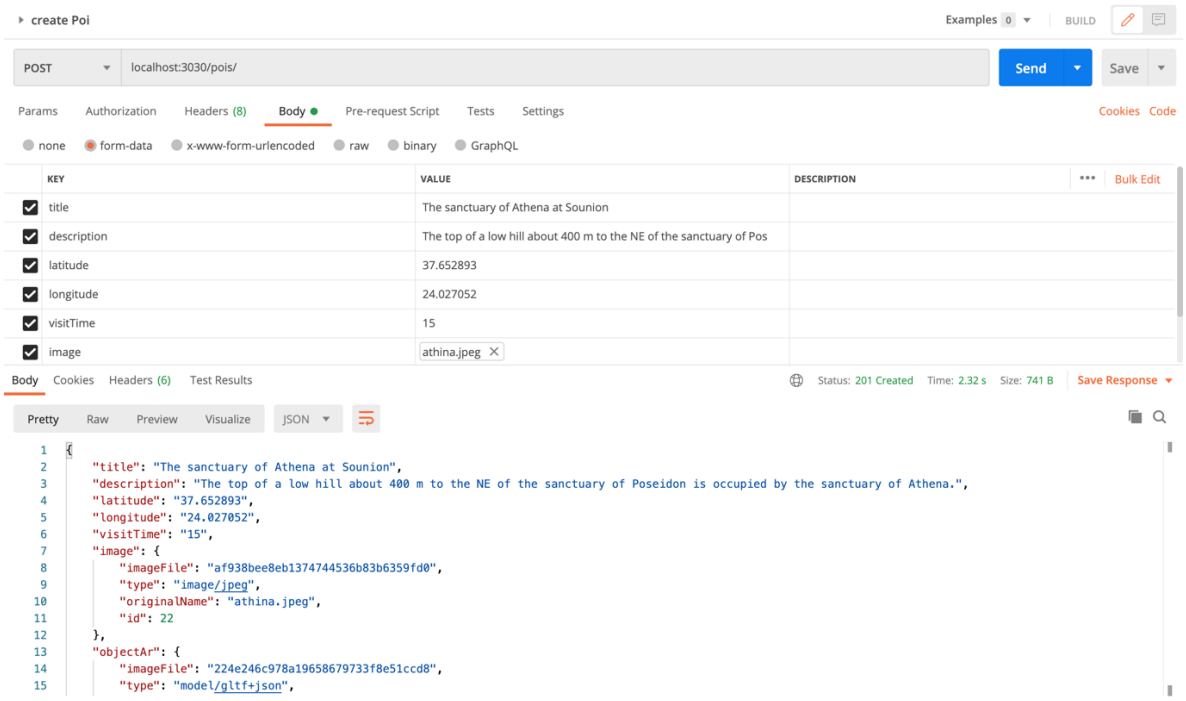
/maps

- Αναζήτηση πόλης με συντεταγμένες

Περισσότερα μπορούν να βρεθούν στον παρακάτω σύνδεσμο https://documenter.getpostman.com/view/4500977/UzBmMSyY

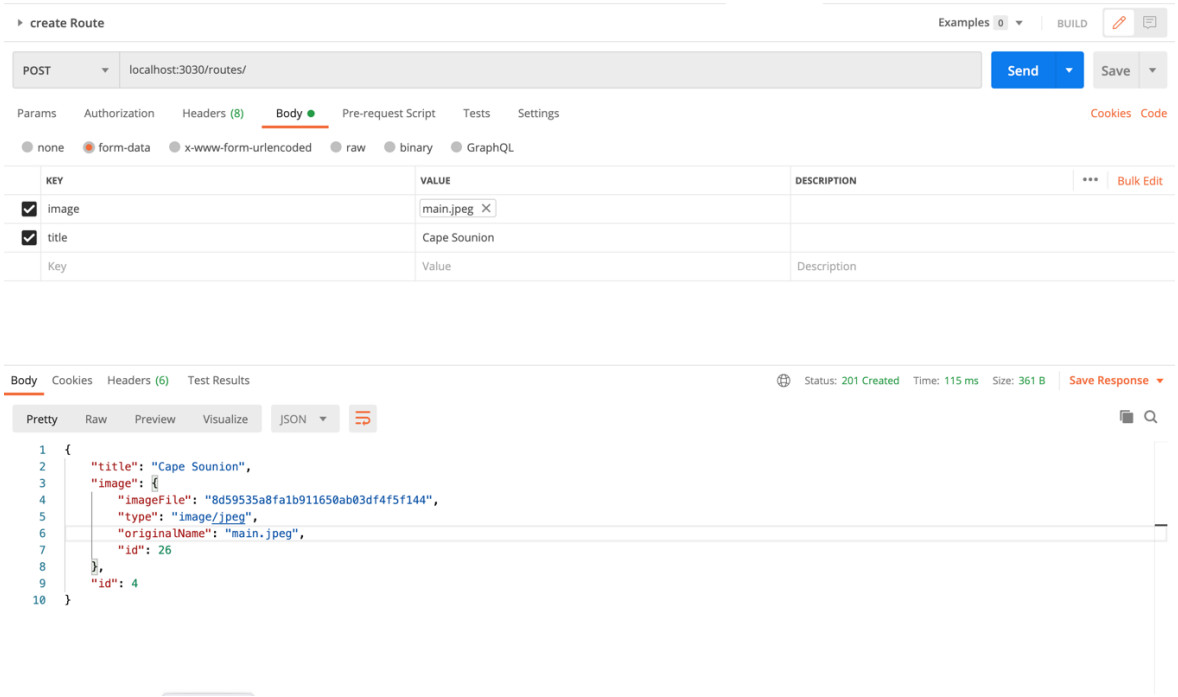
Ο διαχειριστής του συστήματος μπορεί να προσθέσει και να παραμετροποιήσει τις διαδρομές και τα POIS μέσω των κατάλληλων API που προσφέρονται. Οπότε για την δημιουργία διάδρομων ο διαχειριστής πρέπει να ακολουθήσει μια παρόμοια διαδικασία όπως στις παρακάτω εικόνες:

1. Δημιουργία σημείου ενδιαφέροντος



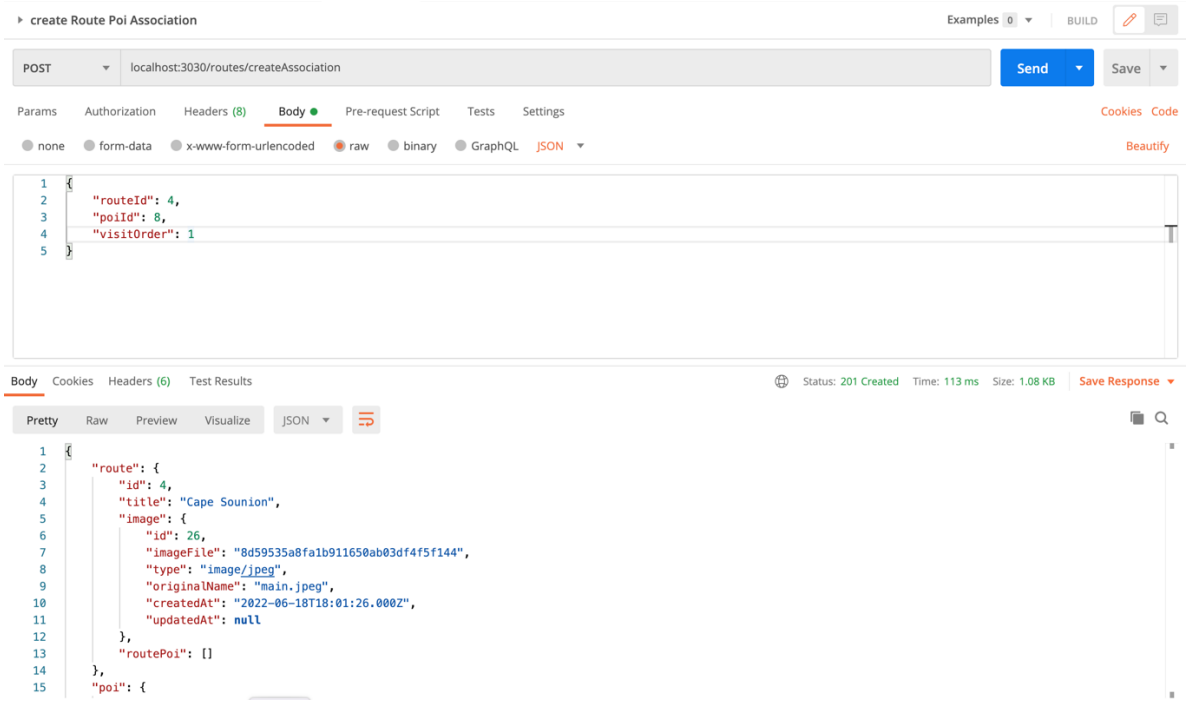
ΕΙΚΟΝΑ 5.4: Δημιουργία σημείου ενδιαφέροντος

2. Δημιουργία διαδρομής



ΕΙΚΟΝΑ 5.5: Δημιουργία διαδρομής

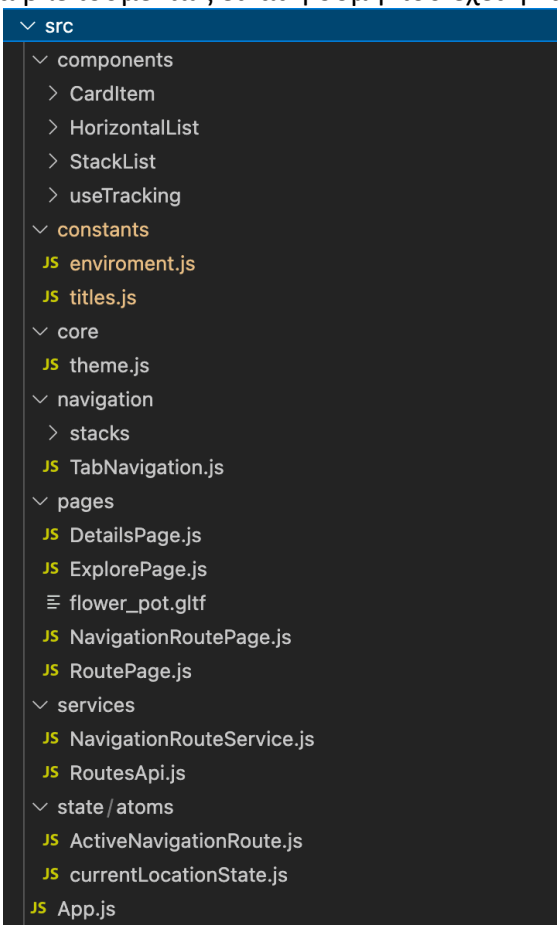
3. Δημιουργία συνδέσμου μεταξύ διαδρομής και σημείο ενδιαφέροντος



ΕΙΚΟΝΑ 5.6: Δημιουργία συνδέσμου μεταξύ διαδρομής και σημείο ενδιαφέροντος

5.4 Εφαρμογή κινητού

Παρακάτω στην εικόνα βλέπουμε πως είναι η δομή που έχει η κινητή εφαρμογή.



ΕΙΚΟΝΑ 5.7: Δομή της React Native εφαρμογής

Κεφάλαιο 6 – Σχεδίαση εφαρμογής λογισμικού

Η εφαρμογή που αναπτύχθηκε μπορεί να χρησιμοποιηθεί από δυο χρήστες τον διαχειριστή, οπού που μπορεί να προσθέσει πληροφορίες στην εφαρμογή, και τον χρήστη της κινητής εφαρμογής ο οποίος καταναλώνει αυτές τις πληροφορίες.

6.1 Περιγραφή περιπτώσεων χρήσης

6.1.1 Περιγραφή περιπτώσεων χρήσης διαχειριστή

Περιγραφή περίπτωσης χρήσης	Δημιουργία διαδρομής
Ρόλος	Διαχειριστής
Προ-συνθήκες	Δυνατότητα να πραγματοποιήσει HTTP αίτημα λόγο μη ύπαρξης γραφικού περιβάλλοντος
Κύρια ροή	1. Δημιουργία σημείου ενδιαφέροντος 2. Δημιουργία διαδρομής 3. Δημιουργία συνδέσμου μεταξύ διαδρομής και σημείο ενδιαφέροντος

6.1.2 Περιγραφή περιπτώσεων χρήσης χρήστη κινητής εφαρμογής

Περιγραφή περίπτωσης χρήσης	Εμφάνιση πληροφοριών διαδρομής
Ρόλος	Χρήστης κινητής εφαρμογής
Προ-συνθήκες	Εγκατεστημένη εφαρμογή στο κινητό, σύνδεση στο διαδίκτυο, συσκευή με κάμερα και GPS
Κύρια ροή	1. Επιλογή διαδρομής 2. Ανάγνωση πληροφοριών διαδρομής

Περιγραφή περίπτωσης χρήσης	Εμφάνιση πληροφοριών σημείου ενδιαφέροντος
Ρόλος	Χρήστης κινητής εφαρμογής
Προ-συνθήκες	Εγκατεστημένη εφαρμογή στο κινητό, σύνδεση στο διαδίκτυο, συσκευή με κάμερα και GPS
Κύρια ροή	1. Επιλογή διαδρομής 2. Επιλογή σημείου ενδιαφέροντος από την διαθέσιμη λίστα 3. Ανάγνωση πληροφοριών για το συγκεκριμένο σημείο ενδιαφέροντος

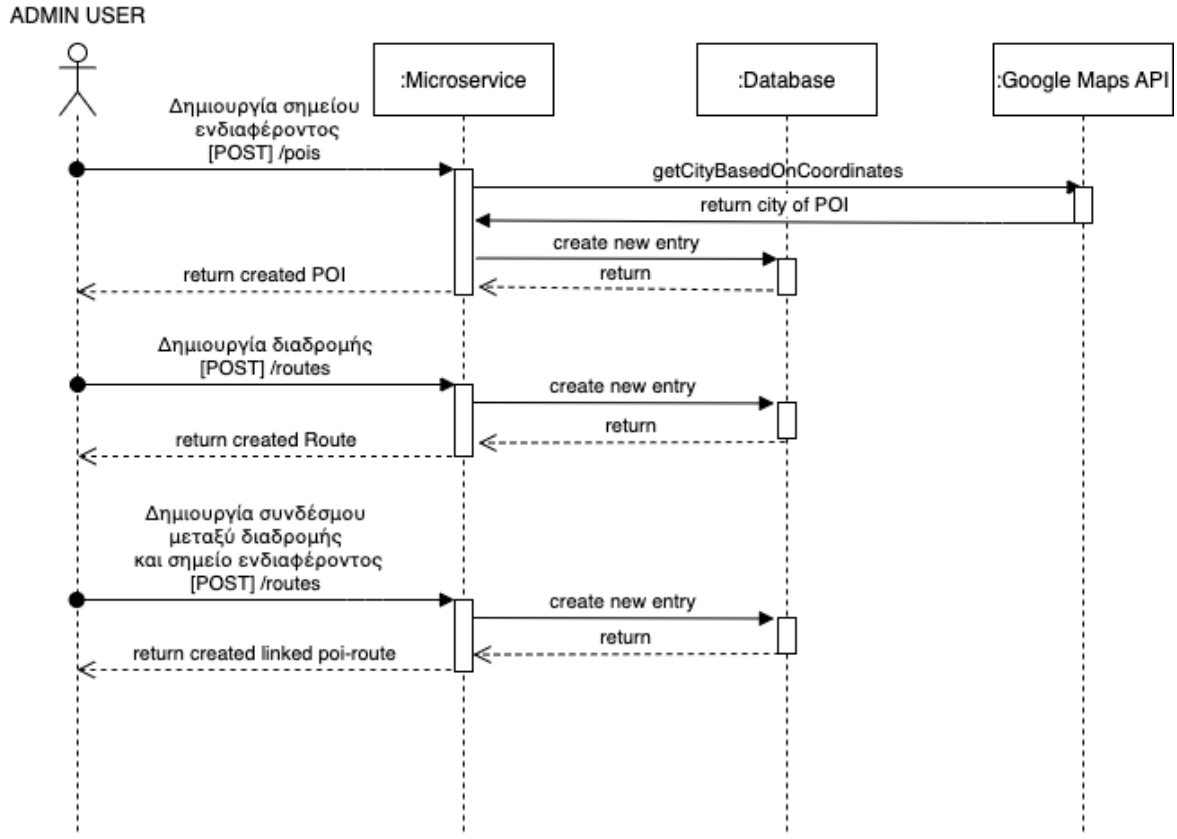
Περιγραφή περίπτωσης χρήσης	Εκτέλεση διαδρομής με χρήση πλοήγησης
Ρόλος	Χρήστης κινητής εφαρμογής
Προ-συνθήκες	Εγκατεστημένη εφαρμογή στο κινητό, σύνδεση στο διαδίκτυο, συσκευή με κάμερα και GPS, εγκατεστημένη εφαρμογή πλοήγησης
Κύρια ροή	1. Επιλογή διαδρομής 2. Επιλογή Start navigation and route 3. Αυτόματο άνοιγμα της διαδρομής στην εφαρμογή πλοήγησης του κινητού 4. Εκτέλεση της διαδρομής 5. Επιλογή End trip για να σταματήσει η εκτέλεση

Περιγραφή περίπτωσης χρήσης	Εκτέλεση διαδρομής χωρίς χρήση πλοήγησης
Ρόλος	Χρήστης κινητής εφαρμογής
Προ-συνθήκες	Εγκατεστημένη εφαρμογή στο κινητό, σύνδεση στο διαδίκτυο, συσκευή με κάμερα και GPS
Κύρια ροή	1. Επιλογή διαδρομής 2. Επιλογή Start route 3. Εκτέλεση της διαδρομής 4. Επιλογή End trip για να σταματήσει η εκτέλεση

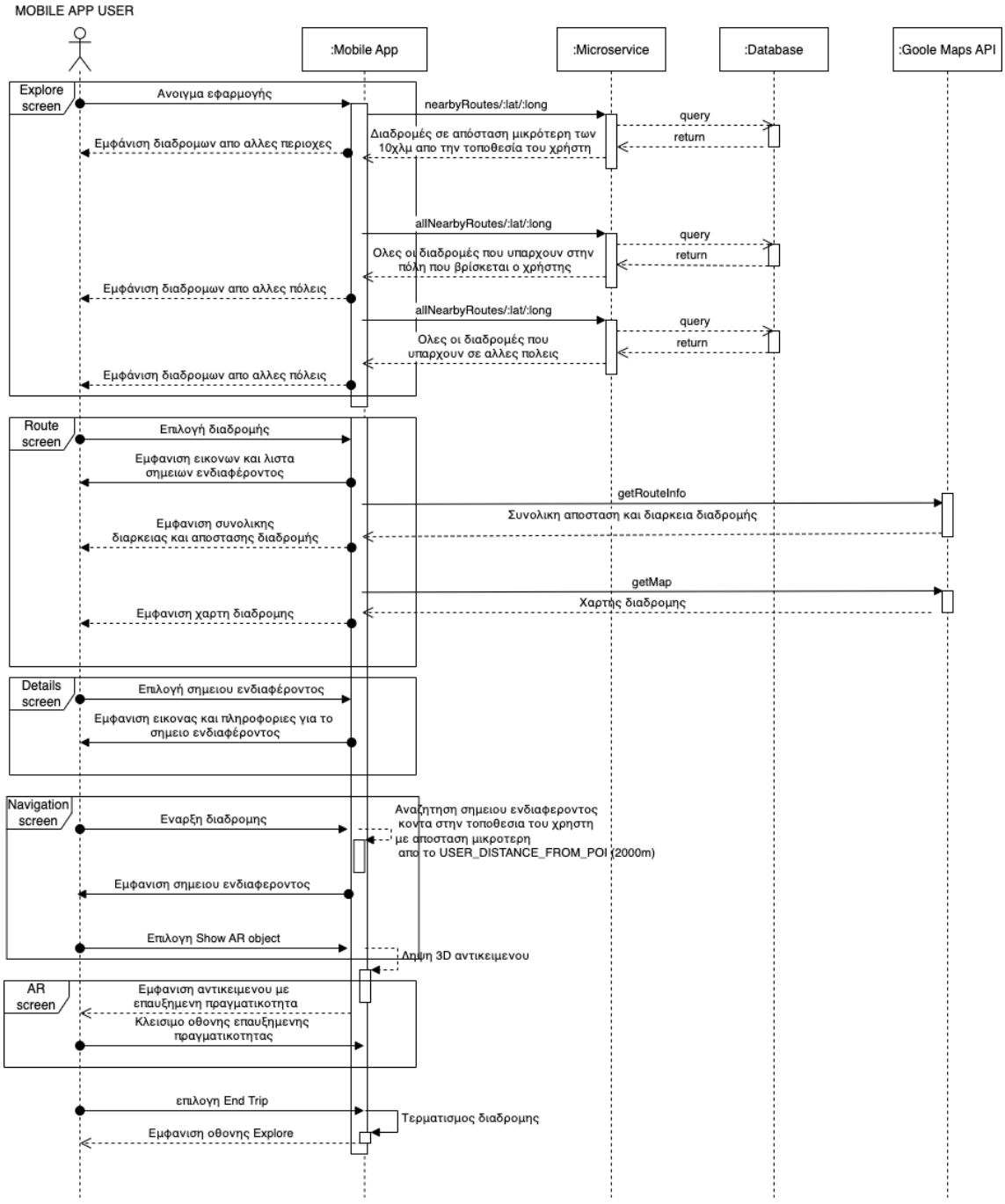
Περιγραφή περίπτωσης χρήσης	Εμφάνιση 3D αντικειμένου με επαυξημένη πραγματικότητα
Ρόλος	Χρήστης κινητής εφαρμογής
Προ-συνθήκες	Εγκατεστημένη εφαρμογή στο κινητό, σύνδεση στο διαδίκτυο, συσκευή με κάμερα και GPS
Κύρια ροή	1. Επιλογή διαδρομής 2. Επιλογή Start route ή Start navigation and route 3. Επιλογή Show AR object κατά την προσέγγιση κάποιου σημείου ενδιαφέροντος 4. Εμφάνιση 3D αντικειμένου μέσω επαυξημένης πραγματικότητας 5. Κλείσιμο οθόνης επαυξημένης πραγματικότητας

6.2 Διαγράμματα αλληλουχίας

6.2.1 Διάγραμμα αλληλουχίας Διαχειριστή



6.2.2 Διάγραμμα αλληλουχίας χρήστη κινητής συσκευής



Κεφάλαιο 7 – Λειτουργία Εφαρμογής

7.1 Χαρακτηριστικά εφαρμογής

Η εφαρμογή που αναπτύχθηκε δίνει την δυνατότητα στον τελικό χρήστη της να διαλέξει και να ακολουθήσει διαδρομές που έχουν δημιουργηθεί από κάποιον διαχειριστή και να εξερευνήσει την περιοχή, την πόλη η ακόμα και την χωρά στην οποία βρίσκεται.

Χρησιμοποιούνται διαφορά εξαρτήματά της συσκευής όπως το GPS και η κάμερα προσφέροντας έτσι στον χρήστη μια εξατομικευμένη εμπειρία.

Μερικά από τα χαρακτηριστικά της εφαρμογής:

1) Εντοπισμός και εμφάνιση κοντινών διαδρομών
2) Εμφάνιση διαδρομών από άλλες περιοχές και πόλεις
3) Υπολογισμός συνολικού χρόνου για την ολοκλήρωση της διαδρομής
4) Υπολογισμός συνολικής απόσταση που πρέπει να διανύσει ο χρήστης
5) Χάρτης με την διαδρομή που πρέπει να ακολουθήσει ο χρήστης
6) Πληροφορίες και εικόνες για το κάθε σημείο ενδιαφέροντος
7) 3D αντικείμενο όταν πλησιάζει ο χρήστης το σημείο ενδιαφέροντος μέσω χρήσης επαυξημένης πραγματικότητας
8) Εύκολο και γρήγορο γραφικό περιβάλλον

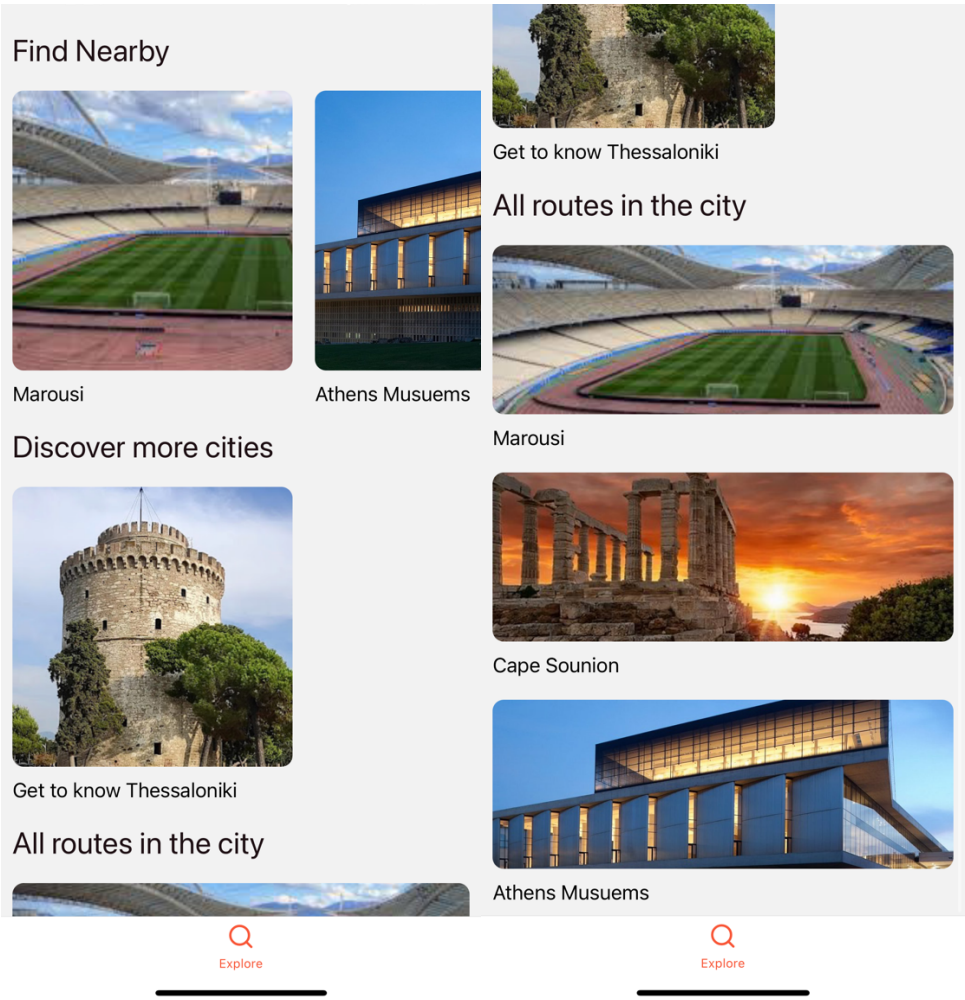
7.2 Οθόνες εφαρμογής

7.2.1 Explore

Η οθόνη Explore είναι η κεντρική σελίδα που εμφανίζεται στον χρήστη και επιτρέπει στον χρήστη να προηγηθεί στις διαθέσιμες διαδρομές που του εμφανίζονται με βάση την γεωγραφική τοποθεσία του.

Οι διαδρομές εμφανίζονται και κατηγοριοποιούνται σε τρεις κατηγορίες:

- Find nearby (περιλαμβάνει τις διαδρομές που είναι λιγότερο από 10km από την τοποθεσία του χρήστη)
- Discover more cities (περιλαμβάνει τις διαδρομές που δεν βρίσκονται στην τωρινή πόλη που βρίσκεται ο χρήστης)
- All routes in the city(περιλαμβάνει όλες τις διαδρομές που βρίσκονται στην πόλη οπού είναι χρήστης)



ΕΙΚΟΝΑ 7.1: Οθόνη Explore

7.2.2 Route

Επιλέγοντας μια διαδρομή από την οθόνη Explore ανοίγει η οθόνη Route η οποία και περιλαμβάνει πληροφορίες και ενέργειες στον χρήστη σχετικά με την επιλεγμένη διαδρομή.

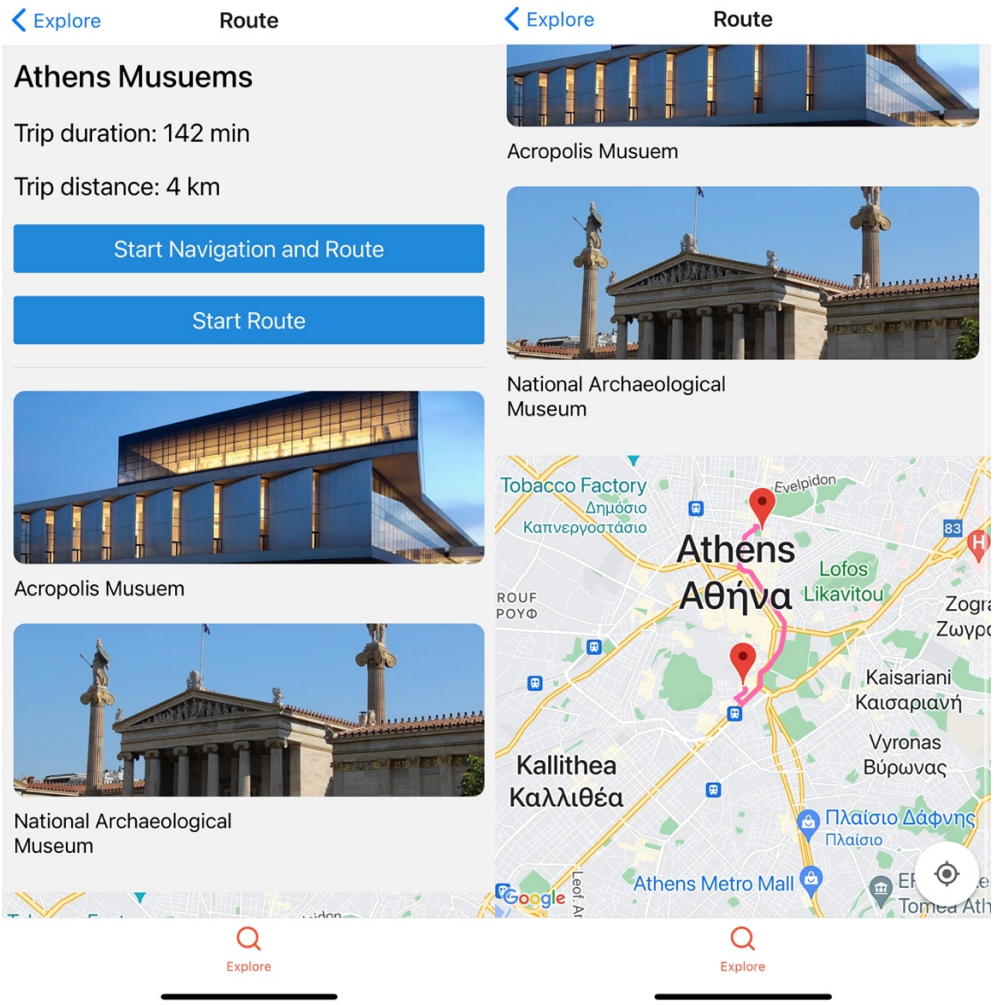
Οι πληροφορίες που περιλαμβάνει η οθόνη είναι:

- Ο τίτλος της διαδρομής
- Η διάρκεια της διαδρομής
- Η συνολική απόσταση που πρέπει να διανύσει ο χρήστης
- Τα σημεία ενδιαφέροντος της διαδρομής
- Χάρτη με την διαδρομή

Επίσης ο χρήστης μπορεί να κάνει τις εξής ενέργειες στην οθόνη:

- Να ξεκινήσει την επιλεγμένη διαδρομή με πλοήγηση

- Να ξεκινήσει την διαδρομή χωρίς πλοήγηση
- Να επιλέξει ένα σημείο ενδιαφέροντος για περισσότερες πληροφορίες



ΕΙΚΟΝΑ 7.2: Οθόνη Routes

7.2.3 Details

Η συγκεκριμένη οθόνη εμφανίζει περισσότερες πληροφορίες σχετικά με το σημείο ενδιαφέροντος όπως εικόνα, τίτλο, περιγραφή και χάρτη με την τοποθεσία.

ΕΙΚΟΝΑ 7.3: Οθόνη Details

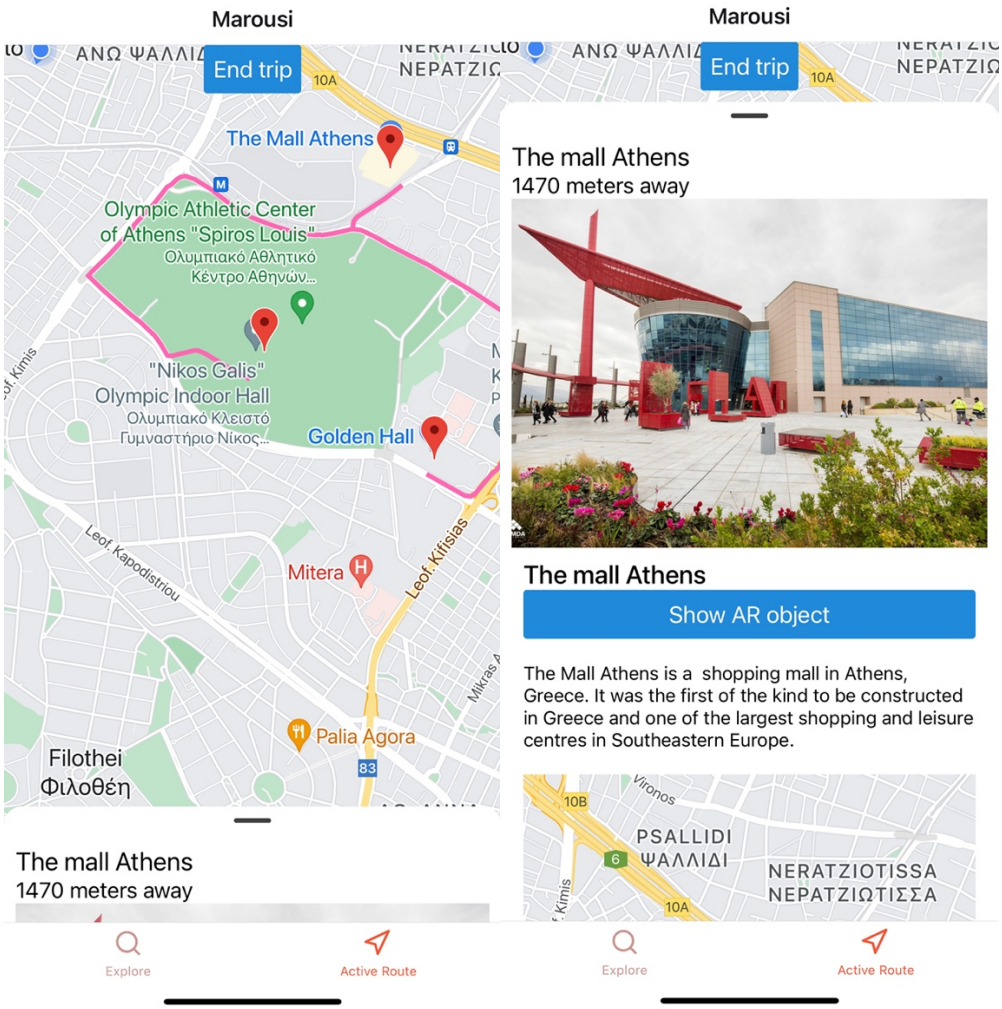
7.2.4 Active route

Με την επιλογή έναρξης διαδρομής από την οθόνη Route εμφανίζεται η οθόνη Active route που περιλαμβάνει τον χάρτη της διαδρομής με το στίγμα του χρήστη αλλά και επίσης καρτέλα με την πληροφορίες για το σημείο ενδιαφέροντος και την απόσταση του χρήστης από αυτό.

Επίσης ο χρήστης μπορεί να προβεί στις παρακάτω ενέργειες:

- Έναρξη επαυξημένης πραγματικότητας (AR) σε περίπτωση που είναι διαθέσιμο
- Τερματισμός διαδρομής

Η επιλογή της έναρξης επαυξημένης πραγματικότητας (AR) εμφανίζεται όταν ο χρήστης είναι κοντά στο συγκεκριμένο σημείο ενδιαφέροντος.



ΕΙΚΟΝΑ 7.4: Οθόνη Active route

ΕΙΚΟΝΑ 7.5: Λειτουργία επαυξημένης πραγματικότητας (AR)

Κεφάλαιο 8 – Συμπεράσματα και βελτιώσεις

Στην διπλωματική εργασία υλοποιήθηκε μια εφαρμογή για ηλεκτρονική περιήγηση μέσω κινητών συσκευών και η αντίστοιχη διαδικτυακή υπηρεσία με την οποία η εφαρμογή αλληλοεπιδρά και αντλεί τα δεδομένα της.

Η εφαρμογή αναπτύχθηκε με τέτοιο τρόπο ώστε να είναι εξατομικευμένη στις ανάγκες του κάθε χρήστη αλλά και επίσης να είναι εύκολη στη χρήση. Η επιλογή της react native για την ανάπτυξη της εφαρμογής έκανε την διαδικασία στις περισσότερες περιπτώσεις πιο εύκολη γιατί υπήρχε γνώση της βιβλιοθήκης που χρησιμοποιείται στις web εφαρμογές (react.js).

Η διαδικτυακή εφαρμογή αναπτύχθηκε έτσι ώστε να μπορεί ευκολά ο διαχειριστής να προσθέτει νέα δεδομένα και δρομολόγια αλλά και επίσης να μπορεί να παραμετροποιήσει τα υπάρχων δρομολόγια ευκολά και γρηγορά μέσω του API που του είναι διαθέσιμο.

Για την επιλογή της τεχνολογικής στοίβας που χρησιμοποιήθηκε έγινε μια μελέτη στις τεχνικές δημιουργίας εφαρμογών για φορητές συσκευές όπου επιλέχθηκε το μοντέλο εφαρμογων μεταγλωτισμένες ανά λειτουργικό σύστημα (cross compiled applications) και στην τεχνολογία των διαδικτυακών υπηρεσιών η επιλογή δημιουργίας ενός RESTful API.

Η μεγαλύτερη δυσκολία που παρουσιάστηκε είναι ότι για την χρήση της AR τεχνολογίας από την εφαρμογή δεν υπήρχαν ενημερωμένες βιβλιοθήκες που υποστηρίζουν ακόμα αυτό το κομμάτι σε σημαντικό βαθμό στην React native.

Συνοπτικά το αποτέλεσμα δείχνει ότι για εφαρμογές με παρουσίαση δεδομένων η παραπάνω τεχνολογική στοίβα είναι κατάλληλη αλλά για την χρήση κάποιον τεχνολογιών όπως AR τότε καλό είναι για την ώρα να χρησιμοποιείται η εγγενής ανάπτυξη εφαρμογών (native apps) γιατί ίσως να μην υπάρχει η αντίστοιχη υποστήριξη από την πλατφόρμα ανάπτυξης (π.χ. React native, flutter).

Μελλοντικές βελτιώσεις είναι η επιλογή διαφορετικής βιβλιοθήκης από αυτήν της google maps για τους χάρτες γιατί δεν υποστηρίζεται η χρήση της πλοήγησης μέσα από την ιδιά εφαρμογή. Ο μονός τρόπος είναι η χρήση της εφαρμογής της Google (google maps)[35] για την πλοήγηση. Μια τέτοια βιβλιοθήκη που προσφέρει κάτι τέτοιο είναι η Mapbox [36]. Επίσης μια άλλη βελτίωση θα ήταν η ανάπτυξη γραφικού περιβάλλοντος για την διαδικτυακή εφαρμογή διαχείρισης έτσι ώστε για να γίνεται ευκολότερη η προβολή και η επεξεργασία των δρομολογίων.

Βιβλιογραφία – Ηλεκτρονικές πηγές

[1] The Importance Of Mobile Applications In Everyday Life!
https://www.hyperlinkinfosystem.com/blog/the-importance-of-mobile-applications-in-everyday-life

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

[3] Matt Neuburg (2021). iOS 15 Programming Fundamentals with Swift. O'Reilly

[4] Human Interface Guidelines
https://developer.apple.com/design/human-interface-guidelines/guidelines/overview/

[5] 5 Reasons to Use Mobile Cross-Platform Development Approach
https://www.velvetech.com/blog/5-reasons-use-mobile-cross-platform-development/

[6] Android
https://el.wikipedia.org/wiki/Android

[7] iOS
https://el.wikipedia.org/wiki/IOS

[8] Comparison of mobile operating systems
https://en.wikipedia.org/wiki/Comparison_of_mobile_operating_systems

[9] ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ – ΕΦΑΡΜΟΓΕΣ
https://smartphoneproject1.weebly.com/lambdaepsiloniotatauomicronupsilonrhogammaiotakappaalpha-sigmaupsilonsigmatauetamualphataualpha-kappaalphaiota-epsilonphialpharhomuomicrongammaepsilonsigma.html

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

[11] Play store - https://play.google.com/

[12] App galery - https://consumer.huawei.com/gr/mobileservices/appgallery/

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

[14] App store - https://www.apple.com/app-store/

[15] Native apps - https://clutch.co/app-developers/resources/pros-cons-native-apps

[16] Pros and cons of native app development
https://decode.agency/article/native-app-development-pros-cons/

[17] What is the Difference Between a Mobile App and a Web App?
https://careerfoundry.com/en/blog/web-development/what-is-the-difference-between-a-mobile-app-and-a-web-app/

[18] What Is a Hybrid App?
https://www.upwork.com/resources/hybrid-app

[19] Cross-compiled applications
https://www.devbridge.com/white-papers/mobile-application-development/cross-compiled-applications/

[20] The Most Popular Cross-platform Mobile Development Frameworks
https://www.thirdrocktechkno.com/blog/best-10-cross-platform-app-frameworks-to-consider-in-2021/

[21] Material Design - https://material.io/design

[22] What Are RESTful Web Services? -
https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

[23] React Native - https://reactnative.dev/architecture/overview

[24] https://www.altexsoft.com/blog/engineering/xamarin-vs-react-native-vs-ionic-vs-nativescript-cross-platform-mobile-frameworks-comparison/

[25] Describing Node.js - https://www.voidcanvas.com/describing-node-js/

[26] Τι ειναι το Node.js
https://www.innoview.gr/el/blog/general/545-ti-einai-to-node-js

[27] nest.js - https://nestjs.com/]

[28] Advantages using MySQL - https://devops.com/8-advantages-using-mysql/

[30] Google Maps developers - https://developers.google.com/maps

[31] GeoLib - npmjs.com/package/geolib]

[32] Viro React - https://github.com/viromedia/viro

[33] React Vector Icons - https://github.com/oblador/react-native-vector-icons

[34] React Native Elements - https://reactnativeelements.com/

[35] Google Maps app Android - https://play.google.com/store/apps/details?id=com.google.android.apps.maps&hl=el&gl=US

[36] Mapbox - https://www.mapbox.com/

[37] Azuma., R., (1997) A Survey of Augmented Reality. Teleoperators and Virtual Environments 6, 4 (August 1997), 355-385.

[38] Augmented reality - https://en.wikipedia.org/wiki/Augmented_reality

[39] Types of AR - https://digitalpromise.org/initiative/360-story-lab/360-production-guide/investigate/augmented-reality/getting-started-with-ar/types-of-ar/

Πηγές εικόνων

2.1 Λογότυπο λειτουργικού android - https://www.android.com/

2.2 Android Studio - https://developer.android.com/studio

2.3 Λογότυπο iOS - https://en.wikipedia.org/wiki/File:IOS_logo.svg

2.4 XCode - https://developer.apple.com/xcode/resources/

2.5 Hybrid app development - https://decode.agency/article/native-app-development-pros-cons/

2.6 REST API architecture - https://medium.com/@dilankam/restful-api-design-best-practices-principles-ded471f573f3

3.1 Επαυξημένη πραγματικότητα μέσω κινητής συσκευής - https://miro.medium.com/max/3200/1*A0Kj20osNQbSuCzJEZK2EQ.jpeg

3.2 Επαυξημένη πραγματικότητα στην γυμναστική - https://cdn.mos.cms.futurecdn.net/fd6e461aa2cb97efd98b0a9759e03ee6-1200-80.jpg

3.3 AR φυσικού δείκτη - https://miro.medium.com/max/3000/0*p79RAZXzBHIJdsR5.jpg

3.4 AR χωρίς φυσικό δείκτη - https://media.istockphoto.com/photos/decorating-apartment-man-holding-digital-tablet-with-ar-interior-picture-id1220765714?k=20&m=1220765714&s=612x612&w=0&h=1XR_neP-Zevvx0SVb9LXNnZPbrilqtyDKueLIO-Xdfg=

3.5 AR με χρήση τοποθεσίας - https://wear-studio.com/wp-content/uploads/2022/06/Location-Based-AR.jpg

4.1 React Native Λογότυπο - https://www.appstud.com/wp-content/uploads/2018/03/React-Native-Titre.png

4.2 React Native Rendering - https://www.altexsoft.com/blog/engineering/xamarin-vs-react-native-vs-ionic-vs-nativescript-cross-platform-mobile-frameworks-comparison/

4.3 Λογότυπο Node.js - https://www.innoview.gr/images/news/nodejs-hosting.jpg

4.4 Παραδοσιακό back-end σύστημα σε σύγκριση με την Node.js - https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

4.5 Μοντέλο επεξεργασίας Node.js - https://www.voidcanvas.com/describing-node-js/

4.6 Λογότυπο Nest.js - https://nestjs.com/

4.7 λογοτυπο MySQL - https://www.mysql.com/

ΠΑΡΑΡΤΗΜΑΤΑ

Κώδικας εφαρμογής React Native

Ο κώδικας της react native εφαρμογής : https://github.com/mariosp/tour-app-mobile

App.js

```
import React, {useEffect} from 'react';
import Icon from 'react-native-vector-icons/Feather';
import TabNavigation from './navigation/TabNavigation';
import {StatusBar, StyleSheet, Text, View} from 'react-native';
import {PERMISSIONS, request, RESULTS, check} from 'react-native-permissions';
import Geolocation from 'react-native-geolocation-service';
import {RecoilRoot} from 'recoil';

Icon.loadFont();

const App = () => {
    useEffect(() => {
        check(PERMISSIONS.IOS.CAMERA)
            .then((result) => {
                switch (result) {
                    case RESULTS.UNAVAILABLE:
                        console.log('This feature is not available (on this
device / in this context)');
                        break;
                    case RESULTS.DENIED:
                        request(PERMISSIONS.IOS.CAMERA).then((result) => {
                            // …
                        });
                        break;
                    case RESULTS.LIMITED:
                        console.log('The permission is limited: some actions
are possible');
                        break;
                    case RESULTS.GRANTED:
                        console.log('The permission is granted');
                        break;
                    case RESULTS.BLOCKED:
                        console.log('The permission is denied and not
requestable anymore');
                        break;
                }
            })
            .catch((error) => {
                //
            });
```

```
        Geolocation.requestAuthorization("whenInUse").then(result => {
            switch (result) {
                case "disabled":
                    console.log('Location disabled');
                    break;
                case "granted":
                    console.log('Location granted');
                    break;
                case "denied":
                    console.log('Location denied');
                    break;
                case "restricted":
                    console.log('Location restricted');
                    break;
            }
        });

    },[]);

  return (
      <>
        <RecoilRoot>
            <StatusBar barStyle = "default" translucent = {false} />
            <TabNavigation />
        </RecoilRoot>
      </>
      );
};

export default App;

const styles = StyleSheet.create({
    container: {
        flex: 1,
        padding: 24,
        backgroundColor: 'grey',
    },
    contentContainer: {
        flex: 1,
        alignItems: 'center',
        backgroundColor: 'white'
    },
});
```

Components>CardItem>CardItem.js

```
import React, {useState} from 'react';
import {Image, Pressable, StyleSheet, Text, View} from 'react-native';
```

```
import {imageRoutesUrl} from '../../constants/enviroment';

const CardItem = ({ item, index, open }) => {
    const [cardOpacity, setCardOpacity] = useState(1.0);

    const reset = () => setCardOpacity(1.0);
    const changeOpacity = ()=> setCardOpacity(0.6);

    return (
        <View style={style.cardContainer} opacity={cardOpacity}>
            <Pressable onPress={()=>open(item)} onPressIn={changeOpacity}
onPressOut={reset}>
                <Image
                    source={{uri: imageRoutesUrl + item.image['imageFile']}}
                    style={style.imageContainer}
                />
                <Text style={style.titleContainer}>{item.title}</Text>
            </Pressable>
        </View>
    );
};

const style = StyleSheet.create({
    cardContainer:{
        margin: 10,
        flex: 1
    },
    imageContainer:{
        width:250,
        height: 250,
        borderRadius: 10,
        flex: 2,
        resizeMode: 'cover'
    },
    titleContainer:{
        marginTop: 10,
        maxWidth: 250,
        flex: 1,
        fontSize: 16
    },
    cardWrapper: {
        width:220,
        height: 220,
        padding: 0
    }
})

export default CardItem;
```

Components>CardItem>FluidCard.js

```
import React, {useState} from 'react';
import {Image, Pressable, StyleSheet, Text, View} from 'react-native';
import {imagePoiUrl} from '../../constants/enviroment';


const FluidCard = ({ item, index, open, imagebaseUrl }) => {
    const [cardOpacity, setCardOpacity] = useState(1.0);

    const reset = () => setCardOpacity(1.0);
    const changeOpacity = ()=> setCardOpacity(0.6);

    return (
        <View style={style.cardContainer} opacity={cardOpacity}>
            <Pressable onPress={()=> open(item)} onPressIn={changeOpacity}
onPressOut={reset}>
                <Image
                    source={{uri: imagebaseUrl + item.image['imageFile']}}
                    style={style.imageContainer}
                />
                <Text style={style.titleContainer}>{item.title}</Text>
            </Pressable>
        </View>
    );
}

const style = StyleSheet.create({
    cardContainer:{
        margin: 10,
        flex: 1
    },
    imageContainer:{
        width: '100%',
        height: 150,
        borderRadius: 10,
        flex: 1,
        resizeMode: 'cover'
    },
    titleContainer:{
        marginTop: 10,
        maxWidth: 250,
        flex: 1,
        fontSize: 16
    },
    cardWrapper: {
        width:220,
        height: 220,
```

```
        padding: 0
    }
})

export default FluidCard;
```

Components>HorizontalList>HorizontalList.js

```
import React from 'react';
import {FlatList, StyleSheet, View} from 'react-native';
import {Text} from 'react-native-elements';
import {TextSize,Color} from '../../core/theme';

const HorizontalList = ({RenderItemComponent, title, data, open}) => {
    return (
        <View>
            <Text style={style.titleContainer}>{title}</Text>
            <FlatList
                data={data}
                keyExtractor={(item) => item.id.toString()}
                renderItem={({item}) => <RenderItemComponent item={item}
open={open} />}
                horizontal={true}
                showsHorizontalScrollIndicator={false}
            />
        </View>
    );
};

const style = StyleSheet.create({
    titleContainer: {
        margin: 10,
        flex: 1,
        fontSize: TextSize.cardHeading,
        color: Color.primary
    },
});

export default HorizontalList;
```

Components>StackList>StackList.js

```
import React from 'react';
import {FlatList, StyleSheet, View} from 'react-native';
import {Text} from 'react-native-elements';
import {TextSize,Color} from '../../core/theme';
import {imageRoutesUrl} from '../../constants/enviroment';


const StackList = ({RenderItemComponent, title, data, open}) => {
```

```
    return (
        <View>
            <Text style={style.titleContainer}>{title}</Text>
            <FlatList
                data={data}
                keyExtractor={(item) => item.id.toString()}
                renderItem={({item}) => <RenderItemComponent item={item}
open={open} imagebaseUrl={imageRoutesUrl} />}
            />
        </View>
    );
};

const style = StyleSheet.create({
    titleContainer: {
        margin: 10,
        flex: 1,
        fontSize: TextSize.cardHeading,
        color: Color.primary
    },
});

export default StackList;
```

Components>useTracking>useTracking.js

```
import React, {useState,useEffect} from 'react';
import Geolocation from 'react-native-geolocation-service';


function useTracking (isActive){
    const [location, setLocation] = useState(null);

    useEffect(()=>{
        if(isActive) startWatchLocation();
        else Geolocation.stopObserving();

        return ()=>{
            Geolocation.stopObserving();
        }
    }, [isActive])



    const startWatchLocation = () =>{
            Geolocation.watchPosition(
                (position) => {
                    console.log("LOCATION WATCH->")
                    console.log(position.coords);
```

```
                    setLocation({
                        latitude: position.coords.latitude,
                        longitude: position.coords.longitude
                    })
                },
                (error) => {
                    // See error code charts below.
                    console.log(error.code, error.message);
                },
                { enableHighAccuracy: true, distanceFilter: 10 }
            );
    }

    return {
        location: location
    }
}


export default useTracking;
```

constants>enviroment.js

```
export const GOOGLE_API_KEY = "AIzaSyBrB1KArl6NBB13w3gsBz-MUx-yVOxWwKs";
export const USER_DISTANCE_FROM_POI = 2000

export const routesApi = "http://192.168.1.18:3030/";
export const imageRoutesUrl = "http://192.168.1.18:3030/uploads/routes/";
export const imagePoiUrl = "http://192.168.1.18:3030/uploads/poi_images/";
```

constants>title.js

```
export const FIND_NEARBY = "Find Nearby";
export const OTHER_PLACES = "Distant routes";
export const ALL_PLACES_IN_TOWN= "All routes in town";
```

core>theme.js

```
export const Color = {
    primary: '#1e1014',
    secondary: '#2978a0ff',
    tertiary: '#cf9893ff',
    success: '#c6ebbeff',
    danger: '#f95738ff'
}

export const TextSize = {
    cardHeading: 24,
}

export const Container = {flex: 1}
```

navigation>TabNavigation.js

```
import React from 'react';
import {Button, Text, View, StyleSheet} from 'react-native';
import {NavigationContainer} from '@react-navigation/native';
import {createBottomTabNavigator} from '@react-navigation/bottom-tabs';
import Icon from 'react-native-vector-icons/Feather';
import {Color} from '../core/theme';
import ExploreStackScreen from './stacks/ExploreStackScreen';
import {ViroARSceneNavigator, ViroText, ViroARScene,  Viro3DSceneNavigator}
from 'react-viro'
import NavigationRouteStackScreen from './stacks/NavigationRouteStackScreen';
import {activeNavigationRouteSate} from
'../state/atoms/ActiveNavigationRoute';
import {useRecoilState} from 'recoil';

export const scenetest =  () => {

    return (
        <ViroARScene onTrackingUpdated={()=> null}>
        <ViroText text="Hello World" position={[0, -.1, -1]} />
        </ViroARScene>
    );
}

const Tab = createBottomTabNavigator();

const TabNavigation = () => {
  const [activeNavigationRoute, setActiveNavigationRoute] =
useRecoilState(activeNavigationRouteSate);

  return (
    <NavigationContainer>
      <Tab.Navigator  tabBarOptions={{
          activeTintColor: Color.danger,
          inactiveTintColor: Color.tertiary,
          safeAreaInsets: { top: 0 }
        }}
      >
        <Tab.Screen
            name="Explore"
            component={ExploreStackScreen}
            options={{
                tabBarIcon: ({ color, size }) => (<Icon name="search"
size={size} color={color}/>)
            }}
        />
          {
```

```
            activeNavigationRoute.isActive &&
              <Tab.Screen
                  name="ActiveRoute"
                  component={NavigationRouteStackScreen}
                  options={{
                      tabBarLabel: " Active Route",
                      tabBarIcon: ({ color, size }) => (<Icon
name="navigation" size={size} color={color}/>)
                  }}
              />
          }
      </Tab.Navigator>
    </NavigationContainer>
  );
};

export default TabNavigation;
```

navigation>stacks>ExploreStackScreen.js

```
import React from 'react';
import {createStackNavigator} from '@react-navigation/stack';
import ExplorePage from '../../pages/ExplorePage';
import RoutePage from '../../pages/RoutePage';
import DetailsPage from '../../pages/DetailsPage';

const ExploreStack = createStackNavigator();

const ExploreStackScreen = ({ navigation }) => {
    return (
        <ExploreStack.Navigator>
            <ExploreStack.Screen name="Explore" component={ExplorePage}
options={{headerShown: false}} />
            <ExploreStack.Screen name="Route" component={RoutePage} />
            <ExploreStack.Screen name="Details" component={DetailsPage}
options={{headerShown: false}} />
        </ExploreStack.Navigator>
    );
}

export default ExploreStackScreen;
```

navigation>stacks>NavigationRouteStackScreen.js

```
import React from 'react';
import {createStackNavigator} from '@react-navigation/stack';
import NavigationRoutePage from '../../pages/NavigationRoutePage';

const NavigationRouteStack = createStackNavigator();
```

```
const NavigationRouteStackScreen = () => {
 return (
     <NavigationRouteStack.Navigator>
         <NavigationRouteStack.Screen name="NavigationRoute"
component={NavigationRoutePage} />

     </NavigationRouteStack.Navigator>
 )
}

export default NavigationRouteStackScreen;
```

pages>DetailsPage.js

```
import React, {useState} from "react";
import {
    Image,
    ImageBackground, Modal,
    Pressable,
    SafeAreaView,
    ScrollView,
    StyleSheet,
    Text,
    TouchableOpacity,
    View,
} from 'react-native';
import {imagePoiUrl, USER_DISTANCE_FROM_POI} from '../constants/enviroment';
import Icon from 'react-native-vector-icons/Feather';
import MapView, {PROVIDER_GOOGLE} from 'react-native-maps';
import {Marker} from 'react-native-maps';
import {Button} from 'react-native-elements';

import {fromLatLngToPoint} from 'mercator-projection';

import {ViroARSceneNavigator, ViroText, ViroARScene,  Viro3DSceneNavigator,
ViroAmbientLight, ViroNode, Viro3DObject} from 'react-viro';
import {useRecoilState} from 'recoil';
import {currentLocationState} from '../state/atoms/currentLocationState';



function _latLongToMerc(lat_deg, lon_deg) {
    let lon_rad = (lon_deg / 180.0 * Math.PI)
    let lat_rad = (lat_deg / 180.0 * Math.PI)
    let sm_a = 6378137.0
    let xmeters  = sm_a * lon_rad
    let ymeters = sm_a * Math.log((Math.sin(lat_rad) + 1) / Math.cos(lat_rad))
    return ({x:xmeters, y:ymeters});
}
```

```
const DetailsPage = ({navigation, route, data, distance}) => {
    const item = route?.params?.data || data;
    const [modalVisible, setModalVisible] = useState(false);
    const [currentLocation] = useRecoilState(currentLocationState);
    const transformPointToAR = (lat, long) => {
        const objPoint = _latLongToMerc(lat, long);

        //TODO:CHANGE WITH this line for current location

        const devicePoint = _latLongToMerc(currentLocation.latitude,
currentLocation.longitude);
        // const devicePoint = _latLongToMerc(38.048429, 23.774418);
        // console.log("objPointZ: " + objPoint.y + ", objPointX: " +
objPoint.x)
        // latitude(north,south) maps to the z axis in AR
        // longitude(east, west) maps to the x axis in AR
        const objFinalPosZ = objPoint.y - devicePoint.y;
        const objFinalPosX = objPoint.x - devicePoint.x;
        return ({x:objFinalPosX, z:-objFinalPosZ});
    }

    //TODO:CHANGE WITH this line for current location
    const currentLocationToWorld = transformPointToAR(38.048423, 23.774414);

    console.log(currentLocationToWorld)


    const scene = () => {
        console.log(item.objectAr)
        return (
            <ViroARScene onTrackingUpdated={()=> null}>
                <ViroAmbientLight color="#FFFFFF" intensity={400}/>
                <ViroNode>
                    <Viro3DObject
                        source={{uri: imagePoiUrl +
item.objectAr['imageFile']}}
                                  position={[currentLocationToWorld.x, 0,
currentLocationToWorld.z-2]}
                                  type="GLTF"
                    />
                </ViroNode>
            </ViroARScene>
        );
    }


    const renderModal = () => {
```

```
        console.log("INIT INIT INIT")
        console.log(item)
        return (
            <Modal
            animationType="slide"
            visible={modalVisible}
            onRequestClose={() => {
                setModalVisible(!modalVisible);
            }}
        >
            <SafeAreaView>
                <Pressable style={styles.buttonWrapper} onPress={() =>
setModalVisible(!modalVisible)}>
                    <Icon
                        name="x"
                        size={30}
                        color="black"
                    />
                </Pressable>
            </SafeAreaView>
            <View style={{flex: 1}}>
                <ViroARSceneNavigator
                    // worldAlignment={'GravityAndHeading'}
                    initialScene={{scene: scene}} style={{flex: 1,
height:500}}/>
            </View>
        </Modal>
        )
    }

   const handleClickShowAR = ()=>{
       setModalVisible(true);
    }


    return (
        <ScrollView contentContainerStyle={styles.scrollContainer}>
                <ImageBackground
                    source={{uri: imagePoiUrl + item.image['imageFile']}}
                    style={styles.imageContainer}
                >
                    <SafeAreaView>
                        {route?.params?.data &&
                        <Pressable style={styles.buttonWrapper} onPress={()=>
navigation.goBack()}>
                            <Icon
                                name="arrow-left"
                                size={30}
```

```
                                color="black"
                            />
                        </Pressable>
                        }
                    </SafeAreaView>
                </ImageBackground>
                <View style={styles.detailsContainer}>
                    <Text style={styles.title}>
                        {item.title}
                    </Text>
                    {distance && item["objectAr"] && <Button
onPress={handleClickShowAR} disabled={distance > USER_DISTANCE_FROM_POI}
title="Show AR object" />}
                    <Text style={styles.description}>
                        {item.description}
                    </Text>
                    <View style={styles.mapWrapper}>
                        <MapView
                            style={styles.map}
                            provider={PROVIDER_GOOGLE}
                            region={{
                                latitude: Number(item.latitude),
                                longitude: Number(item.longitude),
                                latitudeDelta: 0.015,
                                longitudeDelta: 0.0121
                            }}
                            showsUserLocation={true}
                            showsMyLocationButton={true}
                        >
                            <Marker
                                coordinate={{
                                    latitude: Number(item.latitude),
                                    longitude: Number(item.longitude),
                                }}
                                centerOffset={{ x: -18, y: -60 }}
                                anchor={{ x: 0.69, y: 1 }}
                                title={item.title}
                            >
                            </Marker>
                        </MapView>
                    </View>
                </View>
            {distance && renderModal()}
        </ScrollView>
    )
}

const styles = StyleSheet.create({
```

```
    scrollContainer: {
        justifyContent: 'space-between'
    },
    imageContainer:{
        width: '100%',
        height: 300,
        resizeMode: 'cover'
    },
    detailsContainer:{
        margin: 10,
    },
    buttonWrapper:{
        borderWidth:1,
        borderColor:'rgba(0,0,0,0.2)',
        alignItems:'center',
        justifyContent:'center',
        width:40,
        height:40,
        backgroundColor:'#fff',
        borderRadius:50,
        margin: 10,
    },
    title: {
        fontSize: 20,
        fontWeight: '500'
    },
    description: {
        marginTop: 20
    },
    mapWrapper: {
        marginTop: 20,
        height: 400,
        width: '100%',
        justifyContent: 'flex-end',
        alignItems: 'center',
    },
    map: {
        ...StyleSheet.absoluteFillObject,
    },
})
export default DetailsPage;
```

pages>ExplorePage.js

```
import React, {useState, useEffect} from 'react';
import {SafeAreaView, ScrollView, StyleSheet} from 'react-native';
import HorizontalList from '../components/HorizontalList/HorizontalList';
import StackList from '../components/StackList/StackList';
import CardItem from '../components/CardItem/CardItem';
```

```
import { FIND_NEARBY, OTHER_PLACES , ALL_PLACES_IN_TOWN} from
'../constants/titles';
import Geolocation from 'react-native-geolocation-service';
import {currentLocationState} from '../state/atoms/currentLocationState';
import {useRecoilState} from 'recoil';
import {getAllNearbyPlaces, getNearbyPois, getOtherPlaces} from
'../services/RoutesApi';
import FluidCard from '../components/CardItem/FluidCard';

const currentLocationOptions = {
    enableHighAccuracy: false,
    timeout: 5000,
}

const ExplorePage = ({navigation}) => {
    const [currentLocation, setCurrentLocation] =
useRecoilState(currentLocationState);

    const [nearbyRoutes, setNearbyRoutes] = useState([]);

    const [otherPlaces, setOtherPlaces] = useState([]);

    const [allNearByRoutes, setAllNearbyRoutes] = useState([]);

    useEffect(()=> {
        Geolocation.getCurrentPosition((position) => {
            setCurrentLocation((oldState)=> ({
                ...oldState,
                latitude: position.coords.latitude,
                longitude: position.coords.longitude,
                loaded: true
            }))

        }, (err) => {
            console.log(err);
        }, currentLocationOptions);
    }, [])

    useEffect(() => {
        console.log("USE EFFECT CHANGED")
        console.log(currentLocation);
        if(currentLocation?.loaded){
           getNearbyPois(currentLocation.latitude, currentLocation.longitude)
               .then(response => {
                   console.log(response)
                   setNearbyRoutes(response);
               });
```

```
                getOtherPlaces(currentLocation.latitude,
currentLocation.longitude)
                .then(response => {
                    console.log(response)
                    setOtherPlaces(response);
                });

                getAllNearbyPlaces(currentLocation.latitude,
currentLocation.longitude)
                .then(response => {
                    console.log(response)
                    setAllNearbyRoutes(response);
                });
        }
    }, [currentLocation])

    const openRoutePage = (item) => navigation.navigate('Route', { data: item
});


    return(
        <ScrollView contentContainerStyle={styles.scrollContainer}>
            <SafeAreaView>
                <HorizontalList RenderItemComponent={CardItem}
title={FIND_NEARBY} data={nearbyRoutes} open={openRoutePage}/>

                <HorizontalList RenderItemComponent={CardItem}
title={OTHER_PLACES} data={otherPlaces} open={openRoutePage}/>

                <StackList RenderItemComponent={FluidCard}
title={ALL_PLACES_IN_TOWN} data={allNearByRoutes} open={openRoutePage} />

            </SafeAreaView>
        </ScrollView>
    );
}

const styles = StyleSheet.create({
    scrollContainer: {
        flexGrow: 1,
        justifyContent: 'space-between'
    }
})

export default ExplorePage;
```

pages>NavigationRoutePage.js

```
import React, {useState, useEffect,useRef} from 'react';
```

```
import {Image, StyleSheet, Text, View} from 'react-native';
import MapView, {Marker, PROVIDER_GOOGLE} from 'react-native-maps';
import MapViewDirections from "react-native-maps-directions";
import {GOOGLE_API_KEY, imagePoiUrl} from '../constants/enviroment';
import {useRecoilState} from 'recoil';
import {activeNavigationRouteSate} from
'../state/atoms/ActiveNavigationRoute';
import useTracking from '../components/useTracking/useTracking';
import {getNearbyPoi} from '../services/NavigationRouteService';
import BottomSheet from '@gorhom/bottom-sheet';
import DetailsPage from './DetailsPage';
import {Button} from 'react-native-elements';


const renderMap = (routePoisData) => {
    return routePoisData.map(routePoi=> {
        return <Marker
            key={routePoi.poi.id}
            coordinate={{
                latitude: Number(routePoi.poi.latitude),
                longitude: Number(routePoi.poi.longitude)
            }}
            title={routePoi.poi.title}
        />
    })
}

const NavigationRoutePage = ({navigation})=>{
    const bottomSheetRef = useRef<BottomSheet>(null);
    const {location} = useTracking(true);
    const [activeNavigationRoute, setActiveNavigationRoute] =
useRecoilState(activeNavigationRouteSate);
    console.log()
    if(!activeNavigationRoute.isActive) {
        return null;
    }
    navigation.setOptions({ headerTitle: activeNavigationRoute.data["title"],
});

    const [duration, setDuration] = useState(0);
    const [distance, setDistance] = useState(0);
    const [nearbyPoi, setNearbyPoi] = useState(null);

    const data = activeNavigationRoute.data;
    const routePoisData = data['routePoi'];
    const origin = routePoisData[0]['poi'];
    const destination = routePoisData[routePoisData.length-1]['poi'];
```

```
    const waypoints = routePoisData.map((poiRoute)=>{
        return {
            latitude: Number(poiRoute.poi.latitude),
            longitude: Number(poiRoute.poi.longitude)
        }
    })

    useEffect(()=>{
        // console.log(location)
        if(location) {
            setNearbyPoi(getNearbyPoi(location, routePoisData));
        }
    }, [location])

    console.log("AFTER END")

    if(nearbyPoi != null) {
        console.log(bottomSheetRef)
    }

    endTrip = () => {
        console.log("END TRIP CLICKED")
        setActiveNavigationRoute({
            isActive: false,
            data: null,
        });
        setTimeout(()=> navigation.navigate('Explore', { screen: 'Explore' }),
0)
    }

    return (
         <View style={styles.mapWrapper}>
            <MapView
                provider={PROVIDER_GOOGLE}
                style={styles.map}
                initialRegion={{
                    latitude: Number(origin.latitude),
                    longitude: Number(origin.longitude),
                    latitudeDelta: 0.0622,
                    longitudeDelta: 0.0121,
                }}
                showsUserLocation={true}
                showsMyLocationButton={true}
            >
                <MapViewDirections
                    origin={{latitude: Number(origin.latitude), longitude:
Number(origin.longitude)}}
```

```
                    destination={{latitude: Number(destination.latitude),
longitude: Number(destination.longitude)}}
                    waypoints={waypoints}
                    strokeWidth={4}
                    strokeColor="hotpink"
                    apikey={GOOGLE_API_KEY}
                    timePrecision={'now'}
                    precision={'high'}
                    onReady={result => {
                        setDistance(result.distance);
                        setDuration(result.duration);
                        console.log(`Distance: ${distance} km`)
                        console.log(`Duration: ${duration} min`)
                    }}
                />
                {renderMap(routePoisData)}
            </MapView>

            <Button style={styles.buttonEnd} onPress={endTrip} title="End
trip" />

            { nearbyPoi &&
                <BottomSheet
                    index={0}
                    snapPoints={['10%', '90%']}
                >
                    <View style={styles.contentContainer}>
                        <View style={styles.minBottomSheet}>
                            <Text style={styles.minTitle}>
                                {nearbyPoi.poi.title}
                            </Text>
                            <Text style={styles.minSubTitle}>
                                {nearbyPoi.distanceFromUser} meters away
                            </Text>
                        </View>
                    <DetailsPage data={nearbyPoi.poi}
distance={nearbyPoi.distanceFromUser} />
                    </View>
                </BottomSheet>
            }
        </View>
    )
}


const styles = StyleSheet.create({
    scrollContainer: {
        flexGrow: 1,
```

```
        justifyContent: 'space-between'
    },
    title: {
        fontSize: 24,
        padding: 10,
        fontWeight: '500'
    },
    titleTip:{
        fontSize: 20,
        padding: 10,
    },
    divider: {
        margin: 10,
    },
    mapWrapper: {
        flex: 1,
        width: '100%',
        justifyContent: 'flex-start',
        alignItems: 'center',
    },
    map: {
        ...StyleSheet.absoluteFillObject,
        height: '100%'
    },
    navButton:{
        margin: 10
    },
    contentContainer:{
        margin: 10
    },
    minBottomSheet: {
        // flex:1
    },
    minTitle: {
        fontSize: 20
    },
    minSubTitle:{
        fontSize: 17
    },
    minImageContainer:{
        width: '100%',
        height: 150,
        borderRadius: 10,
        flex: 1,
        resizeMode: 'cover'
    },
    buttonEnd: {
        width: '100%',
```

```
    },
})

export default NavigationRoutePage;
```

pages>RoutePage.js

```
import React, {useState} from 'react';
import { FlatList, SafeAreaView, ScrollView, StyleSheet, Text, View} from
'react-native';
import FluidCard from '../components/CardItem/FluidCard';
import {Button, Divider} from 'react-native-elements';
import MapView, {Marker, PROVIDER_GOOGLE} from 'react-native-maps';
import MapViewDirections from 'react-native-maps-directions';
import {GOOGLE_API_KEY} from '../constants/enviroment';
import getDirections from 'react-native-google-maps-directions'
import {activeNavigationRouteSate} from
'../state/atoms/ActiveNavigationRoute';
import {useRecoilState} from 'recoil';
import {imagePoiUrl} from '../constants/enviroment';

const renderMap = (routePoisData) => {
    return routePoisData.map(routePoi=> {
        return <Marker
            key={routePoi.poi.id}
            coordinate={{
                latitude: Number(routePoi.poi.latitude),
                longitude: Number(routePoi.poi.longitude)
            }}
            title={routePoi.poi.title}
        />
    })
}

const RoutePage = ({navigation, route}) => {
    const [activeNavigationRoute, setActiveNavigationRoute] =
useRecoilState(activeNavigationRouteSate);

    const [duration, setDuration] = useState(0);
    const [distance, setDistance] = useState(0);
    const data = route.params.data;
    const routePoisData = data['routePoi'];
    const origin = routePoisData[0]['poi'];
    const destination = routePoisData[routePoisData.length-1]['poi'];

    const waypoints = routePoisData.map((poiRoute)=>{
        return {
            latitude: Number(poiRoute.poi.latitude),
            longitude: Number(poiRoute.poi.longitude)
        }
```

```
    })

    const totalDuration = routePoisData.reduce((accumulator, currentValue) =>
accumulator + currentValue.poi.visitTime , 0) + duration;
    const openDetails =  (item) => navigation.navigate("Details",{data: item})

    const startRoute = () => {
        if(!activeNavigationRoute.isActive) {
            setActiveNavigationRoute({
                isActive: true,
                data: data
            })
            setTimeout(()=> navigation.navigate('ActiveRoute', { screen:
'NavigationRoute' }), 0)
        }
    }

    const startNavigation = () => {
        const data = {
            source: {
                latitude: Number(origin.latitude),
                longitude: Number(origin.longitude)
            },
            destination: {
                latitude: Number(destination.latitude),
                longitude: Number(destination.longitude)
            },
            params: [
                {
                    key: "travelmode",
                    value: "driving"        // may be "walking", "bicycling"
or "transit" as well
                },
                {
                    key: "dir_action",
                    value: "navigate"       // this instantly initializes
navigation using the given travel mode
                }
            ],
            waypoints: [
                ...waypoints
            ]
        }
        startRoute();
        getDirections(data); //navigation to nav app
    }
```

```
    const handleStartRoute = () => startRoute();
    const handleNavigationAndRoute = () => {
        startNavigation();
    }


    return (
            <SafeAreaView style={styles.scrollContainer}>
                {/*<View>*/}
                    <FlatList
                        ListHeaderComponent={
                            <>
                                <Text style={styles.title}>{data.title}</Text>
                                <Text style={styles.titleTip}>Trip duration:
{Math.round(totalDuration)} min</Text>
                                <Text style={styles.titleTip}>Trip distance:
{Math.round(distance)} km</Text>
                                <Button style={styles.navButton} title="Start
Navigation and Route" onPress={()=> handleNavigationAndRoute()} />
                                <Button style={styles.navButton} title="Start
Route" onPress={handleStartRoute} />
                                <Divider style={styles.divider}
orientation="horizontal" />
                            </>
                        }
                        data={routePoisData}
                        keyExtractor={(item) => item.id.toString()}
                        renderItem={({item}) => <FluidCard item={item['poi']}
open={openDetails} imagebaseUrl={imagePoiUrl}/>}
                        nestedScrollEnabled
                        ListFooterComponent={
                            <>
                                <View style={styles.mapWrapper}>
                                    <MapView
                                        provider={PROVIDER_GOOGLE}
                                        style={styles.map}
                                        initialRegion={{
                                            latitude: Number(origin.latitude),
                                            longitude:
Number(origin.longitude),
                                            latitudeDelta: 0.0622,
                                            longitudeDelta: 0.0121,
                                        }}
                                        showsUserLocation={true}
                                        showsMyLocationButton={true}
                                    >
                                        <MapViewDirections
```

```
                                        origin={{latitude:
Number(origin.latitude), longitude: Number(origin.longitude)}}
                                        destination={{latitude:
Number(destination.latitude), longitude: Number(destination.longitude)}}
                                        waypoints={waypoints}
                                        strokeWidth={4}
                                        strokeColor="hotpink"
                                        apikey={GOOGLE_API_KEY}
                                        timePrecision={'now'}
                                        precision={'high'}
                                        onReady={result => {
                                            setDistance(result.distance);
                                            setDuration(result.duration);
                                            console.log(`Distance:
${distance} km`)
                                            console.log(`Duration:
${duration} min`)
                                        }}
                                    />
                                    {renderMap(routePoisData)}
                                </MapView>
                            </View>
                        </>
                    }
                />
                {/*</View>*/}
            </SafeAreaView>
    )
}

const styles = StyleSheet.create({
    scrollContainer: {
        flexGrow: 1,
        justifyContent: 'space-between'
    },
    title: {
        fontSize: 24,
        padding: 10,
        fontWeight: '500'
    },
    titleTip:{
        fontSize: 20,
        padding: 10,
    },
    divider: {
        margin: 10,
    },
    mapWrapper: {
```

```
    marginTop: 20,
        height: 400,
        width: '100%',
        justifyContent: 'flex-end',
        alignItems: 'center',
    },
    map: {
    ...StyleSheet.absoluteFillObject,
    },
    navButton:{
        margin: 10
    }
})

export default RoutePage;
```

services>NavigationRouteService.js

```
import { getDistance } from 'geolib';
import {USER_DISTANCE_FROM_POI} from '../constants/enviroment';

export const getNearbyPoi = (userLocation, pois) => {
    if(userLocation.latitude == 0 && userLocation.longitude == 0) return;

    const newPois = pois.map((poiRoute) => {
        const distance = getDistance(userLocation, poiRoute.poi);
       return {
           ...poiRoute,
           distanceFromUser: distance
       }
    });

        const filtered = newPois.filter(poiRoute => {
        if( poiRoute.distanceFromUser <= USER_DISTANCE_FROM_POI) return true;
        if( poiRoute.distanceFromUser > USER_DISTANCE_FROM_POI) return false;
        return false;
    }).sort((a, b) =>{
        console.log(a)
        if(a.distanceFromUser > b.distanceFromUser) return 1;
        if(a.distanceFromUser < b.distanceFromUser) return -1;
        return 0;
    })

    if(filtered.length === 0 ) return null;
    return filtered[0];

}
```

services>RoutesApi.js

```
import {routesApi} from '../constants/enviroment';

export const getNearbyPois = (lat,long) => {
    return fetch(`${routesApi}routes/nearbyRoutes/${lat}/${long}`)
                .then(response => response.json());
}

export const getOtherPlaces = (lat,long) => {
    return fetch(`${routesApi}routes/exploreCitiesNearYou/${lat}/${long}`)
                .then(response => response.json());
}

export const getAllNearbyPlaces = (lat,long) => {
    return fetch(`${routesApi}routes/allNearbyRoutes/${lat}/${long}`)
                .then(response => response.json());
}
```

state>atoms>ActiveNavigationRoute.js

```
import {atom} from "recoil"

export const ACTIVE_NAVIGATION_ROUTE_STATE = "activeNavigationRouteSate"
export const activeNavigationRouteSate = atom({
    key: ACTIVE_NAVIGATION_ROUTE_STATE,
    default: {
        isActive: false,
        data: null
    },
});
```

state>atoms>currentLovationState.js

```
import {atom} from "recoil"

export const CURRENT_LOCATION_STATE = "currentLocationState"
export const currentLocationState = atom({
    key: CURRENT_LOCATION_STATE,
    default: {
        loaded: false,
        latitude: null,
        longitude: null
    },
});
```

Κώδικας εφαρμογής Web Service

Ο κώδικας του web service: https://github.com/mariosp/tour-app

main.ts

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import {NestExpressApplication} from "@nestjs/platform-express";
import {ConfigService} from "@nestjs/config";
import { join } from 'path';



async function bootstrap() {
  const app = await NestFactory.create<NestExpressApplication>(AppModule);
  // app.connectMicroservice<MicroserviceOptions>(microserviceConfig);
  await app.startAllMicroservicesAsync();
  const configService: ConfigService = app.get('ConfigService');
  app.useStaticAssets(join(__dirname, '..', 'uploads'), {
    index: false,
    prefix: '/uploads',
  });
  await app.listen(configService.get("SERVER_PORT"), ()=>
console.log("MICROSERVICE ROUTES-SERVICE IS LISTENING..."));
}
bootstrap();
```

app.module.ts

```
import {Module} from '@nestjs/common';
import DatabaseConfig from "./config/database.config";
import MapsConfig from "./config/maps.config";
import {ConfigModule, ConfigService} from "@nestjs/config";
import {TypeOrmModule} from "@nestjs/typeorm";
import { RoutesModule } from './routes/routes.module';
import {Route} from "./routes/entities/route.entity";
import {Image} from "./image/image.entity";
import {ImageModule} from "./image/image.module";
import {ServeStaticModule} from "@nestjs/serve-static";
import { join } from 'path';
import { PoisModule } from './pois/pois.module';
import {Poi} from "./pois/entities/poi.entity";
import {RoutePoi} from "./routes/entities/route-poi.entity";
```

```
import { MapsModule } from './maps/maps.module';
import {City} from "./maps/entities/city.entity";

@Module({
  imports: [
    ConfigModule.forRoot({
      isGlobal: true,
      cache: true,
      load: [DatabaseConfig, MapsConfig],
      envFilePath: `./environment/${process.env.NODE_ENV}.env`
    }),
      TypeOrmModule.forRootAsync({
        imports:[ConfigModule],
        useFactory: async (configService: ConfigService) => ({
          type: "mysql",
          username: configService.get("DB_USER"),
          password: configService.get("DB_PASSWORD"),
          database: configService.get("DB_NAME"),
          host: configService.get("DB_HOST"),
          port: configService.get("DB_PORT"),
          entities:[Route, Image, Poi, RoutePoi, City] //__dirname +
'/**/*.entity{.ts,.js}'
        }),
        inject: [ConfigService]
      }),
      ServeStaticModule.forRoot({
        rootPath: join(__dirname, '..', 'uploads'),
      }),
      RoutesModule,
      ImageModule,
      PoisModule,
      MapsModule,
  ]
})
export class AppModule {}
```

config>database.config.ts

```
import {registerAs} from "@nestjs/config";

export const databaseConfig = {
        username: process.env.DB_USER || '',
        password: process.env.DB_PASSWORD || '',
        database: process.env.DB_NAME || '',
        host: process.env.DB_HOST || '127.0.0.1',
        port: Number(process.env.DB_PORT) || 3303,
        type: 'mysql',
        logging: false,
        timezone: process.env.DB_TIMEZONE
```

```
}

export default registerAs('database', () => ({
        ...databaseConfig
}));
```

config>maps.config.ts

```
import {registerAs} from "@nestjs/config";

export default registerAs('maps', () => ({
    apiKey: process.env.GOOGLE_MAPS_API
}));
```

image>image.entity.ts

```
import {BaseEntity, Column, Entity, JoinColumn, OneToOne,
PrimaryGeneratedColumn} from "typeorm";
import {Poi} from "../pois/entities/poi.entity";

@Entity({
    name: "images"
})
export class Image{

    @PrimaryGeneratedColumn()
    id: number;

    @Column({
        name: "image_file",
        nullable: false
    })
    imageFile: string;

    @Column({
        name: "type",
        nullable: false
    })
    type: string;

    @Column({
        name: "original_name",
        nullable: false
    })
    originalName: string;

    @Column({
        name: "created_at",
        nullable: false
```

```
    })
    createdAt: Date;

    @Column({
        name: "updated_at"
    })
    updatedAt: Date;

   // @OneToOne(() => Poi, poi=> poi.image)
   //  poi: Poi

}
```

image>image.module.ts

```
import { Module } from '@nestjs/common';
import { ImageService } from './image.service';
import {TypeOrmModule} from "@nestjs/typeorm";
import {ImageRepository} from "./image.repository";

@Module({
  providers: [ImageService],
  imports: [TypeOrmModule.forFeature([ImageRepository])],
  exports: [ImageService]
})
export class ImageModule {}
```

image>image.repository.ts

```
import {EntityRepository, Repository} from "typeorm";
import {Image} from "./image.entity";

@EntityRepository(Image)
export class ImageRepository extends Repository<Image> {

    public createImage(newImage: Image) : Promise<Image>  {
        return this.save(newImage);
    }
}
```

image>image.service.ts

```
import {Injectable, UploadedFile, UseInterceptors} from '@nestjs/common';
import {Image} from "./image.entity";
import {ImageRepository} from "./image.repository";

@Injectable()
export class ImageService {

    constructor(
        private readonly imageRepository: ImageRepository
```

```
    ) {
    }

    createImageReference(file: Express.Multer.File) : Promise<Image> {
       const  newImage: Image = new Image();
       newImage.imageFile = file.filename;
       newImage.type = file.mimetype;
       newImage.originalName = file.originalname;
       return this.imageRepository.createImage(newImage);
    }
}
```

maps>cities.service.ts

```
import { Injectable } from '@nestjs/common';
import {GeocodingService} from "./geocoding.service";
import {InjectRepository} from "@nestjs/typeorm";
import {City} from "./entities/city.entity";
import {Repository} from "typeorm";

@Injectable()
export class CitiesService {
    constructor(
        @InjectRepository(City) private readonly citiesRepository:
Repository<City>
    ) {
    }

    findByCityName(cityName: string) {
        return this.citiesRepository.findOne({ cityName: cityName});
    }

    createNewCity(city:City) {
        return this.citiesRepository.save(city);
    }
}
```

maps>geocoding.service.ts

```
import { Injectable } from '@nestjs/common';
import {Client, DistanceMatrixRequest, ReverseGeocodeResponse} from
"@googlemaps/google-maps-services-js/dist/client";
import {ConfigService} from "@nestjs/config";

@Injectable()
export class GeocodingService {
    apiKey:string;
    constructor( configService: ConfigService) {
        this.apiKey = configService.get("maps").apiKey;
    }
```

```
    async getReverseCityNameByCoordinates(lat, long): Promise<any> {
       const client = new Client({});
       const repsonse: any = await client.reverseGeocode({
            params: {
                key: this.apiKey,
                latlng: [lat, long]
            }
        }).catch(erro=> console.log(erro));

       for(const resultComponent of repsonse.data.results) {
           for(const component of resultComponent.address_components){
               if(component.types.includes("administrative_area_level_2")){
                   return component.long_name;
               }
           }
           //
if(resultComponent.types.includes("administrative_area_level_2")){
           //  return resultComponent.formatted_address;
           // }
       }

       return null;

    }


    async getDistance(source, destination) {

    }


    getDistanceFromLatLonInKm(lat1, lon1, lat2, lon2): number {
        var R = 6371; // Radius of the earth in km
        var dLat = this.deg2rad(lat2-lat1);  // deg2rad below
        var dLon = this.deg2rad(lon2-lon1);
        var a =
          Math.sin(dLat/2) * Math.sin(dLat/2) +
          Math.cos(this.deg2rad(lat1)) * Math.cos(this.deg2rad(lat2)) *
          Math.sin(dLon/2) * Math.sin(dLon/2)
          ;
        var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
        var d = R * c; // Distance in km
        return d;
      }

    private deg2rad(deg) {
        return deg * (Math.PI/180)
```

```
    }
}
```

maps>maps.controller.ts

```
import {Controller, Get, Param} from '@nestjs/common';
import { MapsService } from './maps.service';
import {GeocodingService} from "./geocoding.service";

@Controller('maps')
export class MapsController {
  constructor(private readonly mapsService: MapsService,
              private readonly geocodingService: GeocodingService) {}

  @Get("getCity/:lat/:long")
  async getCityByCoordinates(@Param("lat") lat: number, @Param("long") long:
number) {
    return {cityName: await
this.geocodingService.getReverseCityNameByCoordinates(lat, long)}
  }
}
```

maps>maps.module.ts

```
import { Module } from '@nestjs/common';
import { MapsService } from './maps.service';
import { MapsController } from './maps.controller';
import {GeocodingService} from "./geocoding.service";
import {CitiesService} from "./cities.service";
import {TypeOrmModule} from "@nestjs/typeorm";
import {City} from "./entities/city.entity";

@Module({
  controllers: [MapsController],
  imports: [
    TypeOrmModule.forFeature([City]),
  ],
  providers: [MapsService, GeocodingService, CitiesService],
  exports: [MapsService, GeocodingService, CitiesService]
})
export class MapsModule {}
```

maps>maps.service.ts

```
import { Injectable } from '@nestjs/common';
import {GeocodingService} from "./geocoding.service";
import {CitiesService} from "./cities.service";
import {City} from "./entities/city.entity";

@Injectable()
export class MapsService {
```

```
    constructor(
        private readonly geocodingService: GeocodingService,
        private readonly citiesService: CitiesService
    ) {
    }

    async findOrCreateNewCity(lat: number, long: number) {
        const cityName = await
this.geocodingService.getReverseCityNameByCoordinates(lat,long);
        if(cityName) {
            const cityEntity: City = await
this.citiesService.findByCityName(cityName);
            if(cityEntity) return cityEntity;
            const newCity = new City()
            newCity.cityName = cityName;
            return this.citiesService.createNewCity(newCity);
        } else {
            throw Error("Error api: cannot find city name with these
coordinates");
        }
    }

    // async getUserCityName(lat: number, long: number) {
    //    const {cityName} =  await
this.geocodingService.getReverseCityNameByCoordinates(lat,long);
    //     if(cityName) {
    //         await this.citiesService.findByCityName(cityName);
    //     } else {
    //         throw Error();
    //     }
    // }

}
```

maps>entities>city.entity.ts

```
import {Column, Entity, PrimaryGeneratedColumn} from "typeorm";

@Entity({name: "cities"})
export class City {
    @PrimaryGeneratedColumn()
    id: number;

    @Column({
        name: "city_name",
        nullable: false
    })
    cityName: string;
}
```

maps>entities>map.entity.ts

```
import {Column, Entity, PrimaryGeneratedColumn} from "typeorm";

@Entity({name: "maps"})
export class Map {
    @PrimaryGeneratedColumn()
    id: number;

    @Column({
        name: "enti",
        nullable: false
    })
    cityName: string;
}
```

pois>pois.controller.ts

```
import {
  Controller,
  Get,
  Post,
  Body,
  Patch,
  Param,
  Delete,
  UseInterceptors,
  UploadedFile,
  UploadedFiles
} from '@nestjs/common';
import { PoisService } from './pois.service';
import { CreatePoisDto } from './dto/create-pois.dto';
import { UpdatePoisDto } from './dto/update-pois.dto';
import {AnyFilesInterceptor, FileFieldsInterceptor, FileInterceptor,
FilesInterceptor} from "@nestjs/platform-express";
import {Image} from "../image/image.entity";
import {ImageService} from "../image/image.service";

@Controller('pois')
export class PoisController {
  constructor(private readonly poisService: PoisService,
              private readonly imageService: ImageService) {}

  @Post()
  @UseInterceptors(
      FileFieldsInterceptor([{name: 'image', maxCount: 1 }, {name: 'object',
maxCount: 1}], {dest: "./uploads/poi_images"}),
  )
```

```
  async create(@UploadedFiles() files: { image?: Express.Multer.File[],
object?: Express.Multer.File[] }, @Body() createPoisDto: CreatePoisDto) {
    const image: Image = files.image && await
this.imageService.createImageReference(files.image[0]);
    const object: Image = files.object && await
this.imageService.createImageReference(files.object[0]);
    createPoisDto.image = image;
    createPoisDto.objectAr = object;
    return this.poisService.create(createPoisDto);
  }

  @Get(':id')
  findOne(@Param('id') id: string) {
    return this.poisService.findOne(+id);
  }

  @Delete(':id')
  remove(@Param('id') id: string) {
    return this.poisService.remove(+id);
  }
}
```

pois>pois.module.ts

```
import { Module } from '@nestjs/common';
import { PoisService } from './pois.service';
import { PoisController } from './pois.controller';
import {TypeOrmModule} from "@nestjs/typeorm";
import {ImageModule} from "../image/image.module";
import {PoisRepository} from "./pois.repository";
import {MapsModule} from "../maps/maps.module";

@Module({
  controllers: [PoisController],
  providers: [PoisService],
  imports: [
    TypeOrmModule.forFeature([PoisRepository]),
    ImageModule,
    MapsModule
  ],
  exports:[PoisService]
})
export class PoisModule {}
```

pois>pois.repository.ts

```
import {EntityRepository, Repository} from "typeorm";
import {Poi} from "./entities/poi.entity";

@EntityRepository(Poi)
```

```
export class PoisRepository extends Repository<Poi> {


    findById(id: number): Promise<Poi> {
        return this.findOneOrFail(id);
    }
    //
    // public findByEmail(email: string): Promise<User> {
    //     return this.createQueryBuilder()
    //         .select("user")
    //         .from(User, "user")
    //         .where("user.email = :email", { email })
    //         .getOneOrFail();
    // }
    //
    createPoi(newPoi: Poi) : Promise<Poi>  {
        return this.save(newPoi);
    }
}
```

pois>pois.service.ts

```
import { Injectable } from '@nestjs/common';
import { CreatePoisDto } from './dto/create-pois.dto';
import { UpdatePoisDto } from './dto/update-pois.dto';
import {Poi} from "./entities/poi.entity";
import {PoisRepository} from "./pois.repository";
import {InjectRepository} from "@nestjs/typeorm";
import {RoutePoi} from "../routes/entities/route-poi.entity";
import {City} from "../maps/entities/city.entity";
import {MapsService} from "../maps/maps.service";

@Injectable()
export class PoisService {

  constructor(
      private readonly poisRepository: PoisRepository,
      private readonly mapsService: MapsService
  ) {
  }

  async create(createPoisDto: CreatePoisDto) {
    let poi = new Poi();
    const cityEntity: City = await
this.mapsService.findOrCreateNewCity(createPoisDto.latitude,createPoisDto.long
itude);
    console.log(cityEntity);
    poi = {...poi, ...createPoisDto, city: cityEntity}
    return this.poisRepository.createPoi(poi);
```

```
  }

  findOne(id: number) {
    return this.poisRepository.findById(id);
  }

  remove(id: number) {
    return this.poisRepository.delete(id);
  }
}
```

pois>entities>poi.entity.ts

```
import {Column, Entity, JoinColumn, ManyToOne, OneToMany, OneToOne,
PrimaryGeneratedColumn} from "typeorm";
import {Image} from "../../image/image.entity";
import {RoutePoi} from "../../routes/entities/route-poi.entity";
import {City} from "../../maps/entities/city.entity";

@Entity({name: "pois"})
export class Poi {
    @PrimaryGeneratedColumn()
    id: number;

    @Column({
        name: "title",
        nullable: false
    })
    title: string;

    @Column({
        name: "description"
    })
    description: string;

    @Column({
        name: "latitude"
    })
    latitude: number;

    @Column({
        name: "longitude"
    })
    longitude: number;

    @Column({
        name: "visitTime"
    })
    visitTime: number;
```

```
    @OneToOne(() => Image, image=> image.imageFile, {eager: true, onDelete:
"CASCADE" })
    @JoinColumn({
        name: "image"
    })
    image: Image;

    @OneToOne(() => Image, image=> image.imageFile, {eager: true, onDelete:
"CASCADE" })
    @JoinColumn({
        name: "object_ar"
    })
    objectAr: Image;

    @ManyToOne(() => City, {eager: true})
    @JoinColumn({
        name: "city_id"
    })
    city: City;


    @OneToMany(() => RoutePoi, routePoi => routePoi.poi)
    routePoi: RoutePoi[];

}
```

routes>routes.controller.ts

```
import {Controller, Get, Post, Body, Patch, Param, Delete, UseInterceptors,
UploadedFile} from '@nestjs/common';
import { RoutesService } from './routes.service';
import { CreateRouteDto } from './dto/create-route.dto';
import { UpdateRouteDto } from './dto/update-route.dto';
import {FileInterceptor} from "@nestjs/platform-express";
import {ImageService} from "../image/image.service";
import {Image} from "../image/image.entity";
import {CreateRoutePoiDto} from "./dto/create-route-poi.dto";

@Controller('routes')
export class RoutesController {
  constructor(
      private readonly routesService: RoutesService,
      private readonly imageService: ImageService,
  ) {}

  @Post()
  @UseInterceptors(FileInterceptor("image", { dest: "./uploads/routes" }))
```

```
  async create(@UploadedFile() file : Express.Multer.File, @Body() data:
CreateRouteDto) {
    const image: Image = await this.imageService.createImageReference(file);
    data.image = image;
    return await this.routesService.create(data);
  }

  @Post("/createAssociation")
  async createAssociation(@Body() data: CreateRoutePoiDto) {
    return await this.routesService.createAssociation(data);
  }

  @Get(':id')
  findOne(@Param('id') id: string) {
    return this.routesService.findOne(+id);
  }

  @Get('nearbyRoutes/:lat/:long')
  findNearby(@Param('lat') lat: number, @Param('long') long: number,) {
    return this.routesService.findNearby(lat,long);
  }

  @Get('allNearbyRoutes/:lat/:long')
  allNearby(@Param('lat') lat: number, @Param('long') long: number,) {
    return this.routesService.allNearby(lat,long);
  }

  @Get('exploreCitiesNearYou/:lat/:long')
  findCitiesNear(@Param('lat') lat: number, @Param('long') long: number,) {
    return this.routesService.findRegionsNear(lat,long);
  }
}
```

routes>routes.inteface.ts

```
import {Image} from "../image/image.entity";

export interface RouteEntity {
    id: number;
    title: string;
    image: Image;
}
```

routes>routes.module.ts

```
import { Module } from '@nestjs/common';
import { RoutesService } from './routes.service';
import { RoutesController } from './routes.controller';
import {TypeOrmModule} from "@nestjs/typeorm";
import {RoutesRepository} from "./routes.repository";
```

```
import {ImageModule} from "../image/image.module";
import {PoisModule} from "../pois/pois.module";
import {RoutePoi} from "./entities/route-poi.entity";
import {MapsModule} from "../maps/maps.module";

@Module({
  controllers: [RoutesController],
  providers: [RoutesService],
  imports: [
      TypeOrmModule.forFeature([RoutesRepository, RoutePoi]),
      ImageModule,
      PoisModule,
      MapsModule
  ]
})
export class RoutesModule {}
```

routes>routes.repository.ts

```
import {EntityRepository, Repository} from "typeorm";
import {Route} from "./entities/route.entity";
import {RoutePoi} from "./entities/route-poi.entity";

@EntityRepository(Route)
export class RoutesRepository extends Repository<Route> {


    findById(id: number): Promise<Route> {
        return this.findOneOrFail(id);
    }
    //
    // public findByEmail(email: string): Promise<User> {
    //     return this.createQueryBuilder()
    //         .select("user")
    //         .from(User, "user")
    //         .where("user.email = :email", { email })
    //         .getOneOrFail();
    // }
    //
    createRoute(newRoute: Route) : Promise<Route>  {
        return this.save(newRoute);
    }

    createAssociation(newRoutePoi: RoutePoi) : Promise<RoutePoi>  {
        return this.save(newRoutePoi);
    }
}
```

routes>routes.service.ts

```
import {HttpException, Injectable} from '@nestjs/common';
import { CreateRouteDto } from './dto/create-route.dto';
import { UpdateRouteDto } from './dto/update-route.dto';
import {RoutesRepository} from "./routes.repository";
import {Route} from "./entities/route.entity";
import {CreateRoutePoiDto} from "./dto/create-route-poi.dto";
import {RoutePoi} from "./entities/route-poi.entity";
import {PoisService} from "../pois/pois.service";
import {InjectRepository} from "@nestjs/typeorm";
import {Repository} from "typeorm";
import {GeocodingService} from "../maps/geocoding.service";
import {CitiesService} from "../maps/cities.service";
import {City} from "../maps/entities/city.entity";
import {HttpErrorByCode} from "@nestjs/common/utils/http-error-by-code.util";

@Injectable()
export class RoutesService {

  constructor(private readonly routesRepository: RoutesRepository,
              private readonly poisService: PoisService,
              private readonly geocodingService: GeocodingService,
              private readonly citiesService: CitiesService,
              @InjectRepository(RoutePoi) private routePoiRepository:
Repository<RoutePoi>
              ) {
  }
  create(createRouteDto: CreateRouteDto):Promise<Route> {
    const newRoute: Route = new Route();
    newRoute.title= createRouteDto.title;
    newRoute.image= createRouteDto.image;
    return this.routesRepository.createRoute(newRoute)
  }

  async createAssociation(createRoutePoiDto: CreateRoutePoiDto):
Promise<RoutePoi> {
    const newRoutePoi: RoutePoi = new RoutePoi();
    newRoutePoi.route = await this.findOne(createRoutePoiDto.routeId);
    newRoutePoi.poi =  await this.poisService.findOne(createRoutePoiDto.poiId)
    newRoutePoi.visitOrder = createRoutePoiDto.visitOrder;
    return this.routePoiRepository.save(newRoutePoi)
  }

  findOne(id: number) {
    return this.routesRepository.findById(id);
  }

  async findNearby(lat, long) {
```

```
    const userCity = await
this.geocodingService.getReverseCityNameByCoordinates(lat, long);
    console.log(userCity)
    if(userCity) {
      const routesNearby: Route[] = await
this.findRoutesWithCityName(userCity);
      const closeRoutes = this.routesCloseToCurrentLocation(routesNearby, lat,
long)
      return closeRoutes ? closeRoutes : new HttpException("No routes found",
404);
    } else {
      return new HttpException("User city not found", 404);
    }
  }

  async allNearby(lat, long) {
    const userCity = await
this.geocodingService.getReverseCityNameByCoordinates(lat, long);
    console.log(userCity)
    if(userCity) {
      const routesNearby: Route[] = await
this.findRoutesWithCityName(userCity);
      return routesNearby ? routesNearby : new HttpException("No routes
found", 404);
    } else {
      return new HttpException("User city not found", 404);
    }
  }

  setFirstPoiImage(routes) {
    return routes.map(route => ({
      ...route,
      image: route.routePoi.poi.image.imageFile
    }))
  }


  async findRoutesWithCityName(cityName: string){
    const cityObject: City = await
this.citiesService.findByCityName(cityName);
    return cityObject && this.routesRepository
        .createQueryBuilder("route")
        .leftJoinAndSelect("route.image", "routeimage")
        .leftJoinAndSelect("route.routePoi", "rp")
        .leftJoinAndSelect("rp.poi", "poi")
        .leftJoinAndSelect("poi.image", "image")
        .leftJoinAndSelect("poi.objectAr", "objectAr")
        .where("poi.city_id = :id", { id: cityObject.id })
```

```
        .orderBy("rp.visitOrder")
        .getMany();
  }


  async findRegionsNear(lat, long) {
    const userCity = await
this.geocodingService.getReverseCityNameByCoordinates(lat, long);
    if(userCity) {
      const routesNearby: Route[] = await this.exploreOtherRegions(userCity);
      return routesNearby ? routesNearby : new HttpException("No routes
found", 400);
    } else {
      return new HttpException("User city not found", 400);
    }
  }

  async exploreOtherRegions(cityName: string){
    const cityObject: City = await
this.citiesService.findByCityName(cityName);
    return cityObject && this.routesRepository
        .createQueryBuilder("route")
        .leftJoinAndSelect("route.image", "routeimage")
        .leftJoinAndSelect("route.routePoi", "rp")
        .leftJoinAndSelect("rp.poi", "poi")
        .leftJoinAndSelect("poi.image", "image")
        .leftJoinAndSelect("poi.objectAr", "objectAr")
        .where("poi.city_id <> :id", { id: cityObject.id })
        .orderBy("rp.visitOrder")
        .getMany();
  }

  //show pois less than 10km
  private routesCloseToCurrentLocation(routesNearby: Route[], lat, long) {
    return routesNearby.filter(element => {
      return this.geocodingService.getDistanceFromLatLonInKm(lat, long,
element.routePoi[0].poi.latitude, element.routePoi[0].poi.longitude) <= 10;
    });
  }

}
```

routes>entities>route-poi.entity.ts

```
import {
    BaseEntity,
    Column,
    Entity,
    JoinColumn,
```

```
    ManyToOne,
    OneToMany,
    OneToOne,
    PrimaryColumn,
    PrimaryGeneratedColumn
} from "typeorm";
import {Poi} from "../../pois/entities/poi.entity";
import {Route} from "./route.entity";

@Entity({
    name: "route_pois"
})
export class RoutePoi{

    @PrimaryGeneratedColumn()
    id: number;

    @Column({name: "visit_order"})
    visitOrder: number;

    @ManyToOne(()=> Route, route=> route.routePoi)
    @JoinColumn({name: "route_id"})
    route: Route;

    @ManyToOne(()=> Poi, poi=> poi.routePoi, {eager:true})
    @JoinColumn({name: "poi_id"})
    poi: Poi;

}
```

routes>entities>route.entity.ts

```
import {RouteEntity} from "../routes.interface";
import {
    Column,
    Entity,
    JoinColumn, OneToMany,
    OneToOne,
    PrimaryGeneratedColumn
} from "typeorm";
import {Image} from "../../image/image.entity";
import {RoutePoi} from "./route-poi.entity";

@Entity({
    name: "routes"
})
export class Route implements RouteEntity {

    @PrimaryGeneratedColumn()
```

```
    id: number;

    @Column({
        name: "title",
        default: "Route"
    })
    title: string;

    @OneToOne(type => Image, {eager: true})
    @JoinColumn({name: 'image'})
    image: Image;

    @OneToMany(() => RoutePoi, routePoi => routePoi.route, {eager: true})
    routePoi: RoutePoi[];
}
```