ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΞΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ ΤΟΥΡΙΣΤΙΚΟΥ ΕΝΔΙΑΦΕΡΟΝΤΟΣ ΑΠΟ ΔΗΜΟΣΙΑ APIS (TOURIST-RELEVANT DATA EXTRACTION FROM PUBLIC APIS)

Πτυχιακή εργασία του

Κωνσταντίνου Μαντζαβέλα (4056)

Επιβλέπων: Δρ. Νικόλαος Πεταλίδης

ΣΕΡΡΕΣ, ΜΑΙΟΣ 2020

Υπεύθυνη δήλωση

Υπεύθυνη Δήλωση: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών του Διεθνούς Πανεπιστημίου της Ελλάδος.

Σύνοψη

Στόχος αυτής της πτυχιακής εργασίας ήταν η εξαγωγή κάποιων πληροφοριών από δημόσια APIs, οι οποίες θα ήταν ιδιαίτερα χρήσιμες για έναν τουρίστα.

Πιο συγκεκριμένα, αναπτύχθηκε ένας web server και μία εφαρμογή για κινητά Android η οποία μπορεί να χρησιμοποιηθεί από έναν τουρίστα στα ταξίδια του. Η εφαρμογή χρησιμοποιώντας τις πληροφορίες που εξήγαγε από κάποια δημόσια APIs έκανε διάφορες προτάσεις στον χρήστη για τις τοποθεσίες που θα μπορούσε να επισκεφθεί βάσει των ενδιαφερόντων του.

Παραδείγματος χάρη, η εφαρμογή προτείνει σε έναν χρήστη τις δημοφιλέστερες τοποθεσίες μιας πόλης βάσει του πλήθους των φωτογραφιών που έχουν τραβηχτεί σε κάθε τοποθεσία. Επίσης, ο χρήστης έχει τη δυνατότητα να αναζητήσει τοποθεσίες βάσει κατηγοριών (όπως αξιοθέατα) ή βάσει της δημοτικότητας των τοποθεσιών ανά εποχή του χρόνου.

Στην πτυχιακή εργασία αναλύονται οι λειτουργίες της εφαρμογής οι οποίες αναπτύχθηκαν καθώς και η ευέλικτη μεθοδολογία με την οποία οργανώθηκε και υλοποιήθηκε το project. Τέλος, αναφέρονται φυσικά και τα APIs τα οποία χρησιμοποιήθηκαν για την εξαγωγή πληροφοριών καθώς και τα συστήματα που βοήθησαν στην απρόσκοπτη ανάπτυξη του project (όπως το Jenkins για CI/CD).

Περιεχόμενα

Υπεύθυνη δήλωση **2**

Σύνοψη **3**

Ευχαριστίες **9**

Ορισμοί **10**

1 Εισαγωγή **11**

1.1 Εφαρμογές τουριστικού περιεχομένου 11

1.2 Κατηγορίες τουριστικών εφαρμογών 11

1.3 Διαδεδομένες εφαρμογές τουριστικού περιεχομένου 12

2 Ανοιχτά Δεδομένα **14**

2.1 Εισαγωγή 14

2.2 Παράδειγμα χρήσης Ανοιχτών Δεδομένων 15

2.3 Χρήση των Ανοιχτών Δεδομένων στο TripAssistant 15

3 Εισαγωγή στα REST APIs **17**

3.1 Ορισμός του API 17

3.2 Ορισμός των REST APIs 17

3.3 Τα δημόσια API 18

3.4 Πλεονεκτήματα των δημοσίων APIs 19

3.5 Μειονεκτήματα των δημοσίων APIs 20

3.6 Τα REST APIs που χρησιμοποιήθηκαν 20

3.6.1 Flickr API 20

3.6.2 Facebook Graph Places Search API 21

3.6.3 DataMuse API 22

4 Διαχείριση Έργου **23**

4.1 Αρχική ερεύνα 23

4.2 Καθορισμός Επαναλήψεων 24

4.3 Τα stories της εφαρμογής 24

4.4 Releases 25

4.5 Συνεχής ενσωμάτωση - Continuous Integration 27

5 Επισκόπηση της Εφαρμογής χρήστη **30**

5.1 Περιγραφή 30

5.2 Λειτουργίες εφαρμογής 30

5.2.1 Αναζήτηση τοποθεσιών βάσει κατηγορίας 30

5.2.2 Δημιουργία λογαριασμού χρήστη 32

5.2.3 Διαχείριση ταξιδιών 32

5.2.4 Οι Δέκα Δημοφιλέστερες τοποθεσίες 35

5.2.5 Δημοφιλείς τοποθεσίες ανά εποχή του χρόνου 36

5.2.6 Ενημέρωση χρήστη για τα επερχόμενα ταξίδια του 37

6 Αρχιτεκτονική της εφαρμογής **38**

6.1 Μοντέλο ανάπτυξης 38

6.1.1 Αρχικό μοντέλο ανάπτυξης 38

6.1.2 Εξέλιξη μοντέλου ανάπτυξης 39

6.2 Τεχνολογίες που χρησιμοποιήθηκαν 41

6.2.1 Java 41

6.2.2 Android 42

6.2.3 Spring Boot Framework 42

6.2.4 Hibernate Framework 42

6.2.5 Spring Security 43

6.2.6 PostgreSQL (Βάση Δεδομένων) 43

6.2.7 RabbitMQ Message Broker 43

6.2.8 Firebase Cloud Messaging 44

6.2.9 JUnit 44

6.3 Διάγραμμα Κλάσεων UML 44
6.3.1 Περιγραφή UML διαγράμματος κλάσεων 44
6.3.2 UML διάγραμμα του TripAssistant 46
6.4 Τεχνική ανάλυση των Λειτουργιών του TripAssistant 48
6.4.1 Εξαγωγή των δέκα δημοφιλέστερων τοποθεσιών 48
6.4.2 Σύστημα σύνδεσης/εγγραφής χρήστη και authentication . . . 49
6.4.3 Αναζήτηση τοποθεσιών βάσει κατηγορίας 49
6.4.4 Κατηγοριοποίηση τοποθεσιών 50
6.4.5 Προβολή δημοφιλών τοποθεσιών ανά εποχή του χρόνου . . . 51
6.4.6 Έναρξη ταξιδιού χρήστη 53
6.4.7 Αποστολή push notifications στον χρήστη 54

7 Πειράματα **56**
7.1 Έλεγχος αλγορίθμου εξαγωγής δέκα δημοφιλέστερων τοποθεσιών . . 56
7.1.1 Ανάλυση πλήθους γεω-αναγραφόμενων φωτογραφιών 57
7.1.2 Αξιολόγηση των αποτελεσμάτων 59
7.2 Έλεγχος προτεινόμενης διαδρομής στο χρήστη για τις τοποθεσίες ενός ταξιδιού 59
7.2.1 Επαλήθευση αποτελεσμάτων διαδρομής 60
7.3 Έλεγχος κατηγοριοποίησης τοποθεσιών 62

8 Συμπεράσματα **64**
8.1 Μελλοντική ανάπτυξη 64
8.1.1 Υποστήριξη περισσότερων πόλεων 64
8.1.2 Βελτίωση της λειτουργίας έναρξης ταξιδιού χρήστη 65
8.1.3 Δημιουργία chat για συνομιλία μεταξύ των χρηστών 65
8.1.4 Σύνδεση με το Wikipedia για λήψη πληροφοριών 66
8.2 Συμπεράσματα 66

Γλωσσάρι **68**

Κατάλογος πινάκων

1.1 Διαδεδομένες εφαρμογές τουριστικού περιεχομένου 13

7.1 Οι 10 δημοφιλέστερες τοποθεσίες της Θεσσαλονίκης 57

7.2 Οι 10 δημοφιλέστερες τοποθεσίες της Θεσσαλονίκης και το ποσοστό των φωτογραφιών που τραβήχτηκαν στην εκάστοτε φωτογραφία επί του συνόλου των φωτογραφιών που τραβήχτηκαν στην πόλη της Θεσσαλονίκης 58

7.3 Δεδομένα γεωγραφικών τοποθεσιών για την επαλήθευση των αποτελεσμάτων του αλγορίθμου πρότασης βέλτιστης διαδρομής 60

7.4 Αποτέλεσμα ανάλυσης βέλτιστης διαδρομής για τις δοσμένες γεωγραφικές συντεταγμένες 62

7.5 Επαλήθευση κατηγοριοποίησης τοποθεσιών 63

Κατάλογος διαγραμμάτων

5.1 Αναζήτηση τοποθεσιών σε ακτίνα 3 χιλιομέτρων για την κατηγορία της "Νυχτερινής Ζωής". 31
5.2 Αποτέλεσμα αναζήτησης τοποθεσιών. 32
5.3 Προτροπή του χρήστη σε σύνδεση με το λογαριασμό του. 33
5.4 Προβολή των μελλοντικών ταξιδιών 34
5.6 Προβολή της προτεινόμενης διαδρομής των τοποθεσιών ενός ταξιδιού. 34
5.5 Προβολή των επερχόμενων ταξιδιών που είναι προγραμματισμένα σε λιγότερο από μία εβδομάδα. 35
5.7 Προβολή των δημοφιλών τοποθεσιών της πόλης που βρίσκεται ο χρήστης, για την τρέχουσα εποχή του χρόνου. 36
5.8 Ειδοποίηση στην Android συσκευή του χρήστη, για ένα επερχόμενο ταξίδι 37

6.1 Αρχική αρχιτεκτονική ανάπτυξης του TripAssistant 39
6.2 Η βελτιωμένη έκδοση της αρχιτεκτονικής του TripAssistant 40
6.3 UML Component Diagram του συστήματος 41
6.4 Παράδειγμα UML Class 45
6.5 UML Διάγραμμα κλάσης της εφαρμογής TripAssistant - Τμήμα 1 . . 46
6.6 UML Διάγραμμα κλάσης της εφαρμογής TripAssistant - Τμήμα 2 . . 47
6.7 UML Διάγραμμα κλάσης της εφαρμογής TripAssistant - Τμήμα 3 . . 48
6.8 Αρχιτεκτονίκη ενός RPC call στο RabbitMQ 52

7.1 Προτεινόμενη διαδρομή μετά από έναρξη ταξιδιού 61
7.2 Βέλτιστη διαδρομή μεταξύ των τοποθεσιών 61

Ευχαριστίες

Θα ήθελα να εκφράσω την βαθιά και ειλικρινή μου ευγνωμοσύνη για τον επιβλέποντα της πτυχιακής εργασίας, Δρ Νικόλαο Πεταλίδη, Καθηγητή, Επιστημονικό Συνεργάτη που μου έδωσε την ευκαιρία να εργαστώ σε ένα τόσο ενδιαφέρον θέμα όπου απέκτησα τριβή για πρώτη φορά με την επιστημονική κοινότητα και ήρθα σε επαφή με τεχνολογίες και πρακτικές που χρησιμοποιούνται ευρέως στα επαγγελματικά περιβάλλοντα. Επίσης, θα ήθελα να τον ευχαριστήσω για την στήριξη και τη βοήθεια που μου παρείχε καθ' όλη τη διάρκεια ανάπτυξης της παρούσας πτυχιακής εργασίας. Η στήριξη του δεν ήταν μόνο στην παρούσα πτυχιακή εργασία, αλλά και κατά τη διάρκεια των σπουδών στα μαθήματα του στα οποία μας στήριζε, μας ενέπνευσε, μας συμβούλευε και μας καθοδηγούσε για θέματα ακαδημαϊκά αλλά και προγραμματιστικά. Ουσιαστικά μας προετοίμασε μέσω των μαθημάτων του για τα πρώτα επαγγελματικά βήματα που θα κάνουμε μετά από τις σπουδές μας. Για όλους αυτούς τους λόγους θα ήθελα να εκφράσω την ευγνωμοσύνη μου και τον θαυμασμό μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου που με στήριξε σε όλα τα στάδια της ζωής μου και έκανε θυσίες ώστε να μπορέσω να σπουδάσω και να κάνω τα πρώτα μου επαγγελματικά βήματα. Επίσης, δεν μπορώ να παραλείψω από τις ευχαριστίες τους κοντινούς μου ανθρώπους και φίλους οι οποίοι με τη σειρά τους με στήριζαν και με ωθούσαν να προοδεύω κατά την διάρκεια των σπουδών μου αλλά και μετέπειτα.

Ορισμοί

Ορισμοί εννοιών που μπορεί να είναι χρήσιμοι. Για παράδειγμα:

Web Service Λογισμικό που επιτρέπει την επικοινωνία δύο συστημάτων μέσω του Παγκόσμιου Ιστού.

XML Γλώσσα σήμανσης που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων

User Story Η απλή περιγραφή μιας λειτουργίας ενός λογισμικού

Product Backlog Μία ανάλυση των εργασιών που πρέπει να γίνουν σε μία ομάδα που εφαρμόζει Scrum.

Push Notification Τα μηνύματα που γίνονται “push” από έναν server ή εφαρμογή σε μία διεπαφή χρήστη.

Framework Λογισμικό που παρέχει μία γενικευμένη λειτουργικότητα που επεκτείνεται από κώδικα που γράφεται χρησιμοποιώντας αυτό.

Endpoint Μία διεπαφή που παρέχεται από ένα μέσο επικοινωνίας ή μία υπηρεσία.

Κεφάλαιο 1

Εισαγωγή

1.1 Εφαρμογές τουριστικού περιεχομένου

Η εξέλιξη των έξυπνων κινητών τηλεφώνων (smartphones) και η συνεχής ανάπτυξη εφαρμογών για αυτά, μεταξύ άλλων έχει διευκολύνει αρκετά τους τουρίστες στην οργάνωση των ταξιδιών τους. Επίσης, η ευκολία εύρεσης αυτών των πληροφοριών οπουδήποτε και οποιαδήποτε στιγμή έχει συμβάλλει στην ανέλιξη χρήσης τέτοιων τουριστικών εφαρμογών σε κινητές συσκευές. Άλλωστε, δεν είναι τυχαίο πως περίπου το 50 τοις εκατό των εφαρμογών τουριστικού περιεχομένου έχουν σχεδιαστεί για κινητές συσκευές. Έχει συμβάλλει σημαντικά στην διευκόλυνση οργάνωσης ενός ταξιδιού. Πλέον, ο ταξιδιώτης έχει στη διάθεσή του ένα μεγάλο αριθμό εφαρμογών στο αποθετήριο της κινητής συσκευής του, με τις οποίες μπορεί να βρει πληροφορίες και να οργανώσει το ταξίδι και τις δραστηριότητες που τον ενδιαφέρουν.

1.2 Κατηγορίες τουριστικών εφαρμογών

Υπάρχουν τέσσερις (4) βασικές κατηγορίες στις οποίες κατατάσσονται οι mobile τουριστικές εφαρμογές. Αυτές είναι οι :

- Online Booking
- Information Resource
- Location-Based Υπηρεσίες

- Trip Journal

Στην κατηγορία “Location-Based Υπηρεσιών” ανήκουν οι εφαρμογές που προσφέρουν στον χρήστη πληροφορίες βάσει της τρέχουσας τοποθεσίας του (πχ πλοήγηση, πληροφορίες για κοντινά νοσοκομεία). [14] Η εφαρμογή της παρούσας πτυχιακής εργασίας ανήκει στην κατηγορία “Location-Based Υπηρεσιών” καθώς προτείνει στον χρήστη διάφορα αξιοθέατα βάσει της τοποθεσίας του.

1.3 Διαδεδομένες εφαρμογές τουριστικού περιεχομένου

Παρομοίως με την εφαρμογή που αναπτύχθηκε για τις ανάγκες της παρούσας πτυχιακής, κυκλοφορούν αρκετές εμπορικές εφαρμογές τουριστικού περιεχομένου. Οι πιο διαδεδομένες εφαρμογές που ανήκουν σε μία ή περισσότερες από τις κατηγορίες που αναφέρθηκαν παραπάνω είναι οι [10]:

Εφαρμογή	Πλατφόρμες	Σύντομη Περιγραφή
Hopper	Android & iOS	Το Hopper συγκεντρώνει κάθε μέρα δισεκατομμύρια τιμές αεροπορικών εισιτηρίων και δωματίων ξενοδοχείων. Επίσης βάσει του ιστορικού του, προτείνει αν πρέπει ο χρήστης να προβεί άμεσα στην κράτηση ή να περιμένει να πέσει η τιμή.
Google Trips	Android & iOS	Το Google Trips δημιουργεί το πρόγραμμα για το ταξίδι (ώρες πτήσης), καθώς και παρέχει διάφορες πληροφορίες όπως την ισοτιμία συναλλάγματος, τηλέφωνα έκτακτης ανάγκης (νοσοκομεία, αστυνομία) καθώς και μέρη που υπάρχει πρόσβαση σε Wi-Fi.

Airbnb	Android & iOS	Το Airbnb παρέχει υπηρεσίες ενοικίασης καταλυμμάτων βάσει διαφόρων φίλτρων. Ο χρήστης επιλέγει ενοικιαζόμενα βάσει το είδος του καταλύματος (δωμάτιο, διαμέρισμα) καθώς και βάσει προσωπικών προτιμήσεων (πισίνα, πλυντήριο, φιλικό προς κατοικίδια).
Hipmunk	Android & iOS	Το Hipmunk προτείνει ταξίδια βάσει κατηγοριών όπως "ρομαντικά", "νυχτερινή ζωή", "απίστευτα νησιά" κλπ.
Rome2rio	Android & iOS	Ο χρήστης επιλέγει μία διεύθυνση, ένα αξιοθέατο ή μία πόλη και το Rome2rio προτείνει στο χρήστη καταλύματα και δραστηριότητες.
TripAdvisor	Android & iOS	Στο TripAdvisor υπάρχουν εκατοντάδες εκατομμύρια κριτικές από εστιατόρια, ξενοδοχεία ή ακόμα και μουσεία.

Πίνακας 1.1: Διαδεδομένες εφαρμογές τουριστικού περιεχομένου

Κεφάλαιο 2

Ανοιχτά Δεδομένα

2.1 Εισαγωγή

Ως ανοιχτά, ορίζονται τα δεδομένα που μπορούν ελεύθερα να χρησιμοποιηθούν, να επαναχρησιμοποιηθούν και να αναδιανεμηθούν από οποιονδήποτε – υπό τον όρο να γίνεται αναφορά στους δημιουργούς και να διατίθενται, με τη σειρά τους, υπό τους ίδιους όρους.

Τα σημαντικότερα στοιχεία των ανοιχτών δεδομένων είναι τα εξής [6]:

- **Διαθεσιμότητα και Προσβασιμότητα**: Τα δεδομένα πρέπει να είναι διαθέσιμα αυτούσια, να έχουν ένα λογικό κόστος αναπαραγωγής, και κατά προτίμηση να είναι διαθέσιμα για λήψη από το Διαδίκτυο. Επίσης, πρέπει να είναι διαθέσιμα σε κάποια πρακτική και παραμετροποιήσιμη μορφή.

- **Επαναχρησιμοποίηση και Αναδιανομή**: Τα δεδομένα θα πρέπει να είναι διαθέσιμα υπό όρους που επιτρέπουν την επαναχρησιμοποίηση και την αναδιανομή τους, συμπεριλαμβανομένης και της ανάμειξης με άλλα σύνολα δεδομένων.

- **Καθολική Συμμετοχή**: Καθένας πρέπει να μπορεί να χρησιμοποιήσει, να επαναχρησιμοποιήσει και να αναδιανείμει τα δεδομένα. Δεν πρέπει αυτά να υπόκεινται σε διακρίσεις με βάση τον τομέα δραστηριότητας ή τα πρόσωπα και τις ομάδες. Για παράδειγμα, περιορισμοί για «μη-εμπορική χρήση» ή περιορισμοί για χρήση μόνο για συγκεκριμένους σκοπούς (π.χ. μόνο στην εκπαίδευση) δεν είναι επιτρεπτοί.

2.2 Παράδειγμα χρήσης Ανοιχτών Δεδομένων

Παρακάτω παρατίθεται ένα παράδειγμα για να αναδειχθεί η χρησιμότητα των Ανοιχτών Δεδομένων. Ας πάρουμε το παράδειγμα ενός κρατικού προϋπολογισμού, ο οποίος αποτελείται από εκατοντάδες διαφορετικούς πίνακες, όλοι τους με μία μικρή διαφοροποίηση στη διάταξη. Για να γίνει μία ανάλυση των κονδυλίων σε διαφορετικές κατηγορίες από αυτές που επέλεξε ο εκδότης, οι πολίτες θα έπρεπε να μεταφέρουν χειροκίνητα όλα αυτά τα δεδομένα σε ένα νέο έγγραφο. Για έναν κρατικό προϋπολογισμό, αυτό θα μπορούσε να είναι αρκετά χρονοβόρο και να διαρκέσει πολλές εβδομάδες. Ακόμη και όταν ολοκληρωθεί η διαδικασία της μεταφοράς αυτών των δεδομένων, οι πολίτες μπορεί να ανακάλυπταν ότι τα δεδομένα προστατεύονται από πνευματικά δικαιώματα όπου απαγορεύεται η ευρεία χρήση αυτών των δεδομένων.

Αυτό το πρόβλημα καλούνται να λύσουν τα ανοιχτά δεδομένα. Μία ανάλυση προϋπολογισμού ενός κράτους, θα επέτρεπε στους πολίτες να χρησιμοποιήσουν αυτά τα δεδομένα με οποιοδήποτε τρόπο επιθυμούν. Οι προγραμματιστές θα μπορούσαν να χρησιμοποιήσουν αυτά τα δεδομένα ώστε να παρέχουν στους πολίτες εφαρμογές που θα τους ήταν περισσότερο χρήσιμες, αξιοποιώντας αυτά τα δεδομένα. Τέλος, οι δυνατότητες της χρήσης αυτών των δεδομένων περιορίζονται μόνο από την φαντασία της κοινότητας που θα τα χρησιμοποιήσει [4].

2.3 Χρήση των Ανοιχτών Δεδομένων στο TripAssistant

Στην εφαρμογή TripAssistant χρησιμοποιήθηκαν ανοιχτά δεδομένα για τις φωτογραφίες που φορτώθηκαν από το API του Flickr. Οι φωτογραφίες αυτές χρησιμοποιήθηκαν στην εφαρμογή για την εξαγωγή διαφόρων στατιστικών που ήταν χρήσιμα για τους χρήστες. Επίσης οι φωτογραφίες αυτές προβλήθηκαν σε κάποια σημεία της εφαρμογής.

Επειδή οι φωτογραφίες που φορτώθηκαν από το API του Flickr ανεβαίνουν όλες από χρήστες, ενδέχεται να υπόκεινται σε πνευματικά δικαιώματα. Η εφαρμογή Trip Assistant, φόρτωσε μέσω του API μόνο φωτογραφίες οι οποίες υπόκεινται στο Attribution License, δηλαδή είναι ανοιχτές προς αντιγραφή, διανομή, προβολή. Οι φωτογραφίες αυτές καθώς και οι άδειες διάθεσής τους περιγράφονται στην ιστοσελίδα `https://www.flickr.com/creativecommons`.

Κεφάλαιο 3

Εισαγωγή στα REST APIs

3.1 Ορισμός του API

Ένα μέρος της συγκεκριμένης πτυχιακής εργασίας είναι η άντληση δεδομένων από δημόσια APIs. Οπότε χρειάζεται να γίνει μία αναφορά στον ορισμό του API. Αρχικά, ένα API είναι ένα σύνολο από μεθόδους που ορίζουν την επικοινωνία μεταξύ συστημάτων. Τα APIs διατίθενται σε προγραμματιστές ώστε να τους διευκολύνουν στην ανάπτυξη προγραμμάτων διαφόρων σκοπών. Τέτοια APIs μπορεί να αφορούν κάποιο λειτουργικό σύστημα, σύστημα βάσεων δεδομένων, υλικό (hardware) υπολογιστών αλλά και κάποιο διαδικτυακό σύστημα.

Κάποια παραδείγματα APIs που χρησιμοποιούνται ευρέως είναι το Windows API και το POSIX. Συνήθως μαζί με το API διατίθεται ένα εγχειρίδιο που περιγράφει τα use cases του API αλλά και οδηγίες για την υλοποίησή του.

3.2 Ορισμός των REST APIs

Το Representational State Transfer (REST) είναι μία αρχή στην αρχιτεκτονική λογισμικού που καθορίζει ένα σύνολο από περιορισμούς που χρησιμοποιούνται κατά την ανάπτυξη Web Services. Τα REST Web Services είναι ένα υποσύνολο του Παγκόσμιου Ιστού (WWW) και είναι βασισμένα στο πρωτόκολλο HTTP.

Τα βασικά χαρακτηριστικά των REST APIs είναι τα εξής [5]:

- χρήση της **αρχιτεκτονικής Client-Server**: Η βασική ιδέα πίσω από την αρχιτεκτονική Client-Server είναι ο διαχωρισμός του προγράμματος σε επιμέρους

ξεχωριστά τμήματα. Διαχωρίζοντας το τμήμα της διεπαφή χρήστη από το τμήμα της αποθήκευσης των δεδομένων επιτυγχάνεται η φορητότητα του προγράμματος και η επεκτασιμότητα.

- **Stateless**: η επικοινωνία μεταξύ client και server πρέπει να είναι stateless, δηλαδή να μην κρατηθεί καμία πληροφορία για την κατάσταση του client. Κάθε request από τον client στον server πρέπει να περιέχει όλες τις απαραίτητες πληροφορίες ώστε ο server να κατανοήσει το request.

- **Cache**: Χρησιμοποιώντας cache στον client βελτιώνεται η απόδοση του δικτύου. Τα δεδομένα που επιστρέφει ο server σε μία απάντηση (response) πρέπει να χαρακτηρίζονται από τον client σαν cacheable ή non-cacheable. Εάν μία απάντηση είναι cacheable, τότε ο client μπορεί να επαναχρησιμοποιήσει αυτά τα δεδομένα αργότερα. Με αυτό τον τρόπο, μειώνεται ο φόρτος του δικτύου μαζί με τις αλληλεπιδράσεις μεταξύ client-server, όπως επίσης βελτιώνεται η εμπειρία χρήστη.

- **Ενιαία Διεπαφή (Uniform Interface)**: Το κύριο χαρακτηριστικό που ξεχωρίζει την αρχιτεκτονική REST από τις υπόλοιπες, είναι η έμφαση που δίνεται στην χρήση μιας ενιαίας διεπαφής μεταξύ των τμημάτων του λογισμικού. Εφαρμόζοντας αυτή την αρχή, η αρχιτεκτονική ενός συστήματος απλουστεύεται σημαντικά. Επίσης, οι εκάστοτε υλοποιήσεις διαχωρίζονται από τις υπηρεσίες που προσφέρουν. Η διεπαφή REST έχει σχεδιαστεί να είναι αποδοτική στη μεταφορά μεγάλων όγκων δεδομένων, κάτι που την καθιστά βέλτιστη λύση για την περίπτωση του Web.

- **Στρωματοποιημένο Σύστημα (Layered System)**: Σε αυτή την αρχιτεκτονική, το κάθε στρώμα αποτελείται από τμήματα (components) που δεν μπορούν να "δουν" πέρα από το στρώμα με το οποίο αλληλεπιδρούν άμεσα. Περιορίζοντας την γνώση του συστήματος σε ένα στρώμα, μπαίνει ένα όριο στην συνολική πολυπλοκότητα του συστήματος.

3.3 Τα δημόσια API

Τα δημόσια APIs (public APIs ή open APIs) είναι οι διεπαφές προγραμματισμού εφαρμογών (APIs) που διατίθενται δωρεάν προς χρήση σε προγραμματιστές. Τέτοια

APIs παρέχουν στους προγραμματιστές που τα χρησιμοποιούν πρόσβαση σε κάποια ιδιωτική εφαρμογή ή κάποιο web service. Τα APIs τα οποία διανέμουν web services υλοποιούνται στη μεγάλη τους πλειοψηφία σε SOAP ή REST. Η παρούσα πτυχιακή χρησιμοποιεί διάφορα REST public APIs.

Οι κύριες διαφορές των SOAP με τα REST web services είναι ότι τα SOAP παρόλο που είναι μια ευρέως διαδεδομένη και παλιά τεχνολογία, έχουν μεγαλύτερο φορτίο δεδομένων σε σχέση με το REST. Επίσης τα SOAP services επιστρέφουν αυστηρώς XML δεδομένα σε αντίθεση με τα REST που είναι πιο ευέλικτα ως προς τον τύπο των δεδομένων που θα επιστρέψουν. Επιπρόσθετα τα SOAP web services χρησιμοποιούν παραπάνω bandwidth καθώς μία απάντηση σε SOAP μπορεί να χρειαστεί μέχρι και 10 φορές περισσότερα byte συγκριτικά με το REST. Τέλος το SOAP είναι το πρότυπο που επικρατεί για τα web services οπότε και έχει καλύτερη υποστήριξη από άλλα πρότυπα (WSDL, WS) [16]

3.4 Πλεονεκτήματα των δημοσίων APIs

Το σημαντικότερο πλεονέκτημα των δημοσίων APIs είναι ότι διατίθενται δωρεάν χωρίς κάποιον περιορισμό στην χρήση τους. Μπορεί δηλαδή μία εταιρεία ή ένας προγραμματιστής να χρησιμοποιήσει ένα δημόσιο API στην εφαρμογή του και να ξεκινήσει την εμπορική διάθεση του λογισμικού του. Παράλληλα ο εκδότης του δημοσίου API κρατά υπό την δικαιοδοσία του τον πηγαίο κώδικα του API ενώ παράλληλα ωφελείται από την αύξηση της βάσεως των χρηστών του.

Παρόλα αυτά τα δημόσια APIs πρέπει να αντιμετωπίζονται από τους εκδότες τους ως τελικό προϊόν καθώς μπορούν να επηρεάσουν τη φήμη του οργανισμού είτε θετικά είτε αρνητικά. Ο οργανισμός που διαθέτει ένα δημόσιο API πρέπει να έχει διασφαλίσει ότι το API:

- Δεν περιέχει bugs
- Δεν έχει κενά ασφαλείας
- Δεν εκθέτει προσωπικά δεδομένα

3.5 Μειονεκτήματα των δημοσίων APIs

Το σημαντικότερο μειονέκτημα των δημοσίων API είναι το γεγονός πως ο οργανισμός που τα εκδίδει μπορεί ανά πάσα στιγμή να αλλάξει τους όρους χρήσης και να ορίσει κάποιο αντίτιμο για την χρήση του API. Σε αυτή την περίπτωση οι προγραμματιστές που το χρησιμοποιούν δεν έχουν άλλη επιλογή παρά να το αποδεχτούν.

3.6 Τα REST APIs που χρησιμοποιήθηκαν

Στην εφαρμογή TripAssistant χρησιμοποιήθηκαν κατά κύριο λόγο τρία δημόσια REST APIs. Σε όλες τις παρακάτω περιπτώσεις χρησιμοποιήθηκε το JSON ως media type.

3.6.1 Flickr API

Το API του flickr χρησιμοποιήθηκε για τον σκοπό της συγκέντρωσης γεωαναγραφόμενων φωτογραφιών από μία συγκεκριμένη πόλη. Συγκεντρώνονται όλες οι φωτογραφίες που έχουν ανέβει στο Flickr τα τελευταία 3 χρόνια και σώζονται σε μία βάση δεδομένων κρατώντας τις γεωγραφικές συντεταγμένες της φωτογραφίες, έναν σύνδεσμο της φωτογραφίας από το flickr μαζί με κάποιες ακόμα πληροφορίες.

Το κεντρικό URL που χρησιμοποιήθηκε για το API του flickr είναι το **https://api.flickr.com** Τα endpoints που χρησιμοποιήθηκαν είναι τα εξής:

- **/services/rest?method=flickr.photos.search&lat=40.23124&lon=21.12342 &has_geo=1&min_upload_date=1449867647**

  Μέθοδος: GET

  Επιστρέφει μία λίστα από φωτογραφίες που πληρούν κάποια κριτήρια. Συγκεκριμένα τα κριτήρια που έχουν τεθεί στο παραπάνω παράδειγμα είναι οι φωτογραφίες που έχουν γεωπληροφορίες, που βρίσκονται κοντά στις ζητούμενες γεωγραφικές συντεταγμένες και η ημερομηνία που μεταφορτώθηκαν είναι μεγαλύτερη της ζητούμενης (δίνεται epoch time φορμάτ).

  Documentation: `https://www.flickr.com/services/api/flickr.photos.search.html`

- **/services/rest?method=flickr.photos.getInfo&photo_id=12345 &secret=ab32cd1ef**

  Μέθοδος: GET

  Επιστρέφει περισσότερες πληροφορίες για την φωτογραφία με το id που ζητήθηκε.

  Documentation: `https://www.flickr.com/services/api/flickr.photos.getInfo.html`

3.6.2 Facebook Graph Places Search API

Το Places Search API του Facebook χρησιμοποιήθηκε για να διασταυρωθούν αλλά και να ενισχυθούν οι υπάρχουσες πληροφορίες τοποθεσιών. Για ένα ζευγάρι γεωγραφικών συντεταγμένων, καλείται το API του Facebook ώστε να ενισχυθούν οι υπάρχουσες πληροφορίες για μία τοποθεσία όπως η περιγραφή, οι κατηγορίες της τοποθεσίας, η επισκεψιμότητα της τοποθεσίας (check-ins κατά το facebook), καθώς και το όνομα της ευρύτερης περιοχής. Το endpoint που χρησιμοποιήθηκε είναι το εξής:

- **https://graph.facebook.com/v5.0/search?type=place &center=40.6152792,22.9607157&distance=50 &fields=id,name,about,description,checkins,location,category_list &access_token=123456789012345|5426dc78123ed241460393f694448fba**

  Μέθοδος: GET

  Επιστρέφει μία λίστα από τις τοποθεσίες που είναι κοντά στα 50 μέτρα από το κέντρο των γεωγραφικών συντεταγμένων που δόθηκαν. Η λίστα περιέχει τοποθεσίες οι οποίες έχουν πληροφορίες όπως όνομα, περιγραφή, αριθμός επισκέψεων (check-ins), τοποθεσία (Πόλη, Χώρα, γεωγραφικές συντεταγμένες) και κατηγορίες.

  Documentation: `https://developers.facebook.com/docs/places/web/search/`

3.6.3 DataMuse API

Το API του DataMuse είναι μία μηχανή εύρεσης λέξεων. Παρέχοντας μία λέξη, επιστρέφει μία λίστα από λέξεις που νοηματικά βρίσκονται κοντά ή είναι ίδιες. Επίσης, μπορούν να καθοριστούν διάφορες παράμετροι όπως ο συλλαβισμός, ο ήχος και το λεξιλόγιο. Το endpoint που χρησιμοποιήθηκε είναι το εξής:

- **https://api.datamuse.com/words?ml=word**

  Μέθοδος: GET

  Επιστρέφει μία λίστα από λέξεις με σημασία παρόμοια με την λέξη word

  Documentation: `https://www.datamuse.com/api/`

Κεφάλαιο 4

Διαχείριση Έργου

4.1 Αρχική ερεύνα

Για την καλύτερη οργάνωση και διαχείριση του συγκεκριμένου project χρησιμοποιήθηκαν κάποιες αρχές της ευέλικτης μεθοδολογίας Scrum.

Το ελάχιστο που χρειάζεται για να ξεκινήσει ένα Scrum project είναι να υπάρχει ένα όραμα και ένα Product Backlog. Συγκεκριμένα το όραμα περιγράφει το λόγο που αναλήφθηκε το project και ποιο είναι το επιθυμητό αποτέλεσμα [13].

Στην αρχή ενός έργου, οι προγραμματιστές και οι πελάτες συζητάνε για το νέο σύστημα με σκοπό να καθορίσουν όλες τις βασικές και σημαντικές λειτουργίες. Ωστόσο, δεν χρειάζεται να καθοριστούν όλες οι λειτουργίες της εφαρμογής. Καθώς το έργο προχωράει, οι πελάτες συνεχίζουν να ανακαλύπτουν καινούργιες λειτουργίες. Καθώς καθορίζεται μία λειτουργία, διασπάται σε ένα ή περισσότερα user stories τα οποία γράφονται το καθένα σε μία καρτέλα. Σε κάθε καρτέλα γράφεται μόνο το όνομα του story όπως για παράδειγμα “Δημιουργία Χρήστη”. Δεν χρειάζονται λεπτομέρειες πάνω στην καρτέλα, παρά μόνο μια υπενθύμιση των λειτουργιών.

Έπειτα, οι προγραμματιστές κάνουν μία εκτίμηση των user stories. Ένας αριθμός “points” (πόντων) γράφεται πάνω σε κάθε καρτέλα story ο οποίος αναπαριστά το σχετικό κόστος του story. Μπορεί να μην είναι ακόμα γνωστό τι χρονική διάρκεια αναπαριστά κάθε story point, αλλά είναι γνωστό ότι ένα story με 8 points (πόντους) θα διαρκέσει 2 φορές παραπάνω από ένα story με 4 points. Όλα αυτά τα stories, στο τέλος τοποθετούνται στο backlog του project μαζί και με τα μελλοντικά stories [9, Chapter 3].

4.2 Καθορισμός Επαναλήψεων

Αφού έχουν καθοριστεί οι βασικές λειτουργίες και τα stories, επόμενο βήμα είναι να καθοριστεί το μέγεθος των επαναλήψεων που θα ακολουθηθεί. Συνήθως, αυτός ο αριθμός είναι 1 ή 2 εβδομάδες, όπου οι πελάτες και οι προγραμματιστές καθορίζουν τα stories που θα μπουν σε κάθε επανάληψη. Σημαντικό είναι σε κάθε επανάληψη να μην επιλεχθούν παραπάνω stories από όσα είναι δυνατόν να έρθουν σε πέρας, σύμφωνα με το velocity της ομάδας. Επίσης οι πελάτες δεν έχουν τη δυνατότητα να αλλάξουν τα stories μιας επανάληψης αφού έχει ήδη ξεκινήσει. Είναι ελεύθεροι να αλλάξουν ή αναδιατάξουν οποιοδήποτε άλλο story μέσα στο project, εκτός από αυτά που έχουν ξεκινήσει να δουλεύουν ήδη οι προγραμματιστές. Επιπλέον, η επανάληψη τελειώνει στην προκαθορισμένη ημερομηνία, ακόμη και αν δεν έχουν ολοκληρωθεί όλα τα stories. Στο τέλος της επανάληψης, επανακαθορίζεται το velocity της ομάδας αθροίζοντας τα story points που ολοκληρώθηκαν. Για παράδειγμα, εαν αυτός ο αριθμός είναι 31 τότε το velocity της ομάδας είναι 31 και στην επόμενη επανάληψη θα πρέπει να προστεθούν stories με 31 story points στο σύνολο. [9, Chapter 3]

4.3 Τα stories της εφαρμογής

Όπως περιγράφηκε προηγουμένως, κατά το ξεκίνημα του project καθορίστηκαν οι βασικές λειτουργίες της εφαρμογής οι οποίες αντιστοιχήθηκαν σε ένα ή περισσότερα stories η καθεμιά. Τα stories που υπήρχαν στο backlog είναι τα παρακάτω.

User Stories:

User Story 1 - Σαν χρήστης θέλω να δω τις 10 δημοφιλέστερες τοποθεσίες τριγύρω μου

User Story 2 - Σαν χρήστης θέλω να δημιουργήσω έναν λογαριασμό χρήστη και να συνδεθώ με αυτόν

User Story 3 - Σαν χρήστης θέλω προτάσεις από τοποθεσίες σε μία ακτίνα τριγύρω μου, βάσει κατηγορίας

User Story 4 - Σαν χρήστης θέλω να προσθέσω τοποθεσίες σε ένα ταξίδι και να το προγραμματίσω για το μέλλον

User Story 5 - Σαν χρήστης θέλω να γράφω αξιολόγηση για μία τοποθεσία που επισκέφθηκα αλλά και να διαβάζω τις αξιολογήσεις των υπόλοιπων χρηστών

User Story 6 - Σαν χρήστης θέλω να έχω τη δυνατότητα να συνομιλήσω μέσω chat με άλλους χρήστες στην ίδια πόλη που βρίσκομαι ώστε να μου προτείνουν μέρη να επισκεφθώ

User Story 7 - Σαν χρήστης θέλω να μου προτείνει η εφαρμογή την βέλτιστη διαδρομή να ακολουθήσω από μία λίστα επιλεγμένων τοποθεσιών

User Story 8 - Σαν χρήστης θέλω να δω τις δημοφιλείς τοποθεσίες για την τρέχουσα εποχή του χρόνου κοντά μου

User Story 9 - Σαν χρήστης θέλω να λάβω ειδοποίηση (push notification) στο κινητό μου για να μου υπενθυμίσει ένα ταξίδι που πλησιάζει

Αξίζει να σημειωθεί πως από τα παραπάνω stories δεν υλοποιήθηκαν όλα, ωστόσο σε επόμενο κεφάλαιο θα αναλυθούν τα user stories και οι λειτουργίες που υλοποιήθηκαν. Επιπλέον, τα περισσότερα από αυτά είχαν καθοριστεί στην αρχή του project όπου και τοποθετήθηκαν στο backlog. Στη συνέχεια, όταν καταγραφόταν ένα καινούργιο user story που αφορούσε την υλοποίηση μίας νέας λειτουργίας, τότε προστίθετο στο backlog με τη σειρά του.

4.4 Releases

Στον μέχρι τώρα κύκλο ζωής του project, κυκλοφόρησαν 6 releases. Όλα τα releases φιλοξενήθηκαν στο αποθετήριο της εφαρμογής που φιλοξενείται στον ιστότοπο του Github στον σύνδεσμο `https://github.com/kostasmantz/trip-assistant/releases`. Ο πηγαίος κώδικας της εφαρμογής διατίθεται υπό την άδεια ανοιχτού λογισμικού Apache License 2.0 . Τα releases και οι λειτουργίες που περιλαμβάνουν πε-

ριγράφονται παρακάτω.

TripAssistant 0.1.0

1. Εξαγωγή των 10 δημοφιλέστερων τοποθεσιών μιας πόλης, απ'το σύνολο των φωτογραφιών που έχουν ανέβει εκεί.
2. Υλοποίηση ενός web service που παρέχει τις 10 δημοφιλέστερες τοποθεσίες μιας πόλης, βάσει των γεωγραφικών συντεταγμένων του χρήστη.

TripAssistant 0.2.0

1. Εγγραφή και σύνδεση των χρηστών στην εφαρμογή.
2. Εύρεση προτάσεων από τοποθεσίες κοντά στον χρήστη βάσει κατηγορίας τοποθεσίας.
3. Χρήση του προσθέτου Jacoco για την παραγωγή αναφορών Unit και Integration tests.

TripAssistant 0.3.0

1. Δημιουργία και προγραμματισμός ενός ταξιδιού για το μέλλον. Τοποθέτηση τοποθεσιών στο μελλοντικό ταξίδι.

TripAssistant 0.4.0

1. Υλοποίηση ειδοποιήσεων push (push notifications) για προγραμματισμένα ταξίδια που πλησιάζουν.
2. Διόρθωση διαφόρων bugs.
3. Μικρές βελτιώσεις στην σχεδίαση.

TripAssistant 0.5.0

1. Εξαγωγή και πρόταση δημοφιλών τοποθεσιών για την τρέχουσα εποχή του χρόνου.
2. Διόρθωση διαφόρων bugs.
3. Βελτιώσεις στον κώδικα.

TripAssistant 0.6.0

1. Χρήση της αναβαθμισμένης έκδοσης του Facebook API για αναζήτηση τοποθεσιών.
2. Ενημέρωση του README αρχείου του project.
3. Χρήση του SonarQube με σκοπό την ανάλυση του πηγαίου κώδικα.
4. Διόρθωση bugs.

Η λογική της δημιουργίας των παραπάνω εκδόσεων ακολουθεί την στρατηγική του Semantic Versioning. Επιλέχθηκε η συγκεκριμένη στρατηγική μεταξύ άλλων, καθώς είναι αρκετά ξεκάθαρη και απλή η εφαρμογή της. Το Semantic Versioning, χρησιμοποιεί μία ακολουθία από τρία ψηφία Major.Minor.Patch (Μείζων.Ελάσσον.Μπάλωμα). Σε αυτή τη στρατηγική ισχύουν οι εξής απλοί κανόνες [11].

1. Η Major έκδοση, αυξάνεται όταν γίνονται ασύμβατες αλλαγές στο API.
2. Η Minor έκδοση, αυξάνεται όταν προστίθενται νέες λειτουργίες με τρόπο οπισθοδρομικά συμβατό (backwards compatible).
3. Η Patch έκδοση αυξάνεται όταν διορθώνονται σφάλματα με τρόπο οπισθοδρομικά συμβατό.

Σε όλες τις εκδόσεις που κυκλοφόρησαν, προσαυξήθηκε μονάχα η minor version καθώς δεν έγινε κάποια αλλαγή η οποία δημιούργησε ασυμβατότητα με τις προηγούμενες εκδόσεις. Οι λειτουργίες που προσθέτονταν ήταν πάντα συμβατές με τις προηγούμενες εκδόσεις και δεν κυκλοφόρησε κάποια έκδοση η οποία αποκλειστικά διόρθωνε σφάλματα. Για τον λόγο αυτό, η πρώτη έκδοση ήταν η 0.1.0 και η τελευταία μέχρι στιγμής έκδοση η 0.6.0

4.5 Συνεχής ενσωμάτωση - Continuous Integration

Η τεχνική της συνεχούς ενσωμάτωσης (continuous integration) συνδέει τακτικά τον πηγαίο κώδικα ενός συστήματος, Επίσης, με το continuous integration διασφαλίζεται μέσω της σουίτας regression tests ότι οι καινούργιες αλλαγές που ενσωματώνο-

νται δεν χαλάνε τις υπάρχουσες λειτουργίες. Επιπρόσθετα, αυτοματοποιημένα συστήματα ανάπτυξης (build systems) επιτρέπουν τη μεταγλώττιση αλλά και εκτέλεση του πηγαίου κώδικα και των tests. Το continuous integration εξασφαλίζει ότι ο κώδικας ενός συστήματος πάντα τεστάρεται διεξοδικά και πως επίσης δεν υπάρχουν προβλήματα ενσωμάτωσης μεταξύ υποσυστημάτων ή τμημάτων του κώδικα. Τα προβλήματα ενοποίησης ανακαλύπτονται πολύ γρήγορα οπότε και το κόστος για την διόρθωση τους είναι πολύ μικρότερο. Επιπλέον, μιας και ανακαλύπτονται άμεσα αυτά τα προβλήματα, είναι γνωστό το μέρος και ο χρόνος δημιουργίας του προβλήματος [8].

Για τους παραπάνω λόγους, εφαρμόστηκε η τεχνική του Continuous Integration και στην ανάπτυξη της εφαρμογής της παρούσας πτυχιακής εργασίας. Για την υλοποίηση του Continuous integration, χρησιμοποιήθηκε ο server αυτοματοποίησης Jenkins. Το Jenkins, όπως προαναφέρθηκε είναι ένας server αυτοματοποίησης ανοιχτού λογισμικού που βοηθά στην αυτοματοποίηση εργασιών όπως η μεταγλώττιση, το testing και η διάθεση του λογισμικού.

Συγκεκριμένα, υλοποιήθηκαν οι 2 παρακάτω διαδικασίες:

- **Μεταγλώττιση, Έλεγχος και Διάθεση του λογισμικού (Build, Test, Deploy)** Με τη βοήθεια ενός script που αναπτύχθηκε, ενορχηστρώθηκαν οι παραπάνω εργασίες οι οποίες εκτελούνται σειριακά. Στο πρώτο βήμα, γίνεται η μεταγλώττιση του κώδικα (compilation), έπειτα εκτελούνται τα unit tests και regression tests που γράφτηκαν. Στο επόμενο βήμα, γίνεται η διανομή του ενημερωμένου λογισμικού στον εξυπηρετητή όπου είναι διαθέσιμο το λογισμικό. Τέλος, παράγονται κάποιες αναφορές όπως το test coverage που έχει επιτευχθεί ή η τάση κάλυψης γραμμών κώδικα με τα διαθέσιμα tests. Οποιοδήποτε από αυτά τα 4 στάδια αποτύχει, τότε ολόκληρη η διαδικασία διακόπτεται και χαρακτηρίζεται σαν αποτυχημένη.

- **Κυκλοφορία νέας έκδοσης** Με τη χρήση του Jenkins αυτοματοποιήθηκε μία ακόμη σημαντική εργασία. Αυτή της δημιουργίας μίας νέας έκδοσης του λογισμικού. Πιο συγκεκριμένα, η εκτέλεση της εργασίας αυτής γίνεται χειροκίνητα, όπου δίνεται από τον χρήστη η παράμετρος της έκδοσης. Η παράμετρος καθορίζει αν η έκδοση που θα δημιουργηθεί θα είναι major, minor ή patch. Για παράδειγμα, αν δοθεί σαν παράμετρος το minor τότε εφόσον η τρέχουσα έκδοση είναι

η 1.2.0, η επόμενη που θα δημιουργηθεί θα είναι η 1.3.0. Στη συνέχεια, εκτελείται η μεταγλώττιση του project που αναπτύχθηκε για τον server και το Android. Εαν αποτύχει η μεταγλώττιση ενός εκ των δύο project, τότε ολόκληρη η εργασία ματαιώνεται. Έπειτα, δημιουργείται ένα Release το οποίο μέσω ενός REST call αποστέλλεται στο Github. Τέλος, τα παραχθέντα αρχεία από τον Java Server και τον Android Client αποστέλλονται και αυτά με τη σειρά τους στο Github, όπου συνδέονται με το σχετικό Release. Με αυτό τον τρόπο, είναι διαθέσιμη η νέα έκδοση του λογισμικού στο ευρύτερο κοινό.

Κεφάλαιο 5

Επισκόπηση της Εφαρμογής χρήστη

5.1 Περιγραφή

Για τους σκοπούς της παρούσας πτυχιακής εργασίας, αναπτύχθηκε μία εφαρμογή για έξυπνες συσκευές Android της οποίας ο βασικός στόχος ήταν η παροχή τουριστικών πληροφοριών. Ένας τουρίστας, χρησιμοποιώντας την εφαρμογή μπορεί να αναζητήσει τις δημοφιλέστερες τοποθεσίες που μπορεί να επισκεφθεί στην πόλη την οποία βρίσκεται. Αυτό επιτυγχάνεται απλά παρέχοντας τις συντεταγμένες του (GPS) και έχοντας σύνδεση στο διαδίκτυο. Επίσης, ο χρήστης μπορεί να αναζητήσει τοποθεσίες βάσει της κατηγορίας που τον ενδιαφέρει, παραδείγματος χάρη καφετέριες ή αξιοθέατα. Με τη χρήση της εφαρμογής, ένας τουρίστας μπορεί να αποθηκεύσει τοποθεσίες για ένα μελλοντικό ταξίδι και να λάβει οδηγίες για την κάθε τοποθεσία.

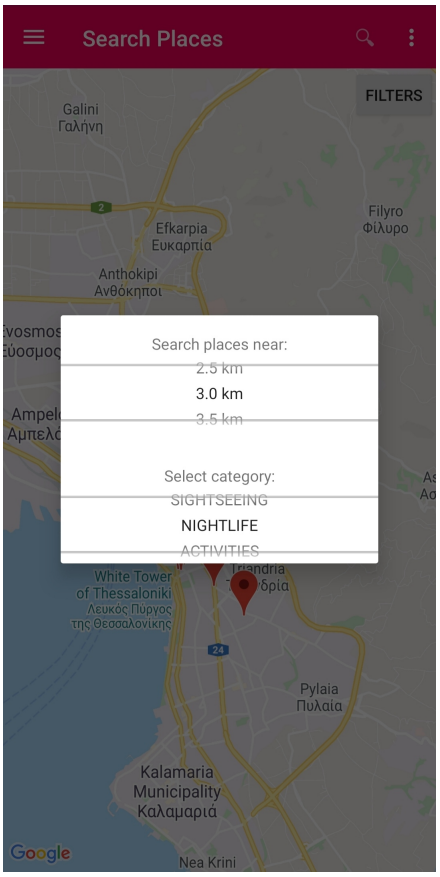
5.2 Λειτουργίες εφαρμογής

5.2.1 Αναζήτηση τοποθεσιών βάσει κατηγορίας

Όπως αναφέρθηκε και προηγουμένως, μία από τις βασικότερες λειτουργίες της εφαρμογής είναι η αναζήτηση τοποθεσιών κοντά στον χρήστη, βάσει κατηγορίας. Παραδείγματος χάρη, ένας χρήστης μπορεί να αναζητήσει για εστιατόρια τριγύρω του σε ακτίνα 3 χιλιομέτρων.

Όπως φαίνεται και στο διάγραμμα 5.1 ο χρήστης έχει στη διάθεση του κάποια φίλτρα για την αναζήτηση τοποθεσιών. Τα φίλτρα που μπορούν να χρησιμοποιηθούν

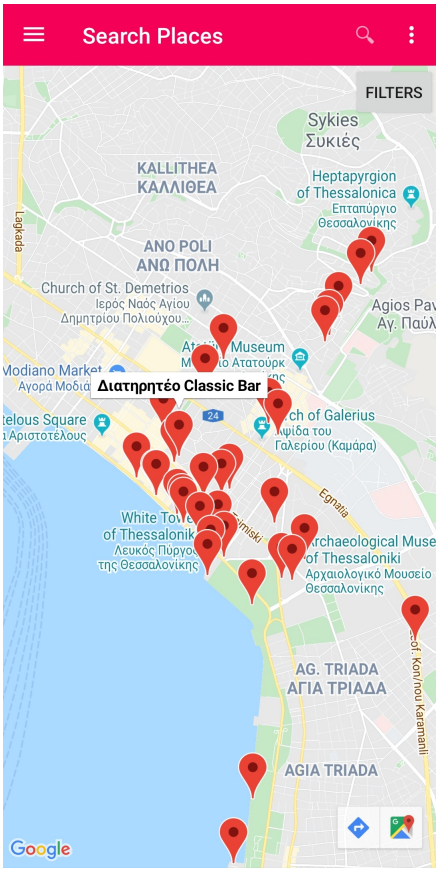
είναι η απόσταση από τις τοποθεσίες και η κατηγορία. Σε περίπτωση που ο χρήστης βρίσκεται σε κάποια πόλη η οποία δεν υποστηρίζεται από την εφαρμογή θα λάβει ένα αντίστοιχο μήνυμα. Επίσης, εάν η αναζήτηση βάσει των φίλτρων του χρήστη δεν επιστρέψει κάποια αποτελέσματα τότε προβάλλεται πάλι ένα αντίστοιχο μήνυμα και προβάλλεται στον χρήστη ένας χάρτης χωρίς κάποιο σημείο. Τέλος, αξίζει να σημειωθεί πως για να προχωρήσει στην χρήση αυτής της λειτουργίας ο χρήστης, απαραίτητη προϋπόθεση είναι να έχει συνδεθεί με το λογαριασμό χρήστη του.



Διάγραμμα 5.1: Αναζήτηση τοποθεσιών σε ακτίνα 3 χιλιομέτρων για την κατηγορία της "Νυχτερινής Ζωής".

Στο διάγραμμα 5.2 φαίνεται το αποτέλεσμα μιας αναζήτησης τοποθεσιών. Κατά την εύρεση των τοποθεσιών, για τη διευκόλυνση του χρήστη γίνεται μεγέθυνση στο χάρτη ώστε να φαίνονται όλες οι τοποθεσίες που πληρούν τα κριτήρια που τέθηκαν. Στα αποτελέσματα που βρέθηκαν, η κάθε τοποθεσία συμβολίζεται με ένα στίγμα στο χάρτη. Μετά την εμφάνιση των αποτελεσμάτων, ο χρήστης πατώντας πάνω σε ένα σημείο στο χάρτη, μπορεί να δει λεπτομέρειες όπως το όνομα της τοποθεσίας. Επιπρόσθετα, ο χρήστης έχει τη δυνατότητα να λάβει οδηγίες από την εφαρμογή χαρτών της Google

ώστε να επισκεφθεί την επιλεγμένη τοποθεσία.



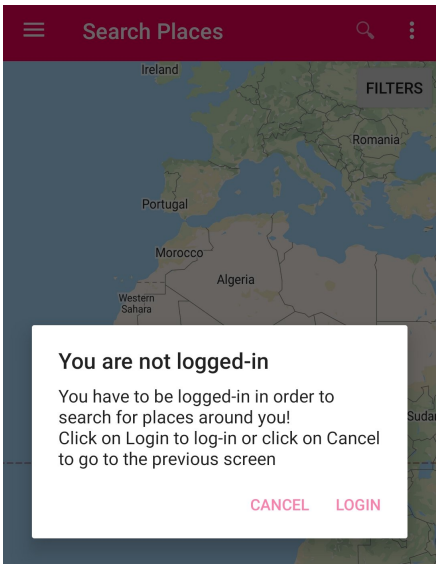
Διάγραμμα 5.2: Αποτέλεσμα αναζήτησης τοποθεσιών.

5.2.2 Δημιουργία λογαριασμού χρήστη

Όπως αναφέρθηκε προηγουμένως, κάποιες λειτουργίες της εφαρμογής απαιτούν από τον χρήστη να είναι συνδεδεμένος με τον λογαριασμό του. Έτσι και στην περίπτωση της "Αναζήτησης τοποθεσιών", εάν ο χρήστης επιλέξει να αναζητήσει τοποθεσίες χωρίς να είναι συνδεδεμένος, θα προβληθεί ένα μήνυμα που προτρέπει τον χρήστη να συνδεθεί ή να δημιουργήσει νέο λογαριασμό. Με την εμφάνιση του μηνύματος αυτού — όπως φαίνεται και στο διάγραμμα 5.3 — αποτρέπεται ο χρήστης στο να προβεί στη χρήση λειτουργιών που είναι διαθέσιμες μόνο για εγγεγραμμένους χρήστες.

5.2.3 Διαχείριση ταξιδιών

Μία ακόμα πολύ σημαντική λειτουργία της εφαρμογής είναι η δυνατότητα που παρέχεται στον χρήστη να δημιουργεί μελλοντικά ταξίδια, στα οποία αποθηκεύει τις

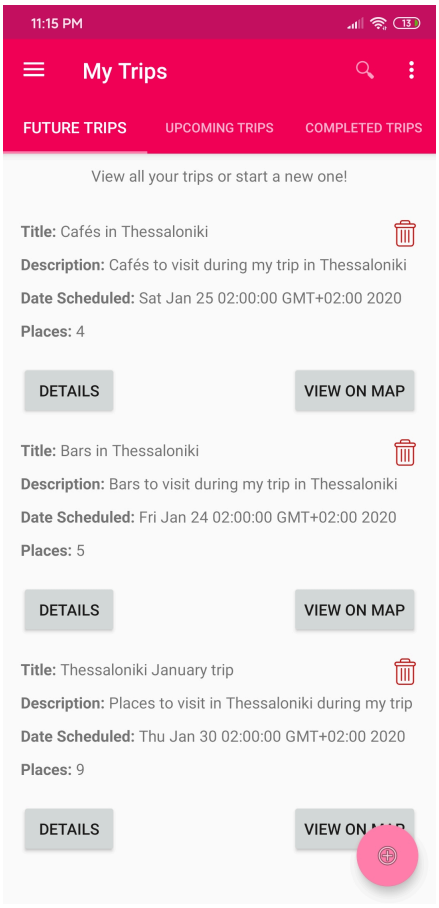


Διάγραμμα 5.3: Προτροπή του χρήστη σε σύνδεση με το λογαριασμό του.

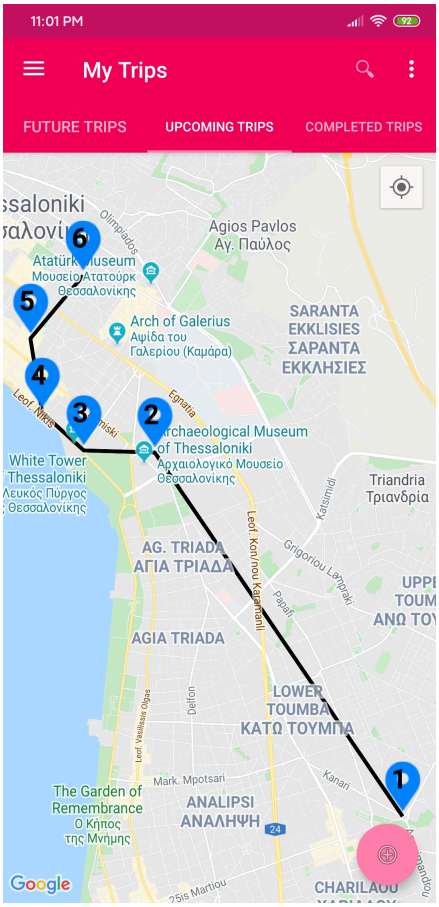
τοποθεσίες που θέλει να επισκεφθεί. Όπως φαίνεται στο διάγραμμα 5.4 ο χρήστης έχει στη διάθεσή του τρεις καρτέλες ώστε να οργανώσει τα ταξίδια του. Στην πρώτη καρτέλα προβάλλονται όλα τα μελλοντικά ταξίδια που έχει προγραμματίσει. Σε κάθε ταξίδι, ο χρήστης έχει την επιλογή να δει τις λεπτομέρειες του ταξιδιού, όπως για παράδειγμα ποιες τοποθεσίες έχει επιλέξει και κάποιες φωτογραφίες από τις τοποθεσίες αυτές. Επιπρόσθετα, υπάρχει η δυνατότητα να προβληθούν τα σημεία ενός ταξιδιού στον χάρτη.

Στην δεύτερη καρτέλα, όπως απεικονίζεται και στο διάγραμμα 5.5 ο χρήστης μπορεί να διαχειριστεί τα ταξίδια του που πλησιάζουν, είναι δηλαδή προγραμματισμένα σε λιγότερο από μία εβδομάδα. Πάλι ο χρήστης, έχει τη δυνατότητα να δει τις λεπτομέρειες του ταξιδιού που πλησιάζει καθώς και να το δει στο χάρτη. Σε αυτή την καρτέλα όμως υπάρχει και η πρόσθετη λειτουργία όπου ο χρήστης μπορεί να ξεκινήσει το ταξίδι του.

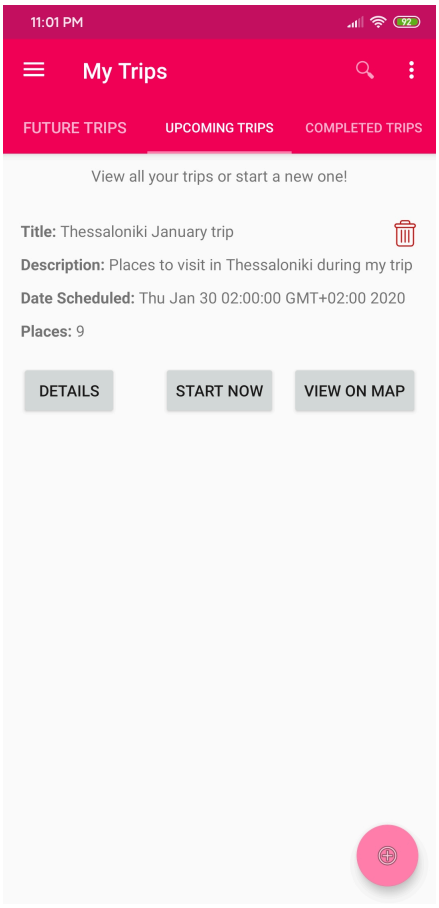
Πατώντας αυτή την επιλογή, εμφανίζεται ο χάρτης με τις τοποθεσίες που είχαν επιλεχθεί όπως στο διάγραμμα 5.6. Στον χάρτη, η κάθε τοποθεσία εμφανίζεται με έναν αριθμό. Οι τοποθεσίες ταξινομούνται βάσει αλγορίθμου από την εφαρμογή του Server, όπου προτείνεται στο χρήστη η βέλτιστη διαδρομή που μπορεί να ακολουθήσει. Κατά αυτόν τον τρόπο, η εφαρμογή προτείνει και σχεδιάζει στον χάρτη τη βέλτιστη διαδρομή που μπορεί να ακολουθήσει ο χρήστης, λαμβάνοντας φυσικά σαν παράμετρο τις τρέχουσες γεωγραφικές του συντεταγμένες.



Διάγραμμα 5.4: Προβολή των μελλοντικών ταξιδιών



Διάγραμμα 5.6: Προβολή της προτεινόμενης διαδρομής των τοποθεσιών ενός ταξιδιού.



Διάγραμμα 5.5: Προβολή των επερχόμενων ταξιδιών που είναι προγραμματισμένα σε λιγότερο από μία εβδομάδα.

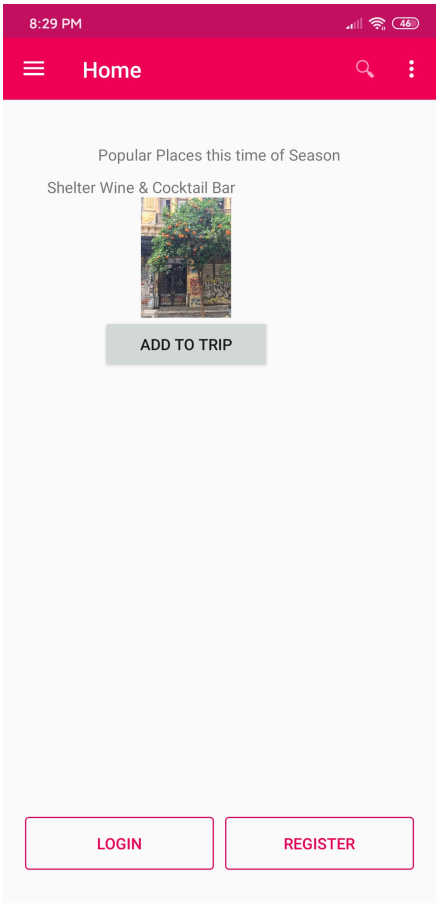
Αφότου τελειώσει ο χρήστης με το ταξίδι του, αυτό πλέον έχει μεταβεί στην κατάσταση του "Ολοκληρωμένου ταξιδιού" όποτε και πλέον μπορεί να βρεθεί μόνο στην τρίτη καρτέλα. Σε αυτή την καρτέλα εμφανίζονται μόνο τα ταξίδια που έχουν ολοκληρωθεί και υπάρχουν κυρίως για να κρατάει ο χρήστης ιστορικό των ταξιδιών του αλλά και των τοποθεσιών που επισκέφθηκε σε αυτά.

5.2.4 Οι Δέκα Δημοφιλέστερες τοποθεσίες

Στο μενού της εφαρμογής, μεταξύ των άλλων λειτουργιών υπάρχει και αυτή στην οποία ο χρήστης έχει τη δυνατότητα να δει τις δέκα δημοφιλέστερες τοποθεσίες της πόλης του. Στον σχετικό χάρτη που εμφανίζεται, υπάρχουν δέκα σημεία με τις πιο δημοφιλείς τοποθεσίες που υπάρχουν στην πόλη που βρίσκεται ο χρήστης. Ο χρήστης, μπορεί να λάβει οδηγίες μέσω των χαρτών της Google ώστε να επισκεφθεί την κάθε τοποθεσία.

5.2.5 Δημοφιλείς τοποθεσίες ανά εποχή του χρόνου

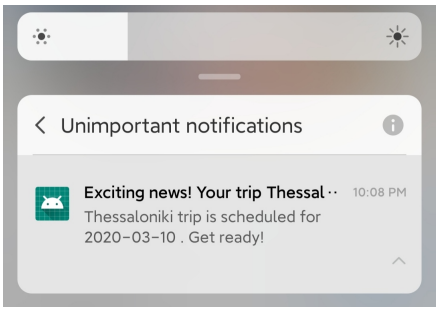
Μία ακόμα ενδιαφέρουσα λειτουργία που είναι διαθέσιμη στην αρχική οθόνη της εφαρμογής, είναι η προβολή των πιο δημοφιλών τοποθεσιών για την τρέχουσα εποχή του χρόνου. Αυτή η λειτουργία είναι διαθέσιμη χωρίς να απαιτεί την ύπαρξη λογαριασμού χρήστη. Κατά αυτό τον τρόπο, όπως απεικονίζεται και στο διάγραμμα 5.7 προβάλλονται στον χρήστη οι 25 δημοφιλέστερες τοποθεσίες για την τρέχουσα εποχή του χρόνου και για την πόλη την οποία βρίσκεται ο χρήστης. Ο χρήστης, "σέρνοντας" την οθόνη με φορά προς τα δεξιά ή αριστερά, μπορεί να πλοηγηθεί ανάμεσα σε αυτές τις τοποθεσίες και να τις προσθέσει ενδεχομένως σε ένα επερχόμενο ταξίδι του.



Διάγραμμα 5.7: Προβολή των δημοφιλών τοποθεσιών της πόλης που βρίσκεται ο χρήστης, για την τρέχουσα εποχή του χρόνου.

5.2.6 Ενημέρωση χρήστη για τα επερχόμενα ταξίδια του

Πέραν των ταξιδιών που μπορεί να προγραμματίσει ο χρήστης, όπου προσθέτει τις τοποθεσίες που θέλει να επισκεφτεί, η εφαρμογή έχει τη δυνατότητα να ενημερώνει αυτόματα τον χρήστη για τα ταξίδια που έχει προγραμματίσει και πλησιάζουν. Πιο συγκεκριμένα, η εφαρμογή στέλνει push notification στις Android συσκευές των χρηστών, όταν για το ταξίδι που έχει προγραμματιστεί μένει λιγότερο από μία εβδομάδα. Επομένως, ο χρήστης λαμβάνει μία ειδοποίηση κάθε εικοσιτέσσερις (24) ώρες, ώστε να γίνει μία υπενθύμιση για το ταξίδι του. Μία τέτοια ειδοποίηση φαίνεται στο διάγραμμα 5.8



Διάγραμμα 5.8: Ειδοποίηση στην Android συσκευή του χρήστη, για ένα επερχόμενο ταξίδι

Στην ειδοποίηση, αναγράφεται η ημερομηνία για την οποία έχει προγραμματιστεί το ταξίδι, καθώς και το όνομα το οποίο έχει δώσει ο χρήστης στο ταξίδι του.

Κεφάλαιο 6

Αρχιτεκτονική της εφαρμογής

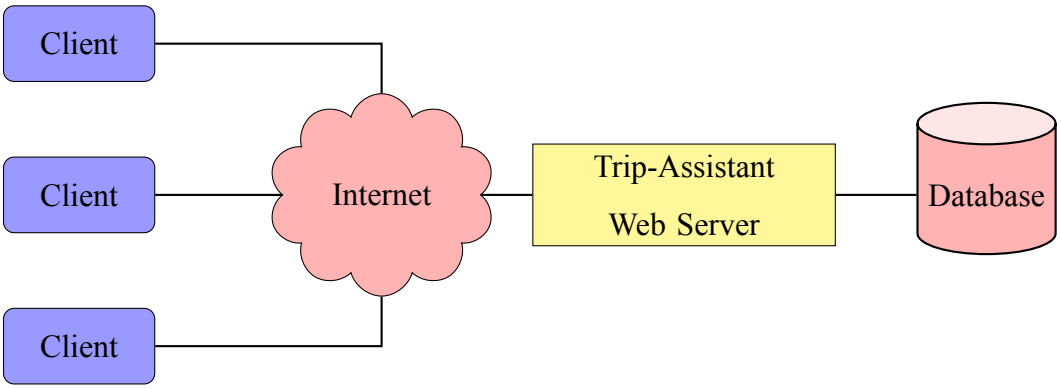
6.1 Μοντέλο ανάπτυξης

6.1.1 Αρχικό μοντέλο ανάπτυξης

Η βασική αρχή κατά την οποία δομήθηκε η εφαρμογή είναι το μοντέλο **Client-Server**. Σε αυτό το μοντέλο, οι εργασίες και ο φόρτος εργασίας διανέμονται μεταξύ των παρόχων μιας υπηρεσίας ή ενός πόρου -που αποκαλούνται servers (εξυπηρετητές)- και των πελατών (clients) που κάνουν χρήση των παρεχόμενων υπηρεσιών. Στο Trip Assistant, για τις ανάγκες του server (εξυπηρετητή), αναπτύχθηκε ένα REST API ώστε να παρέχει όλη τη λειτουργικότητα που χρειάστηκε για την εφαρμογή. Η ανάπτυξη του REST API έγινε με τη γλώσσα Java και τη χρήση του framework Spring. Αντίστοιχα, για την πλευρά του client αναπτύχθηκε μία εφαρμογή για συσκευές Android με τη γλώσσα Java, η οποία εφαρμογή εξυπηρετεί τις ανάγκες του τελικού χρήστη και γίνεται κατανάλωση του REST API που αναπτύχθηκε στην πλευρά του Server.

Στο παρακάτω διάγραμμα 6.1 βρίσκεται η αναπαράσταση της βασικής αρχιτεκτονικής που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής.

Όπως φαίνεται και στην αρχιτεκτονική που περιγράφεται στο παραπάνω διάγραμμα, το κάθε τμήμα του συνόλου της εφαρμογής έχει ξεκάθαρο ρόλο ως προς τις αρμοδιότητες που έχει. Τον ρόλο των Client, έχει η εφαρμογή που αναπτύχθηκε για το Android, της οποίας ο σκοπός είναι να επικοινωνεί με τον Server μέσω του API που αναπτύχθηκε, ώστε να λαμβάνει τα δεδομένα που χρειάζεται για να κάνει προτάσεις τουριστικού περιεχομένου στο χρήστη. Αντίστοιχα, το κομμάτι του server όπως φαίνεται



Διάγραμμα 6.1: Αρχική αρχιτεκτονική ανάπτυξης του TripAssistant

αλληλεπιδρά με την βάση δεδομένων ώστε να αποθηκεύει αλλά και να διαβάζει τα δεδομένα που χρειάζεται. Έπειτα, τα δεδομένα αυτά παρέχονται στους Clients μέσω του API που αναπτύχθηκε και φιλοξενείται στο τμήμα του server.

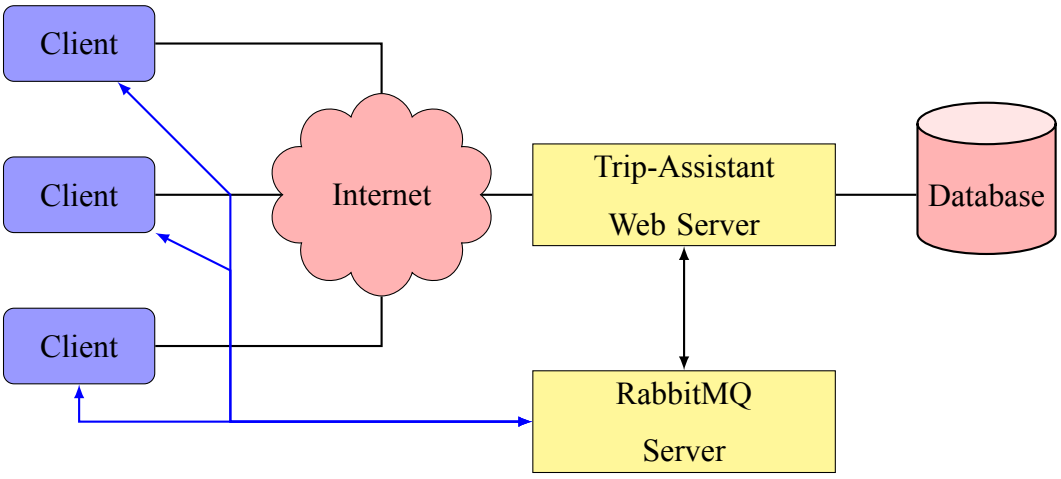
6.1.2 Εξέλιξη μοντέλου ανάπτυξης

Το παραπάνω μοντέλο που περιγράφηκε, στην πορεία ενισχύθηκε προσθέτοντας ένα ενδιάμεσο λογισμικό για μερικές λειτουργίες της εφαρμογής. Αυτό το ενδιάμεσο λογισμικό διευκόλυνε την επικοινωνία μεταξύ του Client και του Server. Το ενδιάμεσο λογισμικό υλοποιήθηκε με το πρωτόκολλο Advanced Message Queuing Protocol (AMQP). Το πρωτόκολλο αυτό διευκολύνει την ανταλλαγή μηνυμάτων μεταξύ δύο μέσων, σε αυτή την περίπτωση του Client και του Server. Ωστόσο, υπήρξε μία περίπτωση χρήσης στην οποία το ενδιάμεσο λογισμικό χρησιμοποιήθηκε εξολοκλήρου στο λογισμικό που αναπτύχθηκε για τον Server.

Ο λόγος που έγινε αυτό ήταν για να διαχωριστούν αλλά και να είναι ξεκάθαρες οι αρμοδιότητες που ανήκουν σε ένα τμήμα της εφαρμογής του Server. Επίσης με την προσθήκη αυτού του ενδιάμεσου λογισμικού, επιλύθηκε το πρόβλημα της εκτέλεσης χρονοβόρων εργασιών μέσω Internet. Παραδείγματος χάρη, ο Client χρειάστηκε να κάνει ένα HTTP request στον Server ώστε να ζητήσει τις πιο δημοφιλείς τοποθεσίες μίας πόλης για την τρέχουσα εποχή του χρόνου. Το συγκεκριμένο αίτημα, απαιτεί από τον Server την εκτέλεση χρονοβόρων διαδικασιών ώστε να συλλέξει τη δεδομένη στιγμή από την βάση δεδομένων τις δημοφιλέστερες τοποθεσίες για την πόλη που ζητήθηκε, έπειτα να τις χωρίσει βάσει εποχής και να κρατήσει τις 25 δημοφιλέστερες για την τρέχουσα εποχή του χρόνου. Η εκτέλεση αυτών των λειτουργιών θα μπορούσε να είναι

αρκετά χρονοβόρα, γεγονός που θα μπορούσε να προκαλέσει ένα timeout στο HTTP request του Client προς τον Server. Χρησιμοποιώντας το πρωτόκολλο AMQP λύνεται αυτό το πρόβλημα. Ο client καταχωρεί ένα μήνυμα στο ενδιάμεσο λογισμικό ζητώντας τις δημοφιλέστερες τοποθεσίες για την τρέχουσα εποχή. Έπειτα, ο server ο οποίος "ακούει" την συγκεκριμένη ουρά, διαβάζει το μήνυμα και ξεκινάει τη διαδικασία του υπολογισμού των ζητούμενων δεδομένων. Όταν υπολογιστεί το αποτέλεσμα, ο Server καταχωρεί στο ενδιάμεσο λογισμικό ένα μήνυμα με το αποτέλεσμα που υπολόγισε, όπου ο Client ο οποίος "ακούει" σε αυτή την ουρά λαμβάνει το μήνυμα του αποτελέσματος και το επεξεργάζεται όπως χρειαστεί. Το συγκεκριμένο μοντέλο ονομάζεται που περιγράφηκε ονομάζεται Remote Procedure Call (RPC).

Η προσθήκη του ενδιάμεσου λογισμικού (middleware) στην αρχιτεκτονική του συστήματος αναπαρίσταται στο διάγραμμα 6.2.

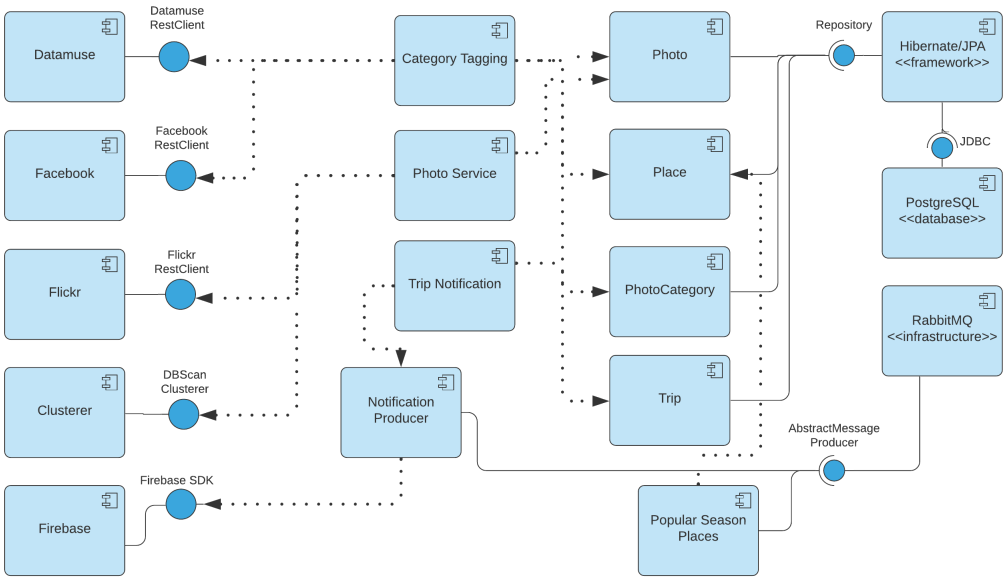


Διάγραμμα 6.2: Η βελτιωμένη έκδοση της αρχιτεκτονικής του TripAssistant

Όπως φαίνεται στο διάγραμμα, ο **Web Server** τροφοδοτεί τον **RabbitMQ Server** με μηνύματα τα οποία είτε προορίζονται για τον ίδιο, είτε για τους clients. Αντίστοιχα οι clients δέχονται μηνύματα από τον **RabbitMQ Server** τα οποία έχουν παραχθεί από τον **Web Server**. Ωστόσο, μόνο στην περίπτωση του RPC request που περιγράφηκε παραπάνω παράγουν μηνύματα με αποδέκτη τον **Web Server**.

Μία πιο λεπτομερής ανάλυση του συστήματος φαίνεται στο UML Component διάγραμμα 6.3. Στο διάγραμμα αυτό απεικονίζονται τα κύρια τμήματα της εφαρμογής και ο τρόπος με τον οποίο αλληλεπιδρούν ή εξαρτώνται μεταξύ τους.

Παραδείγματος χάρη, φαίνεται πως για την υλοποίηση του Trip Notification χρη-



Διάγραμμα 6.3: UML Component Diagram του συστήματος

σιμοποιήθηκε μία ουρά του RabbitMQ η οποία με τη σειρά της καλούσε την υπηρεσία του Firebase Cloud Messaging.

6.2 Τεχνολογίες που χρησιμοποιήθηκαν

Για την ανάπτυξη των δύο εφαρμογών (back-end και front-end) χρησιμοποιήθηκαν οι παρακάτω τεχνολογίες.

6.2.1 Java

Η γλώσσα που χρησιμοποιήθηκε για την υλοποίηση του Web Server και της Android εφαρμογής που χρειάστηκαν για τις ανάγκες της πτυχιακής εργασίας είναι η Java. Η Java είναι μία αρκετά διαδεδομένη και δημοφιλής γλώσσα που χρησιμοποιείται ευρέως σε εφαρμογές διαφόρων μεγεθών. Επίσης, είναι cross-platform που σημαίνει πως μπορεί να τρέξει η εφαρμογή σε διαφορετικές πλατφόρμες και λειτουργικά χωρίς κάποια ιδιαίτερη παραμετροποίηση. Κατ' αυτό τον τρόπο, η εφαρμογή που αναπτύχθηκε για τον Server, έχει τη δυνατότητα να λειτουργήσει σε διάφορα λειτουργικά όπως το Unix αλλά και τα Windows.

6.2.2 Android

Η εφαρμογή που αναπτύχθηκε για το κομμάτι του client ήταν μια εφαρμογή για Android κινητά. Το Android είναι ένα λογισμικό ανοιχτού κώδικα, βασισμένο στο Linux το οποίο δημιουργήθηκε για ένα μεγάλο εύρος συσκευών. Επίσης, το σύνολο των λειτουργιών που παρέχει το λειτουργικό του Android, διατίθεται μέσω APIs γραμμένα στη γλώσσα προγραμματισμού Java [2]. Η ανάπτυξη εφαρμογών σε Android έχει το πλεονέκτημα πως έχει μικρό κόστος για τον εξοπλισμό και χρησιμοποιεί μεταξύ άλλων και την γλώσσα Java ή Kotlin όπως αναφέρθηκε και προηγουμένως. Στην εφαρμογή του Android Client που αναπτύχθηκε χρησιμοποιήθηκε η Java. Επίσης η διάθεση της εφαρμογής έχει αντίστοιχα ένα μικρό κόστος και είναι αρκετά εύκολη και άμεση η διαδικασία.

6.2.3 Spring Boot Framework

Για την ανάπτυξη του REST API στην εφαρμογή του server χρησιμοποιήθηκε το Spring Framework. Το Spring επιταχύνει και διευκολύνει την ανάπτυξη διαδικτυακών εφαρμογών, αφαιρώντας αρκετό από τον απαιτούμενο κώδικα για την παραμετροποίηση της εφαρμογής. Με τη βοήθειά του Spring Boot υλοποιήθηκαν όλα τα endpoints του API που παρείχαν τις λειτουργίες δημιουργίας, διαβάσματος, ενημέρωσης και διαγραφής για κάθε πόρο του συστήματος (CRUD operations). Το Spring Boot, είναι βασισμένο στην αρχή convention over configuration, γεγονός που σημαίνει πως η ανάπτυξη ενός REST API είναι αρκετά γρήγορη, καθώς το Spring Boot παρέχει μία τυπική παραμετροποίηση για το σύστημα χωρίς να χρειάζεται ο προγραμματιστής να αφιερώσει πολύ χρόνο για την παραμετροποίηση.

6.2.4 Hibernate Framework

Για την διασύνδεση της εφαρμογής με τη βάση δεδομένων, χρησιμοποιήθηκε το Hibernate σε συνδυασμό με το Spring. Το Hibernate είναι ένα ORM (Object-Relational Mapping) Framework, το οποίο είναι μία υλοποίηση του JPA (Java Persistence API) και συνεπώς μπορεί να χρησιμοποιηθεί σε οποιοδήποτε περιβάλλον υποστηρίζει το JPA συμπεριλαμβανομένων των Java SE (Standard Edition) και των JAVA EE (Enterprise

Edition) εφαρμογών. [7]

6.2.5 Spring Security

Το Spring Security είναι ένα Framework που προσφέρει ένα επαρκώς παραμετροποιήσιμο σύστημα αυθεντικοποίησης/εξουσιοδότησης (authentication/authorization) και γενικότερα βοηθάει στην διαχείριση της πρόσβασης των χρηστών στις λειτουργίες της εφαρμογής. Το Spring Security είναι πλέον η συνήθης επιλογή για την ασφάλιση εφαρμογών βασισμένων στο Spring. Το Spring Security μεταξύ άλλων, προσφέρει προστασία από τις 10 μεγαλύτερες απειλές κατά τον οργανισμό OWASP όπως το Cross-Site Request Forgery (CSRF), Clickjacking. [15] Όπως αναφέρθηκε, για το authentication και το authorization των χρηστών στην εφαρμογή, χρησιμοποιήθηκε το Spring Security αλλά με την προσθήκη και χρήση των JWT (JSON Web Token). Στη συνέχεια, αφού υλοποιήθηκε το σύστημα ταυτοποίησης των χρηστών, ασφαλίστηκαν όλα τα endpoints της εφαρμογής του Server με το σύστημα αυτό.

6.2.6 PostgreSQL (Βάση Δεδομένων)

Η εφαρμογή που αναπτύχθηκε είχε την ανάγκη να αποθηκευτούν δεδομένα για τους σκοπούς της καταχώρησης χρηστών αλλά και των δεδομένων των φωτογραφιών που στην πορεία επεξεργάζονταν. Για τους σκοπούς αυτούς, χρησιμοποιήθηκε η βάση δεδομένων PostgreSQL η οποία είναι μία σχεσιακή βάση δεδομένων ανοιχτού λογισμικού. Η αλληλεπίδραση με τη βάση δεδομένων γινόταν αποκλειστικά από την εφαρμογή του Server (back-end).

6.2.7 RabbitMQ Message Broker

Μεταξύ των τεχνολογιών που χρησιμοποιήθηκαν είναι και ο Message Broker RabbitMQ. Το RabbitMQ είναι ένας message broker ανοιχτού λογισμικού το οποίο υλοποιεί διάφορα πρωτόκολλα μηνυμάτων, όπου στην συγκεκριμένη περίπτωση χρησιμοποιήθηκε το πρωτόκολλο AMQP. Υποστηρίζει διάφορες λειτουργίες όπως η δρομολόγηση μηνυμάτων σε ουρές, η επιβεβαίωση παράδοσης όπως και πολλά άλλα. Το RabbitMQ χρησιμοποιήθηκε σαν διαμεσολαβητής σε κάποιες από τις επικοινωνίες μεταξύ της εφαρμογής του Server και του Client. Μία περίπτωση που χρησιμοποιήθηκε ήταν για

να διαχωρίσει (decouple) τις εργασίες σε ένα τμήμα του project. Μία άλλη περίπτωση ήταν η χρήση του για την εκτέλεση μιας χρονοβόρας διαδικασίας [12].

6.2.8 Firebase Cloud Messaging

Το Firebase Cloud Messaging είναι μία υπηρεσία της Google, η οποία είναι μία cross-platform υπηρεσία που προσφέρει μία αξιόπιστη λύση για αποστολή μηνυμάτων χωρίς κόστος. Στην περίπτωση της εφαρμογής, χρησιμοποιήθηκε για να στέλνονται push notifications στην Android συσκευή των χρηστών ώστε να ειδοποιούνται για ταξίδια που έχουν προγραμματίσει μέσω της εφαρμογής και πλησιάζουν.

6.2.9 JUnit

Το JUnit είναι ένα framework για που βοηθά στην σύνταξη unit tests και αναπτύχθηκε για την γλώσσα Java. Το JUnit χρησιμοποιήθηκε για να γραφτούν κάποια unit και integration tests στην εφαρμογή του Server ώστε να διασφαλιστεί η σωστή και απρόσκοπτη λειτουργία κάποιων τμημάτων της εφαρμογής.

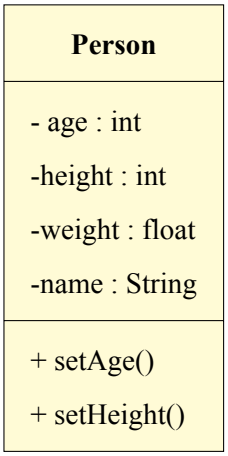
6.3 Διάγραμμα Κλάσεων UML

Την περιγραφή και την ανάλυση της δομής του συστήματος, διευκολύνει το παρακάτω διάγραμμα κλάσεων σε UML. Στο διάγραμμα κλάσεων παρακάτω αναλύονται οι κλάσεις του συστήματος, οι ιδιότητές τους, οι λειτουργίες τους και η σχέση τους με άλλα αντικείμενα.

6.3.1 Περιγραφή UML διαγράμματος κλάσεων

Γενικότερα, η UML είναι το στάνταρ πρότυπο που χρησιμοποιείται για την ανάλυση και σχεδίαση συστημάτων. Ένα από τα σημαντικότερα κομμάτια της UML είναι και τα διαγράμματα κλάσεων, τα οποία μοντελοποιούν την πληροφορία του τομέα ενδιαφέροντος σε αντικείμενα οργανωμένα σε κλάσεις καθορίζοντας και τις μεταξύ τους συσχετίσεις. Το κύριο κομμάτι ενός διαγράμματος κλάσεων είναι η κλάση η ίδια. Μία κλάση σε ένα UML διάγραμμα υποδηλώνει ένα σύνολο από αντικείμενα με κοινά χαρακτηριστικά. Μία κλάση αναπαρίσταται γραφικά από τετράγωνο χωρισμένο σε τρία

μέρη. Το πρώτο μέρος περιέχει το όνομα της κλάσης το οποίο πρέπει να είναι μοναδικό σε ολόκληρο το διάγραμμα. Το δεύτερο μέρος περιέχει τις ιδιότητες μιας κλάσης συνοδευόμενες με το όνομά τους και τον τύπο τους. Το τρίτο μέρος περιέχει όλες τις λειτουργίες της κλάσης οι οποίες σχετίζονται με το αντικείμενο της κλάσης. Τέλος, αξίζει να σημειωθεί πως το δεύτερο και τρίτο μέρος είναι προαιρετικά. Ένα παράδειγμα αναπαράστασης μιας κλάσης βρίσκεται στο διάγραμμα 6.4.

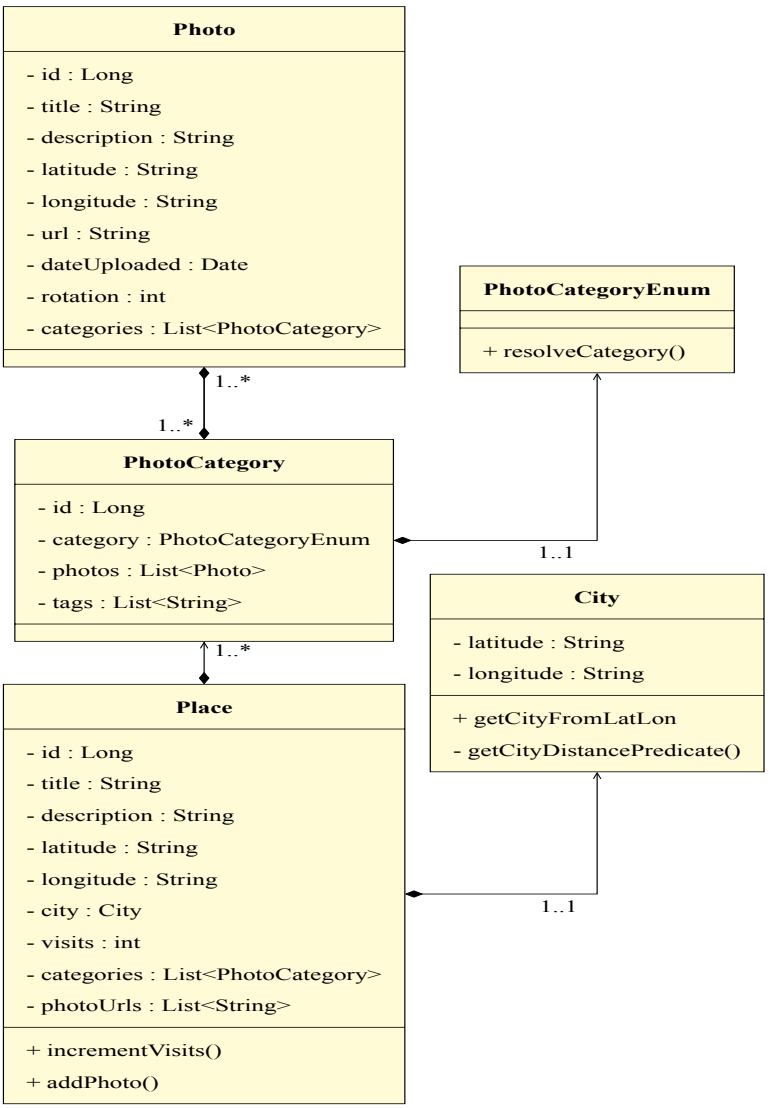


Διάγραμμα 6.4: Παράδειγμα UML Class

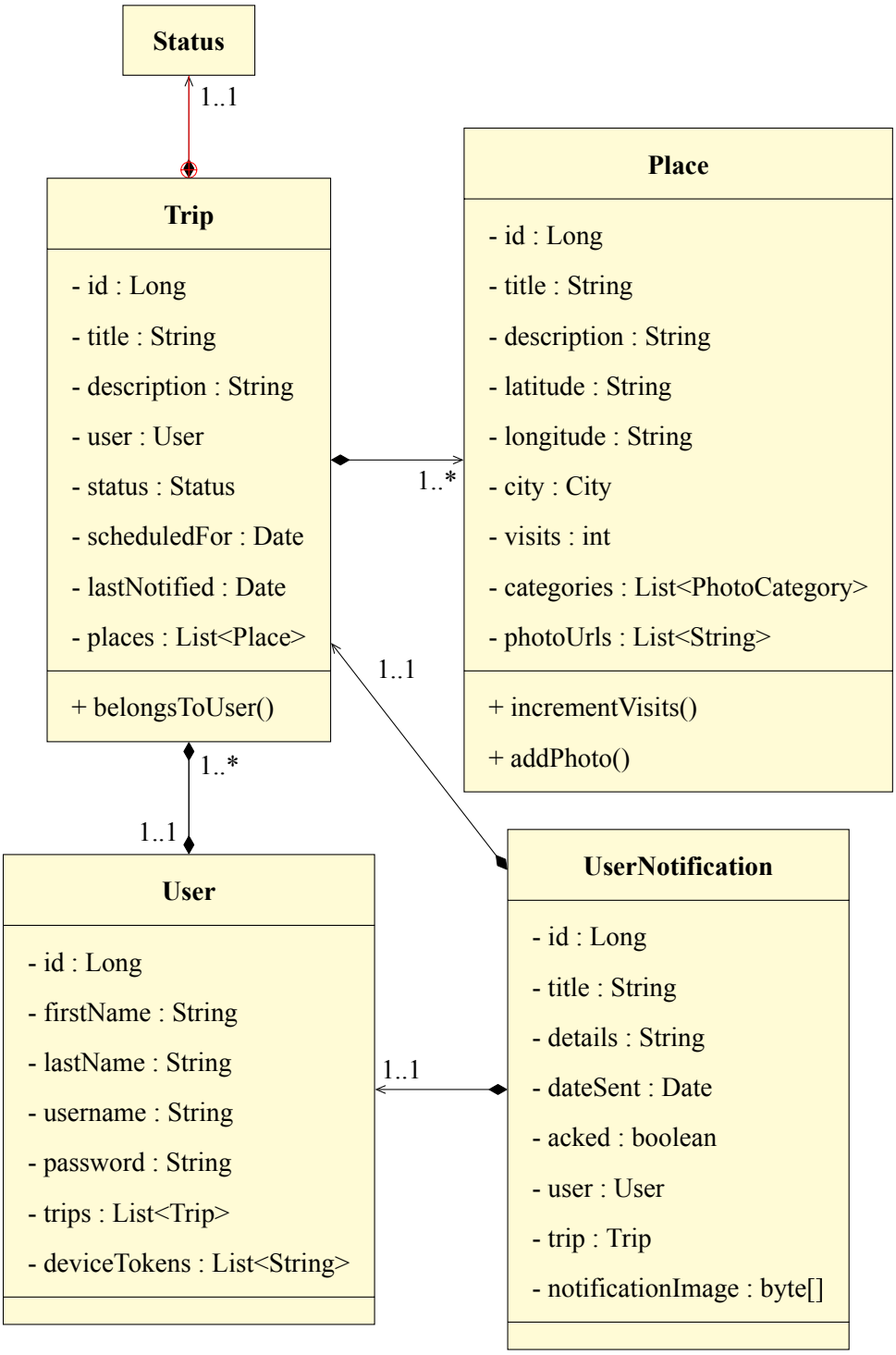
Οι συσχετίσεις σε ένα UML διάγραμμα είναι η σχέση μεταξύ δύο ή περισσότερων κλάσεων. Τα ονόματα των συσχετίσεων -όπως και τα ονόματα των κλάσεων- είναι μοναδικά μέσα σε ένα UML διάγραμμα. Η συσχέτιση μεταξύ δύο κλάσεων αναπαρίσταται γραφικά με μία γραμμή. Επίσης, η πολλαπλότητα σε μία συσχέτιση μεταξύ δύο κλάσεων C_1 και C_2 ορίζεται ως $n_l...n_u$. Η πολλαπλότητα αυτή προσδιορίζει ότι κάθε αντικείμενο της κλάσης C_1 συμμετέχει το λιγότερο n_l φορές και το πολύ n_u φορές στη συσχέτιση. Όταν η πολλαπλότητα παραλείπεται, τότε εννοείται ότι είναι $0...*$ δηλαδή μηδέν ή περισσότερες φορές.

Τέλος, σημαντικό κομμάτι των UML διαγραμμάτων κλάσεων είναι η αναπαράσταση της κληρονομικότητας και των ιεραρχιών. Στην UML μπορεί να αναπαρασταθεί η κληρονομικότητα μεταξύ μίας γονικής κλάσης και μιας υποκλάσης. Κατά αυτό τον τρόπο, τα αντικείμενα της υποκλάσης κληρονομούν τις ιδιότητες της γονικής κλάσης έχοντας τις πρόσθετες ιδιότητες της εκάστοτε υποκλάσης [3].

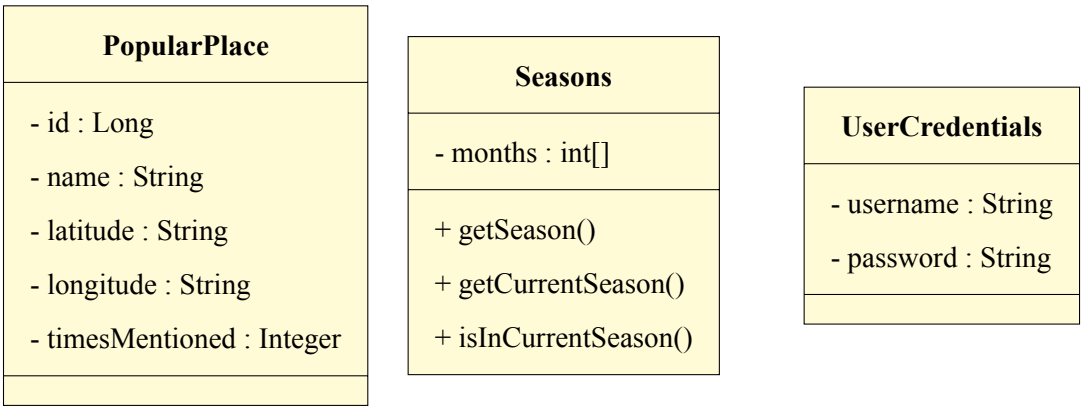
6.3.2 UML διάγραμμα του TripAssistant



Διάγραμμα 6.5: UML Διάγραμμα κλάσης της εφαρμογής TripAssistant - Τμήμα 1



Διάγραμμα 6.6: UML Διάγραμμα κλάσης της εφαρμογής TripAssistant - Τμήμα 2



Διάγραμμα 6.7: UML Διάγραμμα κλάσης της εφαρμογής TripAssistant - Τμήμα 3

6.4 Τεχνική ανάλυση των Λειτουργιών του TripAssistant

6.4.1 Εξαγωγή των δέκα δημοφιλέστερων τοποθεσιών

Η πρώτη λειτουργία που υλοποιήθηκε για την εφαρμογή TripAssistant ήταν η εξαγωγή των δέκα δημοφιλέστερων τοποθεσιών από την πόλη που βρίσκεται ο χρήστης. Αρχικά, η εφαρμογή υποστήριζε μόνο την πόλη της Θεσσαλονίκης, όποτε παίρνοντας αυτό το παράδειγμα, έπρεπε σε πρώτο στάδιο η εφαρμογή να αποθηκεύσει όλες τις φωτογραφίες που τραβήχτηκαν στην πόλη της Θεσσαλονίκης.

Οπότε, υλοποιήθηκε ένα Thread του οποίου η αρμοδιότητα ήταν να ανακτά από το Flickr μέσω του API που παρέχει, τις φωτογραφίες που έχουν τραβηχτεί στην πόλη της Θεσσαλονίκης τα τελευταία τρία χρόνια. Το Thread αυτό έτρεχε μία φορά την ημέρα ώστε να ενημερώνεται η εφαρμογή με τις νεότερες φωτογραφίες. Για την επικοινωνία με το API του Flickr, αναπτύχθηκε στην εφαρμογή του backend ένας Rest Client με τη βοήθεια των εργαλείων που παρέχει το Spring.

Εν συνεχεία, αναπτύχθηκε ακόμα ένα Thread το οποίο έτρεχε και αυτό με τη σειρά του μία φορά την ημέρα και εξήγαγε τις δέκα δημοφιλέστερες τοποθεσίες μιας πόλης. Για να επιτευχθεί αυτό, χρησιμοποιήθηκε ο clustering αλγόριθμος DBSCAN. Δίνοντας στον DBSCAN ένα set απο σημεία (τα οποία αποτελούνταν από τις γεωγραφικές συντεταγμένες των φωτογραφιών), ο αλγόριθμος βγάζει κάποια cluster που το καθένα περιέχει τα σημεία αυτά. Έπειτα, παίρνοντας τα δέκα cluster με τα περισσότερα σημεία έχουμε στη διάθεσή μας και τις δέκα δημοφιλέστερες τοποθεσίες. Τέλος, οι τοποθεσίες αυτές αποθηκεύονται στη βάση δεδομένων στον πίνακα **PopularPlace**.

Στο παραπάνω πρόβλημα, χρησιμοποιήθηκε ο αλγόριθμος DBSCAN καθώς δημιουργεί τα cluster βάσει πυκνότητας, οπότε οι φωτογραφίες οι οποίες έχουν τραβηχτεί στο ίδιο ή σε πολύ κοντινό σημείο συμπεριλαμβάνονται σε ένα cluster καθώς είναι γειτονικές. Αντιθέτως, οι φωτογραφίες οι οποίες είναι διάσπαρτες και αραιές στο χάρτη, θεωρούνται θόρυβος και επομένως παραλείπονται από τα αποτελέσματα.

6.4.2 Σύστημα σύνδεσης/εγγραφής χρήστη και authentication

Όπως αναφέρθηκε σε προηγούμενες ενότητες, οι περισσότερες λειτουργίες της εφαρμογής απαιτούν την ύπαρξη ενός λογαριασμού χρήστη. Για το λόγο αυτό, υλοποιήθηκε το σύστημα εγγραφής/σύνδεσης χρήστη με τη βοήθεια του **Spring Security**.

Κατά αυτό τον τρόπο, οι χρήστες δημιουργούν το λογαριασμό τους όπου παρέχουν ένα όνομα χρήστη και τον κωδικό τους. Ο κωδικός πρόσβασης των χρηστών, για λόγους ασφαλείας δεν αποθηκεύεται σαν κείμενο στη βάση δεδομένων, αλλά δημιουργείται ένα SHA-1 hash του κωδικού πρόσβασης που παρείχε ο χρήστης. Αντίστοιχα, κατά τη σύνδεση του χρήστη, δημιουργείται ένα hash του κωδικού που παρείχε ο χρήστης και στην περίπτωση που ταιριάζει με το hash που υπάρχει στη βάση δεδομένων, η σύνδεση του χρήστη στο λογαριασμό του είναι επιτυχής.

Επιπλέον, τα περισσότερα endpoints του Web Server είχαν ασφαλιστεί, οπότε ένας χρήστης σε κάθε request έπρεπε να παρέχει ένα authentication token ώστε να μπορέσει να ταυτοποιηθεί και να εξυπηρετηθεί. Ο χρήστης, προμηθεύεται το token αυτό κατά την επιτυχή εγγραφή ή σύνδεση του. Το JWT token αυτό ακολουθεί το πρότυπο *RFC7519* και είναι έγκυρο για μία ώρα από τη στιγμή της έκδοσής του. Ο Android client, αποθηκεύει το token αυτό και το χρησιμοποιεί σε όλα τα REST calls που γίνονται προς το API του TripAssistant WebServer, εώς ότου να λήξει.

6.4.3 Αναζήτηση τοποθεσιών βάσει κατηγορίας

Για την αναζήτηση τοποθεσιών υλοποιήθηκε ένα endpoint που χρησιμοποιήθηκε από την Android εφαρμογή. Το endpoint αυτό δέχεται τέσσερις παραμέτρους, οι πρώτες δύο αφορούν τις γεωγραφικές συντεταγμένες του χρήστη, η επόμενη το εύρος (ακτίνα) σε χιλιόμετρα μέχρι το οποίο θα βρίσκονται οι τοποθεσίες. Η τελευταία παράμετρος είναι η κατηγορία των τοποθεσιών.

Για την εύρεση των τοποθεσιών, αρχικά η εφαρμογή ελέγχει αν οι συντεταγμένες του χρήστη βρίσκονται σε κάποια από τις πόλεις που υποστηρίζει η εφαρμογή. Έπειτα αναζητούνται οι τοποθεσίες που βρίσκονται σε αυτή την πόλη. Τέλος, με τη χρήση της φόρμουλας Haversine συγκρίνεται η απόσταση της τοποθεσίας από τις συντεταγμένες του χρήστη και επιστρέφονται αυτές οι οποίες βρίσκονται μέσα στην ακτίνα που ζητήθηκε.

Η φόρμουλα Haversine είναι μία μέθοδος με την οποία υπολογίζεται η απόσταση δύο συντεταγμένων σε έναν δισδιάστατο χάρτη. Η απόσταση αυτή είναι η πραγματική απόσταση των σημείων καθώς λαμβάνεται υπόψιν η σφαιρική τριγωνομετρία της γης, ωστόσο δεν λαμβάνονται υπόψιν τα ύψη των βουνών και τα βάθη των πεδιάδων [1].

Η απόσταση Haversine μεταξύ δύο γεωγραφικών σημείων ορίζεται από την παρακάτω εξίσωση [17]:

$$D = 2R*$$
$$arcsin\left(\sqrt{sin^2(\frac{Lat_1 - Lat_2}{2}) + cos(Lat_1) * cos(Lat_2) * sin^2(\frac{Long_1 - Long_2}{2})}\right) \tag{6.1}$$

Όπου **Lat** είναι το Latitude δηλαδή το γεωγραφικό πλάτος, **Long** το Longitude δηλαδή το γεωγραφικό μήκος και **R** είναι η ακτίνα της γης (χρησιμοποιήθηκε η τιμή *6372.8* ώστε το αποτέλεσμα της απόστασης να είναι σε χιλιόμετρα).

6.4.4 Κατηγοριοποίηση τοποθεσιών

Όπως αναφέρθηκε στην ενότητα 6.4.3 η εφαρμογή προσφέρει τη δυνατότητα στο χρήστη να αναζητήσει τοποθεσίες βάσει κατηγορίας. Για να επιτευχθεί αυτό, η εφαρμογή έχει κατηγοριοποιήσει κάθε τοποθεσία που υπάρχει στη βάση δεδομένων. Η διαδικασία της κατηγοριοποίησης ξεκίνησε προσθέτοντας κάποια tags/λέξεις κλειδιά για κάθε κατηγορία που υποστηρίζει η εφαρμογή

Προσθήκη tags σε κάθε κατηγορία

Για το σκοπό αυτό, δημιουργήθηκε μία λειτουργία η οποία πρόσθετε σε κάθε κατηγορία μία λίστα από λέξεις κλειδιά/tags. Χρησιμοποιώντας το REST call που πε-

ριγράφηκε στην ενότητα 3.6.3 αναζητήθηκαν λέξεις κλειδιά βάσει του ονόματος της κάθε κατηγορίας. Έπειτα τα αποτελέσματα που επέστρεψε το παραπάνω REST call σώζονταν στη λίστα με τα tags της κάθε κατηγορίας. Επίσης υπήρχαν και κάποια βασικά tags για κάθε κατηγορία που υπάρχουν στον πηγαίο κώδικα της εφαρμογής. Και αυτά με τη σειρά τους προστίθενται στα tags της εκάστοτε κατηγορίας. Τέλος, αξίζει να σημειωθεί πως αυτή η διαδικασία προσθήκης tags σε μία κατηγορία γίνεται μία φορά για κάθε κατηγορία.

Προσθήκη κατηγοριών σε μία τοποθεσία

Αφού έχουν δημιουργηθεί τα tags για κάθε κατηγορία, έπειτα γίνεται μία προσπάθεια κατηγοριοποίησης των τοποθεσιών που υπάρχουν στην εφαρμογή. Στην προσπάθεια αυτή, αρχικά δημιουργείται μία λίστα που αποτελείται από τις λέξεις που βρίσκονται στο όνομα και την περιγραφή της τοποθεσίας. Έπειτα, χρησιμοποιείται το REST call που περιγράφηκε στην ενότητα 3.6.2 για να αναζητηθούν επιπλέον πληροφορίες για την εκάστοτε τοποθεσία από το Graph Places API του **Facebook**. Προστίθενται στην προηγούμενη λίστα οι λέξεις που περιλαμβάνονται στο όνομα και την περιγραφή από το αποτέλεσμα που επέστρεψε το API του Facebook. Επιπρόσθετα, κρατείται και η κατηγορία που επέστρεψε το Facebook για την εκάστοτε τοποθεσία.

Τέλος, συγκρίνεται η λίστα με τις λέξεις που συλλέχθηκε για την εκάστοτε φωτογραφία με τα tags της κάθε κατηγορίας που υπάρχει στην εφαρμογή. Σε όποιες κατηγορίες υπάρχει κάποια συσχέτιση, καταχωρείται η κατηγορία με την οποία υπήρξε συσχέτιση στις κατηγορίες της τοποθεσίας. Τέλος, ενημερώνεται η βάση δεδομένων με τις κατηγορίες που προστέθηκαν σε κάθε τοποθεσία.

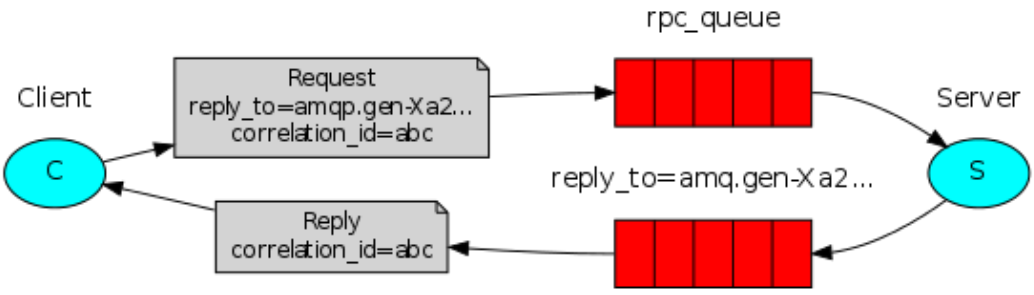
6.4.5 Προβολή δημοφιλών τοποθεσιών ανά εποχή του χρόνου

Όπως αναφέρθηκε και στην ενότητα 5.2.5 μία από τις λειτουργίες της εφαρμογής είναι η προβολή των δημοφιλών τοποθεσιών για την τρέχουσα εποχή του χρόνου. Για την υλοποίηση αυτής της λειτουργίας, έγινε ένα RPC call από την εφαρμογή στον Server. Ο Server λαμβάνοντας τις γεωγραφικές συντεταγμένες, δημιουργεί μία λίστα με όλες τις φωτογραφίες που βρίσκονται σε ακτίνα 50 χιλιομέτρων. Έπειτα, παίρνοντας την ημερομηνία που μεταφορτώθηκε στην εκάστοτε πλατφόρμα (όπως το Flickr)

η κάθε φωτογραφία , κρατούνται οι φωτογραφίες οι οποίες μεταφορτώθηκαν την τρέχουσα εποχή του χρόνου. Έπειτα, αυτές οι φωτογραφίες αντιστοιχίζονται με τις τοποθεσίες που υπάρχουν στη βάση δεδομένων και τελικά επιστρέφονται στην Android εφαρμογή προς προβολή.

Για την υλοποίηση του RPC call, χρησιμοποιήθηκε στις πλευρές του Server και του Android client το RabbitMQ. Ο λόγος που χρησιμοποιήθηκε ένα RPC call αντί για ένα απλό HTTP request είναι πως όλη η διαδικασία εξαγωγής των τοποθεσιών που ζητήθηκαν είναι αρκετά χρονοβόρα. Αν είχε χρησιμοποιηθεί ένα HTTP request τότε θα ήταν αρκετά πιθανό με μεγάλο όγκο δεδομένων στη βάση να υπάρξει timeout στο request.

Η αρχιτεκτονική του RPC call που ακολουθήθηκε παρουσιάζεται στο διάγραμμα 6.8.



Διάγραμμα 6.8: Αρχιτεκτονίκη ενός RPC call στο RabbitMQ

Ο Client στην προκείμενη περίπτωση είναι η εφαρμογή Android η οποία κάνει ένα request σε ένα queue που είναι αφιερωμένο για την λειτουργία της εύρεσης τοποθεσιών για κάθε εποχή του χρόνου. Έπειτα στην εφαρμογή του Server τρέχει ένας worker ο οποίος ακούει στην προαναφερθείσα ουρά. Όταν λάβει ο Server το request, εκτελεί την χρονοβόρα εργασία της εύρεσης των τοποθεσιών για την τρέχουσα εποχή του χρόνου. Όταν ολοκληρωθεί αυτή η εργασία, στέλνει ένα μήνυμα με τα αποτελέσματα στο queue που είχε οριστεί στο **replyTo** του request του Client. Τέλος, ο Client ακούει στο queue που είχε οριστεί στο replyTo και λαμβάνει το αποτέλεσμα από τον Server. Τέλος, ο Client (η εφαρμογή Android) παρουσιάζει το αποτέλεσμα με τις τοποθεσίες στην αρχική σελίδα της εφαρμογής όπως φαίνεται και στο διάγραμμα 5.7.

6.4.6 Έναρξη ταξιδιού χρήστη

Όπως παρουσιάστηκε και στην ενότητα 5.2.3, μία πολύ ενδιαφέρουσα λειτουργία της εφαρμογής Android είναι η δυνατότητα των χρηστών να οργανώνουν τα ταξίδια τους. Ανάμεσα στην οργάνωση των ταξιδιών, ο χρήστης έχει τη δυνατότητα να ξεκινήσει ένα ταξίδι που είχε οργανώσει. Ξεκινώντας ένα ταξίδι, η εφαρμογή προβάλλει στον χρήστη ένα χάρτη με τις τοποθεσίες οι οποίες είναι αριθμημένες. Αυτό αναπαρίσταται στο διάγραμμα 5.6. Η εφαρμογή προτείνει τη βέλτιστη διαδρομή που μπορεί να ακολουθήσει ο χρήστης, εξ ου και η αρίθμηση στις τοποθεσίες.

Για την εύρεση της βέλτιστης διαδρομής, αναπτύχθηκε ένας απλός αλγόριθμος στην εφαρμογή του backend. Ο αλγόριθμος αυτός παίρνει τη λίστα με τις τοποθεσίες και το σημείο εκκίνησης. Αρχικά ξεκινάει συγκρίνοντας την απόσταση του σημείου εκκίνησης από όλα τα υπόλοιπα σημεία. Παίρνει το σημείο με την μικρότερη απόσταση και το προσθέτει στη λίστα της διαδρομής. Έπειτα συγκρίνει την απόσταση του τελευταίου σημείου της διαδρομής με τα υπόλοιπα που μένουν και διαλέγει πάλι αυτό με τη μικρότερη απόσταση το οποίο το προσθέτει στη λίστα της διαδρομής που θα ακολουθηθεί. Κάθε φορά που προστίθεται ένα σημείο στη λίστα των σημείων της προτεινόμενης διαδρομής, τότε αφαιρείται από τη λίστα με τα σημεία. Αυτή η διαδικασία εκτελείται έως ότου η λίστα με τα σημεία μείνει κενή.

Παρακάτω βρίσκεται η υλοποίηση του αλγορίθμου που περιγράφηκε και έγινε στη γλώσσα Java.

```
1  public static void sortPlacesForShortestPath(List<PlaceDto>
       places, PlaceDto startingPoint) {
2
3      if (places == null || places.isEmpty()) {
4          return;
5      }
6      List<PlaceDto> placeDtos = new ArrayList <>(places);
7      List<PlaceDto> visitedPlaces = new LinkedList <>();
8
9      double shortest = Integer.MAX_VALUE;
10     PlaceDto shortestPlace = null;
11     PlaceDto nextPlace = startingPoint;
```

```
12      visitedPlaces.add(startingPoint);
13
14      while (!placeDtos.isEmpty()) {
15          for (PlaceDto placeDto: placeDtos) {
16              double dist = LocationUtil.haversineDistanceInKm(
                    nextPlace.getLatitude(), nextPlace.getLongitude
                    (), placeDto.getLatitude(), placeDto.
                    getLongitude());
17              if (dist < shortest) {
18                  shortestPlace = placeDto;
19                  shortest = dist;
20              }
21          }
22          placeDtos.remove(shortestPlace);
23          visitedPlaces.add(shortestPlace);
24          nextPlace = shortestPlace;
25          shortest = Integer.MAX_VALUE;
26      }
27
28      places.clear();
29      places.addAll(visitedPlaces);
30  }
```

Όπως φαίνεται και στον αλγόριθμο, η μέθοδος δέχεται μία λίστα με τα σημεία και ένα σημείο εκκίνησης και επιστρέφει τη λίστα ταξινομημένη βάσει της βέλτιστης διαδρομής. Οπότε, η εφαρμογή του Android παίρνει αυτή την ταξινομημένη λίστα και αριθμεί τα σημεία της λίστας στο χάρτη. Επίσης και σε αυτή την περίπτωση, για την εύρεση της απόστασης μεταξύ 2 σημείων χρησιμοποιείται η φόρμουλα haversine 6.1 που περιγράφηκε παραπάνω.

6.4.7 Αποστολή push notifications στον χρήστη

Στην ενότητα 5.2.6 αναφέρθηκε η λειτουργία της αποστολής push notifications στην συσκευή Android του χρήστη. Για την αποστολή των push notifications, χρησι-

μοποιήθηκε το Firebase Cloud Messaging, μία υπηρεσία της Google.

Πιο συγκεκριμένα, στην εφαρμογή του back-end αναπτύχθηκε ένα Thread το οποίο κάθε δώδεκα ώρες αναζητεί τα ταξίδια χρηστών τα οποία είναι προγραμματισμένα σε λιγότερο από 7 ημέρες. Έπειτα, δημιουργεί ένα *UserNotification* το οποίο περιλαμβάνει τις πληροφορίες για την ειδοποίηση όπως τον χρήστη, το ταξίδι που αφορά, το κείμενο που θα αναγράφεται στην ειδοποίηση και διάφορες άλλες πληροφορίες. Στη συνέχεια, αυτό το *UserNotification* αποστέλλεται σε μια ουρά του RabbitMQ. Ο Παραλήπτης που "ακούει" σε αυτή την ουρά είναι κι αυτός με τη σειρά του στην εφαρμογή του back-end. Ο παραλήπτης όταν λάβει ένα μήνυμα από την ουρά στέλνει την ειδοποίηση για τον χρήστη στην υπηρεσία του FCM η οποία με την σειρά της στέλνει το push notification στην Android συσκευή του χρήστη. Αξίζει να σημειωθεί πως η εφαρμογή κάνει τους απαραίτητους ελέγχους ώστε ο χρήστης να μην λαμβάνει παραπάνω από μία ειδοποιήσεις την ημέρα, γεγονός που σε αντίθετη περίπτωση ενδεχομένως να ήταν ενοχλητικό. Τέλος, απαραίτητη προϋπόθεση για να σταλεί ένα μήνυμα στην υπηρεσία του FCM είναι να παρέχεται και το device token το οποίο είναι το χαρακτηριστικό ταυτοποίησης της συσκευής Android του χρήστη.

Η εφαρμογή του back-end αποκτά αυτό το device token από την Android εφαρμογή. Κατά την πρώτη εκτέλεση της εφαρμογής, παράγεται ένα event όπου παρέχει στην Android εφαρμογή αυτό το device token, το οποίο αποστέλλεται στην εφαρμογή του Server μέσω του REST API με την προϋπόθεση ότι ο χρήστης έχει πραγματοποιήσει σύνδεση στο λογαριασμό του. Έπειτα, η εφαρμογή του Server αποθηκεύει αυτό το device token στον εκάστοτε χρήστη. Επειδή αυτό το device token αποτελεί ευαίσθητη πληροφορία, κατά την αποθήκευση του στη βάση δεδομένων κρυπτογραφείται. Αντίστοιχα, όταν χρειαστεί να χρησιμοποιηθεί ή να διαβαστεί αποκρυπτογραφείται.

Κεφάλαιο 7

Πειράματα

Η εφαρμογή σε αρκετές λειτουργίες παρουσίασε διάφορα δεδομένα στον χρήστη, όπως για παράδειγμα οι 10 δημοφιλέστερες τοποθεσίες μίας πόλης. Ωστόσο, ήταν αναγκαίο να ελεγχθεί η εγκυρότητα των δεδομένων αυτών που παρουσιάζονται στον χρήστη. Για τους ελέγχους αυτούς χρησιμοποιήθηκαν διάφορες υπηρεσίες και τεχνικές, όπως περιγράφεται παρακάτω.

7.1 Έλεγχος αλγορίθμου εξαγωγής δέκα δημοφιλέστερων τοποθεσιών

Όπως έχει προαναφερθεί, η εφαρμογή μέχρι στιγμής υποστηρίζει την πόλη της Θεσσαλονίκης. Οπότε και στην λειτουργία της εύρεσης των δέκα δημοφιλέστερων τοποθεσιών, βρέθηκαν αυτές για την πόλη της Θεσσαλονίκης.

Συνοπτικά, όπως αναλύθηκε και στην ενότητα 6.4.1 η εύρεση των δέκα δημοφιλέστερων τοποθεσιών έγινε βρίσκοντας τοποθεσίες στις οποίες υπήρχε μεγάλη πυκνότητα φωτογραφιών.

Πιο συγκεκριμένα, οι δέκα δημοφιλέστερες τοποθεσίες βάσει της εφαρμογής για την πόλη της Θεσσαλονίκης είναι οι παρακάτω:

Πόλη	Latitude	Longitude	Περιγραφή
Θεσσαλονίκη	40.626194	22.94778	Λευκός Πύργος
Θεσσαλονίκη	40.625213	22.954108	Αρχαιολογικό Μουσείο Θεσσαλονίκης
Θεσσαλονίκη	40.633313	22.952842	Ροτόντα
Θεσσαλονίκη	40.634863	22.937066	Οδός Αιγύπτου, Πλατεία Λαδάδικα
Θεσσαλονίκη	40.641858	22.95415	Ιερά Μονή Βλατάδων, Άνω Πόλη
Θεσσαλονίκη	40.62699	22.954602	Διεθνής Έκθεση Θεσσαλονίκης
Θεσσαλονίκη	40.632452	22.947036	Ιερός Ναός Αγίας Σοφίας
Θεσσαλονίκη	40.624067	22.949945	Άγαλμα Μεγάλου Αλεξάνδρου, Νέα Παραλία
Θεσσαλονίκη	40.621913	22.951522	Οι Ομπρέλες του Ζογγολόπουλου, Νέα Παραλία
Θεσσαλονίκη	40.602097	22.972908	Zebra espresso bistro &spirits

Πίνακας 7.1: Οι 10 δημοφιλέστερες τοποθεσίες της Θεσσαλονίκης

Από τον πίνακα παραπάνω, φαίνεται πως εκτός από την τελευταία γραμμή που αφορά ένα cafe-bar, όλες οι υπόλοιπες τοποθεσίες είναι διάσημα μνημεία και τοποθεσίες της πόλης της Θεσσαλονίκης. Για παράδειγμα ο "Λευκός Πύργος" είναι το έμβλημα της πόλης και το πιο γνωστό μνημείο της. Αντίστοιχα, εφόσον τα αποτελέσματα παραπάνω εξήχθησαν βάσει του πλήθους των φωτογραφιών που τραβήχτηκαν σε κάθε τοποθεσία, εξηγείται και η τοποθεσία "Οι Ομπρέλες του Ζογγολόπουλου, Νέα Παραλία" η οποία είναι από τα πιο δημοφιλή μέρη για φωτογραφίες στην πόλη.

Στον πίνακα 7.1 φαίνονται οι δημοφιλέστερες τοποθεσίες όπως τις εξήγαγε η εφαρμογή, αφότου εφάρμοσε τον αλγόριθμο DBSCAN στις φωτογραφίες που είχαν συλλεχθεί για την πόλη της Θεσσαλονίκης. Ο αλγόριθμος έβγαλε τα clusters αυτά με την μεγαλύτερη πυκνότητα σε φωτογραφίες. Ως αποτέλεσμα, τα cluster με τα περισσότερα σημεία αφορούσαν και τις πιο δημοφιλείς τοποθεσίες, βάσει των φωτογραφιών που τράβηξαν οι χρήστες.

7.1.1 Ανάλυση πλήθους γεω-αναγραφόμενων φωτογραφιών

Επιπρόσθετα, για να αναλυθούν περαιτέρω τα αποτελέσματα της εφαρμογής και του αλγορίθμου DBSCAN, αναλύθηκαν τα δεδομένα που είχαν συλλεχθεί στη βάση δεδομένων. Πιο συγκεκριμένα, με τη χρήση SQL ερωτημάτων αναλύθηκαν τα δεδομένα

των φωτογραφιών που είχαν συλλεχθεί αφού και ο αλγόριθμος DBSCAN εφαρμόστηκε στις φωτογραφίες αυτές. Πιο συγκεκριμένα, ελέγχθηκαν μία προς μία οι τοποθεσίες που εξήχθησαν ως οι δημοφιλέστερες και με τη χρήση ενός SQL ερωτήματος όπως το 7.1 βγήκε το πλήθος των φωτογραφιών που βρίσκονται πολύ κοντά στην εκάστοτε τοποθεσία. Τα αποτελέσματα της ανάλυσης των δεδομένων της βάσεως δεδομένων βρίσκονται παρακάτω:

Δημοτικότητα	Τοποθεσία	% Φωτογραφιών
1	Zebra espresso bistro &spirits	7.93%
2	Αρχαιολογικό Μουσείο Θεσσαλονίκης	6.27%
3	Λευκός Πύργος	5.87%
4	Διεθνής Έκθεση Θεσσαλονίκης	4.54%
5	Ροτόντα	4.16%
6	Οδός Αιγύπτου, Πλατεία Λαδάδικα	2.97%
7	Ιερός Ναός Αγίας Σοφίας	2.43%
8	Ιερά Μονή Βλατάδων, Άνω Πόλη	1.99%
9	Άγαλμα Μεγάλου Αλεξάνδρου, Νέα Παραλία	1.38%
10	Οι Ομπρέλες του Ζογγολόπουλου, Νέα Παραλία	1.08%
		38.62%

Πίνακας 7.2: Οι 10 δημοφιλέστερες τοποθεσίες της Θεσσαλονίκης και το ποσοστό των φωτογραφιών που τραβήχτηκαν στην εκάστοτε φωτογραφία επί του συνόλου των φωτογραφιών που τραβήχτηκαν στην πόλη της Θεσσαλονίκης

Επίσης, για να γίνει πιο κατανοητή η ανάλυση των παραπάνω δεδομένων αξίζει να σημειωθεί πως το πλήθος των φωτογραφιών που συλλέχθηκαν για την πόλη της Θεσσαλονίκης στην βάση δεδομένων της εφαρμογής, είναι **5718**. Οι φωτογραφίες αυτές συλλέχθηκαν αποκλειστικά από το API του flickr και αφορούν τα τελευταία τρία χρόνια (μέχρι στιγμής).

Παρακάτω βρίσκεται το παραδείγμα ενός SQL ερωτήματος που εκτελέστηκε με την επεξήγησή του.

Listing 7.1: SQL ερώτημα για τις φωτογραφίες με χρήση της φόρμουλας Haversine

```
1     SELECT COUNT(Id) FROM Photo WHERE (6372.8 * 2 * ASIN(SQRT(
```

```
        POWER(SIN(RADIANS(CAST(Latitude AS FLOAT) − CAST(X AS
        FLOAT))/2),2) + POWER(SIN(RADIANS(CAST(Longitude AS
        FLOAT) − CAST(Y AS FLOAT))/2),2) * COS(RADIANS(CAST(X
        AS FLOAT))) * COS(RADIANS(CAST(Y AS FLOAT)))))) < 0.15;
2
3       X: τογεωγραφικόπλάτοςτηςτοποθεσίας
4       Y: τογεωγραφικόμήκος
```

7.1.2 Αξιολόγηση των αποτελεσμάτων

Είναι γεγονός όπως αναφέρθηκε και προηγουμένως πως οι περισσότερες τοποθεσίες που εξήχθησαν από την εφαρμογή σαν δημοφιλείς, αποτελούν είτε αρχαιολογικά μνημεία είτε σημεία μεγάλου ενδιαφέροντος. Ωστόσο, η τοποθεσία που βγήκε ως η πιο δημοφιλής είναι ένα μικρό cafe-bar. Ο λόγος που συνέβη αυτό είναι πως ανέβηκε μεγάλο πλήθος φωτογραφιών από εκείνη την τοποθεσία (454 μέχρι στιγμής για την ακρίβεια). Επομένως, βγαίνει το συμπέρασμα πως τα αποτελέσματα της εφαρμογής TripAssistant επηρεάζονται απόλυτα από τους χρήστες των APIs από τα οποία η εφαρμογή συλλέγει τα δεδομένα της.

7.2 Έλεγχος προτεινόμενης διαδρομής στο χρήστη για τις τοποθεσίες ενός ταξιδιού

Όπως περιγράφηκε στην ενότητα 6.4.6 η εφαρμογή προτείνει στον χρήστη τη βέλτιστη διαδρομή που μπορεί να ακολουθήσει για να επισκεφθεί τις τοποθεσίες που έχει αποθηκεύσει σε ένα ταξίδι του. Αυτό επιτυγχάνεται μέσω του αλγορίθμου που αναπτύχθηκε και περιγράφηκε στην ενότητα 6.4.6. Ο αλγόριθμος φυσικά λαμβάνει υπόψιν του την αρχική θέση του χρήστη πέρα από τις γεωγραφικές συντεταγμένες της κάθε τοποθεσίας. Επειδή ο αλγόριθμος αναπτύχθηκε εξ ολοκλήρου από την αρχή βάσει λογικής και δεν χρησιμοποιήθηκε κάποιος προϋπάρχον, υπάρχει και σε αυτή την περίπτωση η ανάγκη επιβεβαίωσης των αποτελεσμάτων του αλγορίθμου.

Για τον σκοπό αυτό, χρησιμοποιήθηκε μεταξύ άλλων και η υπηρεσία των Google Maps όπως περιγράφεται παρακάτω.

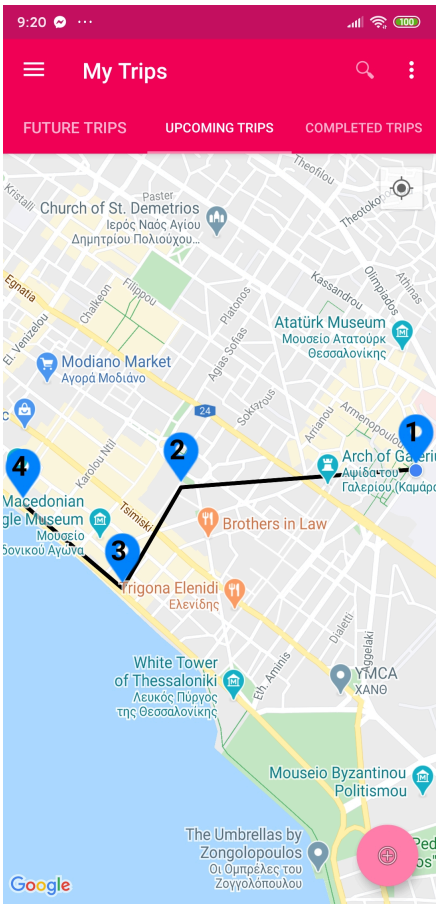
7.2.1 Επαλήθευση αποτελεσμάτων διαδρομής

Για την επαλήθευση των αποτελεσμάτων χρησιμοποιήθηκαν τα δεδομένα του πίνακα 7.3.

#	Latitude	Longitude
A	40.631749939012	22.94103699804
B	40.629498095075	22.944526988045
C	40.63219	22.94658
Αρχική θέση χρήστη		
I	40.632672, 22.954816	

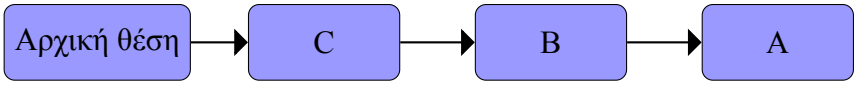
Πίνακας 7.3: Δεδομένα γεωγραφικών τοποθεσιών για την επαλήθευση των αποτελεσμάτων του αλγορίθμου πρότασης βέλτιστης διαδρομής

Για τα δεδομένα του παραπάνω πίνακα, δημιουργήθηκε ένα "ταξίδι" στην εφαρμογή **Trip Assistant** και χρησιμοποιήθηκε η λειτουργία της προτεινόμενης βέλτιστης διαδρομής. Οπότε η εφαρμογή πρότεινε ως βέλτιστη τη διαδρομή που φαίνεται στην φωτογραφία 7.1



Διάγραμμα 7.1: Προτεινόμενη διαδρομή μετά από έναρξη ταξιδιού

Για να βεβαιωθεί ότι κάθε φορά ο αλγόριθμος που περιγράφηκε στην ενότητα 6.4.6 επιστρέφει το ίδιο αποτέλεσμα, εκτελέστηκε 100,000 φορές. Επιβεβαιώθηκε πως κάθε φορά επέστρεψε το ίδιο αποτέλεσμα. Ο αλγόριθμος επέστρεψε μία λίστα με τοποθεσίες ταξινομημένη με την σειρά την οποία πρέπει να επισκεφθεί ο χρήστης την κάθε τοποθεσία. Η λίστα αυτή, φυσικά περιλαμβάνει και την αρχική θέση του χρήστη και αναπαρίσταται από το διάγραμμα 7.2. Επίσης τα σημεία αυτά χρησιμοποιούν ως αναφορά τις εγγραφές του πίνακα 7.3.



Διάγραμμα 7.2: Βέλτιστη διαδρομή μεταξύ των τοποθεσιών

Έπειτα, για να επιβεβαιωθεί ότι η διαδρομή που προτάθηκε ήταν όντως η βέλτιστη, αναλύθηκαν όλες οι πιθανές διαδρομές με τη συνολική τους απόσταση. Τα αποτελέσματα της ανάλυσης περιγράφονται παρακάτω στον πίνακα 7.4.

Τοπ. 1	Τοπ. 2	Τοπ. 3	Τοπ. 4	Συνολική απόσταση
I	A	B	C	2.3km
I	A	C	B	2.5km
I	B	A	C	2.3km
I	B	C	A	2.3km
I	C	A	B	2.0km
I	C	B	A	1.7km

Πίνακας 7.4: Αποτέλεσμα ανάλυσης βέλτιστης διαδρομής για τις δοσμένες γεωγραφικές συντεταγμένες

Όπως φαίνεται και στον πίνακα παραπάνω, η διαδρομή που προτάθηκε από την εφαρμογή είναι και η βέλτιστη. Επιπλέον, αξίζει να σημειωθεί πως για την εξαγωγή της συνολικής απόστασης για κάθε πιθανή διαδρομή χρησιμοποιήθηκε το Directions API από τους χάρτες της Google. Στο συγκεκριμένο API, επιλέχθηκε η μετάβαση στα επιλεγμένα σημεία ως πεζός. Το Directions API λαμβάνει υπόψιν του τους δρόμους και τα εμπόδια που θα εντοπιστούν στην διαδρομή, οπότε προσαρμόζεται ανάλογα η συνολική απόσταση. Αντίθετα, ο αλγόριθμος που αναπτύχθηκε υπολογίζει την απόσταση μεταξύ δύο σημείων σε απόλυτη ευθεία. Σε κάποιες πολύ συγκεκριμένες και λίγες περιπτώσεις, ο αλγόριθμος θα βγάλει λανθασμένο αποτέλεσμα καθώς θα υπάρξουν εμπόδια που θα αναγκάσουν το χρήστη να διανύσει μεγαλύτερη απόσταση.

7.3 Έλεγχος κατηγοριοποίησης τοποθεσιών

Στην ενότητα 6.4.4 περιγράφηκε ο τρόπος κατά τον οποίο γίνεται η κατηγοριοποίηση των τοποθεσιών. Ωστόσο, ήταν αναγκαίο να ελεγχθεί εάν η μέθοδος που χρησιμοποιήθηκε για να μπουν οι κατηγορίες στις τοποθεσίες έβγαζε λογικά και σωστά αποτελέσματα.

Παίρνοντας τα δεδομένα των τοποθεσιών του πίνακα 7.3, οι κατηγορίες για την κάθε τοποθεσία βρίσκονται παρακάτω

#	Τοποθεσία	Latitude	Longitude	Κατηγορία
1	Ernest Hebrard	40.631749939012	22.94103699804	Coffee, Entertainment
2	Esatto	40.629498095075	22.944526988045	Coffee, Nightlife
3	Cin Cin	40.63219	22.94658	Food, Coffee

Πίνακας 7.5: Επαλήθευση κατηγοριοποίησης τοποθεσιών

Για να ελεγχθεί εάν η εφαρμογή πρόσθεσε τις κατάλληλες κατηγορίες στις παραπάνω τοποθεσίες, έγινε μία διασταύρωση με την εφαρμογή του Facebook.

1. Η τοποθεσία #1 του πίνακα 7.5 κατά το Facebook αφορά μία Καφετέρια όπως πολύ σωστά επεσήμανε και η εφαρμογή. Ωστόσο, η εφαρμογή πρόσθεσε στην τοποθεσία αυτή και την κατηγορία "Διασκέδαση". Η κατηγορία αυτή ενδεχομένως να μην αφορά άμεσα την τοποθεσία αυτή.

2. Η τοποθεσία #2 κατά το Facebook αφορά μία Καφετέρια αλλά κατά το TripAdvisor κατατάσσεται στην κατηγορία της Νυχτερινής Ζωής. Οπότε, όπως φαίνεται οι κατηγορίες που πρόσθεσε η εφαρμογή TripAssistant στην τοποθεσία αυτή είναι σωστές.

3. Η τοποθεσία #3 κατά το Facebook αποτελεί ένα Κοκτέιλ Μπαρ, Εστιατόριο και Καφέ. Όπως φαίνεται και σε αυτή την τοποθεσία, οι κατηγορίες που πρόσθεσε η εφαρμογή είναι σωστές και λογικές.

Όπως αναλύθηκε και παραπάνω, η εφαρμογή TripAssistant εφάρμοσε σωστή κατηγοριοποίηση στις τοποθεσίες, με κάποιες μικρές παρεκκλίσεις όπου προστέθηκαν σε μία τοποθεσία κατηγορίες που δεν ήταν σχετικές με την τοποθεσία αυτή.

Κεφάλαιο 8

Συμπεράσματα

8.1 Μελλοντική ανάπτυξη

Είναι γεγονός πως δεν αναπτύχθηκαν όλες οι λειτουργίες που είχαν προγραμματιστεί για την εφαρμογή της πτυχιακής εργασίας, καθώς ο χρόνος δεν επαρκούσε. Οι λειτουργίες που δεν ολοκληρώθηκαν περιγράφονται παρακάτω στα σχέδια για τη μελλοντική ανάπτυξη της εφαρμογής. Κάποιες από αυτές τις λειτουργίες είχαν σχεδιαστεί εξ αρχής να υλοποιηθούν, ενώ κάποιες άλλες προέκυψαν κατά τη διάρκεια της ανάπτυξης.

8.1.1 Υποστήριξη περισσότερων πόλεων

Όπως έχει αναφερθεί και σε προηγούμενες ενότητες, η εφαρμογή την παρούσα στιγμή υποστηρίζει μόνο την πόλη της Θεσσαλονίκης. Σε περίπτωση που χρησιμοποιηθεί η εφαρμογή από χρήστη που βρίσκεται εκτός της Θεσσαλονίκης, τότε θα λάβει τα αντίστοιχα μηνύματα που θα ενημερώνουν πως οι εκάστοτε λειτουργίες δεν είναι διαθέσιμες. Υποστηρίζοντας περισσότερες πόλεις στην εφαρμογή TripAssistant, αυτομάτως διευρύνεται και το κοινό στο οποίο απευθύνεται.

Ο τρόπος που υλοποιήθηκε η υποστήριξη των πόλεων στην εφαρμογή, καθιστά πολύ εύκολη την επέκταση των υποστηριζόμενων πόλεων. Η κύρια αλλαγή είναι να προστεθεί το όνομα της νέας πόλης μαζί με τις γεωγραφικές συντεταγμένες της σε ένα συγκεκριμένο enum. Φυσικά, αυτή η διαδικασία θα μπορούσε να απλουστευθεί περαιτέρω προσθέτοντας νέες πόλεις στην εφαρμογή μέσω ενός πίνακα ελέγχου. Με

αυτό τον τρόπο δεν θα χρειάζονταν νέες αλλαγές στον πηγαίο κώδικα της εφαρμογής καθώς και δημιουργία μίας νέας έκδοσης.

8.1.2 Βελτίωση της λειτουργίας έναρξης ταξιδιού χρήστη

Μία πολύ ενδιαφέρουσα λειτουργία που αναπτύχθηκε είναι αυτή που περιγράφεται στην ενότητα 6.4.6. Ο χρήστης έχει τη δυνατότητα να ξεκινήσει ένα ταξίδι που έχει προγραμματίσει και η εφαρμογή προτείνει την πιο γρήγορη διαδρομή στον χρήστη.

Μία αλλαγή που μπορεί να γίνει στο μέλλον είναι η βελτίωση της περιήγησης στους χάρτες αφότου έχει γίνει έναρξη ενός ταξιδιού. Πιο συγκεκριμένα, αυτή τη στιγμή δεν υπάρχει ζωντανή ενημέρωση για την εξέλιξη του ταξιδιού και για τον αριθμό των τοποθεσιών που έχει επισκεφθεί ήδη ο χρήστης. Αυτό θα μπορούσε να βελτιωθεί δείχνοντας ζωντανά την τρέχουσα τοποθεσία του χρήστη στο χάρτη. Επίσης, οι τοποθεσίες που έχει ήδη επισκεφθεί ο χρήστης θα ήταν χρήσιμο να λαμβάνουν τη σχετική ένδειξη στον χάρτη. Τέλος, η εφαρμογή θα μπορούσε να δώσει την ευκαιρία στον χρήστη να διαγράψει τοποθεσίες από ένα ταξίδι το οποίο έχει ήδη ξεκινήσει.

8.1.3 Δημιουργία chat για συνομιλία μεταξύ των χρηστών

Στον αρχικό σχεδιασμό των λειτουργιών της εφαρμογής, είχε προστεθεί και η δημιουργία chat rooms μέσα στην εφαρμογή ώστε οι χρήστες να συνομιλούν μεταξύ τους και να ανταλλάσσουν ιδέες και προτάσεις για τις τοποθεσίες μίας πόλης. Θα υπάρχει ένα chat room για κάθε πόλη που υποστηρίζει η εφαρμογή, ώστε ένας χρήστης να έχει τη δυνατότητα να ζητήσει προτάσεις από τοποθεσίες που θα μπορούσε να επισκεφθεί με το ταξίδι του στην πόλη που επιθυμεί.

Φυσικά, η συγκεκριμένη λειτουργία μπορεί να αναπτυχθεί και επεκταθεί αρκετά. Μία ιδέα είναι ότι μπορεί να δοθεί η ευκαιρία στον χρήστη να ακολουθήσει τα chat rooms που τον ενδιαφέρουν ώστε να λαμβάνει ειδοποιήσεις από τα μηνύματα που στέλνονται σε αυτό. Επίσης, θα μπορούσε ένας χρήστης να δημιουργήσει μια προσωπική συνομιλία με έναν άλλο χρήστη της εφαρμογής.

8.1.4 Σύνδεση με το Wikipedia για λήψη πληροφοριών

Μία λειτουργία που θα ενίσχυε την εμπειρία του χρήστη είναι και η σύνδεση με το API του Wikipedia ώστε να λαμβάνονται πληροφορίες για τις τοποθεσίες που επιθυμεί να επισκεφτεί ο χρήστης.

Μέχρι στιγμής, ένας χρήστης έχει στη διάθεσή του το όνομα και τις φωτογραφίες της τοποθεσίας. Ωστόσο, αυτές οι πληροφορίες ενδεχομένως να μην είναι αρκετές για να καταλάβει ένας χρήστης εάν τον ενδιαφέρει να επισκεφτεί την εκάστοτε τοποθεσία οπότε ίσως χρειαστεί να αναζητήσει περαιτέρω πληροφορίες για την τοποθεσία από μία τρίτη εφαρμογή. Πέραν τούτου, κατά την διάρκεια της πλοήγησης ενός χρήστη μεταξύ των τοποθεσιών ενός ταξιδιού του θα ήταν χρήσιμο να βλέπει πληροφορίες για τις τοποθεσίες που επισκέπτεται.

Συνδέοντας την εφαρμογή με το API του Wikipedia, θα μπορούσαν να ληφθούν οι σχετικές πληροφορίες για την εκάστοτε τοποθεσία. Για παράδειγμα, ένας χρήστης έχει επισκεφθεί ένα ιστορικό κτήριο και κατά τη διάρκεια της επίσκεψής του διαβάζει μέσω της εφαρμογής TripAssistant πληροφορίες για το κτήριο αυτό.

8.2 Συμπεράσματα

Ύστερα από την περάτωση της παρούσας πτυχιακής εργασίας βγήκαν κάποια ενδιαφέροντα συμπεράσματα για τη χρησιμότητα των δεδομένων από τα δημόσια APIs. Πιο συγκεκριμένα, στα πλαίσια της εφαρμογής της πτυχιακής χρησιμοποιήθηκαν κάποια δεδομένα από δημόσια APIs από τα οποία μπορέσαμε να εξάγουμε πολλές πληροφορίες οι οποίες είναι χρήσιμες για έναν τουρίστα. Με τα δεδομένα αυτά μπορούν να εξαχθούν προτάσεις για μέρη που μπορεί να επισκεφθεί ένας τουρίστας, να ληφθούν πληροφορίες για την ιστορία ενός μνημείου όπως και πολλά άλλα.

Φυσικά, τα δεδομένα από τα δημόσια APIs δεν είναι χρήσιμα μόνο για την εξαγωγή τουριστικών πληροφοριών. Τα δεδομένα που εξάγονται από τα δημόσια APIs μπορούν να χρησιμεύσουν σε πληθώρα εφαρμογών οι οποίες περιορίζονται μόνο από την φαντασία των προγραμματιστών που θα τις αναπτύξουν. Επιπλέον, ο συνδυασμός των δεδομένων που παρέχονται από διαφορετικά δημόσια APIs μπορούν να συνδυαστούν ώστε να δημιουργήσουν ένα μεγαλύτερο και ισχυρότερο σύνολο από δεδομένα.

Μία ακόμα διαπίστωση που έγινε μετά την περάτωση της πτυχιακής εργασίας είναι η χρησιμότητα εφαρμογής των ευέλικτων μεθοδολογιών στην ανάπτυξη ενός project. Πιο συγκεκριμένα, η χρήση της μεθοδολογίας Scrum και η οργάνωση των επαναλήψεων (Sprints) βοήθησε στην σωστή οργάνωση του project αλλά και στη συνεχή παράδοση νέων εκδόσεων και λειτουργιών. Έχοντας κάποιες συγκεκριμένες λειτουργίες προς ανάπτυξη σε κάθε επανάληψη, είναι ορισμένη η δουλειά που χρειάζεται να γίνει και συνήθως αναπτύσσονται οι λειτουργίες στον προβλεπόμενο χρόνο. Αυτό έχει ως αποτέλεσμα τη συνεχή παράδοση νέων εκδόσεων της εφαρμογής, παίρνοντας έτσι συχνά την γνώμη των ιδιοκτητών του project. Σε αντίθετη περίπτωση όπου δεν χρησιμοποιείται μια ευέλικτη μεθοδολογία, η ομάδα και οι προγραμματιστές συνήθως καθυστερούν την παράδοση των εκδόσεων του λογισμικού έχοντας ως αποτέλεσμα πολλές φορές πολλές φορές την μη ικανοποίηση των ιδιοκτητών του project για το τελικό αποτέλεσμα.

Τέλος, πολύ χρήσιμα στον κύκλο ζωής ενός έργου φάνηκαν και τα συστήματα Continuous Integration/Continuous Delivery. Χρησιμοποιώντας ένα τέτοιο σύστημα διασφαλίζεται και βελτιώνεται σημαντικά η ποιότητα του λογισμικού που παραδίδεται.

Γλωσσάρι

API Application Programming Interface

WSDL Web Services Description Language

WS Web Service

RPC Remote Call Procedure

HTTP Hypertext Transfer Protocol

FCM Firebase Cloud Messaging

REST Representational State Transfer

SQL Structured Query Language

JSON JavaScript Object Notation

URL Uniform Resource Locator (Μία διεύθυνση ενός πόρου του παγκοσμίου Ιστού)

Βιβλιογραφία

[1] Cecep Nurul Alam κ.ά. «Implementation of haversine formula for counting event visitor in the radius based on Android application». Στο: *2016 4th International Conference on Cyber and IT Service Management*. IEEE. 2016, σσ. 1–6.

[2] *Android Platform Architecture*. `https://developer.android.com/guide/platform`. Accessed: 2020-02-18.

[3] Daniela Berardi, Diego Calvanese και Giuseppe De Giacomo. «Reasoning on UML class diagrams». Στο: *Artificial intelligence* 168.1-2 (2005), σσ. 70–118.

[4] Tim Davies. «How might open data contribute to good governance». Στο: *Commonwealth Governance Handbook* 13 (2012), σσ. 148–150.

[5] Roy Thomas Fielding. «Architectural Styles and the Design of Network-based Software Architectures». Ph.D. dissertation. University of California, 2000.

[6] Open Knowledge Foundation. *Τι είναι τα Ανοιχτά Δεδομένα;* 2019. URL: `https://opendatahandbook.org/guide/el/what-is-open-data/` (επίσκεψη 11/06/2011).

[7] *Hibernate ORM*. `https://hibernate.org/orm/`. Accessed: 2020-02-22.

[8] Michael Karlesky και Mark Vander Voord. «Agile project management». Στο: *ESC* 247.267 (2008), σ. 4.

[9] Martin Micah Martin C. Robert. *Agile Principles, Patterns, and Practices in C#*. Prentice Hall, 2006.

[10] OprahMagazine. *12 Travel Apps That'll Make Your Next Adventure Easier*. 2019. URL: `https://www.oprahmag.com/life/g25563106/best-travel-apps/` (επίσκεψη 11/06/2011).

[11] Tom Preston-Werner. «Semantic Versioning 2.0. 0». Στο: *ĺinea]. Available: http://semver. org* (2013).

[12] *RabbitMQ Features*. `https://www.rabbitmq.com`. Accessed: 2020-02-23.

[13] Ken Schwaber. *Agile project management with Scrum*. Microsoft press, 2004.

[14] Alexander Smirnov κ.ά. «Mobile application for guiding tourist activities: tourist assistant-tais». Στο: *Proceedings of 16th Conference of Open Innovations Association FRUCT*. IEEE. 2014, σσ. 95–100.

[15] *Spring Security*. `https://spring.io/projects/spring-security`. Accessed: 2020-02-22.

[16] Kishor Wagh και Ravindra Thool. «A comparative study of soap vs rest web services provisioning techniques for mobile host». Στο: *Journal of Information Engineering and Applications* 2.5 (2012), σσ. 12–16.

[17] Ganda Yoga Swara κ.ά. «Implementation of Haversine Formula and Best First Search Method in Searching of Tsunami Evacuation Route». Στο: *IOP Conference Series: Earth and Environmental Science*. Τόμ. 97. 1. 2017, σ. 012004.

Η εργασία αυτή στοιχειοθετήθηκε με το πρόγραμμα XeLaTeX. Για τη στοιχειοθέτηση της βιβλιογραφίας χρησιμοποιήθηκε το πρόγραμμα `biber` και `biblatex`. Οι γραμματοσειρές που χρησιμοποιήθηκαν είναι οι Times New Roman και `Courier New`. Ο πηγαίος κώδικας της εφαρμογή που αναπτύχθηκε για την εργασία αυτή διατίθεται υπό την άδεια ανοιχτού λογισμικού `Apache License 2.0` και βρίσκεται στον σύνδεσμο `github.com/kostasmantz/trip-assistant`.