UNIVERSITY OF TARTU
FACULTY OF SCIENCE AND TECHNOLOGY
Institute of Computer Science
Computer Science Curriculum

Gerrerth Kaur

A Prototype for a Real-time Mobile-based Ridesharing System

Bachelor's Thesis (9 ECTS)

Supervisors:
Satish Narayana Srirama (PhD),
Jakob Mass (BSc)

Tartu 2016

A Prototype for a Real-time Mobile-based Ridesharing System

Abstract:

Nowadays, more and more people are looking for ways how to use existing resources more effectively. In developed countries there is about 1-2 persons per car while the average car capacity is five people. Therefore, it is evident that ridesharing has a huge efficiency potential in this matter. Ridesharing saves a lot of time and fuel by finding the right balance between travel time, travel distance and carried passengers. Such balancing needs a dynamic TSP-type algorithm to offer the most optimized rides to the users. This thesis concentrates on creating a real-time mobile app which includes necessary functionalities for riders and drivers. Drivers can post their rides, see possible options with routes and riders who have booked a seat for their ride. Whereas riders can post their wishes to ride from A to B, book places in rides and see their statuses for all suitable rides. Additionally, the system will have support for push notifications and Google Maps API to maximize the user experience.

Keywords:

ridesharing, carpooling, Android, Parse, Google Maps API

CERCS:

P175 (Informatics, systems theory)

Mobiilipõhise Reaalajalise Sõidujagamissüsteemi Prototüübi Loomine

Lühikokkuvõte:

Tänapäeval otsivad aina enam inimesi uusi võimalusi, kuidas kasutada olemasolevaid ressursse efektiivsemalt. Arenenud maades eksisteerib üks auto iga 1-2 inimese kohta aga autode keskmine mahutavus on viis reisijat. Seega on ilmne, et sõidujagamisel on suur potentsiaal käesoleva probleemi lahendamisel. Sõidujagamine säästab palju aega ja kütust, sest sel puhul leitakse parim tasakaal sõiduaja, sõidudistantsi ja veetud reisijate vahel. Selline balansseerimine nõuab dünaamilist algoritmi, et lahendada tekkivat rändkaupmehe probleemi ning pakkuda kasutajatele kõige optimeeritumaid sõite. Käesolev töö keskendub reaalajalise mobiilirakenduse prototüübi loomisele, mis hõlmab endas funktsionaalsusi nii sõitjatele kui juhtidele. Juhtidel on võimalik postitada sõite, näha süsteemi poolt pakutud võimalusi koos teekonnaga ja näha sõitjaid, kes on reserveerinud koha nende sõitudes. Sõitjad saavad üles postitada oma soove sõitmiseks punktist A punkti B, reserveerida kohti sõitudes ning näha nende endi staatuseid kõikide süsteemi poolt välja pakutud sõitudele. Lisaks toetab süsteem rakendusesisest teadaannete saatmist ja asukohapõhiseid teenuseid, et pakkuda maksimaalset kasutajakogemust kõigile kasutajatele.

Võtmesõnad:

sõidujagamine, autode ühiskasutus, Android, Parse, Google Maps API

CERCS:

P175 (Informaatika, süsteemiteooria)

Acknowledgments

I would like to acknowledge Markus Villig and Iida Kaisa Urm for the contribution to this thesis. They were an influential part of the app development, as they helped to achieve the best possible user experience along with the ever-important design. Furthermore, I would also like to express my gratitude to the Taxify Engineering team whose assistance played a pivotal part in the thesis.

Table of Contents

Acknowledgments 4

List of Abbreviations 7

1. Introduction 8

1.1 Introduction 8

1.2 Motivation 9

1.3 Contribution 9

1.4 Outline 10

2. Background 11

2.1 Ridesharing 11

2.2 Technologies 11

2.2.1 Android 12

2.2.2 Parse 13

2.2.3 Google Maps 17

2.2.4 Crashlytics 18

3. Implementation 19

3.1 Android app 19

3.2 Back-end 20

3.3 Routing 21

3.4 Main flow 22

4. Usability testing 24

4.1 USE Questionnaire 24

4.2 Results 24

4.3 Analysis 24

4.3.1 Rider vs Driver 25

4.3.2 Usage 25

4.3.3 Manual feedback 26
5. Conclusions and future work 27
5.1 Summary 27
5.2 Further work 27
6. References 29
Appendices 33
A. Database model 33
B. Questionnaire 34
C. Questionnaire answers 36
D. Application 37
E. Licence 38

List of Abbreviations

API	Application programming interface
APK	Android application package file
GCM	Google Cloud Messaging
HTTP	Hypertext Transfer Protocol
IDE	Integrated development environment
JSON	JavaScript Object Notation
OS	Operating system
SDK	Software development kit

1. Introduction

1.1 Introduction

We live in a society which endeavours towards a zero emission world where next generations could live a better, more joyful life. Though at the same time the amount of cars is constantly growing [1] [2] [3] [4] [5] and fuel prices are breaking records for being record low for years [6]. Therefore, how can we make transportation more efficient to all of us? Fortunately, we have had a solution for those endless worldwide problems for decades now - it is called carpooling.

Carpooling (also called ridesharing) is a mean of transportation where people can share car journeys with more than one person travelling along. The general purpose of this is to decrease the number of cars by dividing people to other cars which are also headed the same direction, particularly at the same time. The results of this are reduced:

1) environmental impacts;
2) traffic congestions;
3) fuel costs;
4) time costs;
5) need of parking spaces;
6) stress of driving.

In the United States, there is one really good example of carpooling - HOV (high occupancy vehicle) lanes. These are regulated traffic lanes which are solely reserved for vehicles who besides to driver have one or more passengers on board. They have already existed in the States since the 1970s [7]. Although they haven't yet revolutionized the world, they have been a hot topic in the States since the founding of ridesharing companies Uber[1] and Lyft[2] a few years back.

In order to create a software system which provides efficient carpooling, several problems must be solved. The biggest challenge to tackle in this field is to find the right balance of efficiency between miles travelled, travel time spent and number of dispatched passengers. The keyword here is routing. In this field, routing means that after a user input (origin, destination, time etc.) the system calculates better, more efficient routes and trips to go on based on miles travelled, time spent and passengers carried. This can be interpreted as finding the shortest possible path while at the same time minimizing travelling time and maximizing passengers count. In essence, finding shortest possible path is a TSP (Travelling Salesman Problem). TSP is one of the most intensively studied problems in computational mathematics which asks the following question: given a collection of locations and distances between each pair of locations, what is the shortest route that visits every location once and returns to the origin [8] [9]. The uniqueness of this problem is the fact that there is no known algorithm which will always return the shortest possible route.

Nevertheless, reacting to a user's input in such a dynamic way is a great challenge, as a single user's input can potentially turn the whole system's users' routes and rides upside down. For example, when a user changes destination, then the system starts to search for

[1] https://www.uber.com
[2] https://www.lyft.com

the most efficient combined solutions for current user and nearby rides. The diversity of parameters and inputs with the scenario of offering most efficient routes to everybody means that the system has to work through a great amount of exponentially growing options. This is a solid basis for developing a sophisticated algorithm. The aim of the algorithm is to find the best possible options by minimizing travelled miles and time spent on travelling while additionally maximizing the count of passengers.

1.2 Motivation

The motivation for my thesis came from my personal experience of working in a company which offers a taxi-ordering/ridesharing application. As a consequence, I have learned two influential aspects of ridesharing which serve as the motivation for this thesis. To begin with, I believe ridesharing has huge efficiency potential in saving time and fuel. In developed countries there is about 1-2 persons per car on average [10] [11]. Since a prevalent part of cars are equipped with five seats, then on average with every ride it leaves at least three vacant seats into the car which is an excessive waste of resources. Resources like fuel, whichever metals cars are made of and whichever chemical substances are used for building roads. With ridesharing we can all optimize our car usage which will eventually result in a decreased usage of resources, decline of traffic congestions and a downturn to emissions.

Secondly, I believe ridesharing has an immense market potential all around the world. This is evident by the fact of world leading ridesharing companies' impressive market valuations:

- Uber at $68B [12]
- Didi Chuxing[3] at $25B [13]
- Lyft at $5,5B [14]
- Ola[4] at $5B [15]

Additionally, there are several companies who each have market valuations at more than $1B [16] [17]. I think it is reasonable to assume that the total market valuation of ridesharing businesses is around about $110B. Furthermore, we have to consider that this certain market will constantly keep growing, since we are living in the information revolution[5] where every day more and more people are familiarizing themselves with computers, smartphones, tablets etc.

1.3 Contribution

The main aim is to create a ridesharing mobile app that uses a TSP-type algorithm to enhance user experience. This app's main idea is to connect riders to drivers with empty seats. Drivers can post their rides, see possible options with routes and riders who have booked a seat for their ride. On the other hand, riders can post their wishes to ride, book a place and see their status for the ride. Furthermore, users will be notified when another user's decision has affected them. For example, a notification is sent to driver when a rider books a seat in the driver's ride.

[3] http://www.xiaojukeji.com/en/index.html
[4] https://www.olacabs.com
[5] Current economic, social and technological trend

1.4 Outline

In the introductory chapter, the background and current state of ridesharing is described. It is followed by the author's motivation and contribution for the subject as well as for the thesis. Subsequently, a brief review of the thesis' outline is also given.

Chapter 2 gives a more sophisticated overview about the matter. It discusses the problems and benefits of ridesharing while also bringing examples of how have previous applications solved the issue. Additionally, the chapter introduces, explains reason of choice and gives insight about the technologies which are used to achieve the solution.

In Chapter 3, the implementation of the proposed solution is described in detail. The chapter concentrates on answering to two essential questions:

- How does the solution work?
- What and how is used to create the proposed system?

A usability study is conducted in Chapter 4 where emphasis is put on testing the app's user experience. It is followed by a thorough analysis to exteriorize the pros and cons of the built application. In the last chapter of the thesis, conclusions are presented along with future directions and limitations for the built solution.

2. Background

This chapter concentrates on giving a more detailed background of ridesharing and also an extensive overview of technologies needed for the implementation.

2.1 Ridesharing

Ridesharing is an act of sharing a vehicle with more than one passenger to travel from A to B. The general goal is to reduce the number of travels where there are one or more vacant seats available. This can have a hugely beneficial effect on traffic congestions and automobile emissions. The biggest technological challenge here is to synchronously minimize vehicle miles with travel time and to maximize the number of passengers at the same time.

In today's world ridesharing applications can generally be divided into two subcategories. Ones are the applications that solely offer ridesharing as an alternative to on-demand taxi service where people without taxi driver's license can work as ridesharerers whenever they like. These are applications like Uber, Lyft, Taxify[6] etc. However, I think their implementation of ridesharing is incomplete. The way these applications work is that a person orders a taxi to a certain location, the taxi driver picks the client up and chauffeurs client to the desired destination. In this way, all of them are forgetting one of the essential properties of ridesharing – maximizing the number of passengers.

On the other hand, there are applications which follow all the traditional characteristics of the term ridesharing, e.g. BlaBlaCar[7], Hitch-A-Ride[8], Via[9] and Split[10]. Unfortunately, these do also have some downsides. For instance, Via and Split have defined routes based on most used pickup and destination points. This functionality makes them similar with an ordinary public transport in terms of having predefined routes. Although, the two would still offer much more flexible travel times than public transport, they would still force people to take a walk from home to a point on the route. Furthermore, all of the mentioned applications have a concept of preferring demand to overall effectiveness (minimize miles and travel times, maximize number of passengers). It means that for example if six riders want join the ride from various locations and driver randomly accepts four of them. As a result, the route efficiency is low because the route isn't optimized to the best possible. Thus my application would be different in the way that it will prefer overall efficacy to demand.

2.2 Technologies

This system uses several different frameworks and libraries to carry out its main function of helping people to use ridesharing more efficiently. By purpose we can divide the frameworks and libraries into two. Ones are which help the users to get the best possible experience and the others help the developers or system administrators to be more productive. This subsection concentrates on giving a concise overview of what frameworks do I use, what do they give me and why have I specifically chosen them.

[6] http://taxify.eu
[7] https://www.blablacar.com
[8] http://hitcharide.me
[9] http://ridewithvia.com
[10] http://split.us

2.2.1 Android

Android functions as a front-end for this application. It is a mobile OS which is based on the Linux kernel and primarily designed for running touchscreen enabled mobile devices. Via its interface, Android OS allows users to directly manipulate with the system by using touch gestures which correspond to real-world actions [18].

Obviously there are many alternatives to function as this application's front-end, such as iOS, a website, Windows Phone etc. I will briefly explain why a mobile application was preferred to a website. Firstly, people of today tend to always have a smartphone along with them everywhere, but this isn't the case with computers. Computers are far less portable than mobile devices. Secondly, nowadays it is possible to convert your website into an app, but this has some downsides. Converted app is much less responsive to user touch gestures and doesn't follow the guidelines set by different mobile OS manufacturers. These drawbacks undeniably harm the user experience.

Therefore, the front-end choice was between the three biggest mobile OS providers – Google with Android, Apple with iOS and Microsoft with Windows Phone. Here the most credible parameter is the market share of smartphone operation systems. Figure 1 illustrates two decisive points. Android OS is and has clearly been the market leader for the last few years. Additionally, Android OS is slowly but constantly growing its market share. Based on the mentioned arguments, I think Android is the evident winner.

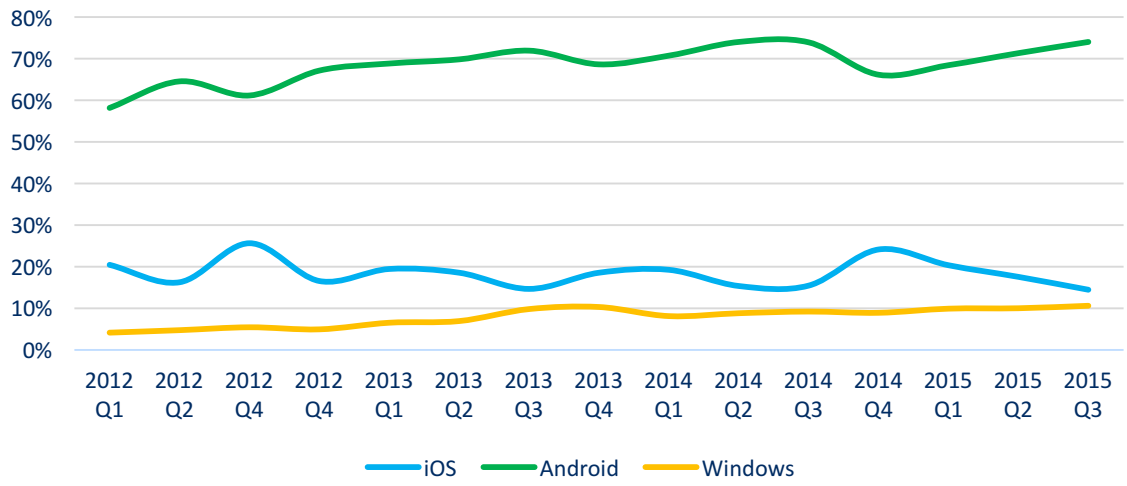

Figure 1. Smartphone OS market share in the following European countries: Germany, United Kingdom, France, Italy and Spain [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29].

Android's success has largely relied on Google's choices of maintaining it as a free-for-everyone open-source OS. This gives the project two significant prosperities. From the users' perspective, it makes mobile devices more attainable by being cheaper and offering mostly free-of-charge applications for downloading. On the other hand, the developers hugely benefit from this as well. Firstly, the OS has all the low-level kit as well as the needed middleware to power and use electronic devices. Secondly, Google has developed a complete application framework where third-party apps can be built and installed – Android Studio [30].

Android Studio is the official IDE for developing Android OS running applications. The IDE is built on the powerful IntelliJ IDEA[11]. For app development explicitly, Android Studio offers several productivity enhancing fundamental features:

1) flexile Gradle[12]-based system;
2) an all-powerful layout editor which is essential for developers because building and installing is excessively time-consuming;
3) specialized tools to dynamically check code usability, performance and version compatibility;
4) code examples for implementing trivial app features;
5) build variants and several ways to generate APKs [31].

2.2.2 Parse

Parse[13] is a cloud-based back-end provider, which allows predominantly mobile developers to easily create back-ends without having to concern about server maintenance and overly complex infrastructure. The provider offers back-end cloud services for storing data in the cloud, handle push notifications, run custom logic in the cloud and manage social identity log-ins. The cloud services are deeply integrated with SDKs for all the major client platforms – Android, iOS, JavaScript, OS X, PHP etc. [32].

Parse consists of numerous tools which help developers, system administrators and even marketing people to be more productive. Developers don't have to involve themselves with low-level database development, e.g. creating tables, views and relations with SQL. Instead, Parse offers a multifunctional website and a cloud-based API written in JavaScript which immensely decreases the work hours put into back-end building. System administrators can monitor system work from logs and Parse's own analytics. For marketing managers, Parse empowers sending push notifications from website and also enables them to analyse users' activities from the analytics. Subsequently, I will give a concise overview of all the various tools *Parse* offers.

2.2.2.1 Alternatives and reasons for choice

Since the beginning of smartphones era, mobile developers have sought for easily usable back-end tools to insure a fully reliable real-time service - a critical part of applications nowadays. Thus the land of back-end tool providers is remarkably diverse which only makes the choice even harder. My application's back-end was required to fulfil the following functionalities:

- data storage;
- a place where to write back-end and business logic;
- ability to send push notifications;
- custom Android library to simplify client-side query handling;
- analytics.

The last two are with lower priorities than the first three, but still necessary parts of the whole back-end experience. As a custom Android library will streamline developers' work

[11] Java IDE for developing computer software
[12] An open source build automation system
[13] https://parse.com

and analytics will give a whole another viewpoint to the application, so that the most significant problems will be put into development. Therefore, I have created a table for comparing the finest back-end services based on the aforementioned compulsory functionalities.

Table 1. Comparison of different back-end services [33] [34] [35] [36] [37] [38] [39] [40] [41].

	Data storage	***Cloud logic***	***Push***	***Analytics***	***Android SDK***	***Pricing (monthly)***
Parse	+	+	+	+	+	Free
Anypresence[14]	+	+	+	+	+	Free trial
AWS Mobile Hub[15]	+	+	+	+	+	Custom
Backendless[16]	+	+	+	+	+	Free
Telerik[17]	+	-	+	+	+	39$

Table 1 clearly indicates that there isn't much difference between the top back-end services, as all of the providers meet the requirements we set before. Nevertheless, there obviously has to be a parameter to distinguish the referred services from one another. In this case, the decisive key is pricing. In that respect, as visible from Table 1, Parse and Backendless differ from others. The others charge a fixed monthly price (e.g. Telerik and Anypresence) or a customised pricing (e.g. AWS Mobile Hub) which depend on the needs of the client. In the race between Parse and Backendless there were two factors that tilted me towards choosing the first one. Firstly, the *Parse* documentation seemed to be better than Backendless', as it was much more organised and the structure was fairly intuitive. Secondly, *Parse* is owned and run by Facebook which assured me that this is an evolving and long-lasting service.

2.2.2.2 Dashboard

Parse provides developers with an interface (onwards called Dashboard), located on Parse's own website, for administrating several functionalities:

1) Database
2) Cloud Code (with Background Jobs)

[14] http://www.anypresence.com
[15] https://aws.amazon.com/mobile/
[16] https://backendless.com
[17] http://www.telerik.com

3) Push notifications
4) Analytics
5) Back-end logs
6) Back-end configuration

2.2.2.3 Database

For storing the necessary data, Parse uses a cloud-based MongoDB database which can be easily accessed through the Dashboard. There one can easily create, edit and delete:

- classes (database tables);
- class fields (database table columns);
- class instances (database table rows).

Parse has predefined specialized classes like *User*, *Installation*, *Rol*e and *Product*. *User* is made unique by having fields like *username*, *password*, *authData*, *email* and *emailVerified*. You can access these fields comfortably through a method created by Parse. *Installation* has an essential role to play in here, because it is responsible for all the handling of push notifications. All device related information is saved into the *Installation* class.

Parse also equips engineers with specialized types for class fields. Some of them are very common to use in database development, e.g. *String*, *Number*, *Boolean*, *Date* and *Array*. Though others are not that trivial to use, e.g. *File*, *GeoPoint*, *Pointer* and *Relation*. All of the non-trivial types have some specific functionalities which make them very handy to use in Cloud Code.

File can be simply created by entering desired name and file extension with an array of byte value numbers or with a base64-encoded string. After that you can use *File* methods for asking the file's URL and name [42]. *GeoPoint*, which consists of latitude and longitude, provides methods that return kilometres, miles or radians between two *GeoPoint*s [43]. Types *Pointer* and *Relation* are for holding dependent class instances of some other classes. This functionality comes in handy when making queries, because it enables to include other classes' instances with one line of code in Cloud Code.

Furthermore, Parse has an intuitive security system for accessing class instances - it is called ACL (Access Control List). ACL is used to control which users can access or modify a particular object. From the ACL object you can ask whether a user has read or write access and also edit read or write access if you have sufficient rights to do so [44]. Whenever Cloud Code needs access to class' private instances, it can use a predefined method for this - called *useMasterKey*.

2.2.2.4 Cloud Code

Cloud Code acts as an API for my application. It is built on the V8 JavaScript Engine[18] and thus uses JavaScript code for functioning. The code runs on the Parse servers to react to changes in data.

[18] https://developers.google.com/v8/

The primary benefit of Cloud Code is the fact that you can move some of the app logic from app to the server. This has multiple benefits. Firstly, it definitely decreases CPU usage of the app which avoids possible performance issues on your smartphone, especially when we need to do CPU-heavy calculations. Secondly, since the logic is in one place, then developers don't have to the change app behaviour on different platforms. Instead, they can only make changes in Cloud Code to have an effect on numerous platforms. Third reason is subsequent from the second – developers can change app logic and add new features without having to publish an update to the app. This is a tremendous perk when developing on mobile platforms because updating an app is an excessively time-consuming activity with all of the reviewing before update and people taking their time with updating after the new app has been released. Secondary benefit is the intuitiveness of JavaScript which I think is a great prosperity for every kind of programmer.

In addition, Parse allows to set up Background Jobs in Cloud Code. Background Jobs are tasks which run in the background and can be scheduled to specific time. It is useful for long running tasks such as communicating with external sites or sending out larger amounts of push notifications [45]. For example, we have a job which is triggered every 10 minutes and it check the departure time of the ride. If the there is less than an hour till the departure, then we send push notifications to driver and all of the riders.

2.2.2.5 Push notifications

Parse enables back-end to send push notifications by using GCM. Push notifications is a way of notifying application's users of events happening in app [46]. One way is to trigger them from inside Cloud Code, where you can target them to specific user(s) or channels, which can be created based on users' activities. Other way is to send notifications manually from the Dashboard. Moreover, the Dashboard enables system administrators to create campaigns and experiments for sending notifications.

2.2.2.6 Parse Analytics

Parse Analytics is a tool on Dashboard for making your app analysing as effortless as possible. From the tool you can examine numerous indicators:

1) Audience – activity of installations and users.
2) Events – total of queries, app openings, pushes and push openings.
3) Data – number of class instances in tables.
4) Retention – % of active users' X days after signup.
5) Performance – total number of queries and failed queries compared to queries' limit.
6) Slow queries – speed of queries.
7) Crashes – information about app stopping working.

In events you can also supervise custom-made events which are created, filled with necessary data and sent from Cloud Code or inside the app after an activity you want to analyse.

2.2.2.7 Parse Android library

Parse also provides a SDK for Android which supports Android 2.3 and higher [47]. The SDK is much alike Cloud Code in terms of functionalities, but it doesn't include any

before or after save triggers. Thus making the Cloud Code usage much more scalable. More importantly, the application implemented in this thesis uses the mentioned SDK's custom objects for handling data more effectively. Most frequently used objects from Parse Android SDK are:

- *ParseObject* – initial class of every object we get from queries.
- *ParseUser* – get current user and includes default methods for getting email, username and password.
- *ParseAnalytics* – enables events sending to Parse Analytics.
- *ParseFile* – predefined methods for getting saved file.

2.2.3 Google Maps

Google Maps[19] is a mapping service which provides maps and route-planning capabilities. The Google Maps APIs gives developers numerous ways of integrating Google Maps into multiple different platforms, e.g. Android, iOS and web browsers. Furthermore, it enables engineers to retrieve every kind of location-related data with an option to extensively customise the given maps [48]. From the various Google Maps APIs this application uses two – Google Maps Android API and Google Maps Directions API.

Around the world there are many different service providers which offer location-related data. The first requirement for this application was that it should display a map inside my app. Meaning that the map provider should also offer me their own Android SDK. Although there were a few promising candidates for this place, like Bing Maps, Foursquare, Waze and Yandex, they all didn't met the previously set requirement. Eventually only two providers met this condition – Google Maps and Here Maps[20]. Though the latter had one unfavourable aspect concerning the usage of its API and Android SDK. The free version for both was only available for 90 days [49]. At the same time Google Maps offers unlimited free usage for its Android SDK and somewhat limited (up to 2500 queries per day) free access to Directions API [50]. Moreover, Google Maps has the most coherent documentation and I had had earlier experience with Google Maps APIs. All in all, that is why Google Maps API was chosen to be my location service provider.

2.2.3.1 Google Maps Android API

Google Maps Android API enables the application to add maps based on Google Maps data. The API handles location data downloading, map displaying, responses to user's gestures on map and access to Google Maps servers [51]. Additionally, the API provides me with the necessary graphics to draw markers and lines onto the map for better visualisation, such as the routes included with every trip.

2.2.3.2 Google Maps Directions API

With the Google Maps Directions API, you can calculate directions between multiple locations by using an HTTP POST[21] request which returns data in JSON [52]. JSON is a human-readable text format to store, organize and transmit information between server

[19] https://maps.google.com
[20] https://maps.here.com
[21] Submits data to be processed to a specified resource

and application [53]. The API enables to search for directions in numerous modes of transportation – driving, transit, cycling and walking. With search you can specify parameters for origin, destination and possible waypoints as text (e.g. “Tallinn”, “Tartu”) or as latitude/longitude coordinates [52]. This application mainly uses driving mode with multiple waypoints which are given in latitude/longitude coordinates.

2.2.4 Crashlytics

Crashlytics is a free cloud-based crash reporting solution offered by Twitter. It aims to give developers as much environmental information about the state of the device when the app crashed [54]. Vital information, such as:

- stack trace;
- app version;
- device production company;
- device OS;
- etc.

3. Implementation

The following chapter describes the implementation of the ridesharing mobile app developed as part of this thesis. It will extensively include explanations to how the various tools, frameworks and libraries are connected with the mobile app and back-end. The Figure 2 illustrates which components are connected to each other.

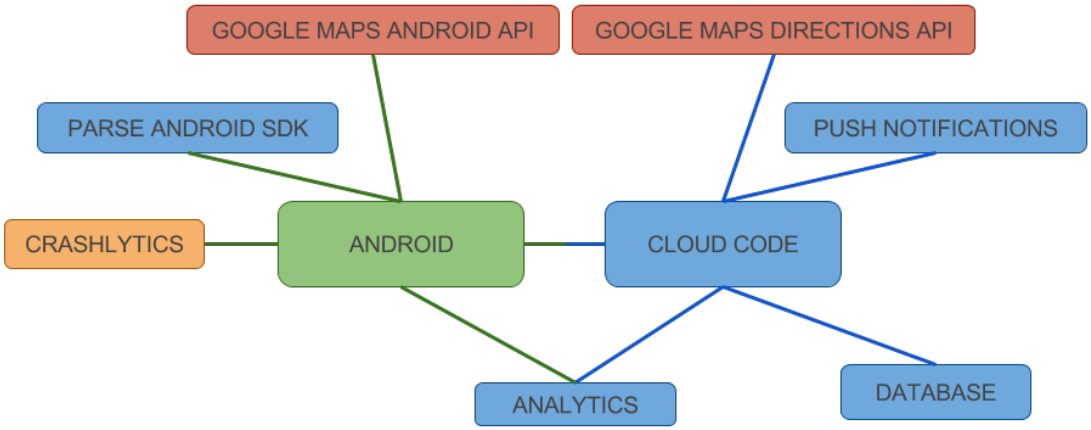

Figure 2. Simplified scheme for displaying how are components connected to one another.

3.1 Android app

The Android app (Appendix D) is responsible for the front-end. It was written Java programming language using Android Studio.

The app covers two major roles: driver and rider. Driver is the application's user who wants to ride from A to B and has at least one empty seat available in their car. Therefore, drivers can post rides in order to share rides with riders. Additionally, drivers can see all of their rides, rides' details and riders of their rides. Rider is the application's user who wants to ride from A to B, but has no car for it. Thus, riders are able to post wishes, after which they are offered the best rides. In this context, wish is a user's request to be involved in rides' riders. Analogously to drivers, riders can see all of their wishes, wishes' details and suitable rides.

Also, there are a few libraries implemented into the mobile app. Ones are external changes which enhance the user experience. For instance, the integration of Google Maps Android API which offers complete handling and visualizing of map-related functionalities, such as displaying a dynamic map, returning marker's latitude/longitude, drawing desired routes etc. Map and marker's location is used in views where we need to set an origin or a destination (Figure 3). Furthermore, map and its functionality of drawing routes with several waypoints and markers is needed in views where the route of ride, wish or option is displayed (Figure 3).

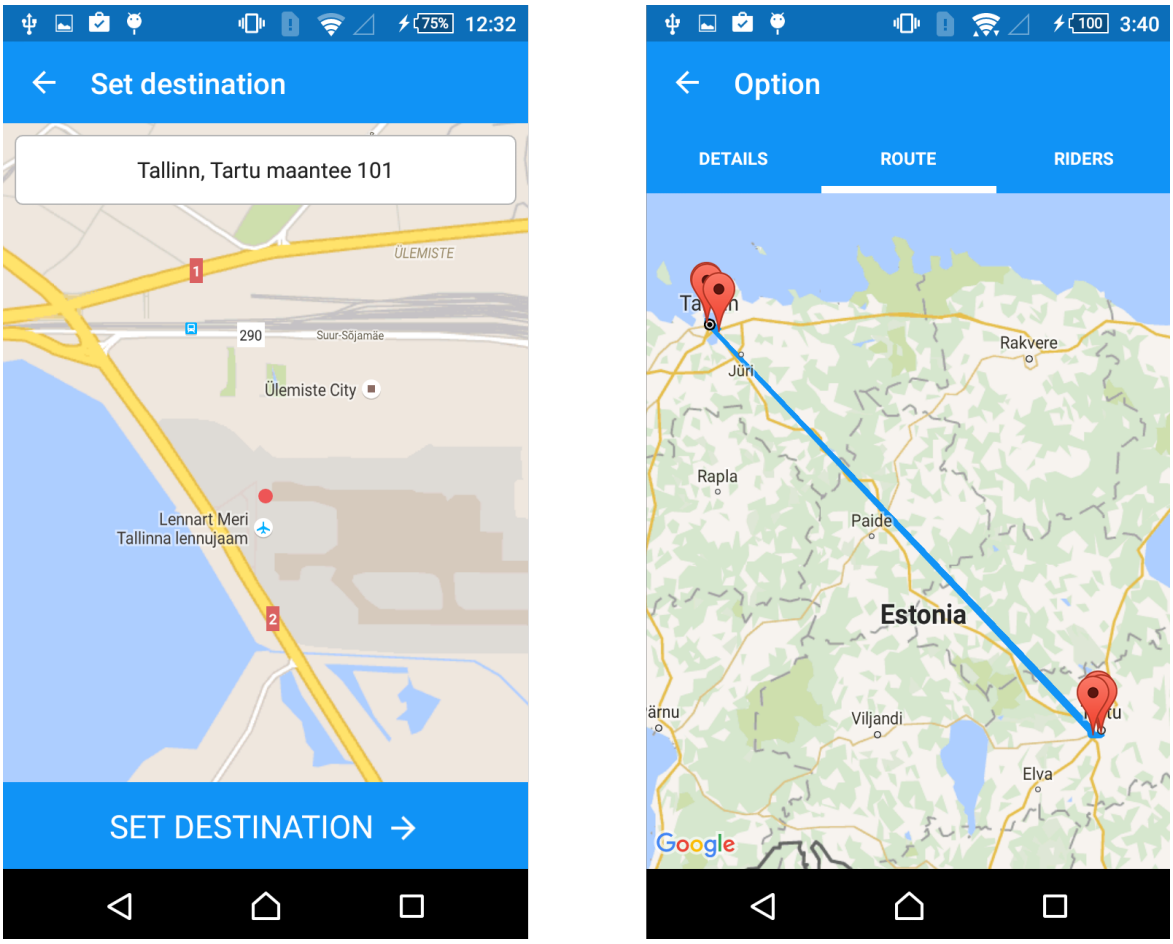

Figure 3. "Set destination" view is on the left and "Route" view is on the right.

The others however, are internal changes which mean that they make developers work more productive. The first from those is Parse's Android library whose biggest responsibility is to execute safe and secure HTTP queries. Additionally, the library provides a broadcast receiver for receiving push notifications. The second is Parse Analytics which is enabled while opening the app. It saves predefined events, such as opening the app, users' activity, installations' activity etc. In addition, you can send custom-made events from app-side with a one liner. The third useful library is Crashlytics which aside from the automatic crash-reporting can also report custom-made crashes called non-fatals.

3.2 Back-end

The server-side consists of the MongoDB type database and the API for it – Cloud Code. The database structure is shown as a diagram in Appendix A in Figure 7. Cloud Code is responsible for the communication between database and app, as all the queries and their logic is written there. Cloud Code was written using Atom[23] as it supports JavaScript.

Push notifications are only triggered from Cloud Code with the help of *Installation* class where all the necessary users' device-related data (e.g. *deviceToken*, *deviceType*, *pushType* etc.) is being held. There are three different pushes that are sent to users. The first one is

[23] https://atom.io

sent to driver after rider joins the ride. The second is sent to rider when driver wants to remind the rider to join the ride. The third push is sent to all of the ride's riders after driver has confirmed the ride.

In the back-end, Google Maps Directions API is extensively used to find route directions for users. A HTTP request is sent to Google API with the following parameters:

- origin latitude and longitude;
- destination latitude and longitude;
- latitudes and longitudes of waypoints the route has to pass;
- mode of transport which is driving in our case;
- API key for our application.

The request returns data about the route directions. Every turn of the route is separately brought out in the data with individual information about distance and duration between the turns. Therefore, one can obtain very multifarious information about the route. For example, totals of duration, distance and price for the whole route. Furthermore, one can receive detailed information about every riders' duration, distance and price for the route they drive with the ride.

As was with the Android app, Parse offers sending customized events to Analytics with one line of code.

3.3 Routing

Routing algorithm is the backbone functionality for the thesis. As a consequence of creating a ride or a wish, back-end calculates the best possible options for the driver or rider, respectively. Option is back-end's way of suggesting the most optimized routes for the drivers or riders to go on. It contains the following information:

- route waypoints;
- total distance, duration and price;
- passengers with personalised data (distance, duration and price).

In order to generate the best routes, the back-end works through several crucial steps:

1. A customized filter sorts out the riders who are inside the default radius - 11,1 km.
2. A matrix of distances is generated for desired locations which include origins and destinations of this ride and previously filtered riders. The distances are calculated based on locations' latitude/longitude values using the Haversine formula which takes into account the sphere shape of the Earth.
3. All combinations of routes are created
4. Shortest routes are found by going through the generated combinations and using distance matrix.
5. For every found route, back-end makes a query to Google Maps Directions API which returns waypoints, total distance and total duration of the route. Additionally, waypoints are a significant source of data, as it gives the back-end the ability to compute total price of the route and some rider-specific fields, such as rider cost, travel distance and travel time for the route.

3.4 Main flow

In this paragraph I will give an illustrated step-by-step example of a use case where all the app's main functionalities are included.

Step 1. Villu, a rider, creates a wish to ride from Tartu to Tallinn. Routing algorithm generates options for this specific wish. Villu sees all the options (Figure 4), but finds none of them suitable.
Step 2. Mati, a driver, creates a ride which drives from Tartu to Tallinn with two vacant seats. Routing algorithm generates options for this specific ride. Mati sees all the options (Figure 4), but opts not to proceed with the options at that time.
Step 3. Kati, a rider, creates a wish to ride from Tartu to Tallinn. Routing algorithm generates options for this specific wish. Amongst the options (Figure 4), there is an option where the driver is Mati and the second rider is Villu. Kati finds this particular option suitable, as it has a low cost to travel time ratio. Therefore, Kati joins the option which triggers a notification sent to the driver – Mati.
Step 4. Mati opens the app from the notification. The particular option's view is opened. Mati sees that Kati has joined but unfortunately Villu hasn't joined yet. Mati alerts Villu with a notification to join the particular option as well.
Step 5. Villu opens the app from push with the particular option's view in front. Villu think that the option is suitable, as the departure time suits him. Villu joins the option after which a notification is sent to the driver – Mati.
Step 6. Mati can now confirm the option and with it the ride because all of the riders have joined and thus the car is full. Mati confirms the ride and notifications are sent to Villu and Kati to remind them that the ride will take place at a certain time.
Step 7. Villu and Kati open the app from push and see that the ride will happen.

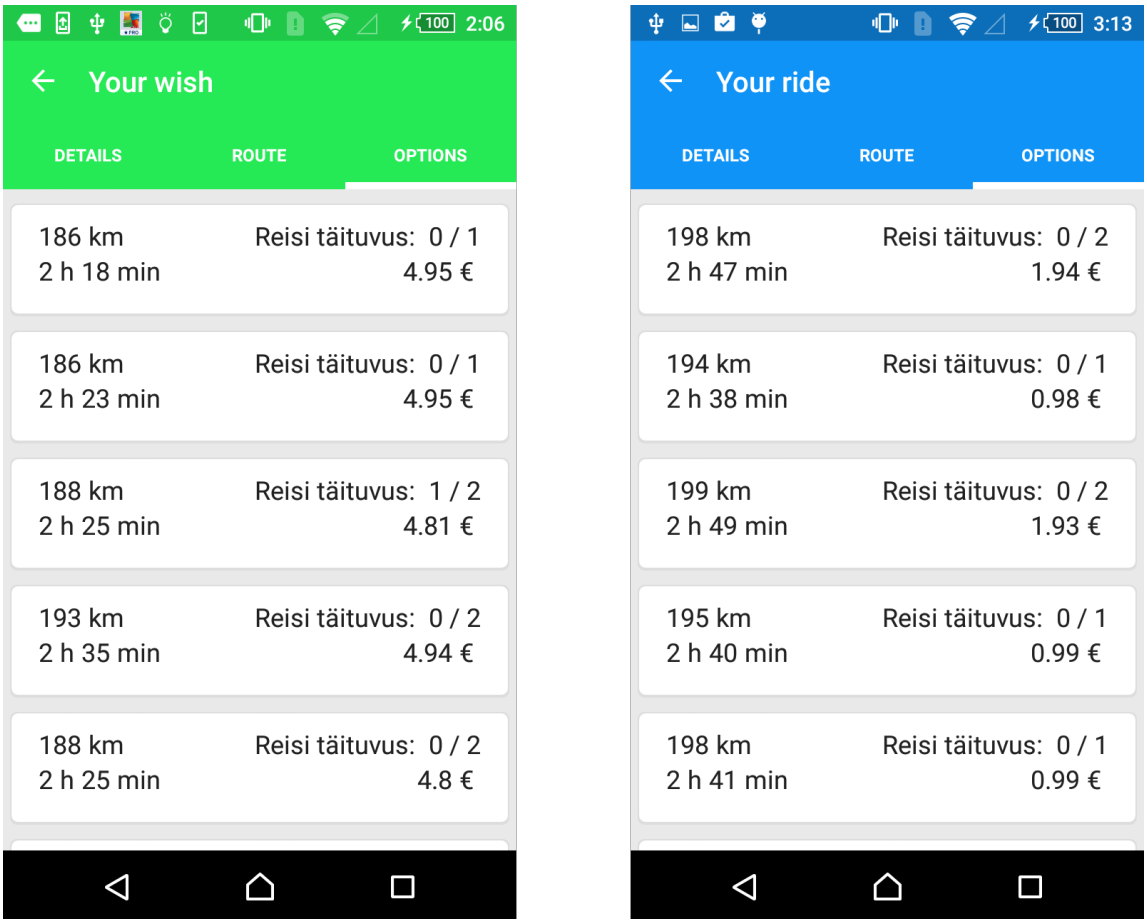

Figure 4. "Wish options" view is on the left and "Ride options" view is on the right.

4. Usability testing

Usability testing is a technique used to assess a product's usability by testing the product directly with the users [55]. This is most commonly used to test user interactions with the product to find potential flaws in the user experience. Nowadays, it is considered to be an inevitable part of developing mobile applications as smaller screens and uncertain performance levels demand the best possible handling of flows and error cases. Mobile applications are much more vulnerable and sensitive to user experience related changes. Whereas with websites, the user experience is very straightforward, because the navigation between pages is simplified due to larger screens.

A survey was conducted to measure the usability of this application. There were 15 participants between the ages of 18 to 46. All of them had an individual smartphone with some of them also having technological backgrounds. All of the testers were told what to accomplish with the app. The participants had to fill the questionnaire for each of the two scenarios:

- user wants a ride from Tallinn to Tartu and joins a suitable ride;
- user drives from Tartu to Tallinn and needs to remind riders with a push notification.

4.1 USE Questionnaire

To be precise, an USE type of questionnaire was created. It was constructed as a five-point Likert rating scale (from negative to positive): strongly disagree, disagree, neither agree nor disagree, agree, strongly agree. The questionnaire is divided into four dimensions: usefulness, ease of use, ease of learning and satisfaction. These dimensions' somewhat influence one another. For instance, ratings of ease of use improve along with the usefulness ratings and vice versa [55]. Additionally, for more detailed analysis the survey includes a few general question including age, gender and technological background. The questionnaire is displayed in Appendix B. It was created using Google Forms[24] due to its ease of use and supplementary functionalities, such as automatically composing analysis for the given answers.

4.2 Results

The results are displayed in Appendix C.

4.3 Analysis

The survey was constructed as a five-point Likert rating scale, ranging from strongly agree to disagree. Alternatively, we can say that the scale ranged from one to five. Therefore, strongly disagree corresponds to one, disagree to two, neither agree nor disagree to three etc. All the graphs were created based on this congruity.

[24] A website where people can create custom surveys, questionnaires, polls etc.

4.3.1 Rider vs Driver

Figure 5 shows average ratings that riders and drivers gave to each of the dimensions. The figure indicates that the usability of the two aforementioned scenarios is approximately equivalent. Even though, the driver scenario is the winner in only one dimension - usefulness. I believe it is fair to say that usefulness measures the potential value and productivity of what the app could be. In other words, it measures the app's main idea. Therefore, instead of rating usefulness, participants actually rate the potentiality. Thus, we can conclude that the driver scenario has more potency than the rider scenario.

On the other hand, the rest of dimensions rather measure the real value and productivity of the app which is in their hands. Figure 5 clearly indicates that in those scenarios the rider scenario is clearly the better one. This means that the usability of the rider flow is much more intuitive and understandable than the driver flow. This is somewhat comprehensible because creating a ride requires twice as much steps as creating a wish. All in all, the driver flow needs more rework than the rider flow.

Additionally, Figure 5 displays that ease of learning has the highest rating amongst riders and drivers. Thus, the application's best usability indicator is how considerably easy it is to learn how the application works. The figure also shows that the satisfaction level is the lowest usability indicator. Hence, we can say that the app's primary idea could be implemented better.

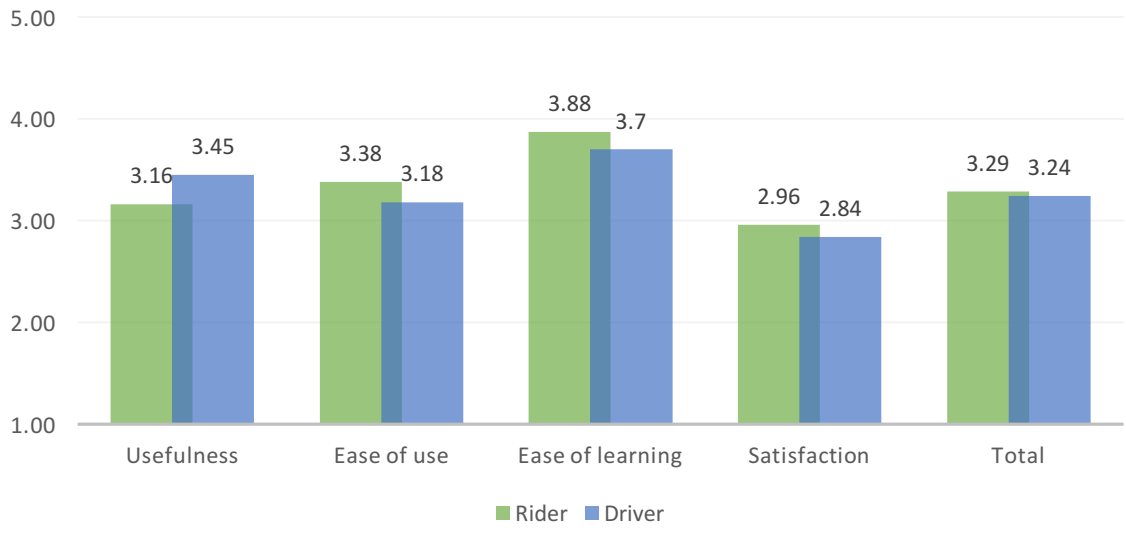

Figure 5. Average rating by dimension

4.3.2 Usage

It is visible from Figure 6 that people with IT-related education have given the highest ratings. This is rather unusual because these people should be the most critical when giving feedback to technological solutions. Nevertheless, it shows that the expertise value of the app is relatively high.

Figure 6 also presents that frequent users of technological devices have given the lowest ratings, even though people with occasional usage of mobile devices are also filtered in this questionnaire. On the one hand, it shows that for beginner level users the simplicity of

the app is quite good. Though on the other hand, the frequent users aren't very satisfied. I think the reason for this are low ratings for ease of use and ease of learning compared to other level users. I believe this implicates that the flow of the app is slightly unusual when we compare it to other booking-based apps, such as Taxify, Airbnb[25] and Airdine[26]. Therefore, the flows needs comparing and reviewing with other similar apps to maximize the user experience of frequent users.

Otherwise, Figure 6 is indicating similar points to the previous figure – Figure 5. The ease of learning has the highest and satisfaction the lowest ratings.

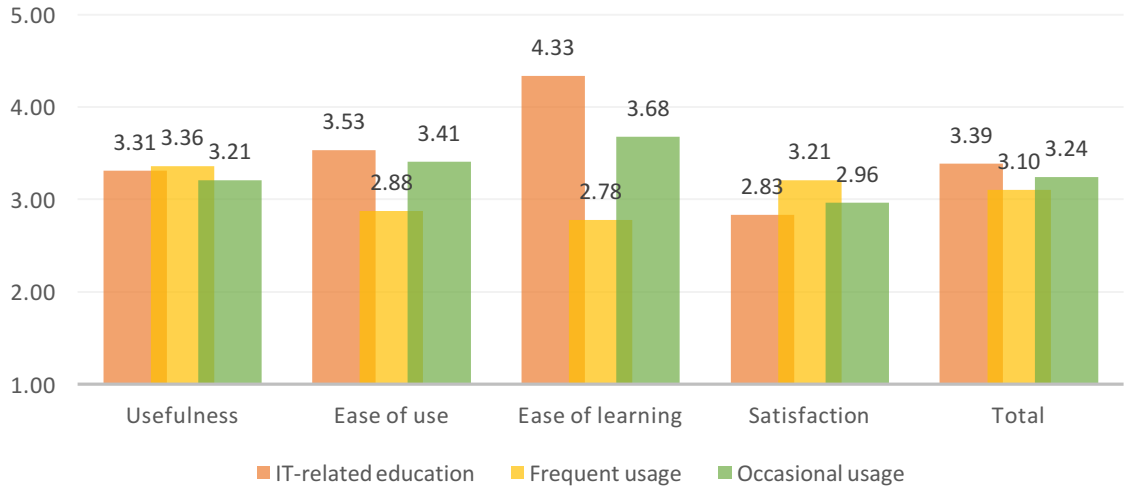

Figure 6. Average rating by technological background

4.3.3 Manual feedback

The survey also included manually written feedback. The following drawbacks were mentioned the most:

- creating a ride flow was too long;
- both riders and drivers weren't sure whether the ride was going to happen at all;
- navigating till the needed result (joining or confirming the ride) was too lengthy;
- info fields weren't undubiously understandable as the designs were incomplete.

These comments also imply to the fact that the flows should be deeply thought through again in order to enhance the user experience.

[25] https://www.airbnb.com
[26] https://airdine.com

5. Conclusions and future work

5.1 Summary

Nowadays, more and more people are looking for ways how to use existing resources more effectively. In developed countries there is about 1-2 persons per car while the average car capacity is five people. It is evident that ridesharing has a huge efficiency potential in this matter to save time and fuel. Therefore, the main aim of the thesis was to create a ridesharing mobile which uses a TSP-type algorithm to enhance user experience. The main idea of this was to connect riders to drivers with empty seats.

An Android app was developed which offers functionalities for two separate but dependant roles: driver and rider. Drivers can post their rides, see possible options with routes and riders who have booked a seat for their ride. Whereas riders can post their wishes to ride from A to B, book places in rides and see their statuses for all suitable rides. Additionally, users will be notified with a notification when another user's decision has affected them. For example, a notification is sent to all of the ride's riders when the driver has confirmed that the ride will happen.

Furthermore, a few additional tools were integrated to the system, in order to make the app better in any way. For instance, Google Maps was implemented into the app to enhance user experience. Also Parse Analytics was enabled to analyse different events that are being sent by the back-end or the app. Additionally, Crashlytics was also enabled to obtain reports about app crashes.

The usability study brought out a few key issues regarding the developed solution. The most severe issue of them was the fact that the app flow is more or less unwonted and definitely too lengthy. Additionally, there were some smaller matters mentioned as well, e.g. users didn't realise what the displayed info indicates.

5.2 Further work

I believe a mobile app is never completely ready. A mobile app can be very close to readiness, but there is always room for improvements, such as adding a new feature, enhancing the user experience, fixing bugs or even refactoring code. Therefore, it is safe to claim that this app is far from perfection. I will a give brief overview of upon what limitations did I stumbled on and which functionalities are needed the most for this solution to work as productively in real-life as possible.

First of all is the routing algorithm. In ideal case, the algorithm should only calculate three of the most optimized options to go on. For this an efficiency parameter should be automated, so the people would see how optimized are the few options that are offered. Additionally, at the moment the algorithm calculates the best options by using locations' latitudes and longitude which means the distances are measured as the crow flies. The algorithm should definitely use real road data offered by various mapping services because then the users would be assured of the most optimized routes. The biggest limitation to the ideal case is the Google Maps' requests limit which is only 2500 for 24 hours (if you use the free version). This algorithm would have exceeded it easily by making just a few request in a couple of seconds.

Furthermore, there is one unavoidable limitation – Parse will be shut down on January 28, 2017. Fortunately, Parse and many other similar frameworks have provided developers with convenient tools to migrate Parse offered services to another services' back-end.

The author will continue to contribute to the application, as it serves a huge efficiency potential amongst our everyday activities.

6. References

[1] P. LeBeau, “Whoa! 1.7 Billion Cars on the Road by 2035,” CNBC, 12 November 2012. [Online]. Available: http://www.cnbc.com/id/49796736. [Accessed 3 May 2016].

[2] “2015 Production Statistics,” OICA, [Online]. Available: http://www.oica.net/category/production-statistics/. [Accessed 16 February 2016].

[3] “2014 Production Statistics,” OICA, [Online]. Available: http://www.oica.net/category/production-statistics/2014-statistics/. [Accessed 16 February 2016].

[4] “2013 Production Statistics,” OICA, [Online]. Available: http://www.oica.net/category/production-statistics/2013-statistics/. [Accessed 16 February 2016].

[5] “2012 Production Statistics,” OICA, [Online]. Available: http://www.oica.net/category/production-statistics/2012-statistics/. [Accessed 16 February 2016].

[6] C. Kraus, “Oil Prices: What’s Behind the Drop? Simple Economics,” The New York Times, 29 April 2016. [Online]. Available: http://www.nytimes.com/interactive/2016/business/energy-environment/oil-prices.html?_r=0. [Accessed 3 May 2016].

[7] J. Kitae, C. Koohong, R. David R and C. Ching-Yao, “Safety Performance of High-Occupancy Vehicle (HOV) Facilities: Evaluation of HOV Lane Configurations in California,” 1 January 2009. [Online]. Available: http://eprints.cdlib.org/uc/item/1cm7z3rd. [Accessed 2 February 2016].

[8] “The Travelling Salesman Problem,” [Online]. Available: http://www.math.uwaterloo.ca/tsp/. [Accessed 3 May 2016].

[9] E. Klarreich, “Computer Scientists Take Road Less Traveled,” Quanta Magazine, 29 January 2013. [Online]. Available: https://www.quantamagazine.org/20130129. [Accessed 3 May 2016].

[10] “Motor vehicles (per 1,000 people),” World Bank Group, [Online]. Available: https://web.archive.org/web/20140209114811/http://data.worldbank.org/indicator/IS.VEH.NVEH.P3. [Accessed 3 May 2016].

[11] “Changes in Vehicles per Capita around the World,” Office of Energy Efficiency & Renewable Energy, 5 April 2010. [Online]. Available: http://energy.gov/eere/vehicles/fact-617-april-5-2010-changes-vehicles-capita-around-world. [Accessed 3 May 2016].

[12] L. Chen, “At $68 Billion Valuation, Uber Will Be Bigger Than GM, Ford, And Honda,” Forbes, 4 December 2015. [Online]. Available: http://www.forbes.com/sites/liyanchen/2015/12/04/at-68-billion-valuation-uber-will-be-bigger-than-gm-ford-and-honda/#e24b8955858a. [Accessed 3 May 2016].

[13] J. Osawa, K. Wu and R. Carew, “China’s Homegrown Rival to Uber Valued at Over $25 Billion,” The Wall Street Journal, 7th April 2016. [Online]. Available: http://www.wsj.com/articles/chinas-homegrown-rival-to-uber-valued-at-over-25-billion-1460017727. [Accessed 3rd May 2016].

[14] E. Newcomer, “GM Invests $500 Million in Lyft,” Bloomberg, 4th January 2016. [Online]. Available: http://www.bloomberg.com/news/articles/2016-01-04/gm-invests-500-million-in-lyft-to-bolster-alliance-against-uber. [Accessed 3rd May 2016].

[15] S. Rai, “Ola, Uber's India Rival, Raises $500M In Funding, Now Valued At $5 Billion,” Forbes, 18th November 2015. [Online]. Available: http://www.forbes.com/sites/saritharai/2015/11/18/ola-ubers-india-rival-raises-500m-in-funding-now-valued-at-over-5-billion/#133f21c47f4b. [Accessed 3rd May 2016].

[16] L. Chen, “Meet Europe's Newest Unicorn: BlaBlaCar Raises $200 Million At $1.6 Billion Valuation,” Forbes, 16th September 2015. [Online]. Available: http://www.forbes.com/sites/liyanchen/2015/09/16/meet-europes-newest-unicorn-blablacar-raises-200-million-at-1-4-billion-valuation/#71b376ff158a. [Accessed 3rd May 2016].

[17] R. Bhagat, “GrabTaxi raising US$200M funding at US$1.5B valuation,” e27, 1st July 2015. [Online]. Available: https://e27.co/grabtaxi-raising-us200m-funding-us1-5b-valuation-20150701/. [Accessed 3rd May 2016].

[18] “Android (operating system),” Wikipedia, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Android_(operating_system)&oldid=718694161. [Accessed 5th May 2016].

[19] “Kantar Worldpanel ComTech’s Smartphone OS market share data Q3 2012,” Kantar Worldpanel Comtech, [Online]. Available: http://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=85. [Accessed 5th May 2016].

[20] “Kantar Worldpanel ComTech’s Smartphone OS market share data Q4 2012,” Kantar Worldpanel Comtech, [Online]. Available: http://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=113. [Accessed 5th May 2016].

[21] “Kantar Worldpanel ComTech’s Smartphone OS market share data Q1 2013,” Kantar Worldpanel Comtech, [Online]. Available: http://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=151. [Accessed 5th May 2016].

[22] “Kantar Worldpanel ComTech’s Smartphone OS market share data Q2 2013,” Kantar Worldpanel Comtech, [Online]. Available: http://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=263. [Accessed 5th May 2016].

[23] “Kantar Worldpanel ComTech’s Smartphone OS market share data Q3 2013,” Kantar Worldpanel Comtech, [Online]. Available: http://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=326. [Accessed 5th May 2016].

[24] “Kantar Worldpanel ComTech’s Smartphone OS market share data Q4 2013,” Kantar Worldpanel Comtech, [Online]. Available: http://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=399. [Accessed 5th May 2016].

[25] “Apple regains momentum as Windows stutters,” Kantar Worldpanel Comtech, 28th April 2014. [Online]. Available: http://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=482. [Accessed 5th May 2016].

[26] “Xiaomi, Huawei & Wiko power Android growth,” Kantar Worldpanel Comtech, 30 July 2014. [Online]. Available: http://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=584. [Accessed 5th May 2016].

[27] “Demand for iPhone 6 boosts Apple's sales,” Kantar Worldpanel ComTech, 29 October 2014. [Online]. Available: http://www.kantarworldpanel.com/global/News/Demand-for-iPhone-6-boosts-Apples-sales. [Accessed 5th May 2016].

[28] “Apple iOS leads US OS share for the first time since Q4 2012,” Kantar Worldpanel ComTech, 4th February 2015. [Online]. Available: http://www.kantarworldpanel.com/global/News/Apple-iOS-leads-US-OS-share-for-the-first-time-since-Q4-2012. [Accessed 5th May 2016].

[29] “Smartphone OS sales market share,” Kantar Worldpanel ComTech, [Online]. Available: http://www.kantarworldpanel.com/global/smartphone-os-market-share/. [Accessed 5th May 2016].

[30] J. Hildenbrand, “What is Android?,” Android Central, 16th May 2015. [Online]. Available: http://www.androidcentral.com/what-android. [Accessed 5th May 2016].

[31] “Android Studio Overview,” Android Developers, [Online]. Available: http://developer.android.com/tools/studio/index.html. [Accessed 5th May 2016].

[32] “Parse,” CrunchBase, [Online]. Available: https://www.crunchbase.com/organization/parse#/entity. [Accessed 30 April 2016].

[33] “Frequently Asked Questions,” Parse, [Online]. Available: https://www.parse.com/faq. [Accessed 30th April 2016].

[34] I. Sukhar, “Parse Pricing: Now Cheaper and Simpler!,” Parse, 30th April 2014. [Online]. Available: http://blog.parse.com/announcements/parse-pricing-now-cheaper-and-simpler/. [Accessed 30th April 2016].

[35] “Features,” Anypresence, [Online]. Available: http://www.anypresence.com/features/. [Accessed 30th April 2016].

[36] “What Is AWS Mobile Hub?,” Amazon Web Services, [Online]. Available: http://docs.aws.amazon.com/mobile-hub/latest/developerguide/overview.html. [Accessed 30th April 2016].

[37] “AWS Mobile Hub Pricing,” Amazon Web Services, [Online]. Available: https://aws.amazon.com/mobile/pricing/. [Accessed 30th April 2016].

[38] “Backendless platform,” Backendless, [Online]. Available: https://backendless.com/products/. [Accessed 30th April 2016].

[39] “Pricing,” Backendless, [Online]. Available: https://backendless.com/pricing/. [Accessed 30th April 2016].

[40] “Telerik Platform,” Telerik, [Online]. Available: http://www.telerik.com/platform#overview. [Accessed 30th April 2016].

[41] “Pricing,” Telerik, [Online]. Available: https://www.telerik.com/purchase/platform. [Accessed 30th April 2016].

[42] “Class Parse.File,” Parse JavaScript SDK, [Online]. Available: https://parse.com/docs/js/api/classes/Parse.File.html. [Accessed 30th April 2016].

[43] “Parse.GeoPoint,” Parse JavaScript SDK, [Online]. Available: https://parse.com/docs/js/api/classes/Parse.GeoPoint.html. [Accessed 30th April 2016].

[44] “Parse.ACL,” Parse JavaScript SDK, [Online]. Available: https://parse.com/docs/js/api/classes/Parse.ACL.html. [Accessed 30th April 2016].

[45] “Cloud Code Guide,” Parse, [Online]. Available: https://parse.com/docs/cloudcode/guide#cloud-code-advanced-background-jobs. [Accessed 30th April 2016].

[46] “Android Push Notifications,” Parse, [Online]. Available: https://parse.com/tutorials/android-push-notifications. [Accessed 30th April 2016].

[47] “Android Guide,” Parse, [Online]. Available: https://parse.com/docs/android/guide#getting-started. [Accessed 30th April 2016].

[48] “FAQ,” Google Developers, [Online]. Available: https://developers.google.com/maps/faq#getting-started. [Accessed 30th April 2016].

[49] “Plans,” Here, [Online]. Available: https://developer.here.com/plans/mobile-sdk. [Accessed 30th April 2016].

[50] “Pricing and Plans,” Google Developers, [Online]. Available: https://developers.google.com/maps/pricing-and-plans/. [Accessed 30th April 2016].

[51] “Introduction to the Google Maps Android API,” Google Developers, [Online]. Available: https://developers.google.com/maps/documentation/android-api/intro. [Accessed 30th April 2016].

[52] “ The Google Maps Directions API,” Google Developers, [Online]. Available: https://developers.google.com/maps/documentation/directions/intro. [Accessed 30th April 2016].

[53] “What is JSON?,” Squarespace Developers, [Online]. Available: http://developers.squarespace.com/what-is-json/. [Accessed 10th May 2016].

[54] D. Rowinski, “Crashlytics Knows Why Your iOS Apps Crash,” ReadWrite, 8th November 2011. [Online]. Available: http://readwrite.com/2011/11/08/crashalytics. [Accessed 9th May 2016].

[55] A. M. Lund, “Measuring usability with the USE questionnaire,” *STC Usability SIG Newsletter,* vol. 8, no. 2, October 2001.

[56] “The Google Maps Geocoding API,” Google Developers, [Online]. Available: https://developers.google.com/maps/documentation/geocoding/intro. [Accessed 9th May 2016].

Appendices

A. Database model

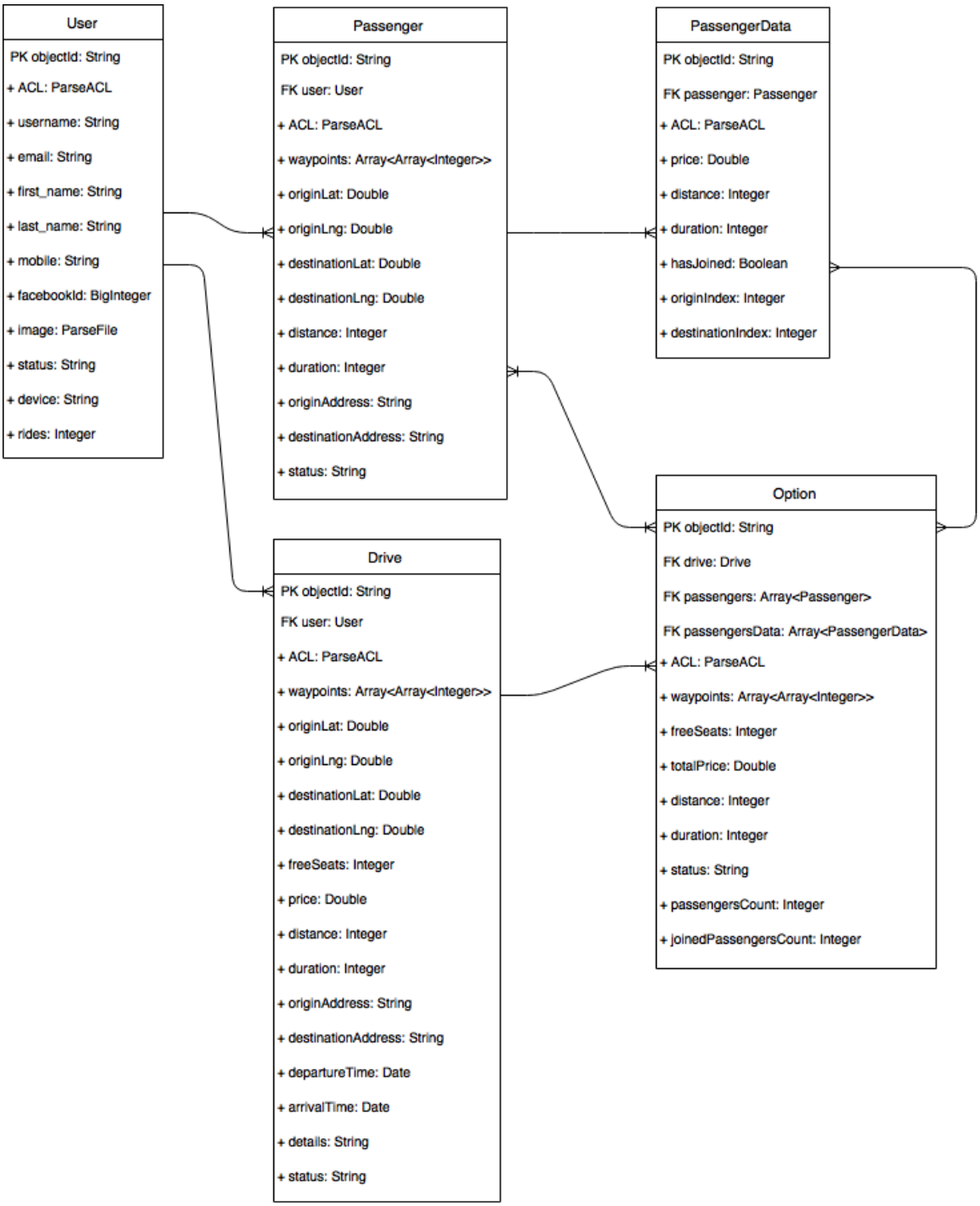

Figure 7. Entity-relationship model of database

B. Questionnaire

The questionnaire is located on the following link http://goo.gl/forms/cwO4XVkIQG.

General questions:

- Age
- Gender
  - Female
  - Male
- Technological background / level of technological knowledge
  - Higher education in a IT-related field
  - Frequently use (every day at least one per hour) devices such computers, tablets or smartphones
  - Occasionally use (less than once per hour every day) devices such computers, tablets or smartphones
  - I don't use any computers, tablets or smartphones

Usability specific questions (answers scale from strongly disagree to strongly agree):

1) Usefulness:
   a. It helps me be more effective
   b. It helps me be more productive
   c. It is useful
   d. It gives me more control over the activities in my life
   e. It makes the things I want to accomplish easier to get done
   f. It saves me time when I use it
   g. It meets my needs
   h. It does everything I would expect it to do
2) Ease of use:
   a. It is intuitive to use
   b. It is user friendly
   c. It requires the fewest steps possible to accomplish what I want to do with it
   d. It is flexible
   e. Using it is effortless
   f. I can use it without written instructions
   g. I don't notice any inconsistencies as I use it
   h. Both occasional and regular users would like it
   i. I can recover from mistakes quickly and easily
   j. I can use it successfully every time
3) Ease of learning:
   a. I learned to use it quickly
   b. I easily remember how to use it
   c. It is easy to learn to use it
   d. I quickly became skilful with it
4) Satisfaction:
   a. I am satisfied with it
   b. I would recommend it to a friend
   c. It is fun to use
   d. It works the way I want it to work

e. It is wonderful
f. I feel I need to have it
g. It is pleasant to use

C. Questionnaire answers

The results of the questionnaire are added to the zipped file which is attached to the thesis.

D. Application

The application APK file is added to the zipped file which is attached to the thesis.

E. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Gerrerth Kaur** (date of birth: 22 September 1994),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   "**A Prototype for a Real-time Mobile-based Ridesharing System"**,
   supervised by **Satish Srirama**,
2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **12/05/2016**