ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εφαρμογή Συνεπιβατισμού για Έξυπνα Τηλέφωνα

Διπλωματική Εργασία

Ζαφείριος Μηλούσης

Επιβλέπων: Γεώργιος Θάνος, Ε.ΔΙ.Π.

Βόλος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εφαρμογή Συνεπιβατισμού για Έξυπνα Τηλέφωνα

Διπλωματική Εργασία

Ζαφείριος Μηλούσης

Επιβλέπων: Γεώργιος Θάνος, Ε.ΔΙ.Π.

Βόλος 2021

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Carpooling Application for Smartphones

Diploma Thesis

Zafeirios Milousis

Supervisor: Georgios Thanos, Laboratory Teaching Staff

Volos 2021

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων/πουσα	**Θάνος Γεώργιος** Ε.ΔΙ.Π., Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Τουσίδου Ελένη** Ε.ΔΙ.Π., Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Φεύγας Αθανάσιος** Ε.ΔΙ.Π., Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 21-09-2021

Στην οικογένεια και τους φίλους μου

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα, αρχικά, να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας, κύριο Γεώργιο Θάνο, μέλος του Εργαστηριακού Διδακτικού Προσωπικού του Τμήματος για τη συνεχή καθοδήγηση και τις συμβουλές του τόσο στο κομμάτι της ανάπτυξης της εφαρμογής αλλά και στην συγγραφή τούτης της εργασίας.

Τέλος, την οικογένεια και τους φίλους μου που μου συμπαραστάθηκαν, με στήριξαν και με παρακινούσαν καθ' όλη τη διάρκεια της φοίτησής μου στο τμήμα. Σας ευχαριστώ όλους θερμά!

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο/Η Δηλών/ούσα

(Υπογραφή)
Ζαφείριος Μηλούσης
Ημερομηνία: 21-09-2021

ΠΕΡΙΛΗΨΗ

Η ραγδαία εξέλιξη της επιστήμης των υπολογιστών και η ενσωμάτωση των αποτελεσμάτων αυτής στην καθημερινότητα των ανθρώπων, έχουν καταστήσει σχεδόν κάθε είδους πληροφορία εύκολη στην αναζήτηση της. Επιπρόσθετα, πλέον οι υπολογιστές μας προσφέρουν πληροφορίες που δεν έχουμε ζητήσει γνωρίζοντας ότι ίσως τις αναζητήσουμε μελλοντικά. Η συγκεκριμένη εφαρμογή συνεπιβασμού για έξυπνα τηλέφωνα αναπτύχθηκε στο πνεύμα της σημερινής εποχής με σκοπό να προσθέσει άλλον έναν τρόπο μετακίνησης των ανθρώπων εντός και εκτός των πόλεων και την μείωση του κόστους των μετακινήσεων αλλά και των ρύπων προς το περιβάλλον. Αυτό επιτυγχάνεται αφού δίνεται η δυνατότητα στους ταξιδιώτες με κοινούς προορισμούς να μοιραστούν τα οχήματά τους και να τα εκμεταλλευτούν με τον βέλτιστο τρόπο. Στην παρούσα εργασία θα αναλυθούν σε βάθος η αρχιτεκτονική του συστήματος που αναπτύχθηκε σε κάθε επίπεδο καθώς και τα εργαλεία που χρησιμοποιήθηκαν προκειμένου να συμπεριληφθούν λειτουργίες εξωτερικών υπηρεσιών. Επιπλέον, θα παρουσιαστούν σενάρια στα οποία η εφαρμογή θα ήταν χρήσιμη προς τους πολίτες και θα συγκριθεί με διαφορετικές λύσεις που έχουν ήδη δημοσιοποιηθεί.

ABSTRACT

The rapid development of computer science and the integration of its results into everyday life of people, have made almost any kind of information easy to search. In addition, our computers now offer information that we have not requested, knowing that we may be searching for it in the future. This carpooling application for smartphones was developed with the needs of modern society in mind to add another way of transporting people in and out of cities and reduce the cost of travel and environmental pollution, since mobile vehicles are reduced. This is achieved by giving travelers with common destinations the ability to share their vehicles and take advantage of them optimally. This paper will analyze in depth the system architecture developed at each level as well as the tools used to include features of external services. In addition, scenarios will be presented in which the application would be useful to people and will be compared with different solutions that are already public.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ ..... *xiii*

ABSTRACT ..... *xiv*

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ ..... *xvi*

ΕΙΣΑΓΩΓΗ ..... *Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.*

ΣΕΝΑΡΙΑ ΛΕΙΤΟΥΡΓΙΩΝ *4*

ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΛΟΓΙΣΜΙΚΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ *9*

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ *13*

ΥΠΟΣΥΣΤΗΜΑΤΑ *15*

Βάση Δεδομένων PostgreSQL **15**

Server-side Laravel (Back-end). **17**

Framework7 Vue (Front-End) ..... **2**Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.

ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ *35*

ΒΙΒΛΙΟΓΡΑΦΙΑ *37*

ΕΙΣΑΓΩΓΗ

Η συγκεκριμένη εφαρμογή αποτελεί μια υπηρεσία συνεπιβατισμού που αφορά ταξίδια μεταξύ διαφορετικών πόλεων. Στο πλαίσιο αυτής της υπηρεσίας αυτοματοποιείται η διαδικασία εύρεσης διαθέσιμου οχήματος που πρόκειται να εκτελέσει την διαδρομή για την οποία έχει εκφράσει ενδιαφέρον ένας χρήστης – επιβάτης. Ο χρήστης έχει την δυνατότητα να αναζητήσει διαδρομές γράφοντας την πόλη εκκίνησης και προορισμού και να πραγματοποιήσει κράτηση θέσης στο αποτέλεσμα που τον εξυπηρετεί ειδοποιώντας επίσης τον οδηγό στην συσκευή του. Επιπλέον, του δίνεται η δυνατότητα να βαθμολογήσει τον οδηγό της διαδρομής ανάλογα με την εμπειρία που του προσέφερε μετά το πέρας του ταξιδιού και να δει το μέσο όρο αλλά και τα σχόλια που έχουν αφήσει προηγούμενοι επιβάτες για τον εν λόγω οδηγό. Ωστόσο, η υπηρεσία αυτή αυτοματοποιεί και την διαδικασία προσφοράς διαδρομών από τους χρήστες – οδηγούς αφού οι ίδιοι μπορούν να δημιουργήσουν και να υποβάλλουν προς κράτηση διαδρομές που πρόκειται να πραγματοποιήσουν. Κατά την δημιουργία μιας διαδρομής μπορούν να ορίσουν τα σημεία εκκίνησης και προορισμού στον χάρτη μέσω της υπηρεσίας Google Maps, τον αριθμό επιβατών που μπορούν να εξυπηρετήσουν όπως επίσης και την ημερομηνία και ώρα έναρξης της διαδρομής από το σημείο εκκίνησης και ένα μέσω επικοινωνίας μαζί του (τηλέφωνο, email). Τέλος, δίνεται η δυνατότητα να ορίσει εάν η ώρα εκκίνησης είναι οριστική ή μπορεί να μεταβληθεί εάν ο επιβάτης επικοινωνήσει μαζί του. Εκτός από την δημιουργία διαδρομής, ο οδηγός μπορεί επίσης να βαθμολογήσει τους επιβάτες του μετά το πέρας της διαδρομής και να δει τις προαναφερθείσες πληροφορίες που τους αφορούν.

Η εφαρμογή αποτελείται από μία εφαρμογή για κάθε είδους χρήστη και δεν χρεώνει τους χρήστες για καμία υπηρεσία ενώ η ίδια επικεντρώνεται στο μακροπρόθεσμο και προγραμματισμένο συνεπιβατισμό για ταξίδια μεταξύ διαφορετικών πόλεων. Ωστόσο, υπάρχουν και άλλες υπηρεσίες που υλοποιούν παρόμοιες λειτουργίες με τον δικό τους τρόπο και υπό διαφορετική οπτική.

Οι παρακάτω υπηρεσίες χρεώνουν τους χρήστες τους με διαφορετικούς τρόπους η κάθε μία και επικεντρώνονται στον άμεσο συνεπιβατισμό εντός των πόλεων και των γύρω περιοχών.

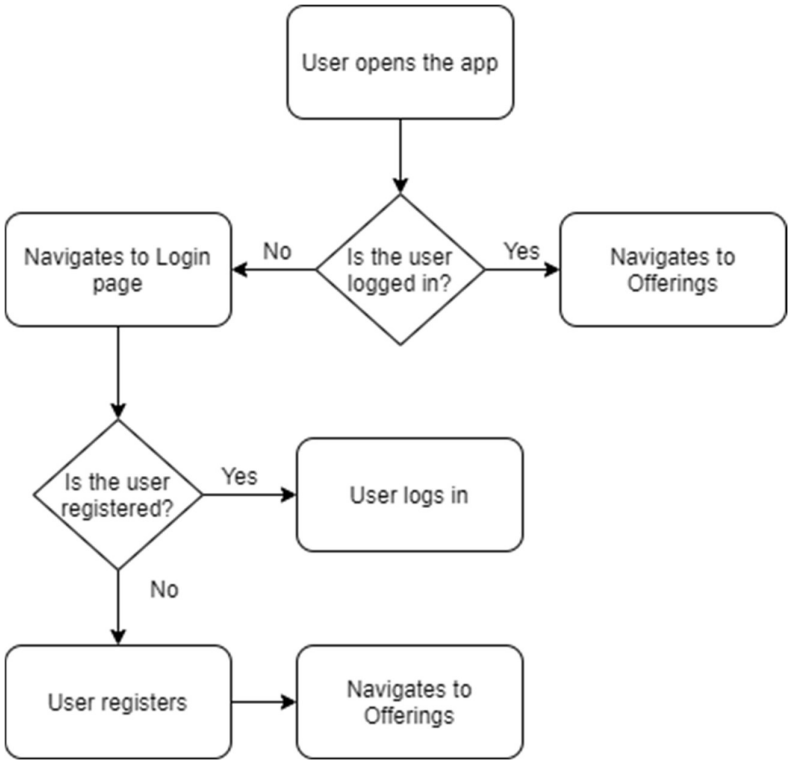
1. **BlaBlaCar**. Η υπηρεσία αυτή έχει τις περισσότερες ομοιότητες με την εφαρμογή σχετικά με το προϊόν που προσφέρει αλλά και με τον τρόπο τον οποίο το προσφέρει. Ιδρύθηκε το 2004 στο Παρίσι της Γαλλίας από τον όμιλο εταιριών Comuto και τους Frederic Mazzela, Francis Nappez, Nicolas Brusson με το όνομα Covoiturage.fr. Πρόκειται επίσης για μια υπηρεσία συνεπιβατισμού μεταξύ χρηστών με στόχο τόσο τις μετακινήσεις εντός πόλης όσο και μεταξύ διαφορετικών πόλεων. Ωστόσο, επικεντρώνεται στην εύρεση διαδρομής την στιγμή της ανάγκης του χρήστη και όχι στον προγραμματισμό μελλοντικών διαδρομών καθώς επίσης οι χρήστες της εφαρμογής που προσφέρουν διαδρομές χρεώνονται με ποσοστό από τα χρήματα που χρέωσαν στους συνεπιβάτες τους. Η υπηρεσία αυτή ωστόσο δεν είχε πάντοτε την μορφή που έχει την χρονική περίοδο ανάπτυξης της εφαρμογής καθώς λειτουργίες της προστίθεντο ανά χρονικά διαστήματα, με σημαντικότερη αυτήν της διαδικτυακής κράτησης στην διαδρομή τον Ιούνιο του 2012. Η προσθήκη αυτής της λειτουργίας βοήθησε την υπηρεσία να επεκταθεί σε περισσότερες χώρες της Ευρώπης τον ίδιο χρόνο, όπως η Ιταλία, η Πολωνία, η Ολλανδία, το Λουξεμβούργο και το Βέλγιο ενώ το 2013 μετονομάστηκε σε BlablaCar.fr.
2. **Uber**. Η πιο διαδεδομένη υπηρεσία συνεπιβατισμού κατά την χρονική περίοδο ανάπτυξης της εφαρμογής. Αποτελείται από πολλές και διαφορετικές υπηρεσίες και μορφές συνεπιβατισμού εκ των οποίων μία συγκλίνει προς την εφαρμογή. Ιδρύθηκε το 2009 από τους Garret Camp και Travis Kalanick στο Σαν Φρανσίσκο των Ηνωμένων Πολιτειών και έχει καταφέρει, τον Ιούνιο του 2021, να εξυπηρετεί 69 χώρες και περίπου 93 εκατομμύρια χρήστες μηνιαίως σε όλον τον κόσμο. Το Uber επικεντρώνεται κυρίως σε επαγγελματίες οδηγούς και εξυπηρετεί τους χρήστες μόνο την στιγμή που έχουν ανάγκη τον συνεπιβατισμό και χρεώνονται το ποσό που θα τους ζητηθεί από τον οδηγό ανάλογα με την απόσταση που διένυσαν, ενώ οι ίδιοι χρεώνονται περίπου το 20% του ποσού που χρέωσαν στους επιβάτες. Ο χρήστης καλεί έναν οδηγό ή ένα ταξί της εταιρίας που είναι διαθέσιμο στην περιοχή που βρίσκεται και αφού ο οδηγός αποδεχτεί το αίτημα πηγαίνει και παραλαμβάνει τον χρήστη από την τοποθεσία του την προκαθορισμένη ώρα. Με αυτόν τον τρόπο, το Uber σκοπεύει να μετατρέψει τους χρήστες σε επαγγελματίες οδηγούς ταξί που ο προορισμός καθορίζεται από τον πελάτη και όχι στο μοίρασμα μιας κοινής διαδρομής ανάμεσα σε δύο χρήστες της εφαρμογής.

3. **Lyft**. Υπηρεσία συνεπιβατισμού στην οποία οι χρήστες διαχωρίζονται ως οδηγοί και επιβάτες και χρησιμοποιούν τις αντίστοιχες εφαρμογές. Ιδρύθηκε το 2012 από τους Logan Green και John Zimmer ως υπηρεσία της Zimride, μιας ακόμα υπηρεσίας συνεπιβατισμού που ίδρυσαν το 2007 και που μετονομάστηκε το 2013 σε Lyft . Το Lyft επίσης επικεντρώνεται κυρίως σε επαγγελματίες οδηγούς και στην εξυπηρέτηση του χρήστη την στιγμή της ανάγκης του για διαδρομές μικρών αποστάσεων εντός των πόλεων. Ένας χρήστης που είναι εγγεγραμμένος ως οδηγός εξυπηρετεί χρήστες που έχουν ανάγκη να μεταβούν σε κάποιο μέρος την στιγμή που είναι διαθέσιμος. Όπως και στο Uber οι χρήστες χρεώνονται ανάλογα με την απόσταση που διένυσαν. Από το 2012 το Lyft δραστηριοποιήθηκε παράλληλα με τον επιχειρηματικό του τομέα και στην έρευνα και ανάπτυξη της τεχνολογίας αυτόνομων αμαξιών, αυτοκινήτων δηλαδή που δεν απαιτούν οδηγό για την χρήση τους και μεταφέρουν τον πελάτη στον προορισμό του αυτόματα. Στα πλαίσια αυτής της έρευνας η Lyft συνεργάστηκε με εταιρίες όπως η General Motors, η NuTonomy, η Ford , η GoMentum Station και η Magna International και το 2021 πούλησε το τμήμα της που δραστηριοποιήθηκε με αυτόν τον τομέα στην Toyota.

ΣΕΝΑΡΙΑ ΛΕΙΤΟΥΡΓΙΩΝ

Στο κεφάλαιο αυτό θα περιγραφούν βασικά αλλά και πολυπλοκότερα σενάρια λειτουργίας του συστήματος.

Σύνδεση και εγγραφή χρήστη: Αρχικά, σε περίπτωση που ο χρήστης δεν έχει πραγματοποιήσει στο παρελθόν σύνδεση στην εφαρμογή ή έχει αποσυνδεθεί, ανοίγοντας την εφαρμογή, ανακατευθύνεται στην σελίδα σύνδεσης χρήστη από όπου επίσης μπορεί να κατευθυνθεί στην σελίδα εγγραφής, σε περίπτωση που επιθυμεί την δημιουργία νέου λογαριασμού. Σε περίπτωση που προσπαθήσει να εγγραφεί και το email του χρησιμοποιείται ήδη από άλλο λογαριασμό, το Server – side κομμάτι θα επιστρέψει το αντίστοιχο μήνυμα αποτυχίας και δεν θα δημιουργήσει τον λογαριασμό. Αντιστοίχως, εάν κατά την σύνδεση του εισάγει λανθασμένο κωδικό πρόσβασης ή email το οποίο δεν αντιστοιχεί σε κάποιον λογαριασμό, θα λάβει το κατάλληλο μήνυμα. Μετά την σύνδεση του χρήστη στην εφαρμογή, ο χρήστης ανακατευθύνεται πάντοτε στην σελίδα προσφερόμενων διαδρομών.

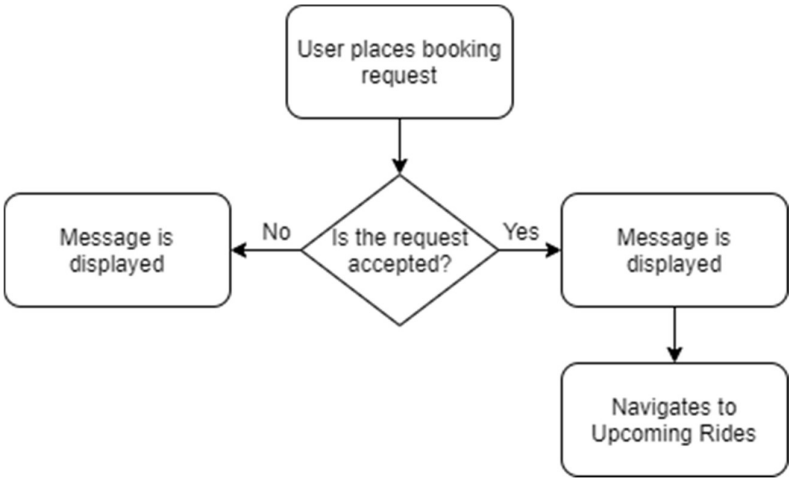


Δημιουργία διαδρομής (προσφοράς και ζήτησης): Ο χρήστης σε κάθε κεντρική σελίδα έχει πρόσβαση στο κουμπί δημιουργίας νέας διαδρομής. Ανάλογα με την επιθυμία του μπορεί να δημιουργήσει είτε μία διαδρομή την οποία θα προσφέρει και στην οποία θα μπορούν οι υπόλοιποι να πραγματοποιήσουν κράτηση, είτε να δημιουργήσει μια διαδρομή προς ζήτηση η οποία θα εμφανίζεται στην αντίστοιχη σελίδα και θα έχει σκοπό να γίνει αντιληπτή από άλλους χρήστες - οδηγούς που τυχόν ενδιαφέρονται να την προσφέρουν. Μετά το πάτημα του κουμπιού δημιουργίας νέας διαδρομής ο χρήστης ανακατευθύνεται στην κατάλληλη φόρμα όπου πρέπει να συμπληρώσει όλα τα πεδία και να την υποβάλλει. Το Server – side κομμάτι της εφαρμογής ελέγχει κατόπιν την εγκυρότητα της υποβολής και επιστρέφει το ανάλογο μήνυμα. Περιπτώσεις που το αίτημα δεν γίνεται δεκτό είναι εάν η ημερομηνία πραγματοποίησης ανήκει στο παρελθόν από την ημέρα υποβολής του αιτήματος και σε περίπτωση ίδιας μέρας υποβολής και πραγματοποίησης, ώρα υποβολής θα πρέπει να είναι τουλάχιστον 40 λεπτά νωρίτερα από την ώρα πραγματοποίησης προκειμένου να υπάρχει χρόνος να γίνουν κρατήσεις από άλλους χρήστες και να προσέλθουν στο σημείο αναχώρησης.

Αναζήτηση διαδρομής: Οι χρήστες έχουν την δυνατότητα να αναζητήσουν διαδρομές επίσης συμπληρώνοντας την κατάλληλη φόρμα η οποία είναι πάντοτε προσβάσιμη με το αντίστοιχο κουμπί στο επάνω – δεξιά μέρος της οθόνης. Η διαδικασία απαιτεί μόνον την συμπλήρωση όσων πεδίων επιθυμεί ο χρήστης και περιλαμβάνει τον όρο αναζήτησης, επιλογή ως προς τον τρόπο αξιοποίησης του όρου δηλαδή εάν θα χρησιμοποιηθεί για την αναζήτηση βάση των σημείων αναχώρησης ή άφιξης, το τύπο των διαδρομών (προσφερόμενες ή προς ζήτηση), ημερομηνία πραγματοποίησης και αριθμό επιβατών. Σε περίπτωση που κάποιο πεδίο δεν συμπληρωθεί τότε το Server – side κομμάτι της εφαρμογής επιστρέφει αποτελέσματα με οποιαδήποτε τιμή σε εκείνο το πεδίο. Συνεπώς, δεν υπάρχει περίπτωση αποτυχίας της λειτουργίας αφού χείριστο σενάριο είναι να μην υπάρχουν αποτελέσματα με κάποιο συνδυασμό επιλογών.

Κράτηση διαδρομής: Στην σελίδα επισκόπησης μιας διαδρομής, εάν η διαδρομή που προβάλλεται δεν έχει υποβληθεί από τον ίδιο χρήστη που εκτελεί την επισκόπηση και εάν δεν πρόκειται για μια διαδρομή της οποίας η ημερομηνία πραγματοποίησης δεν έχει παρέλθει, εμφανίζεται το κουμπί κράτησης. Το κουμπί αυτό δίνει στο χρήστη την

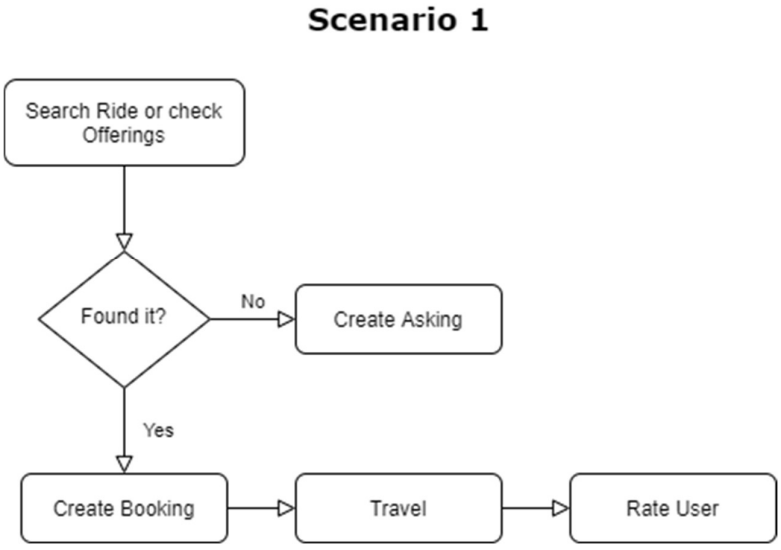
δυνατότητα να πραγματοποιήσει κράτηση για την συγκεκριμένη διαδρομή εάν αυτό είναι δυνατό και να σταλεί ειδοποίηση push στον χρήστη που την προσφέρει. Σε αντίθετη περίπτωση που η διαδρομή ανήκει στον χρήστη προσφέρεται το κουμπί ελέγχου κρατήσεων που εμφανίζει την αντίστοιχη λίστα. Περιπτώσεις που το αίτημα για κράτηση δεν θα γίνει δεκτό είναι εάν απομένουν λιγότερο από 40 λεπτά για την πραγματοποίηση της διαδρομής ή εάν την στιγμή του αιτήματος έχουν συμπληρωθεί ο αριθμός διαθέσιμων θέσεων για κράτηση. Σε αυτή τη περίπτωση επιστρέφεται το αντίστοιχο μήνυμα. Σε περίπτωση που το αίτημα γίνει δεκτό, επιστρέφεται το αντίστοιχο μήνυμα και πλέον η συγκεκριμένη διαδρομή εμφανίζεται στην σελίδα επερχόμενων διαδρομών του χρήστη.



Βαθμολόγηση χρηστών: Μετά το πέρας μιας διαδρομής, ο ένας χρήστης έχει την δυνατότητα να βαθμολογήσει τον άλλο και να γράψει και ένα σχετικό σχόλιο. Η διαδικασία αυτή καθίσταται δυνατή από την εφαρμογή κάνοντας επισκόπηση μιας διαδρομής από την λίστα του ιστορικού και είτε πατώντας στο όνομα του χρήστη που προσέφερε την διαδρομή είτε από την λίστα κρατήσεων με τα ονόματα των αντίστοιχων χρηστών. Σε κάθε περίπτωση ο χρήστης ανακατευθύνεται στην σελίδα προβολής του προφίλ του χρήστη που επέλεξε και εκεί εμφανίζονται ο μέσος όρος των βαθμολογιών του καθώς και η κάθε αξιολόγηση που έχει γίνει για αυτόν. Εάν η προϋπόθεση που τέθηκε στην αρχή είναι αληθής και η στιγμή πραγματοποίησης της συγκεκριμένης διαδρομής από την οποία ανακατευθύνθηκε ο χρήστης έχει παρέλθει, τότε εμφανίζεται το κουμπί δημιουργίας αξιολόγησης όπου ο χρήστης συμπληρώνει την αντίστοιχη φόρμα με τον βαθμό και το σχόλιο του και να την υποβάλλει. Μοναδική περίπτωση στην οποία μια

αξιολόγηση δεν θα γίνει δεκτή είναι εάν ο χρήστης δεν έχει υποβάλλει μια τιμή στο πεδίο βαθμού.

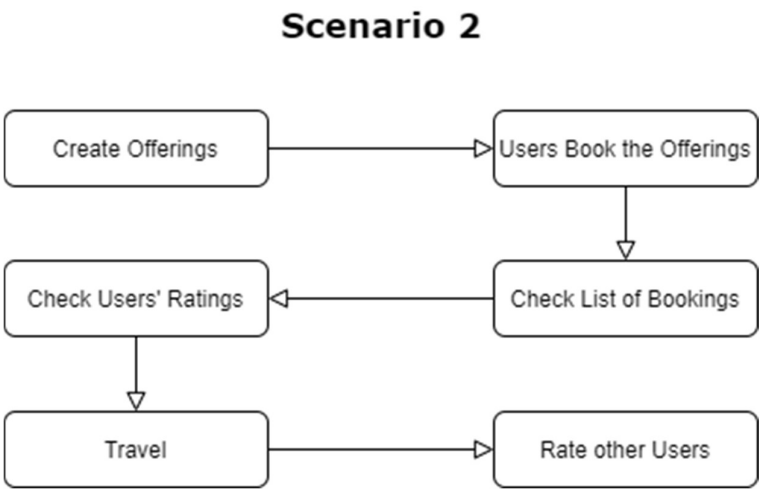
1ος Συνδυασμός σεναρίων: Ο χρήστης ενδιαφέρεται για να βρει διαδρομή για να μεταβεί από την πόλη της Θεσσαλονίκης στο Βόλο. Ο σκοπός του μπορεί να επιτευχθεί είτε μέσω της λειτουργίας αναζήτησης είτε μέσω της λίστας προσφερόμενων διαδρομών. Εάν αυτοί οι δύο τρόποι αποτύχουν, επειδή η συγκεκριμένη διαδρομή δεν προσφέρεται από άλλο χρήστη, τότε μπορεί να χρησιμοποιήσει την λειτουργία δημιουργίας νέας διαδρομής ζήτησης και να δημιουργήσει ζήτηση για την εν λόγω διαδρομή προτρέποντας άλλους χρήστες να την προσφέρουν ή να επικοινωνήσουν μαζί του. Τέλος, σε περίπτωση που ο χρήστης βρει μια προσφερόμενη διαδρομή, μπορεί να κάνει κράτηση σε αυτήν προκειμένου να ειδοποιηθεί με ειδοποίηση push ο χρήστης που την προσφέρει, μετά από το πέρας αυτής, μπορεί να μεταβεί στο ιστορικό του και να βαθμολογήσει τον χρήστη που του την προσέφερε.



Εικόνα 1: Διάγραμμα ροής 1ου σεναρίου χρήσης της εφαρμογής.

2ος Συνδυασμός σεναρίων: Ο χρήστης ταξιδεύει συχνά με το όχημα του σε διάφορες πόλεις και επιθυμεί να προγραμματίζει νωρίς την κάθε διαδρομή του και να μοιράζεται τα έξοδα αυτής με άλλους πολίτες που ταξιδεύουν προς τον ίδιο προορισμό. Έτσι αποφασίζει να προσφέρει όλες τις διαδρομές του στην εφαρμογή μέσω της λειτουργίας δημιουργίας νέας διαδρομής προσφέροντας το πολύ 3 θέσεις για άλλο κόσμο σε κάθε διαδρομή του.

Οι ενδιαφερόμενοι χρήστες προβαίνουν σε κρατήσεις και ο χρήστης έχει στην διάθεσή του λίστα με τα άτομα που έκαναν κράτηση σε κάθε διαδρομή. Για την ασφάλεια του, μεταβαίνει από την εν λόγω λίστα στην σελίδα με την βαθμολογία κάθε χρήστη και μπορεί να διαβάσει παλαιότερες αξιολογήσεις για τον καθένα. Τέλος, εκτελεί τις διαδρομές και αξιολογεί και αυτός τους επιβάτες του.



Εικόνα 2: Διάγραμμα ροής 2ου σεναρίου χρήσης της εφαρμογής.

ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΛΟΓΙΣΜΙΚΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

1. **Framework7 Vue**. Το Framework7 Vue είναι ένα υβριδικό mobile front-end πλαίσιο ανάπτυξης που αποτελείται από τα Framework7 και Vue.js [1]. Το Framework7 είναι ένα δωρεάν και ανοικτού κώδικα πλαίσιο ανάπτυξης για την ανάπτυξη εφαρμογών για πολλαπλές πλατφόρμες, όπως το iOS και το Android, δίνοντας ωστόσο στην εφαρμογή την εγγενή αισθητική της εκάστοτε πλατφόρμας. Ακόμη, χρησιμοποιεί το Apache Cordova, λογισμικό το οποίο δίνει στην τελική εφαρμογή τη δυνατότητα να έχει πρόσβαση στην προγραμματιστική διεπαφή της πλατφόρμας και να χρησιμοποιεί υλικά συστήματα της συσκευής όπως η κάμερα, το GPS, το γυροσκόπιο ή το NFC. Το Framework7 προτιμήθηκε έναντι άλλων εναλλακτικών για την συνεργασία του με Vue.js στην ανάπτυξη τόσο του υβριδικού Framework7 Vue όσο και για τα HTML Components που διαθέτει τα οποία έχουν αναπτυχθεί με το Vue.js δίνοντας μεγαλύτερη ευελιξία στο προγραμματιστή που τα χρησιμοποιεί. Απλά στοιχεία όπως ένα πεδίο εισόδου σε μία φόρμα υπάρχουν ενισχυμένα λειτουργικά στο Framework7 Vue μέσω του Vue.js. Το Vue.js είναι επίσης ένα δωρεάν και ανοικτού κώδικα front-end πλαίσιο ανάπτυξης βασισμένο στην γλώσσα προγραμματισμού JavaScript που στοχεύει στην δημιουργία αλληλεπιδράσεων μεταξύ χρηστών και εφαρμογής [2]. Το Vue.js είναι το πιο ανερχόμενο και ραγδαία αναπτυσσόμενο front-end πλαίσιο ανάπτυξης σε σύγκριση με τα ανταγωνιστικά του Angular.js και React.js και προτιμήθηκε λόγω της μεγάλης ελευθερίας που παρέχει σε τροποποιήσεις καθώς και για τον τρόπο σύνταξής του καθώς παρέχει την ελευθερία συγγραφής ενός Component με όλα τα στοιχεία του (template, script, style) σε ένα αρχείο (Single File Components), βοηθώντας έτσι στην οργάνωση του κώδικα και στην ευκολία ανάγνωσής του.
2. **Laravel**. Ένα δωρεάν ανοικτού κώδικα πλαίσιο ανάπτυξης βασισμένο στην γλώσσα προγραμματισμού PHP [3]. Βασισμένο στο αρχιτεκτονικό μοτίβο ανάπτυξης διαδικτυακών εφαρμογών Model – View – Controller (MVC), το Laravel είναι ικανό να υποστηρίξει την ανάπτυξη μιας διαδικτυακής εφαρμογής σε οποιοδήποτε επίπεδο. Χρησιμοποιήθηκε στην εφαρμογή εξ ολοκλήρου στο back-end επίπεδο ως διεπαφή – μεσολαβητής μεταξύ της εφαρμογής πού εκτελείται στην πλατφόρμα του χρήστη που φτιάχτηκε με το Framework7 Vue και της βάσης δεδομένων PostgreSQL. Το Laravel

διαθέτει εσωτερικές διεπαφές που καθιστούν την επικοινωνία των υποσυστημάτων που εκπροσωπεί στο μοντέλο MVC, στο οποίο θα αναφερθούμε σε επόμενο κεφάλαιο, εύκολη και βέλτιστη καθώς επίσης διαθέτει και έτοιμες πολύπλοκες λειτουργίες, οι οποίες μπορούν να τροποποιηθούν για την εκάστοτε περίπτωση, για παράδειγμα για την συλλογή δεδομένων από την βάση δεδομένων μαζικά ή και μεμονωμένα, την δημιουργία πινάκων και στηλών βάσεων δεδομένων ή την επικοινωνία του συστήματος με άλλα συστήματα μέσω του πρωτοκόλλου επικοινωνίας HTTP. Πρόκειται λοιπόν για ένα πλήρες πλαίσιο ανάπτυξης ικανό να προσφέρει λειτουργικότητα σε όλα τα επίπεδα για την μετάδοση της πληροφορίας με απλό και οργανωμένο τρόπο.

3. **PostgreSQL**. Σχεσιακή βάση δεδομένων ανοικτού κώδικα [4]. Η PostgreSQL υποστηρίζει όλα τα λειτουργικά συστήματα βασισμένα στο Linux και Unix καθώς και σε Windows. Επιπλέον υποστηρίζει σχεδόν όλους τους SQL92 και SQL99 χαρακτήρες και επιτρέπει την αποθήκευση μεγάλων σε όγκο αρχείων όπως εικόνας, ήχου και βίντεο. Προτιμήθηκε αντί άλλων βάσεων δεδομένων λόγω της υποστήριξης ζώνης ώρας στον τύπο δεδομένων timestamp.

4. **Firebase Cloud Messaging**. Το Firebase Cloud Messaging ή εν συντομία FCM είναι μια πλατφόρμα παροχής υπηρεσιών για ειδοποιήσεις push [5]. Οι ειδοποιήσεις push είναι διαδικτυακό μοντέλο επικοινωνίας συστημάτων, όπου το αίτημα για την εκκίνηση της μετάδοσης δεδομένων ξεκινάει από τον κεντρικό διακομιστή (server) και όχι από τον πελάτη (client) όπως γίνεται στα περισσότερα άλλα μοντέλα όπως το HTTP. Αυτό επιτυγχάνεται μέσω tokens μοναδικά για κάθε χρήστη της εφαρμογής και βρίσκονται καταγεγραμμένα στην βάση δεδομένων PostgreSQL. Έτσι όταν το back-end επίπεδο, που εκπροσωπείται μέσω της εφαρμογής που αναπτύχθηκε με το Laravel, απαιτείται να ειδοποιήσει για κάποιο λόγο κάποια συσκευή ενός χρήστη, αποστέλλει τα επιθυμητά δεδομένα στο, μοναδικό για την εφαρμογή, url που έχει αποδοθεί από το FCM και αυτό τα προωθεί στην συσκευή του χρήστη, μέσω της προγραμματιστικής διεπαφής της πλατφόρμας (iOS, Android). Η μεταφορά αυτή επιτυγχάνεται με την βοήθεια του εργαλείου cURL. Τέλος, προτιμήθηκε μεταξύ άλλων για την κοινή τιμολόγηση του με το Google Maps και την κοινή πλατφόρμα ελέγχου που μοιράζονται καθώς επίσης για την ευκολία στην ρύθμιση του και την ένταξή του σε ένα σύστημα

και την υψηλή διασημότητά του και συνεπώς την άμεση υποστήριξη που παρέχεται στο διαδίκτυο.

5. **Google Maps**. Πλατφόρμα παροχής γεωγραφικών πληροφοριών και χαρτών. Χρησιμοποιήθηκε για την αναζήτηση διευθύνσεων των σημείων αναχώρησης και άφιξης των διαδρομών από τον χρήστη και την απεικόνιση αυτών σε χάρτη εντός της εφαρμογής [6]. Πρόκειται για την διασημότερη και πληρέστερη πλατφόρμα παροχής αυτών των υπηρεσιών. Το Google Maps πέρα από τις πληροφορίες που προσφέρει, προσφέρει και ψηφιακούς χάρτες εκτός της δικής του εφαρμογής προκειμένου να ενσωματωθούν σε άλλα συστήματα και να απεικονισθεί η πληροφορία σε περιβάλλον και αισθητική τα οποία είναι ήδη οικεία στο χρήστη. Επιπλέον το Apache Cordova πλαίσιο ανάπτυξης που δίνει στον προγραμματιστή και στις εφαρμογές του πρόσβαση στο GPS της πλατφόρμας συνεργάζεται άψογα με το Google Maps και προσφέρει ένα ολοκληρωμένο Component για την απεικόνιση των χαρτών με πλήρη λειτουργικότητα.

6. **NGINX**. Δωρεάν και ανοικτού κώδικα διακομιστής ιστού (server) και ισορροπιστής φορτίου [7]. Χρησιμοποιείται για να εκθέτει στο διαδίκτυο το back-end επίπεδο της εφαρμογής καθιστώντας δυνατή την μεταφορά δεδομένων από το back-end στο front-end και αντίστροφα μέσω του πρωτοκόλλου HTTP. Προτιμήθηκε λόγω της απευθείας συνεργασίας και συμβατότητάς του με το Laravel για την γρήγορη και ασφαλή έκθεση της εφαρμογής στο διαδίκτυο καθώς επίσης και για τις λειτουργίες του ρύθμισης του φόρτου αιτημάτων HTTP χωρίς την ανάγκη περαιτέρω παραμετροποιήσεων από τον προγραμματιστή.

7. **Git**. Λογισμικό ελέγχου και διανομής εκδόσεων λογισμικού [8]. Το Git είναι το εργαλείο που χρησιμοποιείται πλέον από κάθε προγραμματιστή για τον έλεγχο του κώδικα και την ιχνηλάτηση των βημάτων που πραγματοποιήθηκαν για την ανάπτυξή του. Το βασικότερο πλεονέκτημα που προσφέρει είναι το σώσιμο κάθε βήματος ξεχωριστά και η αναίρεση του κάνοντας την αποσφαλμάτωση πολύ ευκολότερη.

8. **Visual Studio Code**. Λογισμικό επεξεργασίας πηγαίου κώδικα που αναπτύχθηκε από την εταιρία Microsoft και είναι από τα διασημότερα εργαλεία αυτού του τύπου [9]. Προσφέρει πληθώρα συντομεύσεων από το πληκτρολόγιο διευκολύνοντας τον χρήστη να γράφει κώδικα ταχύτερα, καθώς επίσης και την δημιουργία καινούργιων. Επιπρόσθετα, μπορεί να παραμετροποιηθεί σε βάθος σχεδόν για κάθε λεπτομέρεια

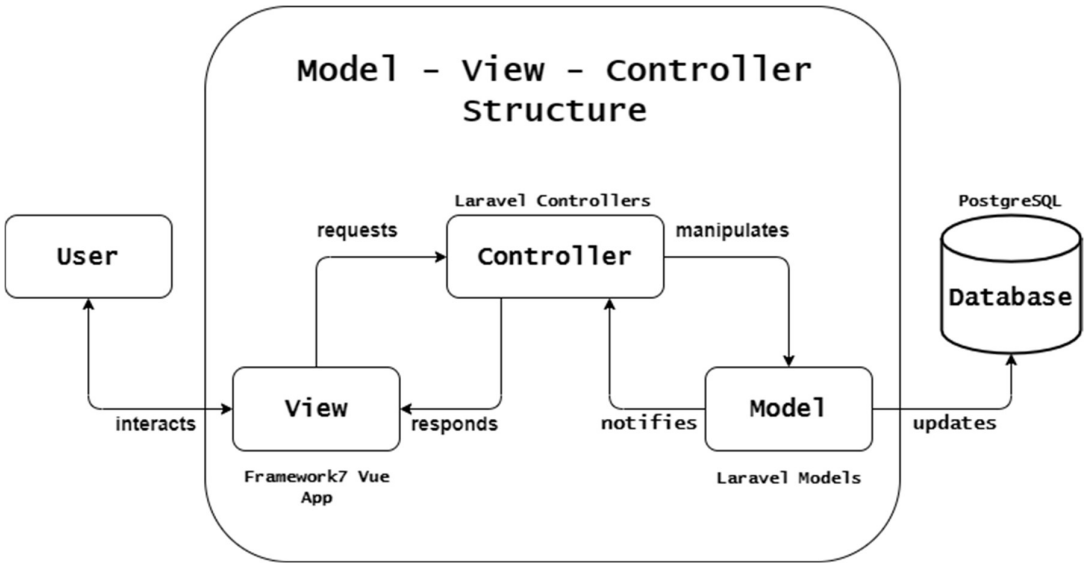
που μπορεί να απασχολήσει τον προγραμματιστή κατά την διάρκεια της πληκτρολόγησης. Οι τρόποι παραμετροποίησης ωστόσο δεν σταματούν εκεί καθώς υπάρχει μεγάλο πλήθος πρόσθετων λογισμικών που προσφέρουν επιπλέον δυνατότητες. Τέλος, προσφέρει καλαίσθητη αισθητική καθώς και μια μεγάλη γκάμα από θέματα εμφάνισης.

9. **Android Tools**. Πλατφόρμα εργαλείων που παρέχεται δωρεάν από την εταιρία Google [10]. Χρησιμοποιήθηκε για την μετατροπή του πηγαίου κώδικα του τελικού ενιαίου JavaScript αρχείου της εφαρμογής σε εκτελέσιμο για την πλατφόρμα Android.

10. **cURL.** Δωρεάν και ανοικτού κώδικα λογισμικό για την μεταφορά δεδομένων με την χρήση διάφορων διαδικτυακών πρωτοκόλλων [11]. Το cURL αποτελείται από την βιβλιοθήκη libcurl και το εργαλείο γραμμής εντολών curl. Η βιβλιοθήκη libcurl υποστηρίζει πρωτόκολλα όπως το HTTPS, FTP, IMAP, POP3. Το εργαλείο curl δέχεται και αποστέλλει δεδομένα χρησιμοποιώντας το συντακτικό URL και εφόσον χρησιμοποιεί την libcurl υποστηρίζει κάθε πρωτόκολλο που υποστηρίζει η προαναφερθείσα βιβλιοθήκη. Χρησιμοποιήθηκε για την επικοινωνία του back-end επιπέδου με τους διακομιστές του FCM για την μεταφορά δεδομένων με σκοπό την υλοποίηση ειδοποιήσεων push.

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

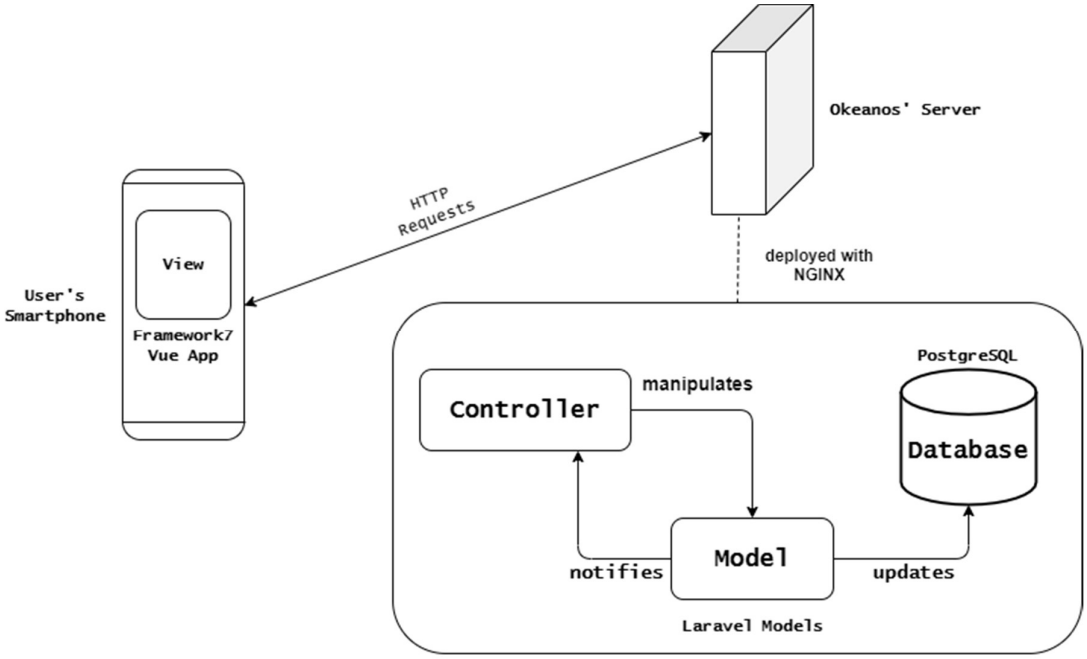
Όπως προαναφέρθηκε σε προηγούμενο κεφάλαιο, η εφαρμογή έχει βασιστεί στο αρχιτεκτονικό μοτίβο ανάπτυξης διαδικτυακών εφαρμογών Model – View – Controller ή εν συντομία MVC. Τα εργαλεία που χρησιμοποιήθηκαν συνέβαλαν στην ανάπτυξη επί μέρους προγραμμάτων τα οποία αντιπροσωπεύουν στοιχεία του MVC και αλληλοεπιδρούν μεταξύ τους συνθέτοντας την εφαρμογή. Στην Εικόνα 1 απεικονίζεται το σχεδιάγραμμα της αρχιτεκτονικής MVC με τα επί μέρους τμήματα του και συνοπτικά τον τρόπο με τον οποίο αυτά αλληλοεπιδρούν.



Εικόνα 3: Δομή αρχιτεκτονικής MVC.

Ο χρήστης αρχικά αλληλοεπιδρά με την εφαρμογή στο έξυπνο τηλέφωνό του, μέσω της οθόνης αφής που αυτό διαθέτει, η οποία αντιπροσωπεύει το τμήμα View του συστήματος. Η εφαρμογή, ανάλογα με την επιθυμία του χρήστη στέλνει το αντίστοιχο αίτημα για δεδομένα στο τμήμα Controller μέσω του πρωτοκόλλου HTTP, το οποίο αντιπροσωπεύεται από την εφαρμογή που έχει αναπτυχθεί μέσω του πλαισίου ανάπτυξης Laravel και εκτίθεται στο διαδίκτυο μέσω του διακομιστή ιστού NGINX που εκτελείται σε εξυπηρετητή ιστού του δικτύου okeanos.grnet.gr, όπως φαίνεται στην Εικόνα 2. Κατόπιν, το τμήμα Controller εκτελεί οποιονδήποτε υπολογισμό απαιτείται και επικοινωνεί με το τμήμα Model μέσω των εσωτερικών διεπαφών που διαθέτει το Laravel καθώς έχει την

δυνατότητα να αντιπροσωπεύει και τα δυο αυτά τμήματα. Ο Controller επικοινωνεί είτε για να ενημερώσει το Model πως απαιτείται συλλογή δεδομένων από τη βάση δεδομένων, είτε για να κάνει αλλαγές στα υπάρχοντα δεδομένα, είτε για να εισάγει καινούργια δεδομένα τα οποία έστειλε ο χρήστης μέσω του HTTP αιτήματος του. Σημειώνεται πως ο Controller εκτελεί τυχόν υπολογισμούς και επικοινωνεί με το Model με οποιαδήποτε σειρά αυτό είναι απαραίτητο, ανάλογα την περίπτωση. Τέλος, το Model εκτελεί οτιδήποτε ζητήθηκε από τον Controller και επιστρέφει το αντίστοιχο αποτέλεσμα. Αντίστοιχα, ο Controller επιστρέφει οτιδήποτε του ζητήθηκε από αίτημα του χρήστη στο τμήμα View, το οποίο αναπαριστά την πληροφορία στο χρήστη σε αναγνώσιμη μορφή.



Εικόνα 4: Διαίρεση του MVC μεταξύ συσκευών.

Το MVC είναι το πιο διαδεδομένο μοντέλο αρχιτεκτονικής διαδικτυακών εφαρμογών και γνώρισε τεράστια δημοσιότητα μέσω της γλώσσας προγραμματισμού Java η οποία είναι από τις διασημότερες γλώσσες προγραμματισμού για το διαδίκτυο. Στα επόμενα κεφάλαια, θα παρουσιαστούν τα επιμέρους υποσυστήματα που απαρτίζουν την εφαρμογή.

ΥΠΟΣΥΣΤΗΜΑΤΑ

Βάση Δεδομένων PostgreSQL. Η Βάση Δεδομένων είναι συνδεδεμένη με τα Laravel Models του συστήματος τα οποία διαχειρίζονται τα δεδομένα κάθε table της βάσης καθώς επίσης δείχνουν και τις σχέσεις μεταξύ των δεδομένων κάθε table αφού πρόκειται για μια σχεσιακή βάση δεδομένων. Επιπλέον, το Laravel παρέχει το πολύ χρήσιμο εργαλείο των migrations. Στα migrations περιγράφεται το κάθε table που θέλουμε να δημιουργηθεί στην βάση μας και να ελεγχθεί από το αντίστοιχο Model. Δηλώνεται το όνομα κάθε πεδίου του table, ο τύπος δεδομένων του, η μοναδικότητα κάθε εγγραφής στο table καθώς και αρχικές τιμές ή αν ένα πεδίο σε μια εγγραφή μπορεί να πάρει την τιμή null. Τα migrations που χρησιμοποιήθηκαν είναι τα εξής:

Users: id (μοναδικό, το αναλαμβάνει το Laravel), string name, string email μοναδικό, float rating_avg αρχικοποιημένο με την τιμή 0, timestamp email_verified_at nullable (για μελοντική χρήση), string fcm_token, string password.

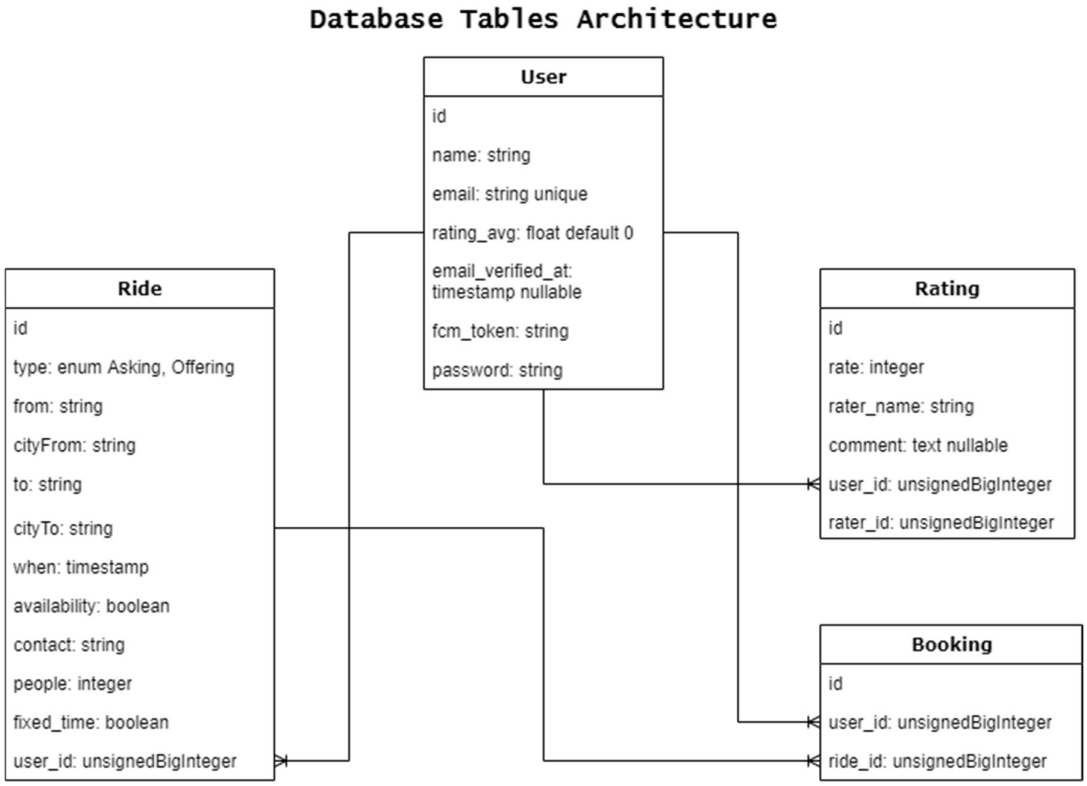
Rides: id, enum type με τιμες Asking και Offering, string from (σημείο εκκίνησης διαδρομής), string cityFrom (πόλη εκκίνησης), string to και string cityTo (τα αντίστοιχα για τον προορισμό της διαδρομής), timestamp when, boolean availability, string contact, integer people, boolean fixed_time, unsignedBigInteger user_id που δείχνει στο id του χρήστη που δημιούργησε την διαδρομή.

Booking: id, unsignedBigInteger user_id που δείχνει στο id του χρήστη που δημιούργησε την διαδρομή, unsignedBigInteger ride_id που δείχνει στο id της διαδρομής στην οποία ανήκει η κράτηση.

Rating: id, integer rate, string rater_name, text comment nullable, unsignedBigInteger user_id που δείχνει στο id του χρήστη για τον οποίο έγινε η αξιολόγηση, unsignedBigInteger rater_id που δείχνει στο id του χρήστη ο οποίος έκανε την αξιολόγηση.

Όπως είδαμε σε προηγούμενο κεφάλαιο, τα Models χρησιμοποιούνται από τους Controllers για να επικοινωνήσουν με την βάση δεδομένων και σε αυτά ορίζονται και οι σχέσεις μεταξύ των tables της βάσεις. Οι σχέσεις αυτές είναι οι εξής:

User: one-to-many Rides, one-to-many Bookings, one-to-many Ratings.
Ride: one-to-many Bookings, belongsTo User. Η σχέση belongsTo είναι μια σχέση που ορίζεται εσωτερικά στο Laravel και υλοποιεί το αντίστροφο της one-to-many σχέσης δίνοντας πρόσβαση κατευθείαν στον χρήστη στον οποίο ανήκει η διαδρομή.
Booking: belongsTo Ride, belongsTo User.
Rating: belongsTo User. Η σχέση αυτή χρησιμοποιείται για την εύρεση του χρήστη για τον οποίο έγινε η αξιολόγηση.



Εικόνα 5: Αρχιτεκτονική των tables της βάσης δεδομένων.

Server-side Laravel (Back-end). Πέρα από τα Models και τα Migrations που χρησιμοποιεί το Laravel για την επικοινωνία με την βάση δεδομένων, υπάρχουν και οι Controllers. Οι Controllers είναι κλάσεις με μεθόδους τις οποίες η εφαρμογή χρησιμοποιεί όταν λάβει HTTP Requests σε προκαθορισμένα routes. Το μοντέλο λογισμικού που μόλις περιγράφηκε ονομάζεται Application Programming Interface (API). Τα routes είναι λογισμικό που κατευθύνει τα HTTP Requests που λαμβάνει η εφαρμογή σε προκαθορισμένα URLs στην αντίστοιχη μέθοδο του Controller της επιλογής μας προκειμένου να το επεξεργαστεί και να δημιουργήσει την αντίστοιχη απάντηση (response). Η επικοινωνία ανάμεσα στο Server - side κομμάτι και το Front-end κομμάτι γίνεται με την χρήση μόνο των μεθόδων HTTP Post και Get και στην πρώτη περίπτωση τα δεδομένα αποτελούν ένα αντικείμενο τύπου json ενώ στην δεύτερη δίνεται μία παράμετρος ως id στο URL που γίνεται η κλήση της μεθόδου. Τέλος, αξίζει να σημειωθεί ότι το Laravel κατά την δημιουργία ενός Model δημιουργεί αυτόματα και τον αντίστοιχο Controller. Συνεπώς, οι Controllers που διαθέτει η εφαρμογή είναι:

UserController: Διαθέτει δύο μεθόδους για την εγγραφή και την είσοδο του χρήστη στην εφαρμογή ενώ επίσης κατά την είσοδο του χρήστη ενημερώνει το fcm token του χρήστη προκειμένου να μπορεί να δεχτεί ειδοποιήσεις push.

1. **register:** Λαμβάνει ένα αντικείμενο τύπου json που περιέχει τις πληροφορίες του χρήστη που πρόκειται να εγγραφεί δηλαδή email, username και password. Κατόπιν, επιβεβαιώνει την μη ύπαρξη ενός χρήστη με το συγκεκριμένο email, δημιουργεί έναν καινούργιο χρήστη και αποστέλλει ως απάντηση ένα αντικείμενο json με μήνυμα επιτυχίας και το αντικείμενο του χρήστη που δημιούργησε.

   Αντικείμενο που παραλαμβάνει:

   {

   “name”: value,

   “username”: value,

   “password”: value,

   “c_password”: value

   }

Το πεδίο c_password αποτελεί το πεδίο της φόρμας όπου ζητείται από τον χρήστη να εισάγει για δεύτερη φορά τον κωδικό του και χρησιμοποιείται στο Server - side κομμάτι για επιβεβαίωση του κωδικού πριν δημιουργηθεί ο χρήστης και δεν αποθηκεύεται στην βάση δεδομένων.

Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση επιτυχίας:

```
{
    “success”: true,
    “message”: ‘User Registration Successful’,
    “data”: $user
}
```

Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση αποτυχίας:

```
{
    “success”: false,
    “message”: ‘Validation Error’ + $validator->errors()
}
```

Το αντικείμενο $validator του Laravel ελέγχει τα ορίσματα κατά την δημιουργία του χρήστη, για παράδειγμα εάν έχει δοθεί τιμή σε όλα τα πεδία ή εάν το πεδίο password και c_password συμφωνούν και παράγει τα αντίστοιχα μηνύματα τα οποία επιστρέφονται με την μέθοδο errors.

2. **details:** Λαμβάνει ένα αντικείμενο json με το token του χρήστη και κατόπιν με την χρήση της εσωτερικής μεθόδου auth του Laravel λαμβάνει και τον χρήστη του συγκεκριμένου session. Αμέσως μετά, ελέγχει εάν το token που έλαβε είναι το ίδιο με το token στο αντικείμενο του χρήστη και εάν όχι, το ανανεώνει και το αποθηκεύει στη βάση. Τέλος, αποστέλλει πίσω το ανανεωμένο αντικείμενο του χρήστη μαζί με μήνυμα επιτυχίας.

   Αντικείμενο που παραλαμβάνει:

   ```
   {
       “fcm_token”: value,
   }
   ```

Αντικείμενο που αποστέλλει ως απάντηση:

```
{
    “success”: value,
    “user”: $user
}
```

Αξίζει να σημειωθεί πως η διαδικασία σύνδεσης του χρήστη δεν γίνεται στο συγκεκριμένο Controller αλλά πραγματοποιείται με εσωτερικό λογισμικό (middleware) του Laravel όπου η διαδικασία αναγνώρισης έχει ως εξής: Η εφαρμογή του χρήστη έχει αποθηκευμένο ένα κωδικό μοναδικό για την εφαρμογή του Laravel που βρίσκεται στο Server – side τμήμα. Κατά την σύνδεση χρήστη αποστέλλεται HTTP Request με το username και τον κωδικό του χρήστη και σε περίπτωση που ο κωδικός είναι σωστός, τότε το Server – side τμήμα αποστέλλει πίσω ένα μοναδικό token για τον συγκεκριμένο χρήστη το οποίο αποστέλλεται πάντοτε μαζί με κάθε άλλο HTTP Request από την εφαρμογή του Framework7 Vue. Αποτέλεσμα αυτής της διαδικασίας είναι το Server – side τμήμα να γνωρίζει πάντοτε εάν έχει προηγηθεί σύνδεση από τον χρήστη στο παρελθόν. Ως εκ τούτου, δεν απαιτείται και μέθοδος από τον Controller για την αποσύνδεση του χρήστη καθώς η εφαρμογή του Framework7 Vue, πατώντας το κουμπί της αποσύνδεση, θέτει την τιμή null στις μεταβλητές με τα στοιχεία του χρήστη και του token του.

RideController: Διαθέτει εννέα μεθόδους.

1. **myRides:** Λαμβάνει σαν όρισμα το id του χρήστη και επιστρέφει τις διαδρομές που έχει δημιουργήσει ο χρήστης και η ημερομηνία πραγματοποίησης τους ξεπερνάει την ημερομηνία που λαμβάνεται HTTP Request.

   Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση επιτυχίας:

   ```
   {
       “success”: true,
       “message”: ‘Rides retrieved successfully’,
       “data”: $rides
   }
   ```

Η ίδια απάντηση αποστέλλεται και στις παρακάτω μεθόδους του συγκεκριμένου Controller εκτός των περιπτώσεων μου αναφέρεται διαφορετικά.

2. **history:** Λαμβάνει σαν όρισμα το id του χρήστη και επιστρέφει τις διαδρομές που έχει δημιουργήσει ή συμμετάσχει ο χρήστης μέσω κράτησης και η ημερομηνίες πραγματοποίησης τους είναι αυστηρά μικρότερη από την ημερομηνία που λαμβάνεται το HTTP Request.
3. **offering:** Επιστρέφει όλες τις διαδρομές με type offering, η ημερομηνία πραγματοποίησης τους ξεπερνάει την ημερομηνία που λαμβάνεται HTTP Request και το πεδίο availability είναι true (δεν είναι πλήρης από κρατήσεις).
4. **asking:** Επιστρέφει όλες τις διαδρομές με type asking, η ημερομηνία πραγματοποίησης τους ξεπερνάει την ημερομηνία που λαμβάνεται HTTP Request και το πεδίο availability είναι true (δεν είναι πλήρης από κρατήσεις).
5. **search:** Δέχεται ένα αντικείμενο που περιέχει ένα string για την τοποθεσία που θέλουμε να αναζητήσουμε και σχετικά φίλτρα για την αναζήτηση όπως ημερομηνία, διαθέσιμες θέσεις, τρόπος χρήσης του string τη τοποθεσίας (εάν θέλουμε να το χρησιμοποιήσουμε ως σημείο αναχώρησης ή προορισμού) αλλά και του τύπου της διαδρομής (Offering ή Asking).

   Αντικείμενο που παραλαμβάνει:

   ```
   {
       “ride_type”: value,
       “location_filters”: value,
       “when”: value,
       “people”: value,
       “searchTerm”: value
   }
   ```

6. **store:** Μέθοδος για την αποθήκευση μιας νέας διαδρομής στην βάση δεδομένων.

   Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση επιτυχίας:

   ```
   {
       “success”: true,
       “message”: ‘Ride created successfully’,
   ```

}

Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση αποτυχίας:

{

“success”: false,

“message”: ‘Validation Error’ + $validator->errors()

}

7. **upcomingRides:** Λαμβάνει σαν όρισμα το id του χρήστη και επιστρέφει τις διαδρομές που έχει δημιουργήσει ο χρήστης ή θα συμμετάσχει ο χρήστης μέσω κράτησης και η ημερομηνία πραγματοποίησης τους ξεπερνάει την ημερομηνία που λαμβάνεται HTTP Request.
8. **ridesUser:** Λαμβάνει σαν όρισμα το id μιας διαδρομής και επιστρέφει σε array το id, το name και το rating_avg του χρήστη που την δημιούργησε.

   Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση επιτυχίας:

   {

   “success”: true,

   “message”: ‘User retrieved successfully’,

   “data”: [id, name, rating_avg]

   }
9. **suggestedRides:** Λαμβάνει ένα αντικείμενο το οποίο περιέχει το id και την πόλη στην οποία βρίσκεται ο χρήστης σε string, κατόπιν βρίσκει τις πόλεις από τους 20 τελευταίους προορισμούς του χρήστη, είτε από δικές του διαδρομές είτε από τις κρατήσεις του, και επιστρέφει διαδρομές με σημείο εκκίνησης την πόλη που βρίσκεται ο χρήστης και προορισμούς τις 20 πόλεις που βρήκε στο προηγούμενο βήμα.

   Αντικείμενο που παραλαμβάνει:

   {

   “id”: value,

   “userCity”: value

   }

BookingController: Διαθέτει τέσσερεις μεθόδους για την διαχείριση των κρατήσεων αλλά και των push ειδοποιήσεων σχετικά με τις κρατήσεις.

1. **myBookings:** Λαμβάνει ως όρισμα το id του χρήστη και επιστρέφει όλες τις κρατήσεις τις οποίες έχει κάνει.

   Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση επιτυχίας:

   {

   “success”: true,

   “message”: ‘Bookings retrieved successfully’,

   “data”: $bookings

   }

   Η ίδια απάντηση αποστέλλεται και στις παρακάτω μεθόδους του συγκεκριμένου Controller εκτός των περιπτώσεων μου αναφέρεται διαφορετικά.

2. **store:** Λαμβάνει ως ορίσματα το id του χρήστη και της διαδρομής για την οποία ενδιαφέρεται ο χρήστης να κάνει κράτηση. Κατόπιν, δημιουργεί την σχετική κράτηση εάν υπάρχουν διαθέσιμες θέσεις και μόνο εάν η ώρα που θα ληφθεί το αίτημα είναι τουλάχιστον 40 λεπτά νωρίτερα από την ώρα πραγματοποίησης της διαδρομής ώστε ο χρήστης να έχει αρκετό χρόνο να φτάσει εγκαίρως στο σημείο αναχώρησης. Αμέσως μετά, ελέγχει εάν μετά από αυτή την κράτηση η διαδρομή είναι πλήρης και ενημερώνει το σχετικό πεδίο της διαδρομής (availability). Τέλος σε κάθε περίπτωση καλεί την μέθοδο sendPushNotification με το κατάλληλο μήνυμα (είτε νέας κράτησης είτε ότι η κράτηση είναι πλέον πλήρης) και το fcm token του χρήστη που δημιούργησε την διαδρομή.

   Αντικείμενο που παραλαμβάνει:

   {

   “ride_id”: value,

   “user_id”: value

   }

   Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση επιτυχίας:

   {

   “success”: true,

   “message”: ‘Booking created successfully’

   }

Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση αποτυχίας:

{

“success”: false,

“message”: ‘You have already booked this ride!’

}

Σημειώνεται πως σε άλλες περιπτώσεις αποτυχίας επίσης το μοναδικό πεδίο που διαφοροποιείται είναι αυτό του μηνύματος με το κατάλληλο μήνυμα.

3. **ridesBookings:** Λαμβάνει ως όρισμα το id μιας διαδρομής και επιστρέφει όλους τους χρήστες που έχουν κάνει κράτηση σε αυτήν τη διαδρομή.
4. **sendPushNotification:** Λαμβάνει ως ορίσματα δύο strings, το μήνυμα που επιθυμούμε να στείλουμε και το fcm token του παραλήπτη χρήστη. Δημιουργεί το κατάλληλο object και αποστέλλει το HTTP Request στο FCM μέσω του εργαλείου cURL.

   Αντικείμενο που αποστέλλει:

   {

   “apiKey”: value (δίδεται από το FCM και είναι μοναδικό για την εφαρμογή),

   “notifications”: [‘body’: (όνομα της εφαρμογής), ‘title’: $message],

   “notification_foreground”: true (Καθορίζει ένα η μορφή της ειδοποίησης θα είναι εκτεταμένη ή ελαχιστοποιημένη κατά την παράδοση)

   }

RatingController: Διαθέτει τρεις μεθόδους για την διαχείριση των αξιολογήσεων αλλά και των push ειδοποιήσεων σχετικά με τις αξιολογήσεις.

1. **store:** Αποθηκεύει μια καινούργια αξιολόγηση για κάποιον χρήστη. Λαμβάνει ως ορίσματα το id του χρήστη για τον οποίο γίνεται η αξιολόγηση, το id του χρήστη που κάνει την αξιολόγηση, τον βαθμό (ακέραιος) και το σχόλιο. Ελέγχει εάν το id του αξιολογητή και του αξιολογούμενου ταυτίζονται αποτρέποντας κάποιον χρήστη από το να αξιολογήσει τον εαυτό του και κατόπιν ενημερώνει τον μέσω όρο του αξιολογούμενου χρήστη κατάλληλα. Τέλος, καλεί την δική

του sendPushNotification προκειμένου να ενημερώσει τον αξιολογούμενο ότι έχει μια καινούργια αξιολόγηση.

Αντικείμενο που παραλαμβάνει:

```
{
    “rater_id”: value,
    “rater_name”: value,
    “user_id”: value,
    “rate”: value,
    “comment”: value
}
```

Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση επιτυχίας:

```
{
    “success”: true,
    “message”: ‘Rate submitted successfully’
}
```

Αντικείμενο που αποστέλλει ως απάντηση σε περίπτωση αποτυχίας:

```
{
    “success”: false,
    “message”: ‘You cannot rate yourself!’
}
```

Σημειώνεται πως σε άλλες περιπτώσεις αποτυχίας επίσης το μοναδικό πεδίο που διαφοροποιείται είναι αυτό του μηνύματος με το κατάλληλο μήνυμα.

2. **userRatings:** Λαμβάνει ως όρισμα το id ενός όρισμα και επιστρέφει όλες τις αξιολογήσεις που έχουν γίνει για αυτόν.

   Αντικείμενο που αποστέλλει ως απάντηση:

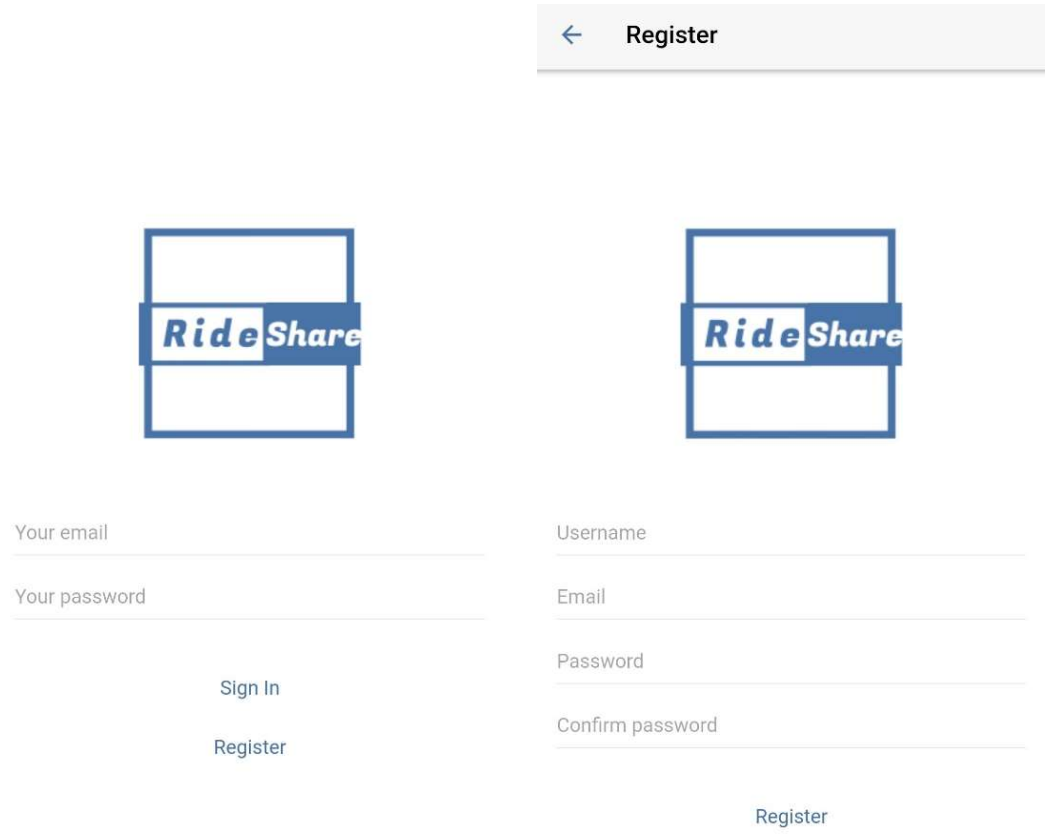
```
{
    “success”: true,
    “message”: ‘Ratings retrieved successfully’,
    “data”: $ratings
}
```

3. **sendPushNotification.** Λαμβάνει ως όρισμα το fcm token του χρήστη για τον οποίο έχει γίνει μια αξιολόγηση και μέσω του cURL αποστέλλει το σχετικό μήνυμα στο FCM. Η μορφή του είναι ίδια με αυτή της αντίστοιχης μεθόδου του BookingController.

Framework7 Vue (Front-End). Όπως αναλύθηκε σε προηγούμενο κεφάλαιο, η εφαρμογή που βρίσκεται στο έξυπνο τηλέφωνο του εκάστοτε χρήστη και εκπροσωπεί το τμήμα View του μοντέλου MVC, αποτελείται από δύο πλαίσια ανάπτυξης τα οποία έχουν συνδυαστεί μεταξύ τους και λειτουργούν ως ένα. Το Framework7 είναι υπεύθυνο για την μετατροπή του κώδικα σε αρχείο εγκατάστασης για την εκάστοτε πλατφόρμα, την επικοινωνία της εφαρμογής με την προγραμματιστική διεπαφή της πλατφόρμας αλλά και για την ενσωμάτωση της εγγενούς αισθητικής της πλατφόρμας στην εφαρμογή. Επιπλέον, διαθέτει και ενισχυμένα λειτουργικά components. Τα components είναι επαναχρησιμοποιούμενα στοιχεία τα οποία είτε διατίθενται έτοιμα από το πλαίσιο ανάπτυξης είτε αναπτύσσονται από τον προγραμματιστή προκειμένου να προσφέρουν συγκεκριμένες λειτουργίες, απλές αλλά και πιο σύνθετες. Αποτελούνται από τρία τμήματα κώδικα, HTML JavaScript και CSS προσφέροντας λειτουργικότητα και απεικόνιση των λειτουργιών. Έτσι τοποθετώντας σε μια σελίδα τα κατάλληλα components δημιουργείται μια ολοκληρωμένη οπτικά και λειτουργικά σελίδα από κώδικα καλύτερα οργανωμένο, που μπορεί να χρησιμοποιηθεί ξανά και σε άλλες σελίδες, χωρίς να γραφτεί από την αρχή. Τα components λειτουργούν παράλληλα με τους μηχανισμούς του Vue όπως το Vuex για την ευκολότερη ροή δεδομένων ανάμεσά τους. Το Vuex αποτελεί μηχανισμό state management, δηλαδή πρόκειται για λογισμικό που ελέγχει την κατάσταση των components σχετικά με τα δεδομένα που απεικονίζουν και επεξεργάζονται. Τα δεδομένα αντί να μεταδίδονται από component σε component και κάθε ενημέρωση τους να αποτελεί μία περίπλοκη διαδικασία, το Vuex αναλαμβάνει να τα αποθηκεύσει και να ενημερώσει όλα τα components που τα χρησιμοποιούν, χωρίς τα ίδια να χρειάζεται να κάνουν re-rendering, δηλαδή να μεταγλωττιστούν από την αρχή σε εκτελέσιμα αρχεία με τα νέα δεδομένα και να απεικονιστούν. Κατά αυτό τον τρόπο η εφαρμογή γίνεται ελαφρύτερη, γρηγορότερη και λιγότερο πολύπλοκη. Τέλος, το Vue είναι υπεύθυνο για την συλλογή των δεδομένων από τα διάφορα components και την δημιουργία HTTP Requests για την επικοινωνία της εφαρμογής με τον server διάφορα συμβάντα (events). Αναλυτικότερα, η δομή της εφαρμογής αποτελείται από ένα κύριο component του Framework7 το οποίο είναι η ρίζα όλων των υπολοίπων και μέσα του διαθέτει το component side-panel και ένα page component στο οποίο το Framework7 φορτώνει την σελίδα που θα διαλέξει ο χρήστης από το side-panel. Το μοντέλο που περιγράφηκε είναι γνωστό ως Single-Page Application καθώς η εφαρμογή δεν χρειάζεται να φορτώσει

πολλαπλές σελίδες σε κάθε ανακατεύθυνση αλλά αντίθετα διαθέτει πάντοτε μία και φορτώνει τα components που χρειάζεται σε αυτήν. Τα components που αναπτύχθηκαν για την εφαρμογή είναι συνολικά είκοσι (20) εκ των οποίων δεκαπέντε (15) είναι components αυτόνομα που φορτώνονται στο page component της εφαρμογής ενώ τα υπόλοιπα πέντε (5) είναι components που προσφέρουν λειτουργίες στα αυτόνομα.

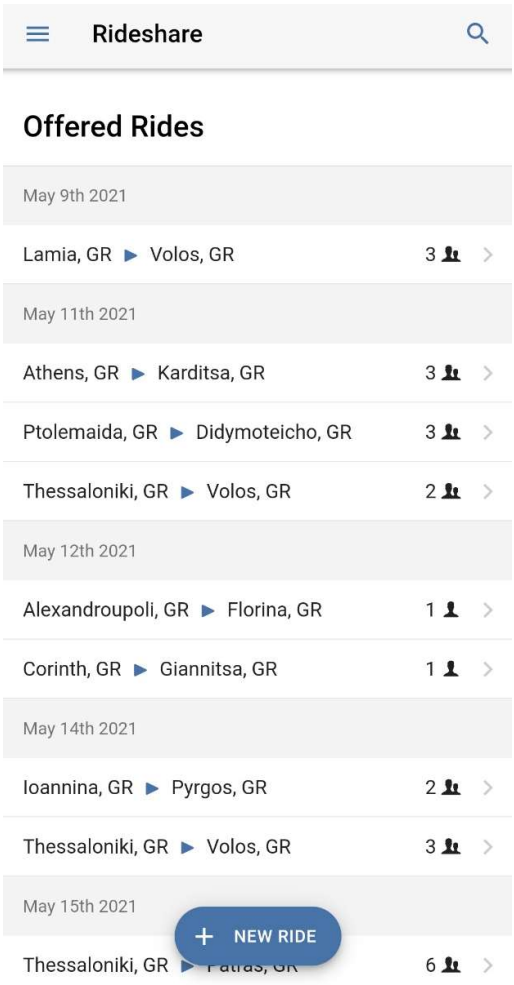
Login & Register components. Πρόκειται για δύο components με φόρμες συμπλήρωσης προκειμένου να πραγματοποιηθούν η εγγραφή και η είσοδος του χρήστη. Τα δεδομένα του χρήστη βρίσκονται αποθηκευμένα στο Vuex και ο μηχανισμός ανακατεύθυνσης φορτώνει το component Login εάν δεν υπάρχουν τα στοιχεία του χρήστη, ενώ σε αντίθετη περίπτωση φορτώνει το component offering όπου βρίσκονται όλες οι προσφερόμενες διαδρομές. Τέλος κατά την εγγραφή ή είσοδο αιτείται καινούργιο fcm token από το FCM για τον χρήστη και ενημερώνει την βάση δεδομένων σχετικά μέσω HTTP Request, προκειμένου ο χρήστης να λαμβάνει ειδοποιήσεις push.



Εικόνες 6 & 7: Σελίδες σύνδεσης και εγγραφής χρήστη.

Offering & Asking components. Δύο λειτουργικά αλλά και εμφανισιακά ίδια components τα οποία εμφανίζουν όλες τις προσφερόμενες διαδρομές αλλά και αυτές προς ζήτηση

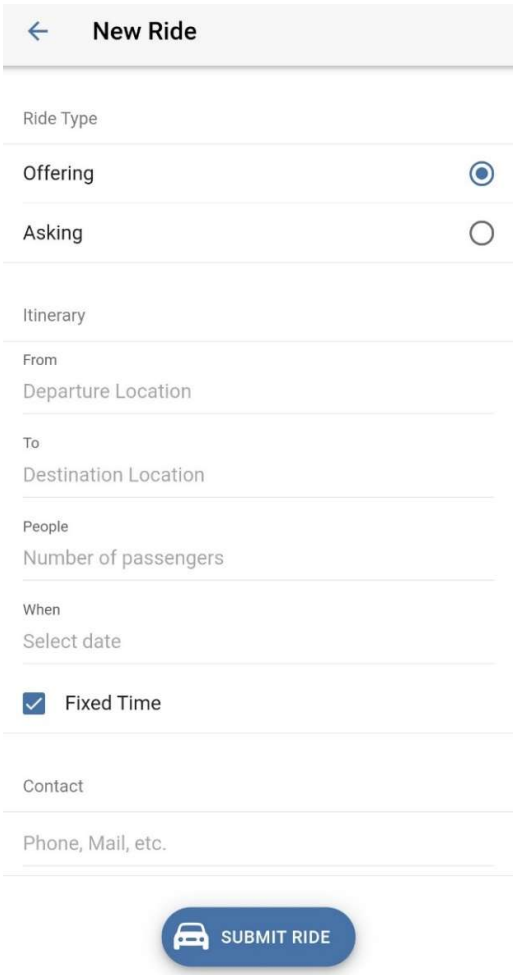
αντίστοιχα. Στα κάτω μέρος τους, όπως και στων υπολοίπων αυτόνομων components, υπάρχει το κουμπί νέας διαδρομής που φορτώνει το αυτόνομο component «New Ride». Τέλος, πατώντας σε μία διαδρομή από τις εικονιζόμενες, φορτώνεται το component Show Ride.



Εικόνες 8: Σελίδα προσφερόμενων διαδρομών και αναζήτησης.

New Ride component. Αποτελείται από μια φόρμα συμπλήρωσης για την δημιουργία μιας καινούργιας διαδρομής. Διαθέτει πεδίο radio για την επιλογή του τύπου της διαδρομής και πεδία εισαγωγής για τα σημεία αναχώρησης και άφιξης, τα οποία πατώντας επάνω τους ανοίγουν τα pop-over components setMapLocationDeparture και setMapLocationArrival, τα οποία αρχικοποιούνται με την τοποθεσία του χρήστη μέσω του

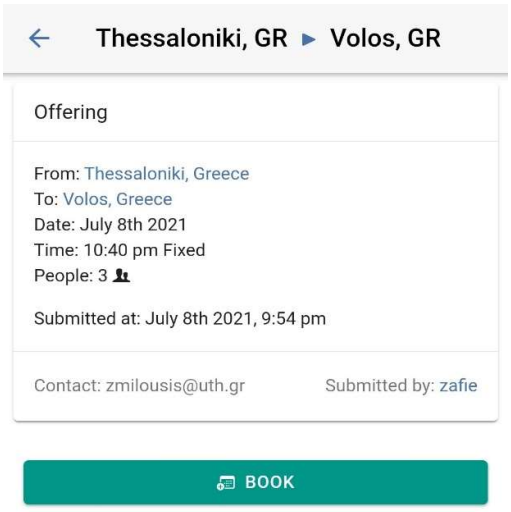
GPS της συσκευής και ο χρήστης μπορεί να τοποθετήσει πινέζα στο Google Maps με την τοποθεσία που επιθυμεί ή να αναζητήσει μία τοποθεσία γραπτώς και να δει το αποτέλεσμα στον χάρτη. Κλείνοντας τα εν λόγω components, η τοποθεσία που επιλέχθηκε συμπληρώνεται αυτόματα στο αντίστοιχο πεδίο εισαγωγής της φόρμας. Τέλος, η φόρμα διαθέτει πεδία εισαγωγής των διαθέσιμων θέσεων για επιβάτες στην διαδρομή, ημερολόγιο επιλογής ημερομηνίας και ώρας πραγματοποίησης της διαδρομής, πεδίο επιλογής τελικής ή μεταβλητής ώρας και πεδίο επικοινωνίας.



Εικόνα 9: Σελίδα δημιουργίας νέας διαδρομής.

Show Ride component. Πατώντας επάνω σε μια διαδρομή, από τα components που απεικονίζουν διαδρομές για κάποια κατηγορία, μεταφερόμαστε στο εν λόγω component όπου απεικονίζονται όλες οι διαθέσιμες πληροφορίες για αυτήν της διαδρομή. Πατώντας στα σημεία όπου αναγράφεται το σημείο εκκίνησης και άφιξης εμφανίζονται τα

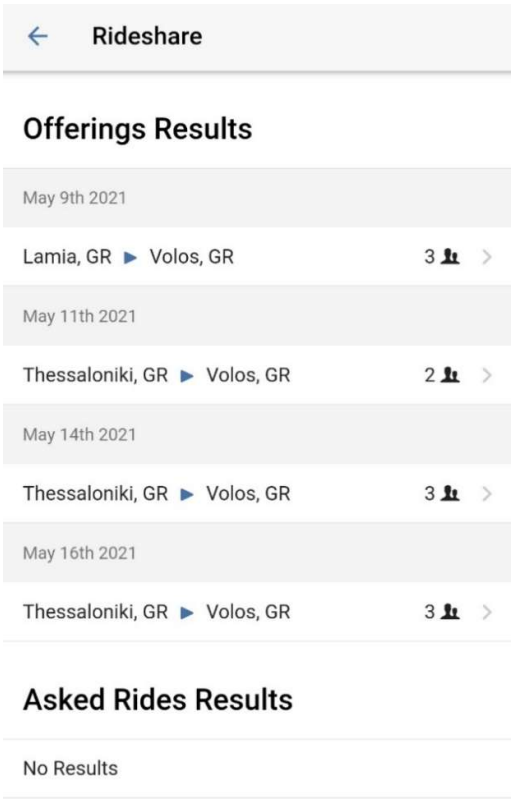
components showLocationDeparture και showLocationArrival αντίστοιχα όπου ο χρήστης μπορεί να δει στο Google Maps τις εν λόγω τοποθεσίες. Τέλος, κάτω από την κάρτα πληροφοριών της διαδρομής, εάν η διαδρομή που απεικονίζεται προσφέρεται από τον χρήστη που ζήτησε να απεικονιστεί, εμφανίζεται το κουμπί «Check Bookings» που φορτώνει το component Show Bookings ενώ σε αντίθετη περίπτωση, και εάν η ημερομηνία και ώρα πραγματοποίησης δεν έχει παρέλθει, εμφανίζεται το κουμπί «Book» το οποίο στέλνει HTTP Request στον server προκειμένου ο χρήστης να κάνει κράτηση στην διαδρομή εάν αυτό είναι δυνατό.

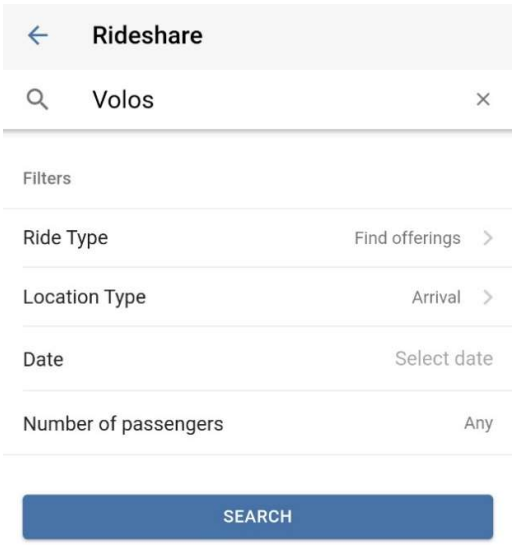


Εικόνα 10: Σελίδα προσφερόμενης διαδρομής από την οποία ο χρήστης μπορεί να πραγματοποιήσει κράτηση.

Search Ride component. Επάνω αριστερά στο nav-bar στοιχείο κάθε αυτόνομου component βρίσκεται το κουμπί search με το εικονίδιο ενός μεγεθυντικού φακού. Πατώντας το εν λόγω κουμπί φορτώνεται το component Search Ride με την φόρμα αναζήτησης διαδρομής. Η φόρμα αυτή διαθέτει πεδίο εισαγωγής όπου ο χρήστης μπορεί να γράψει την τοποθεσία που επιθυμεί να αναζητήσει ενώ επίσης προσφέρονται πεδία επιλογής φίλτρων, όπως τύπου διαδρομής, τύπου τοποθεσίας που δείχνει εάν η τοποθεσία που γράφτηκε είναι επιθυμητό να αναζητηθεί ως τοποθεσία εκκίνησης ή άφιξης, ημερολόγιο επιλογής ημερομηνίας, και πεδίο επιλογής διαθέσιμων θέσεων. Τέλος, στο κάτω μέρος υπάρχει το κουμπί «Search» που αποστέλλει στον server τις επιλογές μας και φορτώνει το component Search Results προκειμένου να απεικονιστούν τα αποτελέσματα.

Search Results component. Εδώ απεικονίζονται τα αποτελέσματα αναζήτησης. Απεικονίζονται σε λίστα όπως σε κάθε άλλο component και είναι χωρισμένα σε δύο λίστες ανάλογα με τον τύπο της διαδρομής και ταξινομημένα κατά ημερομηνία ξεκινώντας από την κοντινότερη, όπως και στα υπόλοιπα components.

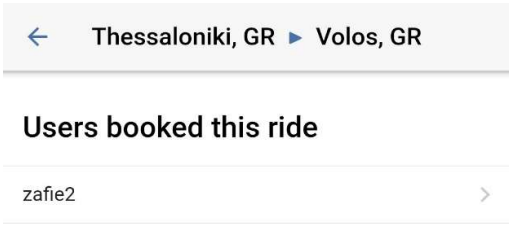




Εικόνες 11 & 12: Σελίδες αναζήτησης και αποτελεσμάτων αναζήτησης.

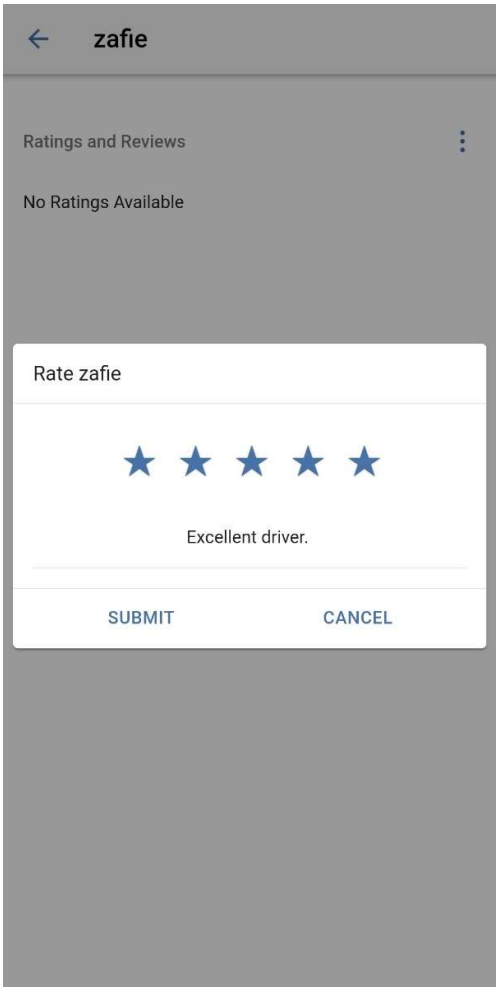
Upcoming Rides, My Rides, Suggested Rides & History components. Τρία αυτόνομα components απεικόνισης κατηγορίας διαδρομών. Το Upcoming Rides απεικονίζει όλες τις διαδρομές στις οποίες πρόκειται να συμμετάσχει στο μέλλον ο χρήστης είτε επειδή τις προσφέρει είτε επειδή έχει κάνει κράτηση σε αυτές. Το My Rides απεικονίζεις όλες τις διαδρομές που προσφέρει ή ζητάει ο χρήστης μελλοντικά. Το Suggested Rides απεικονίζει όλες τις διαδρομές που προτείνονται από τον server, όπως εξηγήθηκε στο κεφάλαιο του Server Side Laravel, για τον συγκεκριμένο χρήστη. Τέλος, το History απεικονίζει όλες τις διαδρομές στις οποίες έχει συμμετάσχει ο χρήστης στο παρελθόν είτε επειδή τις προσέφερε είτε επειδή είχε κάνει κράτηση σε αυτές.

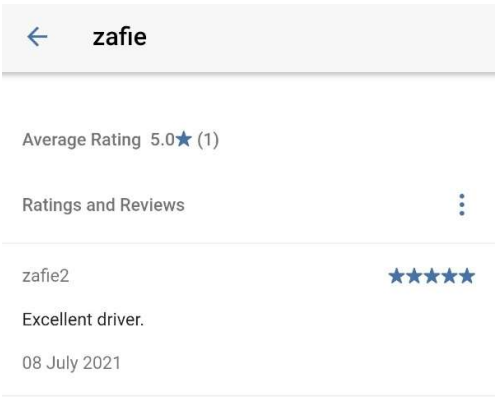
Show Ride Bookings component. Απεικονίζει το username όλων των χρηστών που είχαν κάνει κράτηση στην διαλεγμένη διαδρομή. Πατώντας επάνω στο username τους φορτώνεται το component Show User Ratings.



Εικόνα 13: Σελίδα λίστας χρηστών που πραγματοποίησαν κράτηση σε μία διαδρομή.

Show User Ratings component. Απεικονίζει το μέσο όρο αξιολογήσεων από τους χρήστες για τον διαλεγμένο χρήστη και κατόπιν απεικονίζει ξεχωριστά όλες τις αξιολογήσεις που έχει δεχθεί ο χρήστης μαζί το σχόλιο, εάν αυτό δόθηκε. Τέλος, ελέγχει εάν η διαδρομή από την οποία κατευθυνθήκαμε στον συγκεκριμένο χρήστη ανήκει στο παρελθόν ή πρόκειται για ανερχόμενη διαδρομή. Στην πρώτη περίπτωση που ανήκει στο παρελθόν εμφανίζει και το κουμπί «Rate» που εμφανίζει το component rateUser το οποίο δίνει στο χρήστη την δυνατότητα να διαλέξει βαθμολογία από το ένα έως το πέντε, μέσω αστεριών, και να γράψει κάποιο σχόλιο για τον χρήστη.





Εικόνες 14 & 15: Σελίδες βαθμολόγησης χρήστη.

Settings component. Φορτώνεται μόνο από το side-panel και δίνει στον χρήστη τη δυνατότητα να μεταβάλλει την κατάσταση του σκοτεινού θέματος της εφαρμογής μέσω κουμπιού-διακόπτη. Τέλος, διαθέτει κουμπί αποσύνδεσης του χρήστη από την εφαρμογή. Κατά την αποσύνδεση του χρήστη ενημερώνεται και η βάση δεδομένων μέσω HTTP Request και σβήνει το fcm token αυτού του χρήστη προκειμένου να μη λαμβάνει πλέον ειδοποιήσεις push.

ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Λειτουργίες που θα προστεθούν στο μέλλον για περαιτέρω διευκόλυνση των χρηστών στη χρήση της εφαρμογής είναι η επιβεβαίωση του email που δήλωσε ο χρήστης κατά την εγγραφή του μέσω αυτοματοποιημένου email από το Server - side τμήμα της εφαρμογής. Ως προέκταση αυτής τη λειτουργίας και αφού αυτή προστεθεί, θα προστεθούν ακόμη η λειτουργία υπενθύμισης κωδικού πρόσβασης του χρήστη επίσης μέσω αυτοματοποιημένου email καθώς και εγγραφή και είσοδος χρήστη μέσω λογαριασμών Google και Facebook. Οι επεκτάσεις αυτές θα αποδώσουν επιπλέον ασφάλεια και ευκολία χρήσης της εφαρμογής στους χρήστες. Εκτός από τις αυτές τις επεκτάσεις, υπάρχουν και μεγαλύτερης κλίμακας προσθήκες νέων λειτουργιών που πρόκειται να υλοποιηθούν. Η εξυπηρέτηση των χρηστών εντός τον πόλεων είναι η επόμενη μεγάλη αυτοματοποίηση που θα δώσει λύσεις στους πολίτες στην καθημερινότητά τους. Όταν ένας χρήστης έχει ανάγκη τον συνεπιβατισμό μέσα στην μέρα του για να κινηθεί εντός την πόλης του τότε η εύρεση της διαδρομής εκείνη την χρονική στιγμή είναι αυτό που ζητάει και όχι ο σχεδιασμός του ταξιδιού του στην πιο βολική ώρα και επιλογή της διαδρομής του ανάμεσα πολλές διαφορετικές προσφορές. Με αυτό κατά νου, θα υπάρξει ένας νέος τύπος διαδρομών, αυτός των διαδρομών εντός πόλης. Ο χρήστης επιλέγοντας αυτήν την κατηγορία στο έξυπνο τηλέφωνό του θα εισάγει μόνο στο χάρτη την διεύθυνση και το σημείο στο οποίο επιθυμεί να μεταβεί άμεσα και θα αποστέλλεται αίτημα στο Server – side τμήμα της εφαρμογής μαζί με την τοποθεσία που βρίσκεται εκείνη την στιγμή. Αφού διαπιστωθεί ότι οι δύο τοποθεσίες βρίσκονται όντως στην ίδια περιοχή τότε θα αναζητήσει προσφερόμενες διαδρομές επίσης εντός πόλης που τον εξυπηρετούν και θα πραγματοποιήσει κράτηση για αυτήν με ώρα πραγματοποίησης κοντινότερη στην στιγμή του αιτήματος του και θα επιστρέψει την διαδρομή αυτή. Η λειτουργία αυτή θα έρθει αμέσως μετά από την προσθήκη της επόμενης λειτουργίας και θα συνδυαστεί με αυτήν προκειμένου να επιτύχει τον σκοπό της. Η λειτουργία αυτή με την οποία θα συνδυαστεί είναι η δυνατότητα ο χρήστης να πραγματοποιεί κράτηση σε διαδρομές που βρίσκονται σε εξέλιξη. Αυτό θα γίνει μόνο εφόσον έχει ο ορίσει ο χρήστης που προσφέρει την διαδρομή ότι δέχεται να κρατήσεις κατά την διάρκεια της διαδρομής συμπληρώνοντας το αντίστοιχο πεδίο κατά την δημιουργία της. Έτσι όταν ο χρήστης – πελάτης αποστείλει

αίτημα είτε για άμεση διαδρομή εντός πόλης είτε κανονικής, το Server – side κομμάτι θα αναζητήσει και διαδρομές στις οποίες η τοποθεσία του πελάτη βρίσκεται μέσα σε κομμάτι της διαδρομής και εάν πρόκειται για εντός πόλης αίτημα θα ακολουθήσει την αντίστοιχη διαδικασία. Σε αντίθετη περίπτωση θα επιστρέψει τα αποτελέσματα της αναζήτησης στον χρήστη. Οι δύο αυτές επεκτάσεις σε συνδυασμό με τις προσθήκες λειτουργικότητας που περιγράφηκαν στην αρχή θα δώσουν μια πιο άμεση και ολοκληρωμένη και ασφαλή εμπειρία χρήσης στους χρήστες της εφαρμογής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Framework7 Vue, Framework7 Vue, 2017 [online]. Available: https://www.framework7.io/vue/

[2] Vue.js, Vue.js, 2014 [online]. Available: https://www.vuejs.org

[3] Laravel, Laravel, 2011 [online]. Available: https://www.laravel.com

[4] PostgreSQL, PostgreSQL, 1996 [online]. Available: https://www.postgresql.org/

[5] Firebase, Firebase, 2010 [online]. Available: https://firebase.google.com/docs/cloud-messaging

[6] Google Maps, Google Maps, 2005 [online]. Available: https://maps.google.com

[7] NGINX, NGINX, 2004 [online]. Available: https://www.nginx.com

[8] Git, Git, 2005 [online]. Available: https://git-scm.com

[9] Visual Studio Code, Visual Studio Code, 2015 [online]. Available: https://code.visualstudio.com/

[10] Android Tools, Android Tools, 2008 [online]. Available: https://developer.android.com/studio

[11] cURL, cURL, 1997 [online]. Available: https://curl.se/