LAPIN AMK
Lapland University of Applied Sciences

Android Client-server application "Travelling Together"

Android application development and its monetization

Valerii Kan
Ivan Shabunin

Thesis
Lapland University of Applied Sciences
Degree Programme in Information Technology

2016

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Technology, Communication and Transport
Degree Programme in Information Technology
Bachelor Degree

Authors	Valerii Kan, Ivan Shabunin	Year	2016
Supervisor	Aku Kesti		
Title of Thesis	Android Client-Server Application "Travelling Together"		
No. of pages + app.	51 + 41		

Nowadays people are travelling a lot compared to the past. The main reason is a wide variety of the means of transport. The most common type is travelling by car. However, today not everyone has a car. Moreover, often people want to go to other city and they do not want to go alone. Therefore, the search of fellow travelers is widely used today. The goal of the thesis was to design and implement an Android application that would help drivers and passengers to cooperate in case they have the same destination and organize a joint ride. This system gives a chance for everyone to visit other cities and find friends.

In the theoretical part of the thesis, the research of Android environment was presented. The concept of application development was explained as well. Next, the design and the implementation of a "Travelling Together" application was presented. The step by step process was discussed. The main part of the application is the interaction between a server and a client. Furthermore, the work process with Google Maps, Google APIs and the parsers was presented. The final section included the research on the business opportunities of the Android applications in the IT world. The process of gaining profits from the application was analyzed. A descriptive research method was used for analyzing the information from the web sources and books.

The main result of the thesis is that the "Travelling Together" application for searching the fellow travellers was developed. The application project and the APK file can be downloaded from the website http://travellingtogether.ru. This application can be used by everyone all over the world. The principles of the monetization were analysed and will be applied to the application as well. Further study is needed to add extra features to the application. Moreover, the implementation of the iOS version is planned in the near future.

Key words application, activity, fragment, JSON

CONTENTS

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

SDK	Software Development Kit
APP	Application
JSON	JavaScript Object Notation
Inc.	Incorporated
JDK	Java Development Kit
JRE	Java Runtime Environment
OS	Operational System
IDE	Integrated Development Environment
SQL	Structured Query Language
UI	User Interface
API	Application Programming Interface
PHP	Hypertext Preprocessor

1 INTRODUCTION

Nowadays travelling has become an important part of modern life. Millions of people all over the world travel about their own countries and to foreign countries as well. At the same time, modern technologies are developing rapidly.

Knowledge of the Java programming language and the main concepts of Object-oriented programming encouraged to make a useful Android application. Furthermore, familiarity with the databases and the SQL queries prompt to explore the implementation of connection between server and client. Moreover, the basic understanding of business marketing awakes interest in gaining profits from the Android applications. The limits of the thesis are to acquire the basic understanding of the Android concepts and explore their usage on a real application.

The topic of the thesis is an exploration of the Android environment and making a real Android application for searching of fellow travellers. The main reason for choosing this topic is to simplify the search of drivers and passengers with the use of Android application. The Android platform was chosen due to the fact that majority of European population use Android Smartphones. The popularity of the Android OS all over the world is developing rapidly.

The objectives of this thesis are to make a research of the Android environment and design and implement a server-client application which would help drivers and passengers to cooperate in case they have the same destination and organize a joint ride. In order to explore the Android environment, the real application development process is described and analyzed in that thesis. This app will give a chance for everyone to visit other cities and find friends. Furthermore, the purpose of this thesis is to explore how to gain profits from apps.

2 ANDROID ENVIRONMENT

2.1 Android History

In October 2003 the team of four friends: Andy Rubin, Rich Miner, Nick Sears and Chris White founded a Android Corporation. Firstly, this company wanted to develop an operating system for the digital cameras. However, the market of these devices was small. Therefore, the company made a Smartphone operating system which would be a competitor to a Symbian OS and Microsoft Windows Mobile. (HeatWare 2014.)

In July 2005, Android Inc. was bought by the Google company for $50 million. 4 leaders of Android stayed at the company after its acquisition. From the beginning they made a mobile device platform powered by Linux kernel. On November 5, 2007 Android presented that platform as their first product. After that the first Smartphone (HTC Dream) running the Android OS was produced on October 22, 2008. (TechNotification 2014.) Since 2008, the Android company presented the new versions of the Android OS adding the new features and fixing bugs in the previous releases (Martinez 2014). The Android operational system became one of the most functional and acknowledged.

2.2 Versions of Android

The first Android version, Android 1.0, was released on September 23, 2008 (ComputerHope 2016). After that almost each year the Android company presented the new versions. Every new version obtained new features. For instance, with the appearance of Android 4.1 the extended notifications were added. The playback controls were displayed in a music player and a "missed call" notification provided buttons for calling back or sending the caller an SMS message. (Rohini 2014.) New Android versions fix bugs of the previous ones.

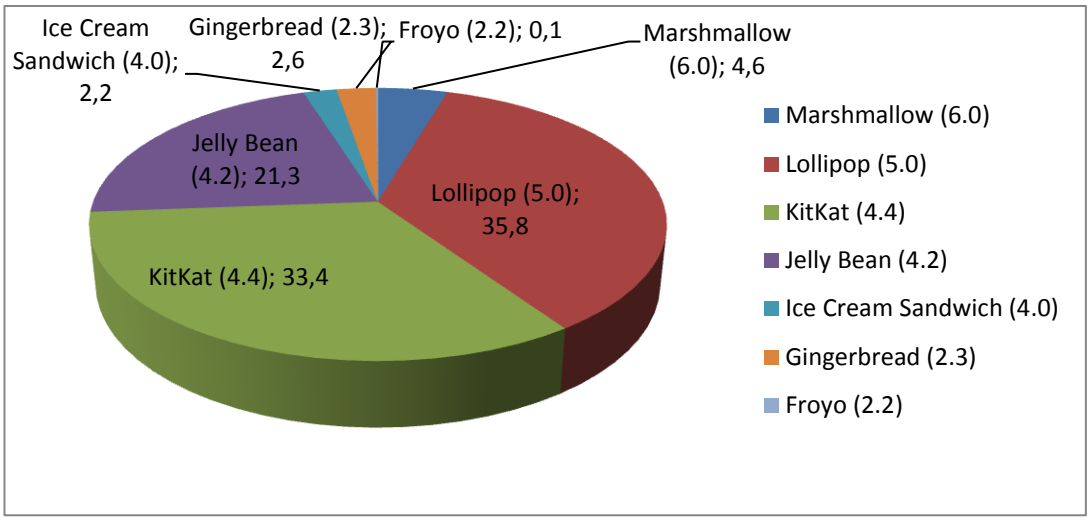

Figure 1. Usage of Android versions by April 2016 (Android Developers 2016a)

In Figure 1 the statistics of usage of Android versions by April 2016 is presented. The most used version is Android Kitkat 4.4, approximately 33,4 % of the world population are using it. The old versions (Froyo and Gingerbread) are in use nowadays, however, they do not support the main features and the functions of the modern devices. (Android Developers 2016a.)

2.3 Android Application Development

The Android apps connect the users around the globe in different innovative ways. The mobile apps are used every day and changed the way people communicate and consume entertainment. Therefore, the Android application development is a very important field because it broadens the horizons of personal qualities and awakes the interest in the exploration of the new creative apps that simplify the people life (Stadd 2015). Every phase of the app development is important.

The first phase is a setup of all necessary tools. The detailed information about the tools is described in the section 2.4. The next step is a development phase where the application is built with source codes. The third phase is when the application is run in the debug mode, tested and this shows if the application is ready for the usage. In that stage all application possibilities are checked and the possible errors are detected. The final step is a publishing the app. For

instance, Google Play market is a store of the various games and the applications for office, entertainment and tourism. Every person is able to publish an application there and get profits from that in case the application is widely used in society. (Quora 2016.)

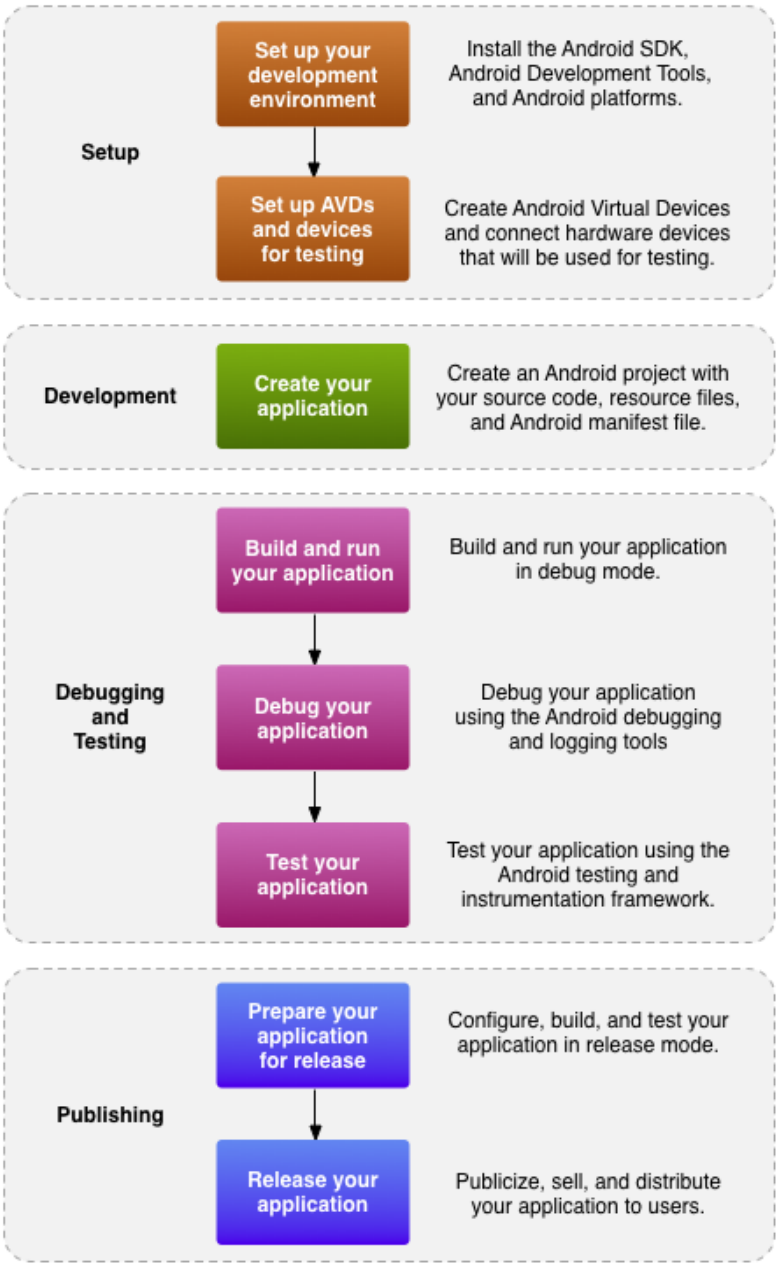

Figure 2. Android application development (Quora 2016)

Figure 2 presents an overall picture of the Android application development process. Following that scheme the modern functional applications and the games are created. (Quora 2016.)

2.4 Tools

Today the Android applications are written using the Android software development kit (SDK) and the Java programming language. For the Android Development Android Studio and Java Development Kit are needed to be installed. From 2014 Android Studio is the main open source development environment for implementing the Android apps (Android Developers 2016b). Until that Eclipse ADT was used by developers. However, the programme performance was slow and the computer did not respond for a long time. Android Studio improved these disadvantages and became the most used Android IDE in the world.

2.4.1 JDK

In order to start a work in the Android environment Java Development Kit is needed to be installed from the Oracle official site. This kit is for Java developers and includes a complete JRE and tools for developing, debugging and monitoring the Android applications. (Oracle 2016.)

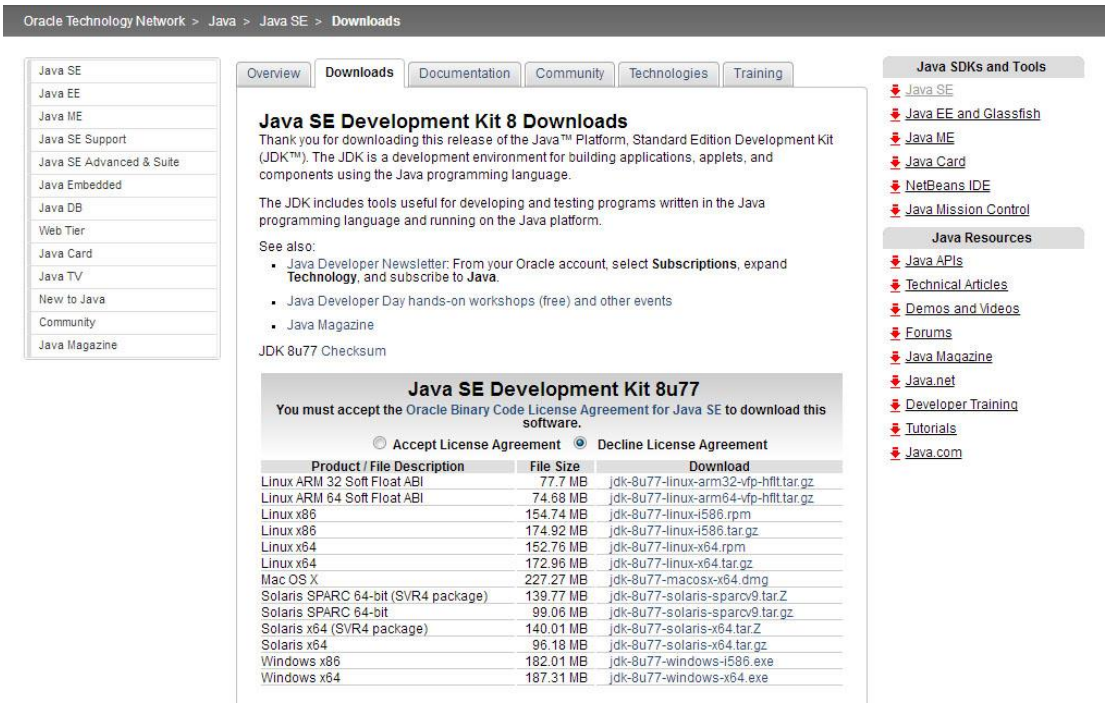

Figure 3. Download of a JDK (Oracle 2016)

On Figure 3 the download of a JDK is presented. The type of the computer operational system is chosen and JDK is installed. JDK is a building platform for installing the android environment system.

2.4.2 Android Studio

The next step is to download and install Android Studio from the official site. On Figure 4 the download of Android Studio is shown. It is the main programme for designing and implementing the applications.

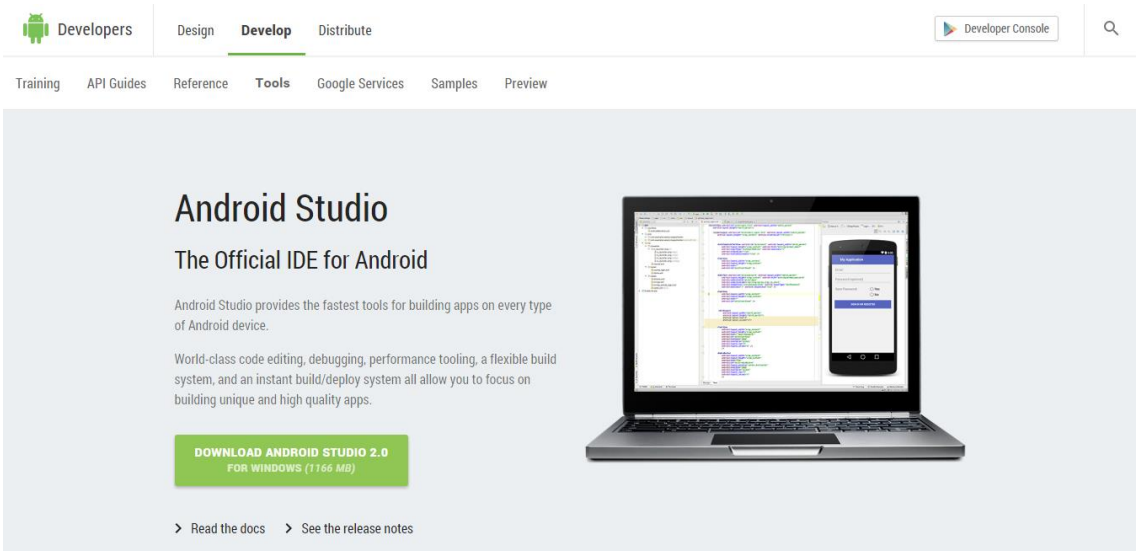

Figure 4. Download of Android Studio (Android Developers 2016c)

From December 2014 Android Studio is an official IDE for the Android platform development (Android Developers 2016c). Before that Eclipse was used as a common environment for designing and implementing the Android apps. However, Android Studio has some advantages regarding Eclipse. The comparison between Android Studio and Eclipse ADT is presented in Figure 5. (Wikipedia 2014a.)

Feature	Android Studio	Eclipse ADT
Build system	Gradle	Apache Ant
Maven-based build dependencies	Yes	No
Build variants and multiple-APK generation	Yes	No
Advanced Android code completion and refactoring	Yes	No
Graphical layout editor	Yes	Yes
APK signing and keystore management	Yes	Yes
NDK support	Yes	Yes

Figure 5. Android Studio vs. Eclipse ADT comparison(Wikipedia 2014)

The Android Studio build system is a Gradle which is used for building the application automatically. This system assemble the app to work properly. The main advantage of Android Studio is that the code completion and the refactoring is included in that IDE. This features remind the developer about all possible methods that can be used in a definite case and restructure existing computer code without changing its external behaviour. Furthermore, the lint tools are used for catching the performance, the usability and the version compatibility. (Wikipedia 2014.)

ProGuard tools are integrated to Android Studio for shrinking, optimising and obfuscating the Java code. This software detect and remove the unused instructions as well. Android Studio contains the template-based wizards for creating the common Android designs and the components. (Android Developers 2016d.)

At the same time Android Studio IDE has a layout editor that is used for drag-and-dropping the user interface components. The option of previewing the layout on the multiple screen configurations is included also. Android Studio supports the Android Wear applications. Moreover, the possibility of the integration with Google Cloud Messaging and App Engine is contained. (Wikipedia 2014.)

After the installation of Android Studio it is necessary to configure SDK Manager in the starting menu and download the tools that are needed for the development:

- Android SDK Tools
- Android SDK Platform-tools
- Android SDK Build-tools
- SDK Platform for all versions of Android starting from 4.0
- Android Support Library
- Local Maven repository for Support Libraries
- Google Play services
- Google Repository
- Google USB Driver (Android Developers 2016e.)

2.4.3 Genymotion Emulator

For developing and testing the applications without using a physical device the emulators are used. The mobile device emulator is a virtual mobile device that runs on a computer. The Android virtual device emulator is configured by default in Android Studio. The main objective of the emulator is to find the bugs and design the imperfections before the release of the app. However, the default Android emulator performance is not fast enough (StackOverFlow 2014).

Therefore, it is advised to use the fast Android emulator called Genymotion. It needs to be integrated to Android Studio directly. The Genymotion emulator is based on Oracle Virtualbox and provides a huge number of the different screen configurations of the Android devices. Furthermore, the feature of simulating phone calls or text is included in order to see how the app reacts. (Genymotion 2016.) The opinion of the IT engineer on Genymotion is:

"Genymotion is the best Android emulator ever. Everything is in one place, it works fast, and the support service is really good. It really simplified our work." (Iliushin 2015.)

Figure 6 represents the Genymotion Emulator main screen.

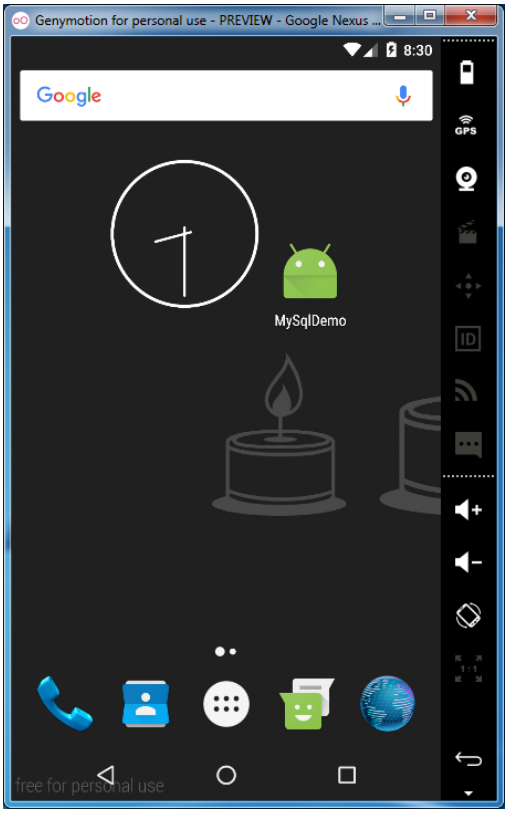

Figure 6. Genymotion Emulator (Genymotion 2016)

2.5 The structure of the Android project

Every Android project has its own structure that consists of three folders (manifests, java and res) and Gradle Scripts (Tools Android 2014). The structure is presented on Figure 7. In the Manifests folder the Android application must have a manifest file in a root directory. This file presents an essential information about the app to the Android system. Only after getting the information the system can execute the code of the app.

AndroidManifest file is a key file that works as a bridge between the Android developer and the Android platform. It contains the information about the

permissions to access the Android device capabilities. For instance, the internet access permission is determined there. Moreover, the manifest file describes all activities and services that are used in the application. It declares the minimum level of the Android application programming interface. (LearnCertification 2015.)

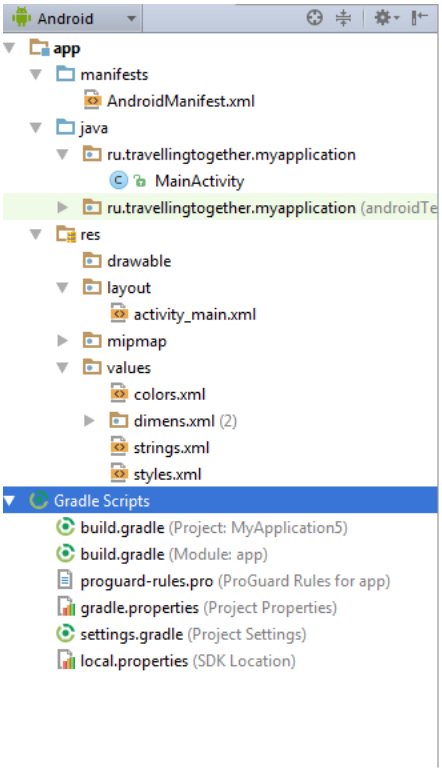

Figure 7. The structure of the Android project

Java directory contains all classes (activities, fragments) which are used in the application.

The res folder is used for storing the file resources of different types. This folder contains the subfolders to keep the resources based on its type. The subfolder "drawable" stores different bitmap drawables for medium, high and extra high density screens. The subfolder "layout" contains the layouts to be used in the application. A layout resource defines the architecture for the user interface in an Activity or a component of the UI. The subfolder "mipmap" stores icons. The subfolder "values" used to define strings, colors, dimensions, styles and static arrays of the strings and the integers. (Compile Time Error 2012.)

The Gradle Scripts stores files for building the application. Moreover, these scripts have the information about the compatibility of the Android app. These files check the application on the errors during debugging and running it. The assembling function builds the app to perform properly. (Vogella 2016e.)

2.6 Android General Concepts

The Android application development includes three general concepts that are used almost in all complicated apps. Every application consists of at least one activity, has a user interface layout and can contain the fragments. The fragment is a new feature that simplifies the performance of the application.

2.6.1 Activity

The Activity is a building block of the user interface written in the Java code. The simple applications consist of one activity. More complex applications contain multiple activities that interact with each other. The activity that starts first is the main. Other activities are launched from the main activity. (Android Developers 2016f.)

Generally the activity takes up the whole screen of the device. However, translucent and floating activities are widely used. In order to create an activity, two steps need to be done. The first one is to inherit the Activity class and the second step is to call the method *onCreate*(). As a result, the blank screen is created. There is no use of such screen. Therefore, various components and the fragments are added into the activity layout. (StartAndroid 2013.)

The Activity has a lifecycle. It can be in four different states depending how it is interacting with the user. When activity is in the foreground of the screen this state is called *active* or *running*. If the activity lost focus but is still visible, for instance, the transparent activity focused on the top of that activity, this state is called *paused.* If the activity is completely closed by another activity, it is *stopped*. The final state is *destroyed,* if the activity is dropped from memory by

the system. For managing the lifecycle of the activity, the main callback methods are implemented. (Vogella 2015b.) Figure 8 shows the main state paths of the activity. According to Android Developers source (2016f) the main methods are:

onCreate() method is called when the start activity is created.

onResume() method is launched when the activity starts to interact with the user.

onPause() method is called when the system is ready for resuming the previous activity.

onDestroy() method is called when the activity is destroyed. The activity is destroyed when it is finishing or due to the capacity of the system. (Android Developers 2016f.)

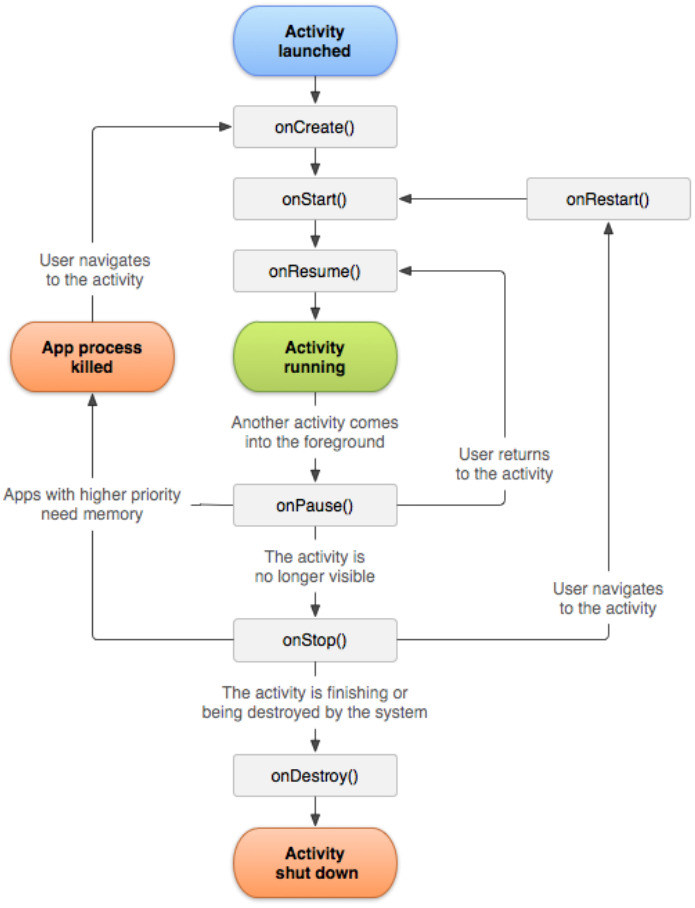

Figure 8. Activity Lifecycle (Android Developers 2016f)

2.6.2 Fragment

The Fragment is a portion of an user interface that can be implemented in different parts of the application. The Fragment has a connection with a activity. Therefore, the fragments are not able to exist separately. They function only with the activity. (Jones 2012.)

For the interaction between fragments special class *FragmentManager* is used. With the use of the method *findFragmentById(int id)* the manager finds a fragment by the identifier. For different states of the fragments the methods of the class *FragmentTransaction* are used.

> *"add() method adds fragment to activity.*
>
> *remove() method removes fragment from activity.*
>
> *replace() method changes one fragment to another one.*
>
> *hide() method makes fragment to be invisible.*
>
> *show() method shows invisible fragment on a screen."*
>
> (Android Developers 2016g)

The methods *remove*() and *replace*() are not applicable to the static fragments. Figure 9 shows the lifecycle of the fragments. It is analogous to the lifecycle of the activities. (Jones 2012.)

The main advantange of using the fragments in the Android application is that they simplifies the task of creating uthe ser interface for the multiple screen sizes. Furthermore, the fragments streamline the reuse of the components in different layouts. At the same time the fragments can be defined without an user interface and called as the headless fragments. (Vogella 2015d.)

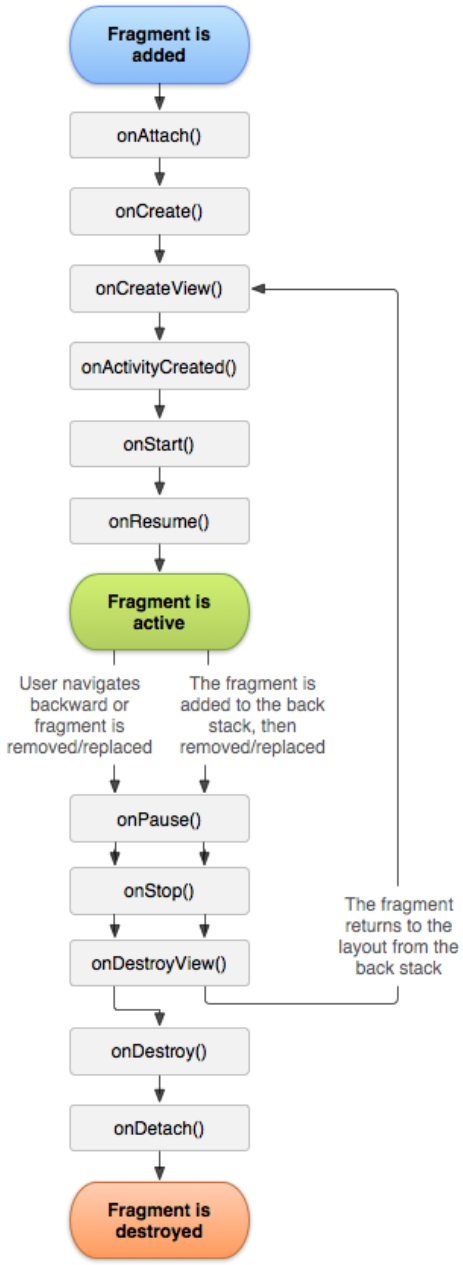

Figure 9. Lifecycle of fragments (Android Developers 2016g)

2.6.3 Android UI Layouts

A layout is a visual template for the user interface of the application which allows to manage the control of the elements, their properties and location. In Android Studio 6 standard types of the layout are presented. (Android Developers 2016k.)

2.6.3.1 Linear Layout

A Linear layout is a layout that arranges all children elements in one single direction: vertical or horizontal. The orientation is assigned by the attribute *android:orientation*. The alignment of all child elements is specified by the gravity. Figure 10 presents an example of using the nested linear layout. One LinearLayout container is used inside another. Both of the layouts have different orientations. (TutorialsPoint 2016f.)

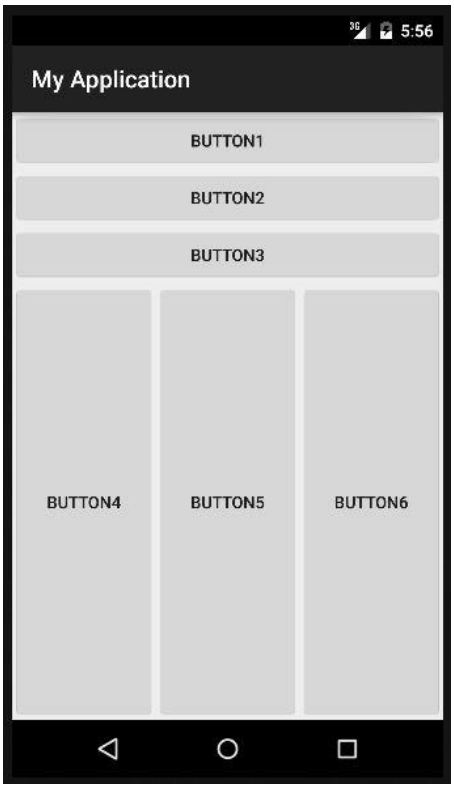

Figure 10. The example of the Linear Layout (EazyTutz 2016)

2.6.3.2 Frame Layout

A Frame layout is the simplest type of the layout. Generally, the frame layout is an empty space on the screen, that can be filled by child elements *View* and *ViewGroup.* All child elements of the frame layout are attached to the upper left corner of the screen. The different position for child elements cannot be defined in this type of the layout. The subsequent child view elements are simply drawn on the top of the previous components. (TutorialsPoint 2016e.) The Frame layout example is presented in Figure 11.

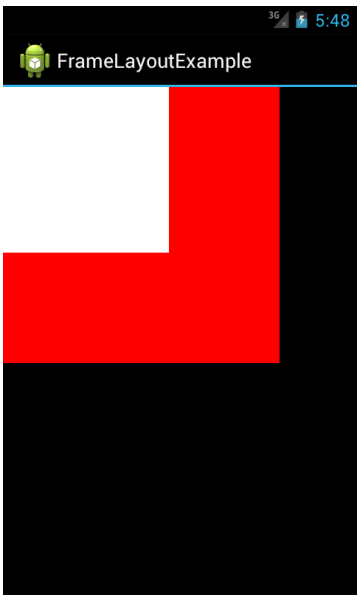

Figure 11. The example of the Frame Layout (KarimVarela 2012)

2.6.3.3 Table Layout

A Table layout positions its child elements into the rows and the columns, not displaying the border lines for the rows, the columns and the cells. This type of the layout can have the rows with a different number of cells. *TableRow* elements are used for the rows markup. Figure 12 presents the example of using the table layout. (TutorialPoint 2016h.)

Figure 12. The example of the Table Layout (NESTED IF 2014)

2.6.3.4 Relative Layout

A Relative layout allows the child elements to determine its positions relatively the parents elements or relatively to the adjacent child elements. In the relative layout the child elements are arranged in a definite way, that if the first element is located at the centre of the screen, other elements, which are aligned relative to the first element, will be aligned relative to the centre of the screen. Therefore, with this arrangement in the layout the element, that will be referenced for positioning other elements must be declared before the other elements refer to it. Figure 13 displays the example of the Relative layout. (TutorialsPoint 2016g.)

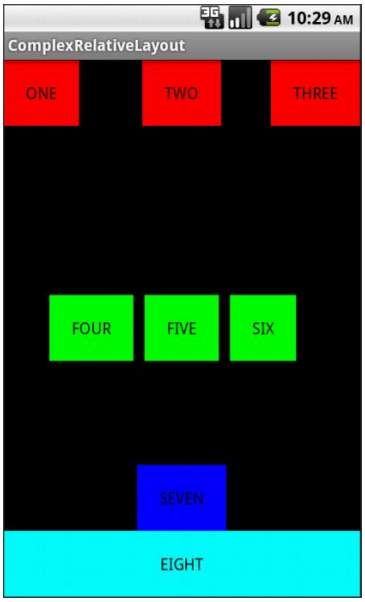

Figure 13. The example of the Relative Layout (AndroidBook 2013)

2.6.3.5 Grid Layout

A Grid layout is similar to the table layout. However, it is much more comfortable and functional. In the Grid layout the row and the column can be specified for any element, and in this table it will be located. To specify the element for each cell is not necessary. The elements can extend over several cells of the table. (Techotopia 2016b.) The Figure 14 shows the example of the simple Grid layout.

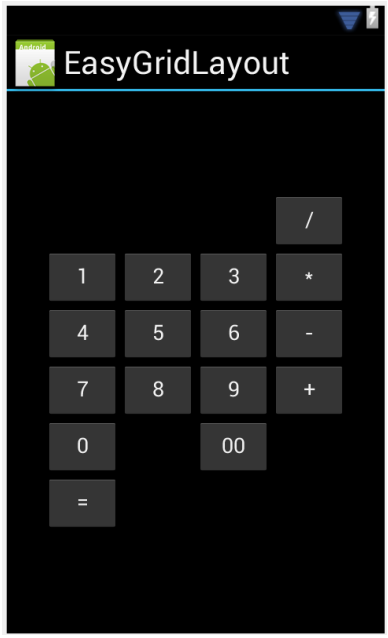

Figure 14. The example of the Grid Layout (Shane Conder 2013)

2.6.3.6 Absolute Layout

An Absolute layout is an outdated type of the layout. The principle of that layout is that the exact location (x/y coordinates) of its children is specified. The Absolute layouts are less flexible and harder to maintain than other types of layouts without the absolute positioning. In Figure 15 example of the absolute layout is presented. (Yadav 2012.)

Figure 15. The example of the Absolute Layout (TutorialsPoint 2012d)

2.6.4 Layout Parameters

Every type of the layout contains its own parameters. The main parameters are a width and a height of layout. For their definition the constants *match_parent* and *wrap_content* are used. The *Match_parent* means that the element will occupy all the available width/height in the parent element. The *Wrap_content* defines that width/height is located by its content. (Android Developers 2016h.)

One more parameter which is used in layouts is a weight. By default all views have the zero weight. This parameter indicates that the free space is distributed between the elements in the proportion to their weight values. (Beta recalll 2016.)

Moreover, a gravity is used as a parameter in creating the layout. The *Android:gravity* allows to change the location of the elements on the screen with the gravity in a relation to the parent element. A layout margin is a parameter that defines a margin from the element in a relation to another element. (CodeProject 2014.)

2.7 Android main methods and features

Android Studio provides a wide range of the useful methods and the features that are used in the development of the applications. This section describes the main ones for the basic understanding.

2.7.1 Event Handling Methods

Events in Android are generally generated in the response to the user interaction with the input controls. In order to handle the event the view needs to have an event listener. In the Android framework 6 event listeners are used. (TutorialsPoint 2016a.)

onClickListener is used for detecting the click style events by the user and releasing an area of the device display occupied by a view. *onLongClickListener* is operated for clicking or focusing on a view component for more than one second. onTouchListener performs a touch action including the motion gesture on the screen. (Techotopia 2016a.)

onCreateContextMenuListener is used for the creation of a context menu as a result of a long click. *onFocusChangeListener* detects the navigation onto or away from an item. *onKeyListener* check when a key on a device is pressed while the view has a focus. (Android Developers 2016z.)

The most common event listener is *onClickListener* that is used for controlling buttons. The button is the most used element is programming and refers to an event. For instance, for confirming actions and transitions between the activities or the fragments buttons are defined. (Techotopia 2016a.)

2.7.2 Android Debugging with Logs

Debugging is a significant aspect in the Android development. Android Studio SDK mechanism contains a set of tools for debugging. With the use of *logcat* command logs the buffer is displayed. That feature allows a developer to understand the principle of not working app. The Class *android.util.log* splits log messages into the categories depending on the importance. (Vogella 2015c.) For categorization special methods are used that are easy to remember by the first letter indicating the category:

" *Log.v - verbose*

Log.d - debug

Log.i - information

Log.w - warning

Log.e - error " (Developer.Android 2016l)

2.7.3 Android Toasts

In the Android development toasts are used for attracting the attention of an user. The toast is a pop-up notification that appears on the surface of the application window filling the necessary amount of the space required for the message. The main advantage of that feature is that the current activity of the application remains functional for the user. Within few seconds the message closes smoothly. Furthermore, a toast notification can be created by a service running in a background mode. (JavaTechig 2013.)

However, the main function of toasts is a displaying short text messages. The toast location is changed by the gravity method. Moreover, the picture can be added to a message in a toast notification for attracting the user attention in a more beautiful way. (Jenkov 2014.)

2.7.4 Android Context Menu

Android context menu is a floating menu that appears when the user performs a long-click on a element. It provides actions that affect the selected content or the context frame. This menu displays a vertical list of similar items that is bounded to the view that invoked the menu. For example, the context menu is used in a mailbox for presenting the messages. Moreover, this menu is for extended actions that relate to content areas in the operation. (CompileTimeError 2014.)

2.7.5 Android Intents

Intents are widely used in the Android applications. The intent is a mechanism for describing an operation - selecting a photo, sending an email, making a call, launching a browser or navigating to the specified address. (Android Developers 2016m.)

The most common scenario of usage intents is defined in a starting another activity in the application. The Android apps generally contain few activities. Therefore, the activity switch is operated using intents. (TutorialsPoint 2016b.) Furthermore, the intents are used for the announcements about the launch of the activity or the service that is aimed at the implementation of any actions. The intents are selected for sending notifications that the action happened. (Vogella 2014a.)

The Android broadcast intents to announce the system events as changes in the network connection status or the battery level. The system apps in Android register the components tracking the given intents and react accordingly to them. For instance, receiving a new SMS message is the intent. (Android Developers 2016m.)

2.7.6 Android Shared Preferences

Shared Preferences is a permanent storage on the Android platform that is used by the applications to store their settings and the data. This storage is relatively constant, that the user can open the app settings and clear the data of the app. thereby clearing all the data in the repository. The method *getSharedPreferences* is called for returning a shared preference instance pointing to the file that contains the values of preferences. For saving the data in Shared Preferences the editor class is used. That class contains methods for the manipulation of the data inside the shared preferences. (TutorialsPoint 2016c.)

3 DESIGN AND IMPLEMENTATION OF ANDROID APP "TRAVELLING TOGETHER"

3.1 Background

Nowadays search of fellow travellers is widely used. The idea was to create a server-client application which would help drivers and passengers to cooperate in case they have the same destination and organize a joint ride. The application was called "Travelling Together".

Studying the process of the Android application development led to the idea about implementing the real application. Web resources and books were used for choosing the best concepts for creating the app.

3.2 Design of the application

The first step of the application development process was to create a design of the application. The Navigation Drawer activity template was used to create a side menu of the application. It is a panel that displays the main navigation options on the left edge of the screen. Use cases in which the user is involved are displayed in Figure 16.

The Navigation panel is opened when the user swipes a finger from the left edge of the screen or touches the icon in the action bar. However, for most of the time it is hidden. The navigation panel consists of three items for the user choice. Figure 17 displays the navigation panel of the "Travelling Together application. Launching the panel the user selects *Driver menu, Passenger menu* or *App information.*

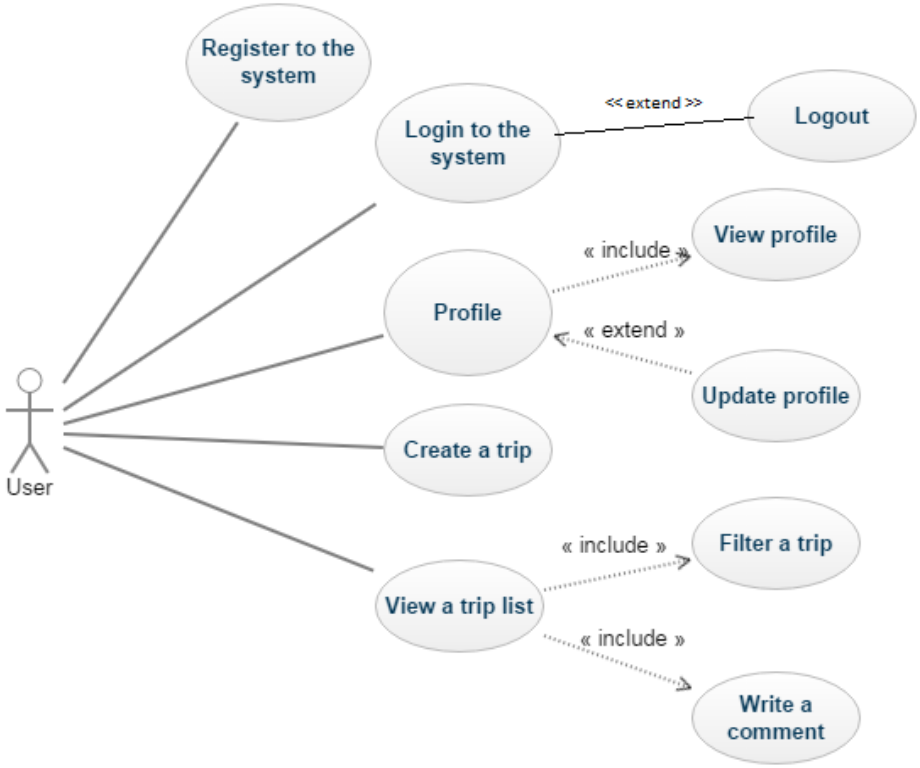

Figure 16. Use case diagram

The main screen of the application and the fragments' markup are developed with the use of the Relative layout. This type of the layout was chosen for the best readability of the application by the users. The elements of the screen are mainly located in the centre of the screen for the improvement of the legibility.

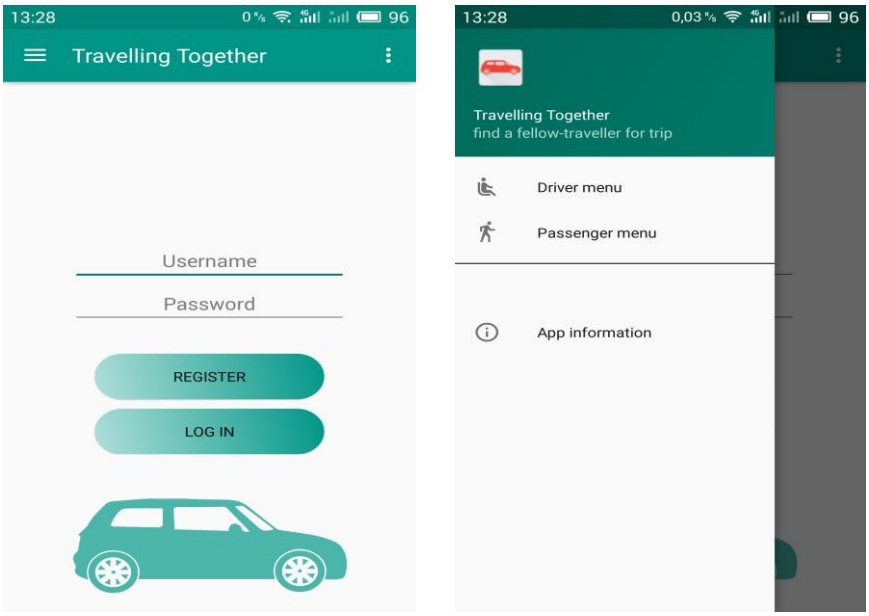

Figure 17. Design of the application

3.3 Structure of the "Travelling Together" application

The next step of the application development process was to develop the structure of the application. Figure 18 displays the detailed architecture of the application states. The overall structure consists of the start page and 7 fragments . All the fragments that are used in the application are directly connected in a logical sequence.

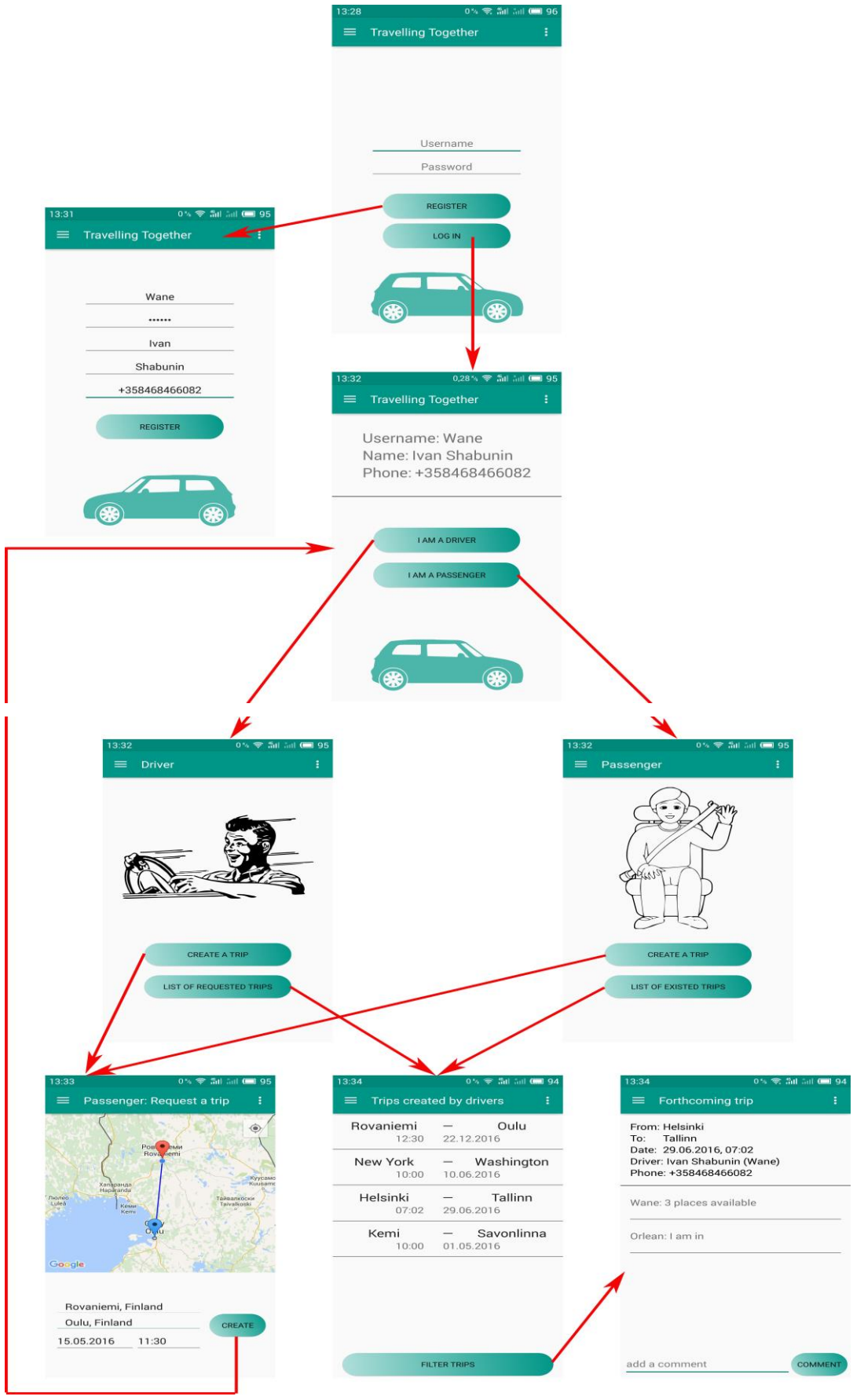

Figure 18. Structure of "Travelling Together" app

3.3.1 Start page

Launching the application the user is offered to login to the application. The main screen of the app contains username and password *EditText* views and two buttons *REGISTER* and *Log In.* In order to use the application the user must register pressing the button *REGISTER*. After the registration the user is able to use the username and the password to log in the next launching time.

The source code of the start page is presented in Appendix 1. The inserted data is transmitted to a server database by the *LoginBW* parser (Appendix 2). Afterwards the app receives a server answer. If the answer is positive, the *FragmentUser* is opened and the user data acquire the JSON format. Otherwise, the application displays an error.

3.3.2 Registration fragment

Every user should complete the fast registration stage that is presented on Figure 19, entering username, password, first name and family name and clicking the *REGISTER* button. After the registration the user is able to log in immediately inserting the username and password.

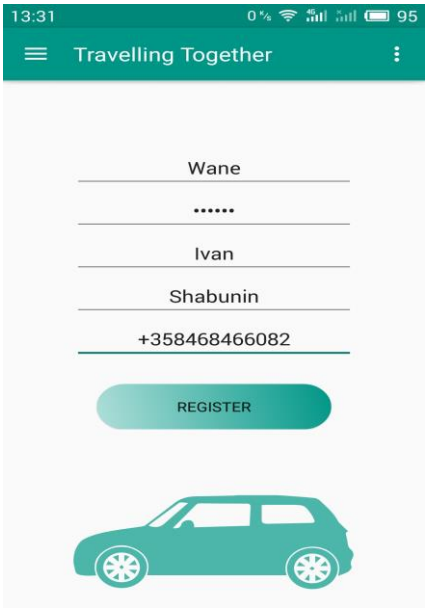

Figure 19. Registration stage

The detailed code is shown in Appendix 3. All the user data is retrieved from the *EditText* fields and is sent to the server by the *RegisterBW* parser. (Appendix 4) Thereafter, the server database responses to the application and shows the answer in a log. If it is *successful*, the FragmentUser executes. On the contrary, the app logs an error.

3.3.3 User fragment

The next stage after the main screen and the registration page is the users page that displays the name, the username and the phone number of the user. The user data is extracted to the *TextViews* from the JSON file that was created in the login page. Furthermore, the screen contains two buttons that the user selects depending on the status if the customer is a driver or a passenger. To understand the principle of the user fragment Appendix 5 is included.

Selecting *I AM a DRIVER* button transfers the user to *FragmentDriver*. Clicking *I AM a PASSENGER* opens *FragmentPassenger*. Analogously, the same fragments can be opened from the navigation panel. The user fragment is displayed in Figure 20.

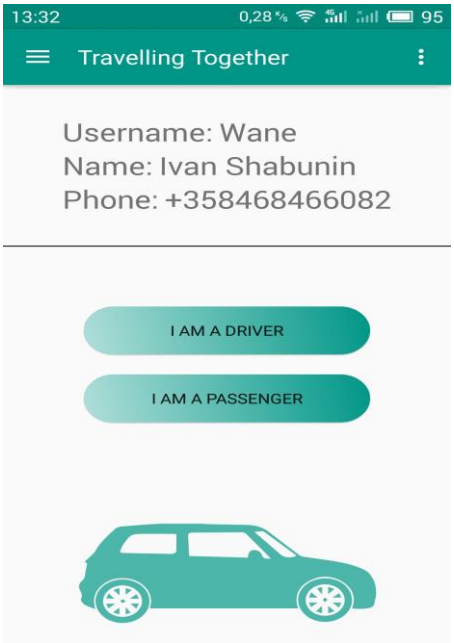

Figure 20. User fragment

3.3.4 Driver fragment

The driver fragment is created for the users who offer a trip to other users. The source code is presented in Appendix 6. The fragment consists of two buttons *CREATE a TRIP* and *LIST of REQUESTED TRIPS.* If the user wants to suggest a trip, the *CREATE a TRIP* button is selected. The driver fragment is contained in Figure 21.

Choosing the *LIST of REQUESTED TRIPS* executes the *TripPassengerJSONParser* parser and checks the status of the user (driver or passenger). The source code of the parser is shown in Appendix 7. Thereafter, if the user is a driver, the application displays a trip list that was created only by passengers. As a result, the *FragmentTripList* is opened.

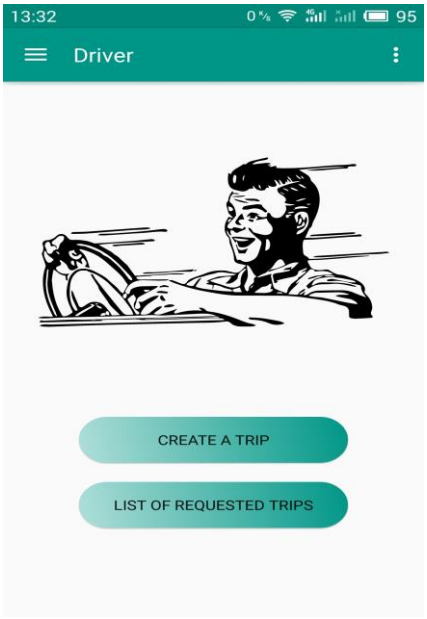

Figure 21. Driver fragment

3.3.5 Passenger fragment

The passenger fragment is executed for the users who search for the trip. The fragment consists of two buttons by the analogy with the driver fragment. Hence, it the user searches for a driver, the *CREATE a TRIP* button is chosen.

Selecting the *LIST of EXISTED TRIPS* launches the *TripDriverJSONParser* parser and inspects the status of the user. Afterwards, if the user is a passenger, the list of the trips created only by the drivers is displayed. The *FragmentTripList* is launched. The *FragmentPassenger* code is described in Appendix 8. The passenger fragment is depicted in Figure 22.

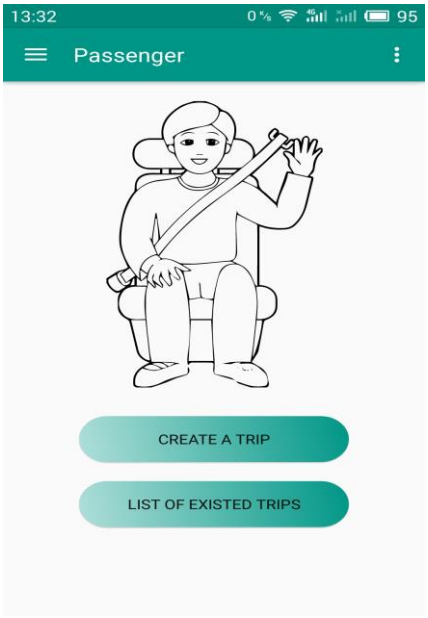

Figure 22. Passenger fragment

3.3.6 Map fragment

The user is able to request a trip in the *FragmentMap* that is displayed in Figure 23 inserting the route of the planned trip, the date and the time. The map view is launched by assigning a map value to a variable *googleMap* with the use of the *createMapView* method. The map displays the user the current location. The detailed structure of the map fragment is explained in Appendix 9.

AutoCompleteTextViews are used for making a request of the trip. The user fills the route and the information is used for receiving the latitude and the longitude of the current location and the planned destination by the *Geocoder*. The markers are drawn by the latitude and the longitude of the places. The line is created between two markers in case two cities are found in Google Maps. The *Geocoder* extracts the name of the cities that will be used in a trip list.

DatePickerDialog and *TimePickerDialog* retrieve the date(number of a day, month, year) and the time(hours and minutes) respectively.

Provided that all data is inserted, the information is sent to the server database by the *TripCreateBW* parser. If the trip is saved the FragmentUser launches. Otherwise, an error is logged.

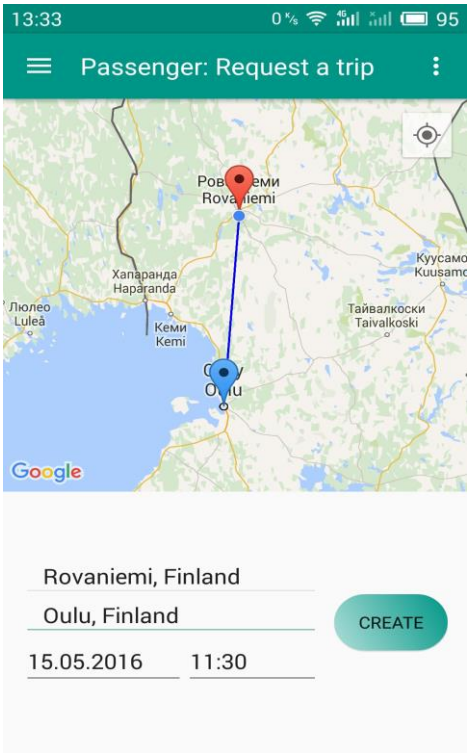

Figure 23. Map fragment

3.3.7 Trip list and trip details fragments

The *FragmentTripList* displays the list of the planned trips created by the users. However, the trip list made by the driver is visible exclusively for the passengers. On the contrary, the passenger list is displayed only for the drivers. The user data is sorted by the name, the username and the phone number with the use of the *TripCreatorJSONParser*. The parser extracts the user data to the *TextViews.* The source code of the fragment is presented in Appendix 10.

Furthermore, the fragment contains the discussion area where the user is able to add comments in order to connect with the driver or the passenger. The

comments are extracted to the *ListViews* by the *AddCommentBW* parser and displayed on the screen. Moreover, the user is able to filter the search of the trip by pressing the *Filter* button. The trip fragment and the trip details fragment are depicted in Figure 24 respectively.

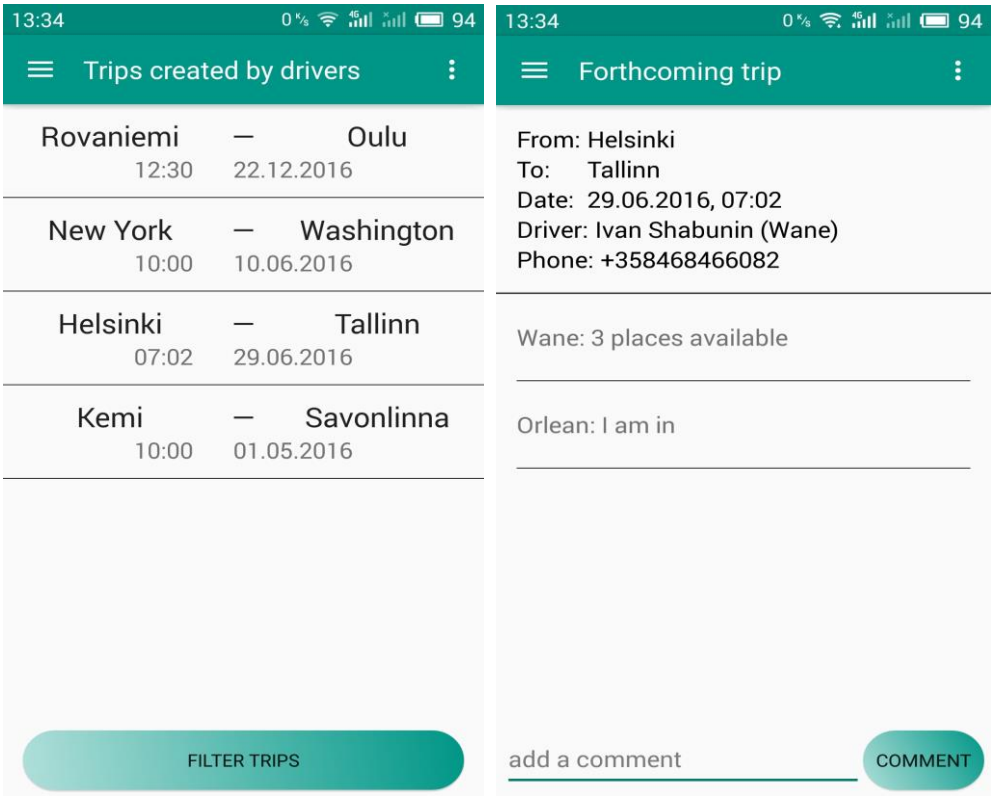

Figure 24. Trip fragment and trip details fragment

3.4 Shared Preferences

The "Travelling Together" application uses shared preferences to release the session data. The user data is saved to the shared preferences after the users registration. The information about the user can be updated by the method *FragmentUpdateInfo*.

The values of the shared preferences are zeroed after the user logout. If the user do not logout, the next session is launched with the saved shared preferences and the *FragmentUser* is started. On the contrary, the *FragmentLogin* is launched in case the values of the shared preferences are equalled to zeroes.

3.5 The principle of the application work

The most common type of the Android application is an app that is based on connection between a user and a server. The direct connection from the MySQL database to the application is not possible. Therefore, special server-side scripting language PHP was used.

The structure of client-server application is described in Figure 25. In the first step the user creates a request via HTTP to the server. After that the php script connects to the MySQL database server. In the next step all SQL data is taken by php script. The final step is an adding of the SQL data to the JSON array that is parsed by the application and displaying the required data to the user. (Bartolome 2015.)

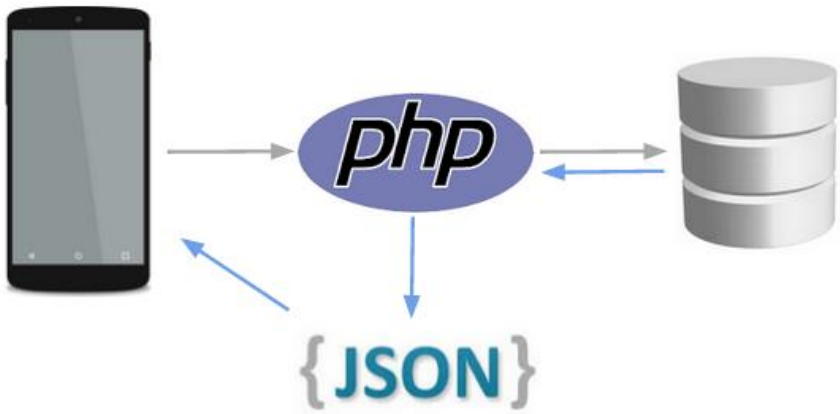

Figure 25. Structure of a client-server application (Bartolome 2015)

3.6 Data storage

All user information stores in the server MySQL database. The database consists of three tables: Users, Comments and Trips. The database diagram is presented in Figure 26. The tables in the database are connected by the username.

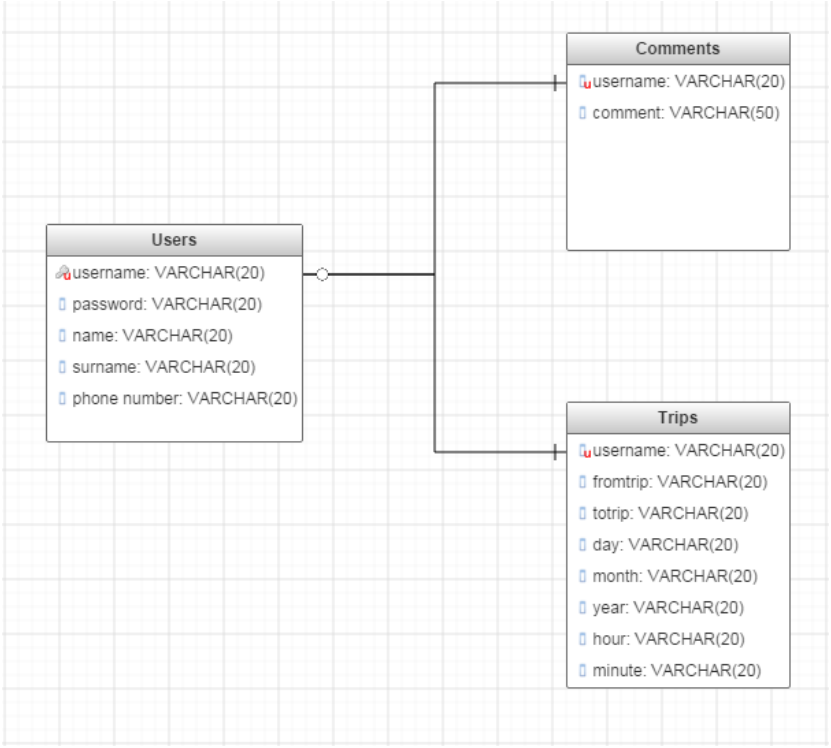

Figure 26. MySQL database diagram

The Wamp server programme was used as a server storage. The main feature of the Wamp is that it supports phpMyAdmin tool. This tool is intended to handle the administration of the MySQL database. The Wamp server was used for testing the application and fixing issues in it. Finally, the real web server replaced the Wamp.

3.7 Application testing

Testing is a significant part of the development process that checks the functionality of the product. The "Travelling Together" application was tested with use of the Genymotion emulator, Samsung Galaxy S3 and Meizu MX5. Different Android versions were tested in order to check the compatibility of the application. All posible states and transitions were tested. The main problem of the testing was the slow performance of the Internet connection that led to the errors in launching the fragments. The testing was considered successful: all features worked properly and promptly.

4 MONETIZATION OF ANDROID APP

4.1 Monetization process

Monetization is a process of acquiring profits from the application. Nowadays a big variety of the Android applications and games are stored in a Google Play Market. This store is the main market where the developer can introduce the application to the users. In order to gain profits the developer needs to think over the target audience and users interest in the application. The main rule of the monetization is not to annoy the users.

4.2 Google Play Store

Google Play Store is the official application store for the distribution of Android applications (Wikipedia 2016b). Furthermore, users are able to buy music, magazines, books and movies. Figure 27 indicates the number of available applications on Google Play Store. Current number of all Android apps (red line) by April 2016 is 2 million 100 thousands applications. The blue line shows the low quality apps that are not used by the users. The green line indicates the regular apps that are downloaded constantly.

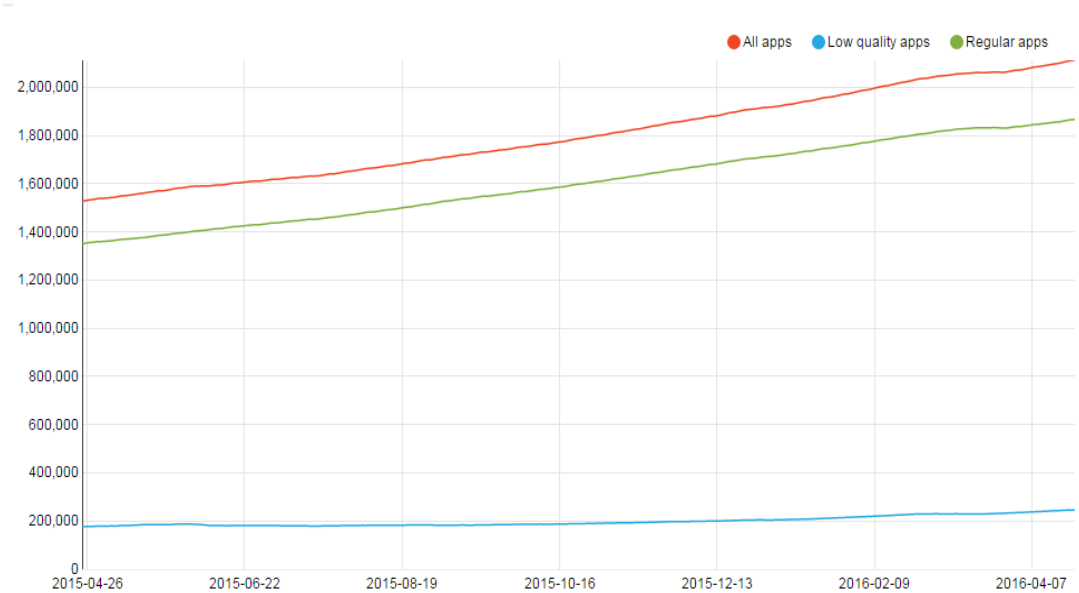

Figure 27. The total number of the Android apps in Google Play Store (AppBrain 2016a)

Google Play Store has a categorization system for the Android apps. The total number of the Android apps per category (purple column) is presented in Figure 28. The free apps are indicated as a red column and the paid apps as a green column. The most popular category is the application for education.

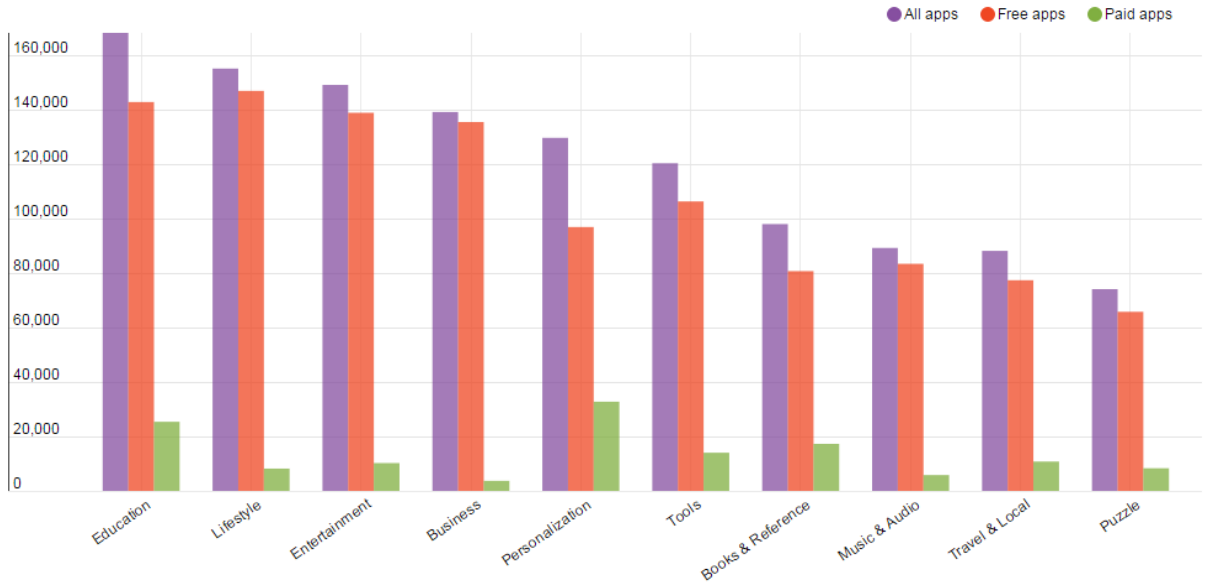

Figure 28. Top 10 Google Play Store categories (AppBrain 2016b)

4.3 Application publishment to Google Play Store

The first step to publish the application is to create an account in Google Play Developer Console paying 25$ as a registration fee. The main reason of that fee is to keep out duplicate accounts in the market. After the account creation the developer is able to publish the unlimited amount of Android applications.

The application is published as a APK file. The developer creates that file format in Android Studio by debugging the app. Screenshots of the user interface should be attached to introduce the app to the customers of Google Play Store. Furthermore, a description part needs to contain the detailed list of the application features. The contact information of the developer is filled as well. Finally, after the application approval by Google Play Store the app can be downloaded by users all over the world.

4.4 Competition in Google Play Store

To increase the popularity of the application the analysis of the competitors is advised to be conducted. Google Play Store generally contains applications with similar functionality. In order to compete with them the developer explores the features of that apps and analyzes the advantages and disadvantages. The development of new worthwhile features is the main principle of the rival with other developers.

Translation of the application to other languages can be a benefit of the application in relation to others. Furthermore, the YouTube video about the application promotes the app. In Google Play Store customers leave a feedback for every application in comparison to the similar apps. The users feedback assists the developer to fix bugs and improve features in the application.

4.5 Alternative app stores

To enhance the chances of the monetization for the Android application the developer is able to publish the app in alternative app stores all over the world. The main competitor to Google Play Store is Amazon App Store. That store suggests the developers to publish paid and free download versions of the Android applications. However, the main feature of Amazon App Store is that one paid application can be downloaded for free every day.

Opera mobile store is a popular alternative among the Android users. Approximately 60 million users visit that market every month (AppFlood 2016). Samsung Apps Mobile store is a global market that provides high-quality apps. The customers are persuaded that the apps are safe to download because the market check the applications for malware and device-compatibility.

4.6 Monetization methods

Today successful monetization is the main purpose for every developer. A lot of methods are used to gain profits. Analysing the market the common monetization method is adding the advertisement to the application. However, choice of the method depends on the necessity of the application in real life.

4.6.1 Monetization with advertisement

The idea of this method is to offer the application for free and displaying the advertisement in the app to generate the income. When a user taps on the ad, the money is earned. This method is the most common among the developers of the mobile applications. The internal advertising works due to the display of advertising at previously allocated space in the interface that gives a profit in calculation for a certain number of viewings and transitions. This method effectively works in games, news applications, messengers and entertaining apps. However, the main disadvantage is that the application should be used often and constantly for gaining money.

4.6.2 Monetization with in-app purchases

Monetization with in-app purchases is widely used in games. In-app purchases allow the user to buy additional features in the application. For instance, extra levels or virtual currency (coins can be used in the game for upgrades). In the applications monetization bonuses are considered as in-app purchases.

4.6.3 Paid downloading

This principle works due to the payment by the user of all application cost beforehand. That approach perfectly works in games, entertaining apps. However, the more users need to pay from the beginning, the less they accept

advertising or need to buy in addition certain functions. The best option is to offer free and paid application. However, in paid app additional content will be presented and advertisement will be minimized or excluded.

4.7 Advertising platforms

Nowadays a lot of advertising companies offer a developer to earn money in the application displaying the advertisement. The most common monetization platforms are AdMob, InMobi, Tapjoy and Chartboost. These platforms are integrated into more than 300,000 games and applications by 2016 (Wikipedia 2016). The monetization platforms provide various ad formats as banners, video ads. The filtering system is included in the platform to define in which cases the advertisement needs to be displayed. The developer's income is generated by the amount of the advertisement viewings.

4.8 The possible monetization of "Travelling Together" application

The first step for the successful monetization of the "Travelling Together" application is to analyze the necessity of that application in a world. By the fact, that travelling is popular nowadays, application for searching fellow travellers is worthwhile.

The next step is to find competitors of this application in app stores. Google Play Store offers about 15 applications on "fellow-travellers" category. "Direct ridesharing" app is the main competitor to the "Travelling Together" application. "Direct ridesharing" searches fellow travellers that want to drive from one place to other place of the same city. However, the "Travelling Together" application allows the users to travel all over the world.

Finally, to gain profits the application is planned to be published in Google Play Store as a free application with advertisement. In order not to annoy the users the application is intended to contain a limited amount of ads.

5 DISCUSSION

The thesis shows the way of studying the Android application development from the beginning. As a result, the real Android app "Travelling Together" for the search of fellow travellers was implemented. The goals of the thesis were achieved.

During the thesis the Android programming language was studied. The project learnt to implement various types of the Android applications. Familiarization with the databases, parsers extended the knowledge of the client-server connection. Google API's assisted to create different routes and developed the working with the maps. Furthermore, the process of the monetization was researched and the analysis of the Google Play Store was made.

During the thesis various difficulties occurred. The main problem was the low performance of the portable computers. The code errors occurred constantly especially errors with the deprecated methods. The problems happened in different devices. For instance, Geocoder did not work on Meizu MX5 and maps errors occurred on Samsung Galaxy S3.

The future plans are to publish the first version of the "Travelling Together" application in the Google Play Store. The data security level is planned to be increased as well. Furthermore, the design of the application is intended to be improved and additional features are expected to be imported to the app. For instance, the personal messenger is planned to be added. Moreover, the iOS version of the "Travelling Together" app is intended.

Taking everything into account, the thesis project was cognitive and brought an unforgettable experience that will be a benefit in the future.

BIBLIOGRAPHY

Android Developers 2016a. Dashboards. Accessed 15 February 2016 http://developer.android.com/about/dashboards/index.html.

Android Developers 2016b. SDK Manager. Accessed 16 February 2016 http://developer.android.com/tools/help/sdk-manager.html.

Android Developers 2016c. Android Studio. Accessed 15 March 2016 http://developer.android.com/sdk/index.html.

Android Developers 2016d. Build System Overview. Accessed 18 March 2016 http://developer.android.com/sdk/installing/studio-build.html.

Android Developers 2016e. Application Fundamentals. Accessed 22 February 2016 http://developer.android.com/guide/components/fundamentals.html.

Android Developers 2016f. Activity. Accessed 26 February 2016 http://developer.android.com/intl/ru/reference/android/app/Activity.html.

Android Developers 2016g. Fragments. Accessed 22 March 2016 http://developer.android.com/guide/components/fragments.html.

Android Developers 2016h. GridLayout.LayoutParams. Accessed 8 March 2016 http://developer.android.com/reference/android/widget/GridLayout.LayoutParams.html.

Android Developers 2016k. Layouts. Accessed 25 March 2016 http://developer.android.com/guide/topics/ui/declaring-layout.html.

Android Developers 2016l. Reading and Writing Logs. Accessed 22 March 2016 http://developer.android.com/tools/debugging/debugging-log.html.

Android Developers 2016m. Intents and Intent Filters. Accessed 1 March 2016 http://developer.android.com/guide/components/intents-filters.html.

Android Developers 2016z. Input events. Accessed 29 February 2016 http://developer.android.com/guide/topics/ui/ui-events.html.

AndroidBook 2013. Pleasures of Relative Layout - A boon for UI designers. Accessed 23 February 2016 http://www.androidbook.com/akc/display?url=DisplayNoteIMPURL&reportId=4170&downerUserId=deepak.

AppBrain 2016a. Number of Android application. Accessed 15 March 2016 http://www.appbrain.com/stats/number-of-android-apps.

AppBrain 2016b. Most popular Google Play categories. Accessed 16 March 2016. http://www.appbrain.com/stats/number-of-android-apps

AppFlood 2016. 10 Google Play alternatives to boost Android app installs. http://appflood.com/blog/ten-alternative-android-app-stores.

Bartolome, R. 2015. A simple web service using PHP, JSON and MySQL. Accessed 15 March 2016 http://robertobartolome.com/a-simple-web-service-using-php-json-and-mysql.

Beta recalll 2016. Layout Params. Accessed 8 March 2016 https://recalll.co/app/?q=android%20-%20Programmatically%20set%20weight%20of%20children%20of%20LinearLayout.

CodeProject 2014. Let's create the Scren - Android UI Layout and Controls. Accessed 10 March 2016 http://www.codeproject.com/Articles/803688/Lets-create-the-Screen-Android-UI-Layout-and-Contr.

CompileTimeError 2014. Context menu in Android with example. Accessed 15 March 2016 http://www.compiletimeerror.com/2013/09/context-menu-in-android-with-example.html#.Vx0S2HuzC30.

ComputerHope 2016. Android. Accessed 13 February 2016 http://www.computerhope.com/jargon/a/android.htm.

Conder, S. 2013. Android User Interface Design: Creating a Numeric Keypad withGridLayout. Accessed 23 Februay 2016 http://code.tutsplus.com/tutorials/android-user-interface-design-creating-a-numeric-keypad-with-gridlayout--mobile-8677.

EazyTutz 2016. Android LinearLayout. Accessed 23 February 2016 http://www.eazytutz.com/android/android-linearlayout.

Genymotion 2016. Genymotion Features. Accessed 18 February 2016 https://www.genymotion.com.

HeatWare 2014. A Brief History of Android OS. Accessed 13 February 2016 http://www.heatware.net/linux-unix/brief-history-android.

Javatechig 2013. Android Toast Example. Accessed 13 March 2016 http://javatechig.com/android/android-toast-example.

Jenkov, J. 2013. Android Toast. Accessed 14 March 2016 http://tutorials.jenkov.com/android/toast.html.

Jones, K. 2012. Android Fragments Tutorial. Accessed 5 March 2016 https://newcircle.com/s/post/1250/android_fragments_tutorial.

LearnCertification 2015. The Android Manifest File. Accessed 20 February 2016 http://www.learncertification.com/study-material/the-android-manifest-file-58.

Martinez, M. 2014. History of Android OS and Settings. Accessed 13 February 2016 https://prezi.com/aj7pinr-xnfb/history-of-android-os-and-settings.

NESTED IF 2014. The example of the Table Layout. Accessed 23 February 2016 http://nestedif.com/android-development/android-tablelayout-example.

Oracle. 2016. Java SE Downloads. Accessed 13 February 2016 http://www.oracle.com/technetwork/articles/javase/index-jsp-138363.html.

Quora 2016. What is android app development. Accessed 16 February 2016 https://www.quora.com/What-is-android-app-development.

Rohini, M. 2014. What is Android? What are its Features and Applications? Accessed 14 February 2016 http://tutsdaddy.com/what-is-android.html.

StackOverFlow 2014. Difference between emulator and AVD. Accessed 18 February 2016 http://stackoverflow.com/questions/11574601/dfference-between-emulator-and-avd/11575321.

Stadd, A. 2015. How to Become an Android Developer. Accessed 14 February 2016 http://blog.udacity.com/2015/05/become-android-developer.html.

StartAndroid 2013. Creating and starting an Activity. Accessed 1 March 2016 http://startandroid.ru/en/lessons/complete-list/227-lesson-21-creating-and-starting-an-activity.html.

TechNotification 2014. What is Android operating system? Accessed 13 February 2016 http://www.technotification.com/2014/02/what-is-android-operating-system.html.

Techotopia 2016a. An Overview and Android Studio Example of Android Event Handling. Accessed 10 March 2016 http://www.techotopia.com/index.php/An_Overview_and_Android_Studio_Example_of_Android_Event_Handling.

Techotopia 2016b. Working with the Android GridLayout in XML Layout Resources. Accessed 6 March 2016 http://www.techotopia.com/index.php/Working_with_the_Android_GridLayout_in_XML_Layout_Resources.

Tools Android 2014. Gradle Plugin User Guide. Accessed 20 February 2016 http://tools.android.com/tech-docs/new-build-system/user-guide.

TutorialsPoint 2016a. Android - Event Handling. Accessed 10 March 2016 http://www.tutorialspoint.com/android/android_event_handling.htm.

TutorialsPoint 2016b. Android - Intents and Filters. Accessed 15 March 2016 http://www.tutorialspoint.com/android/android_intents_filters.htm.

TutorialsPoint 2016c. Android - Shared Preferences Tutorial. Accessed 18 March 2016 http://www.tutorialspoint.com/android/android_shared_preferences.htm.

TutorialsPoint 2016d. Android Absolute Layout. Accessed 23 February 2016 http://www.tutorialspoint.com/android/android_absolute_layout.htm.

TutorialsPoint 2016e. Android Frame Layout. Accessed 6 March 2016 http://www.tutorialspoint.com/android/android_frame_layout.htm.

TutorialsPoint 2016f. Android Linear Layout. Accessed 6 March 2016 http://www.tutorialspoint.com/android/android_linear_layout.htm.

TutorialsPoint 2016g. Android Relative Layout. Accessed 6 March 2016 http://www.tutorialspoint.com/android/android_relative_layout.htm.

TutorialsPoint 2016h. Android Table Layout. Accessed 6 March 2016 http://www.tutorialspoint.com/android/android_table_layout.htm.

Varela, K. 2012. Android FrameLayout example. Accessed 23 February 2016 http://karimvarela.com/2012/05/31/android-framelayout.

Vogella 2014a. Android Intents - Tutorial. Accessed 15 March 2016 http://www.vogella.com/tutorials/AndroidIntent/article.html.

Vogella 2015b. Android application and activity life cycle - Tutorial. Accessed 25 February 2016 http://www.vogella.com/tutorials/AndroidLifeCycle/article.html.

Vogella 2015c. Android Logging - Tutorial. Accessed 12 March 2016 http://www.vogella.com/tutorials/AndroidLogging/article.html.

Vogella 2015d. Multi-pane development in Android with Fragments - Tutorial. Accessed 5 March 2016 http://www.vogella.com/tutorials/AndroidFragments/article.html.

Vogella 2016e. The Gradle build system - Tutorial. Accessed 20 February 2016 http://www.vogella.com/tutorials/Gradle/article.html.

Wikipedia 2014a. Android Studio. Accessed 13 February 2016 https://en.wikipedia.org/wiki/Android_Studio.

Wikipedia 2016b. Google Play. Accessed 15 March 2016 https://en.wikipedia.org/wiki/Google_Play.

Yadav, K. 2012. Absolute Layout In Android. Accessed 7 March 2016 http://www.learn-android-easily.com/2013/05/absolute-layout-in-android.html.

APPENDICES

FragmentLogin

```
package ru.travellingtogether.travellingtogether.fragments;

import android.app.FragmentTransaction;
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import ru.travellingtogether.travellingtogether.MainActivity;
import ru.travellingtogether.travellingtogether.parsers.LoginBW;
import
ru.travellingtogether.travellingtogether.parsers.LoginJSONParser;
import ru.travellingtogether.travellingtogether.R;

public class FragmentLogin extends android.app.Fragment {

    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    // declaration of variables for fragment ets
    EditText etLogUsername, etLogPassword;

    // declaration of strings for login stage
    String username, password;

    public FragmentLogin() {
        // Required empty public constructor
    }

    public static FragmentLogin newInstance(String param1, String
param2) {
        FragmentLogin fragment = new FragmentLogin();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
```

```
}

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_login, container,
false);
        etLogUsername = (EditText) v.findViewById(R.id.etLogUsername);
        etLogPassword = (EditText) v.findViewById(R.id.etLogPassword);

    //    ImageView logoImg = (ImageView)
v.findViewById(R.id.logoImg);
    //    logoImg.setImageResource(R.drawable.car);

        // Register button
        Button btnLogRegister = (Button)
v.findViewById(R.id.logRegister);
        btnLogRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // open FragmentRegister
                FragmentRegister fregister = new FragmentRegister();
                FragmentTransaction ftRegister =
getFragmentManager().beginTransaction();
                ftRegister.replace(R.id.container, fregister);
                ftRegister.addToBackStack(null);
                ftRegister.commit();
            }
        });

        // Login button
        Button btnLogin = (Button) v.findViewById(R.id.login);
        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                username = etLogUsername.getText().toString();
                password = etLogPassword.getText().toString();

                if (isOnline()) {
                    // send username:password to database and check is
it correct
                    LoginBW loginBW = new LoginBW(FragmentLogin.this,
getActivity());
                    loginBW.execute(username, password);
                } else {
                    Toast.makeText(getActivity(), R.string.noInternet,
Toast.LENGTH_SHORT).show();
                }
            }
        });

        return v;
    }

    @Override
    public void onResume() {
        super.onResume();
        ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.TravellingToget
her);
        etLogUsername.setText("");
```

```
            etLogPassword.setText("");
            MainActivity.jsonMarker = null;
            }

        // internet connection state check
        public boolean isOnline() {
            ConnectivityManager cm = (ConnectivityManager)
getActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo netInfo = cm.getActiveNetworkInfo();
            if (netInfo != null && netInfo.isConnectedOrConnecting()) {
                return true;
            }
            return false;
        }

        // if username:password are correct, get user data in json format
        public void loginBWPE() {
            LoginJSONParser loginJSONParser = new
LoginJSONParser(FragmentLogin.this);
            loginJSONParser.execute(username, password);
        }

        // open FragmentUser
        public void loginJSONPE(String json) {
            FragmentUser fuser = new FragmentUser();
            FragmentTransaction ftUser =
getFragmentManager().beginTransaction();
            ftUser.replace(R.id.container, fuser);
            ftUser.commit();
        }

        public void onButtonPressed(Uri uri) {
            if (mListener != null) {
                mListener.onFragmentInteraction(uri);
            }
        }

        @Override
        public void onDetach() {
            super.onDetach();
            mListener = null;
        }

        public interface OnFragmentInteractionListener {
            void onFragmentInteraction(Uri uri);
        }
}
```

LoginBW parser

```
package ru.travellingtogether.travellingtogether.parsers;

import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;

import
ru.travellingtogether.travellingtogether.fragments.FragmentLogin;

// sending JSON request to server and getting answer
// Login stage: check if username:password are correct
public class LoginBW extends AsyncTask<String,Void,String> {
    public FragmentLogin source = null;
    Context context;
    ProgressDialog loading;

    public LoginBW(FragmentLogin fl, Context ctx) {
        source = fl;
        context = ctx;
    }

    @Override
    protected String doInBackground(String... params) {
        String login_url = "http://travellingtogether.ru/login.php";

        try {
            String username = params[0];
            String password = params[1];
            URL url = new URL(login_url);

            // creating an http connection to communicate with url
            HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
            httpURLConnection.setRequestMethod("POST");
            httpURLConnection.setDoOutput(true);
            httpURLConnection.setDoInput(true);

            // sending a request to server to check is
username:password pair correct
            OutputStream outputStream =
httpURLConnection.getOutputStream();
            BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
            String post_data = URLEncoder.encode("username","UTF-
8")+"="+URLEncoder.encode(username,"UTF-8")+"&"
                    +URLEncoder.encode("password","UTF-
8")+"="+URLEncoder.encode(password,"UTF-8");
```

```
            bufferedWriter.write(post_data);
            bufferedWriter.flush();
            bufferedWriter.close();
            outputStream.close();

            // reading answer from server
             InputStream inputStream =
httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream,"iso-8859-1"));
            String result="";
            String line="";
            while ((line = bufferedReader.readLine())!=null) {
                result = line;
            }
            bufferedReader.close();
            inputStream.close();

            httpURLConnection.disconnect();
            return result;

        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPreExecute() {
        loading = ProgressDialog.show(context, "Please
wait...",null,true,true);
    }

    @Override
    protected void onPostExecute(String result) {
        loading.dismiss();
        Toast.makeText(context, result, Toast.LENGTH_SHORT).show();
        if (result.equals(" Login success ")) {
        source.loginBWPE();
        }
    }

    @Override
    protected void onProgressUpdate(Void... values) {
        super.onProgressUpdate(values);
    }
}
```

FragmentRegister

```
package ru.travellingtogether.travellingtogether.fragments;

import android.app.FragmentManager;
import android.content.Context;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import ru.travellingtogether.travellingtogether.MainActivity;
import ru.travellingtogether.travellingtogether.R;
import ru.travellingtogether.travellingtogether.parsers.RegisterBW;

public class FragmentRegister extends android.app.Fragment {

    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    // declaration of variables for fragment ets
    EditText etRegUsername, etRegPassword, etRegName, etRegSurname,
etRegPhone;

    // declaration of strings for register stage
    String username, password, name, surname, phonenumber;

    // SharedPreferences to contain login session data
    SharedPreferences sPref;

    public FragmentRegister() {
        // Required empty public constructor
    }

    public static FragmentRegister newInstance(String param1, String
param2) {
        FragmentRegister fragment = new FragmentRegister();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
```

```
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_register,
container, false);
        etRegUsername = (EditText) v.findViewById(R.id.etRegUsername);
        etRegPassword = (EditText) v.findViewById(R.id.etRegPassword);
        etRegName = (EditText) v.findViewById(R.id.etRegName);
        etRegSurname = (EditText) v.findViewById(R.id.etRegSurname);
        etRegPhone = (EditText) v.findViewById(R.id.etRegPhone);

        // Register button
        Button btnRegRegister = (Button)
v.findViewById(R.id.regRegister);
        btnRegRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                username = etRegUsername.getText().toString();
                password = etRegPassword.getText().toString();
                name = etRegName.getText().toString();
                surname = etRegSurname.getText().toString();
                phonenumber = etRegPhone.getText().toString();

                if (isOnline()) {
                    // send user inserted data to database
                    RegisterBW registerBW = new
RegisterBW(FragmentRegister.this, getActivity());
                    registerBW.execute(username, password, name,
surname, phonenumber);
                } else {
                    Toast.makeText(getActivity(), R.string.noInternet,
Toast.LENGTH_SHORT).show();
                }
            }
        });

        return v;
    }

    public void onResume() {
        super.onResume();
        ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.TravellingToget
her);
    }

    // internet connection state check
    public boolean isOnline() {
        ConnectivityManager cm = (ConnectivityManager)
getActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if (netInfo != null && netInfo.isConnectedOrConnecting()) {
            return true;
        }
        return false;
    }

    // open FragmentUser, fill sPref with inserted data
    public void registerBWPE() {
```

```
        MainActivity.regMarker = "regmarker";

        sPref = this.getActivity().getSharedPreferences("logdata",
Context.MODE_PRIVATE);
        SharedPreferences.Editor ed = sPref.edit();
        ed.putString(MainActivity.USERNAME, username);
        ed.putString(MainActivity.NAME, name);
        ed.putString(MainActivity.SURNAME, surname);
        ed.putString(MainActivity.PHONENUMBER, phonenumber);
        ed.commit();

        FragmentUser fuser = new FragmentUser();
        FragmentManager fm = getFragmentManager();
        fm.popBackStack(null,
FragmentManager.POP_BACK_STACK_INCLUSIVE);
        fm.beginTransaction().replace(R.id.container, fuser).commit();
    }

    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    public interface OnFragmentInteractionListener {
        void onFragmentInteraction(Uri uri);
    }
```

RegisterBW parser

```
package ru.travellingtogether.travellingtogether.parsers;

import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;

import
ru.travellingtogether.travellingtogether.fragments.FragmentRegister;

// sending JSON request to server and getting answer
// Registration stage: send new user data to database
public class RegisterBW extends AsyncTask<String,Void,String> {
    Context context;
    ProgressDialog loading;
    public FragmentRegister source = null;

    public RegisterBW(FragmentRegister fr, Context ctx) {
        source = fr;
        context = ctx;
    }

    @Override
    protected String doInBackground(String... params) {
        String register_url =
        "http://travellingtogether.ru/register.php";

        try {
            String username = params[0];
            String password = params[1];
            String name = params[2];
            String surname = params[3];
            String phonenumber = params [4];
            URL url = new URL(register_url);

            // creating an http connection to communicate with url
            HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
            httpURLConnection.setRequestMethod("POST");
            httpURLConnection.setDoOutput(true);
            httpURLConnection.setDoInput(true);

            // sending a request to server to insert data in database
            OutputStream outputStream =
httpURLConnection.getOutputStream();
            BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
            String post_data = URLEncoder.encode("username","UTF-
```

```
8")+"="+URLEncoder.encode(username,"UTF-8")+"&"
                    +URLEncoder.encode("password","UTF-
8")+"="+URLEncoder.encode(password,"UTF-8")+"&"
                    +URLEncoder.encode("name","UTF-
8")+"="+URLEncoder.encode(name,"UTF-8")+"&"
                    +URLEncoder.encode("surname","UTF-
8")+"="+URLEncoder.encode(surname,"UTF-8")+"&"
                    +URLEncoder.encode("phonenumber","UTF-
8")+"="+URLEncoder.encode(phonenumber,"UTF-8");
            bufferedWriter.write(post_data);
            bufferedWriter.flush();
            bufferedWriter.close();
            outputStream.close();

            // reading answer from server
            InputStream inputStream =
httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream,"iso-8859-1"));
            String result="";
            String line="";
            while ((line = bufferedReader.readLine())!=null) {
                result = line;
            }
            bufferedReader.close();
            inputStream.close();

            httpURLConnection.disconnect();
            return result;

        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPreExecute() {
        loading = ProgressDialog.show(context, "Please
wait...",null,true,true);
    }

    @Override
    protected void onPostExecute(String result) {
        loading.dismiss();
        Toast.makeText(context, result, Toast.LENGTH_SHORT).show();
        if (result.equals(" Registration successful ")) {
            source.registerBWPE();
        }
    }

    @Override
    protected void onProgressUpdate(Void... values) {
        super.onProgressUpdate(values);
    }
}
```

FragmentUser

```
package ru.travellingtogether.travellingtogether.fragments;

import android.app.FragmentTransaction;
import android.content.Context;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import ru.travellingtogether.travellingtogether.MainActivity;
import
ru.travellingtogether.travellingtogether.parsers.LoginJSONParser;
import ru.travellingtogether.travellingtogether.R;

public class FragmentUser extends android.app.Fragment {

    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    // declaration of variables for fragment tvs
    TextView tvUserUsername, tvUserName, tvUserPhonenumber,
navHeaderUsername, navHeaderName;

    // SharedPreferences to contain login session data
    SharedPreferences sPref;

    // string for username and user status called in other fragments
    public static String username = null;
    public static String userStatus = null;

    // strings for extractJSON()
    private static final String JSON_ARRAY ="result";
    private static final String JSON_USERNAME = "username";
    private static final String JSON_NAME= "name";
    private static final String JSON_SURNAME = "surname";
    private static final String JSON_PHONENUMBER = "phonenumber";

    public FragmentUser() {
        // Required empty public constructor
    }

    public static FragmentUser newInstance(String param1, String
param2) {
        FragmentUser fragment = new FragmentUser();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
```

```
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_user, container,
false);
        tvUserUsername = (TextView)
v.findViewById(R.id.tvUserUsername);
        tvUserName = (TextView) v.findViewById(R.id.tvUserName);
        tvUserPhonenumber = (TextView)
v.findViewById(R.id.tvUserPhonenumber);
        navHeaderUsername =
(TextView)getActivity().findViewById(R.id.navHeaderUsername);
        navHeaderName =
(TextView)getActivity().findViewById(R.id.navHeaderName);

        // Driver button
        Button btnDriver = (Button) v.findViewById(R.id.driver);
        btnDriver.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // open FragmentDriver
                FragmentDriver fdriver = new FragmentDriver();
                FragmentTransaction ftDriver =
getFragmentManager().beginTransaction();
                ftDriver.replace(R.id.container, fdriver);
                ftDriver.addToBackStack(null);
                ftDriver.commit();
            }
        });

        // Passenger button
        Button btnPassenger = (Button) v.findViewById(R.id.passenger);
        btnPassenger.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // open FragmentPassenger
                FragmentPassenger fpassenger = new
FragmentPassenger();
                FragmentTransaction ftPassenger =
getFragmentManager().beginTransaction();
                ftPassenger.replace(R.id.container, fpassenger);
                ftPassenger.addToBackStack(null);
                ftPassenger.commit();
            }
        });

        return v;
```

```
    }

    public void onResume() {
        super.onResume();
        ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.TravellingToget
her);

        // if not logged id, fill sPref with data from received json
        if (MainActivity.loggedMarker == null &&
MainActivity.regMarker == null && MainActivity.updMarker == null &&
MainActivity.jsonMarker == null) {
            extractJSON();
        }

        // read SharedPreferences data
        sPref = this.getActivity().getSharedPreferences("logdata",
MainActivity.MODE_PRIVATE);
        String usernamePref = sPref.getString(MainActivity.USERNAME,
"");
        String namePref = sPref.getString(MainActivity.NAME, "");
        String surnamePref = sPref.getString(MainActivity.SURNAME,
"");
        String phonePref = sPref.getString(MainActivity.PHONENUMBER,
"");

        // fill username, tvs and header elements with sPref data
        username = usernamePref;
        tvUserUsername.setText(usernamePref);
        tvUserName.setText(namePref + " " + surnamePref);
        tvUserPhonenumber.setText(phonePref);
        if (MainActivity.loggedMarker == null) {
            navHeaderName.setText(namePref + " " + surnamePref);
            navHeaderUsername.setText(usernamePref);
        }
    }

    // extracting json, getting data and inserting it sPref
    private void extractJSON(){
        try {
            JSONObject jsonResult = new
JSONObject(LoginJSONParser.jsonResult);
            JSONArray userinfo = jsonResult.getJSONArray(JSON_ARRAY);
            JSONObject jsonObject = userinfo.getJSONObject(0);

            sPref = getActivity().getSharedPreferences("logdata",
Context.MODE_PRIVATE);
            SharedPreferences.Editor ed = sPref.edit();
            ed.putString(MainActivity.USERNAME,
jsonObject.getString(JSON_USERNAME));
            ed.putString(MainActivity.NAME,
jsonObject.getString(JSON_NAME));
            ed.putString(MainActivity.SURNAME,
jsonObject.getString(JSON_SURNAME));
            ed.putString(MainActivity.PHONENUMBER,
jsonObject.getString(JSON_PHONENUMBER));
            ed.commit();

        } catch (JSONException e) {
            e.printStackTrace();
        }
```

```
    }

    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    public interface OnFragmentInteractionListener {
        void onFragmentInteraction(Uri uri);
    }
}
```

FragmentDriver

```
package ru.travellingtogether.travellingtogether.fragments;

import android.app.FragmentTransaction;
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

import ru.travellingtogether.travellingtogether.R;
import
ru.travellingtogether.travellingtogether.parsers.TripPassengerJSONPars
er;

public class FragmentDriver extends android.app.Fragment {

    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    public FragmentDriver() {
        // Required empty public constructor
    }

    public static FragmentDriver newInstance(String param1, String
param2) {
        FragmentDriver fragment = new FragmentDriver();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_driver, container,
false);
```

```
        // "Create a trip" button
        Button btnDriverTrip = (Button)
v.findViewById(R.id.btnDriverCreate);
        btnDriverTrip.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // open FragmentMap
                FragmentUser.userStatus = "driver";
                FragmentMap fmap = new FragmentMap();
                FragmentTransaction ftMap =
getFragmentManager().beginTransaction();
                ftMap.replace(R.id.container, fmap);
                ftMap.addToBackStack(null);
                ftMap.commit();
            }
        });

        // Requested trip list button
        Button btnPassengerList = (Button)
v.findViewById(R.id.btnPassengerList);
        btnPassengerList.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                if (isOnline()) {
                    // get from database list of trips created by
passenger
                    FragmentUser.userStatus = "driver";
                    TripPassengerJSONParser tripPassengerJSONParser =
new TripPassengerJSONParser(FragmentDriver.this, getActivity());

tripPassengerJSONParser.execute(FragmentUser.userStatus);
                } else {
                    Toast.makeText(getActivity(), R.string.noInternet,
Toast.LENGTH_SHORT).show();
                }
            }
        });

        return v;
    }

    @Override
    public void onResume() {
        super.onResume();
        ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.driver);
    }

    // internet connection state check
    public boolean isOnline() {
        ConnectivityManager cm = (ConnectivityManager)
getActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if (netInfo != null && netInfo.isConnectedOrConnecting()) {
            return true;
        }
        return false;
    }

    // open FragmentTrips
    public void passengerJSONPE(String json) {
```

```
        FragmentTripList ftlist = new FragmentTripList();
        FragmentTransaction ftrans =
getFragmentManager().beginTransaction();
        ftrans.replace(R.id.container, ftlist);
        ftrans.addToBackStack(null);
        ftrans.commit();
    }

    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    public interface OnFragmentInteractionListener {
        void onFragmentInteraction(Uri uri);
    }
}
```

TripPassengerJSONParser

```
package ru.travellingtogether.travellingtogether.parsers;

import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;

import
ru.travellingtogether.travellingtogether.fragments.FragmentDriver;

// sending JSON request to server and getting JsonString answer
// Trip list: get list of trips created by passengers
public class TripPassengerJSONParser extends
AsyncTask<String,Void,String> {
    public static String jsonResult = null;
    public FragmentDriver source = null;
    Context context;
    ProgressDialog loading;
    public TripPassengerJSONParser(FragmentDriver fp, Context ctx) {
        source = fp;
        context = ctx;
    }

    @Override
    protected String doInBackground(String... params) {
        String gettriplist_url =
"http://travellingtogether.ru/gettriplist.php";


        try {
            String userstatus = params[0];
            URL url = new URL(gettriplist_url);

            // creating an http connection to communicate with url
            HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
            httpURLConnection.setRequestMethod("POST");
            httpURLConnection.setDoOutput(true);
            httpURLConnection.setDoInput(true);

            // sending a request to server to get trip list
            OutputStream outputStream =
httpURLConnection.getOutputStream();
            BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
            String post_data = URLEncoder.encode("type", "UTF-8") +
"=" + URLEncoder.encode(userstatus, "UTF-8");
            bufferedWriter.write(post data);
            bufferedWriter.flush();
```

```
            bufferedWriter.close();
            outputStream.close();

            // reading answer from server
            InputStream inputStream =
httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));
            jsonResult = bufferedReader.readLine();
            bufferedReader.close();
            inputStream.close();

            httpURLConnection.disconnect();
            return jsonResult;

        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPreExecute() {
        loading = ProgressDialog.show(context, "Please
wait...",null,true,true);
    }

    @Override
    protected void onPostExecute(String jsonResult) {
        loading.dismiss();
        source.passengerJSONPE(jsonResult);
    }
}
```

FragmentPassenger

```
package ru.travellingtogether.travellingtogether.fragments;

import android.app.FragmentTransaction;
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

import ru.travellingtogether.travellingtogether.R;
import
ru.travellingtogether.travellingtogether.parsers.TripDriverJSONParser;

public class FragmentPassenger extends android.app.Fragment {

    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    public FragmentPassenger() {
        // Required empty public constructor
    }

    public static FragmentPassenger newInstance(String param1, String
param2) {
        FragmentPassenger fragment = new FragmentPassenger();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v=inflater.inflate(R.layout.fragment_passenger,
container, false);

        // "Create a trip" button
```

```
        Button btnPassengerTrip = (Button)
v.findViewById(R.id.btnPassengerCreate);
        btnPassengerTrip.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                // open FragmentMap
                FragmentUser.userStatus = "passenger";
                FragmentMap fmap = new FragmentMap();
                FragmentTransaction ftMap =
getFragmentManager().beginTransaction();
                ftMap.replace(R.id.container, fmap);
                ftMap.addToBackStack(null);
                ftMap.commit();
            }
        });

        // Existed trip list button
        Button btnDriverList = (Button)
v.findViewById(R.id.btnDriverList);
        btnDriverList.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (isOnline()) {
                    // get from database list of trips created by
drivers
                    FragmentUser.userStatus = "passenger";
                    TripDriverJSONParser tripDriverJSONParser = new
TripDriverJSONParser(FragmentPassenger.this, getActivity());

tripDriverJSONParser.execute(FragmentUser.userStatus);
                } else {
                    Toast.makeText(getActivity(), R.string.noInternet,
Toast.LENGTH_SHORT).show();
                }
            }
        });

        return v;
    }

    @Override
    public void onResume() {
        super.onResume();
        ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.passenger);
    }

    // internet connection state check
    public boolean isOnline() {
        ConnectivityManager cm = (ConnectivityManager)
getActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if (netInfo != null && netInfo.isConnectedOrConnecting()) {
            return true;
        }
        return false;
    }

    // open FragmentTripList
    public void driverJSONPE(String json) {
        FragmentTripList ftlist = new FragmentTripList();
```

```
        FragmentTransaction ftrans =
getFragmentManager().beginTransaction();
        ftrans.replace(R.id.container, ftlist);
        ftrans.addToBackStack(null);
        ftrans.commit();
    }

    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    public interface OnFragmentInteractionListener {
        void onFragmentInteraction(Uri uri);
    }
}
```

FragmentMap

```
package ru.travellingtogether.travellingtogether.fragments;

import android.Manifest;
import android.app.DatePickerDialog;

import android.app.FragmentManager;
import android.app.TimePickerDialog;
import android.content.Context;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.location.Address;
import android.location.Geocoder;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.AdapterView;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TimePicker;
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.util.Calendar;
import java.util.List;

import ru.travellingtogether.travellingtogether.R;
import ru.travellingtogether.travellingtogether.parsers.MyGeocoder;
import ru.travellingtogether.travellingtogether.parsers.TripCreateBW;
import ru.travellingtogether.travellingtogether.parsers.PlacesBW;

public class FragmentMap extends android.app.Fragment {

    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";
```

```
    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    // variable for mapView
    GoogleMap googleMap;

    // variables for markers and polyline
    final Marker[] markerFrom = {null};
    final Marker[] markerTo = {null};
    final LatLng[] latLngFrom = {null};
    final LatLng[] latLngTo = {null};
    final Polyline[] line = {null};

    // variables for fragment views
    EditText etDate, etTime;
    AutoCompleteTextView atvFrom, atvTo;

    // call for Google Places Api Web Service parser
    PlacesBW placesBW;

    // variables to send to database
    String tripDay = null, tripMonth = null, tripYear = null, tripHour
= null, tripMinute = null;
    String tripFrom = null, tripTo = null;

    // helper variables to get text from AutoCompleteTextViews
    String locationFrom = null, locationTo = null;

    // helper variables for MyGeocoder proper work order
    int atvFromMarker = 0;
    int atvToMarker = 0;

    public FragmentMap() {
        // Required empty public constructor
    }

    public static FragmentMap newInstance(String param1, String
param2) {
        FragmentMap fragment = new FragmentMap();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_map, container,
```

```
false);

        // getting current date and time to show it in TimePicker and
DatePickerDialogs
        final Calendar cal = Calendar.getInstance();
        final int currentYear = cal.get(Calendar.YEAR);
        final int currentMonth = cal.get(Calendar.MONTH);
        final int currentDay = cal.get(Calendar.DAY_OF_MONTH);
        final int currentHour = cal.get(Calendar.HOUR_OF_DAY);
        final int currentMinute = cal.get(Calendar.MINUTE);

        // attaching map to googleMap variable
        createMapView();

        // attaching ets to variables
        etDate = (EditText) v.findViewById(R.id.etDate);
        etTime = (EditText) v.findViewById(R.id.etTime);

        // attaching AutoCompleteTV to atvFrom variable, calling
Google Places parser
        atvFrom = (AutoCompleteTextView) v.findViewById(R.id.atvFrom);
        atvFrom.setThreshold(2);
        atvFrom.addTextChangedListener(new TextWatcher() {
            @Override
            public void onTextChanged(CharSequence s, int start, int
before, int count) {
                placesBW = new PlacesBW(atvFrom, getActivity());
                placesBW.execute(s.toString());

                // erase locationFrom value, remove marker and
polyline
                locationFrom = null;
                if (markerFrom[0] != null) {
                    markerFrom[0].remove();
                    markerFrom[0] = null;
                }
                if (line[0] != null) {
                    line[0].remove();
                    line[0] = null;
                }
            }

            @Override
            public void beforeTextChanged(CharSequence s, int start,
int count,
                                          int after) {
                // TODO Auto-generated method stub
            }

            @Override
            public void afterTextChanged(Editable s) {
                // TODO Auto-generated method stub
            }
        });

        // on item click: set value to tripFrom, create marker on map,
polyline(optional)
        atvFrom.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
```

```
                locationFrom = atvFrom.getText().toString();
                List<Address> addressListFrom = null;
                Geocoder geocoder = new Geocoder(getActivity());
                atvFromMarker = 1;
                atvToMarker = 0;

                hideKeyboard();

                try {
                    if (markerFrom[0] != null) {
                        markerFrom[0].remove();
                    }
                    addressListFrom =
geocoder.getFromLocationName(locationFrom, 1);

                    // check if Geocoder works or not (does not work
on Meizu MX5), if yes - keep on
                    if (addressListFrom.size() > 0) {
                        Address addressFrom = addressListFrom.get(0);
                        latLngFrom[0] = new
LatLng(addressFrom.getLatitude(), addressFrom.getLongitude());
                        tripFrom =
addressListFrom.get(0).getFeatureName();
                        markerFrom[0] = googleMap.addMarker(new
MarkerOptions().position(latLngFrom[0]).title("From"));

googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLngFrom[0]));

                        if (markerTo[0] != null) {
                            if (line[0] != null) {
                                line[0].remove();
                            }
                            line[0] = googleMap.addPolyline(new
PolylineOptions().add(latLngFrom[0],
latLngTo[0]).width(5).color(Color.BLUE));
                        }

                    } else {
                        if (isOnline()) {
                            // if Geocoder does not work, call for
self-created one using Google Maps Geocoding API
                            MyGeocoder myGeocoder = new
MyGeocoder(FragmentMap.this, getActivity());
                            myGeocoder.execute(locationFrom);
                        } else {
                            Toast.makeText(getActivity(), "Your
internet connection is turned off", Toast.LENGTH_SHORT).show();
                        }

                    }

                } catch (IOException e) {
                    Toast.makeText(getActivity(), "Error",
Toast.LENGTH_SHORT).show();
                }
            }
        });

        // attaching AutoCompleteTV to atvTo variable, calling Google
Places parser
        atvTo = (AutoCompleteTextView) v.findViewById(R.id.atvTo);
        atvTo.setThreshold(2);
```

```
        atvTo.addTextChangedListener(new TextWatcher() {

            @Override
            public void onTextChanged(CharSequence s, int start, int
before, int count) {
                placesBW = new PlacesBW(atvTo, getActivity());
                placesBW.execute(s.toString());

                // erase locationTo value, remove marker and polyline
                locationTo = null;
                if (markerTo[0] != null) {
                    markerTo[0].remove();
                    markerTo[0] = null;
                }
                if (line[0] != null) {
                    line[0].remove();
                    line[0] = null;
                }
            }

            @Override
            public void beforeTextChanged(CharSequence s, int start,
int count,
                                          int after) {
                // TODO Auto-generated method stub
            }

            @Override
            public void afterTextChanged(Editable s) {
                // TODO Auto-generated method stub
            }
        });

        // on item click: set value to tripTo, create marker on map,
polyline(optional)
        atvTo.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
                locationTo = atvTo.getText().toString();
                List<Address> addressListTo = null;
                Geocoder geocoder = new Geocoder(getActivity());
                atvFromMarker = 0;
                atvToMarker = 1;

                hideKeyboard();

                try {
                    if (markerTo[0] != null) {
                        markerTo[0].remove();
                    }
                    addressListTo =
geocoder.getFromLocationName(locationTo, 1);

                    // check if Geocoder works or not (does not work
on Meizu MX5), if yes - keep on
                    if (addressListTo.size() > 0) {
                        Address addressTo = addressListTo.get(0);
                        latLngTo[0] = new
LatLng(addressTo.getLatitude(), addressTo.getLongitude());
```

```
                        tripTo =
addressListTo.get(0).getFeatureName();
                        markerTo[0] = googleMap.addMarker(new
MarkerOptions().position(latLngTo[0]).title("To").icon(BitmapDescripto
rFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));

googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLngTo[0]));

                        if (markerFrom[0] != null) {
                            if (line[0] != null) {
                                line[0].remove();
                            }
                            line[0] = googleMap.addPolyline(new
PolylineOptions().add(latLngFrom[0],
latLngTo[0]).width(5).color(Color.BLUE));
                        }

                    } else {
                        if (isOnline()) {
                            // if Geocoder does not work, call for
self-created one using Google Maps Geocoding API
                            MyGeocoder myGeocoder = new
MyGeocoder(FragmentMap.this, getActivity());
                            myGeocoder.execute(locationTo);
                        } else {
                            Toast.makeText(getActivity(),
R.string.noInternet, Toast.LENGTH_SHORT).show();
                        }
                    }

                } catch (IOException e) {
                    //e.printStackTrace();
                    Toast.makeText(getActivity(), "Error",
Toast.LENGTH_SHORT).show();
                }
            }
        });

        // set date
        etDate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                DatePickerDialog dialog = new
DatePickerDialog(getActivity(), datePickerListener, currentYear,
currentMonth, currentDay);
                dialog.show();
            }
        });

        // set time
        etTime.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                TimePickerDialog dialog = new
TimePickerDialog(getActivity(), timePickerListener, currentHour,
currentMinute, true);
                dialog.show();
            }
        });

        // button create
        Button btnCreate = (Button)
```

```
v.findViewById(R.id.btnCreateTrip);
        btnCreate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // if From and To points are taken with AutoComplete
places, send trip values to database
                if ((tripFrom != null) && (tripTo != null)) {

                    if (tripDay != null && tripMonth != null &&
tripYear != null && tripHour != null && tripMinute != null && tripFrom
!= null && tripTo != null) {

                        if (isOnline()) {
                            TripCreateBW tripCreateBW = new
TripCreateBW(FragmentMap.this, getActivity());

tripCreateBW.execute(FragmentUser.userStatus, FragmentUser.username,
tripFrom, tripTo, tripDay, tripMonth, tripYear, tripHour, tripMinute);
                        } else {
                            Toast.makeText(getActivity(),
R.string.noInternet, Toast.LENGTH_SHORT).show();
                        }


                    } else {
                        // From and To points have values, but Date or
Time not
                        Toast.makeText(getActivity(),
R.string.fillDateTime, Toast.LENGTH_SHORT).show();
                    }
                }

                // if not, get tripFrom and tripTo values from
TextViews
                else {
                    locationFrom = atvFrom.getText().toString();
                    locationTo = atvTo.getText().toString();

                    if (!locationFrom.equals("") &&
!locationTo.equals("")) {
                        List<Address> addressListFrom = null;
                        List<Address> addressListTo = null;
                        Geocoder geocoder = new
Geocoder(getActivity());
                        try {
                            addressListFrom =
geocoder.getFromLocationName(locationFrom, 1);
                            addressListTo =
geocoder.getFromLocationName(locationTo, 1);

                            // check if Geocoder works or not (does
not work on Meizu MX5), if yes - keep on
                            if ((addressListFrom.size() > 0) &&
(addressListTo.size() > 0)) {
                                Address addressFrom =
addressListFrom.get(0);
                                latLngFrom[0] = new
LatLng(addressFrom.getLatitude(), addressFrom.getLongitude());
                                tripFrom =
addressListFrom.get(0).getFeatureName();

                                Address addressTo =
```

```
addressListTo.get(0);
                                latLngTo[0] = new
LatLng(addressTo.getLatitude(), addressTo.getLongitude());
                                tripTo =
addressListTo.get(0).getFeatureName();

                                // when From, To, Date, Time has
values, send it to database and create markers with polyline on map
                                if (tripDay != null && tripMonth !=
null && tripYear != null && tripHour != null && tripMinute != null &&
tripFrom != null && tripTo != null) {
                                    googleMap.addMarker(new
MarkerOptions().position(latLngFrom[0]).title("From"));
                                    googleMap.addMarker(new
MarkerOptions().position(latLngTo[0]).title("To").icon(BitmapDescripto
rFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));

googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLngFrom[0]));
                                    Polyline line =
googleMap.addPolyline(new PolylineOptions().add(latLngFrom[0],
latLngTo[0]).width(5).color(Color.BLUE));

                                    if (isOnline()) {
                                        TripCreateBW tripCreateBW =
new TripCreateBW(FragmentMap.this, getActivity());

tripCreateBW.execute(FragmentUser.userStatus, FragmentUser.username,
tripFrom, tripTo, tripDay, tripMonth, tripYear, tripHour, tripMinute);
                                    } else {
                                        Toast.makeText(getActivity(),
R.string.noInternet, Toast.LENGTH_SHORT).show();
                                    }

                                } else {
                                    // if some field is empty or
contain wrong value
                                    Toast.makeText(getActivity(),
R.string.fillFields, Toast.LENGTH_SHORT).show();
                                }

                            } else {
                                // if Geocoder does not work, call for
self-created one using Google Maps Geocoding API
                                atvFromMarker = 2;
                                atvToMarker = 0;
                                if (isOnline()) {
                                    MyGeocoder myGeocoderFrom = new
MyGeocoder(FragmentMap.this, getActivity());

myGeocoderFrom.execute(locationFrom);
                                } else {
                                    Toast.makeText(getActivity(),
R.string.noInternet, Toast.LENGTH_SHORT).show();
                                }

                            }

                        } catch (IOException e) {
                            Toast.makeText(getActivity(), "Error",
Toast.LENGTH_SHORT).show();
                        } catch (IndexOutOfBoundsException e) {
                            // if From or To views contain wrong
```

```
values that impossible to find on map
                            Toast.makeText(getActivity(),
R.string.noPlace, Toast.LENGTH_SHORT).show();
                        }

                    } else {
                        // if From or To view is empty
                        Toast.makeText(getActivity(),
R.string.fillFromTo, Toast.LENGTH_SHORT).show();
                    }
                }
            }
        });
        return v;
    }

    public void onResume() {
        super.onResume();
        if (FragmentUser.userStatus.equals("driver")) {
            ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.createTripDrive
r);
        }
        if (FragmentUser.userStatus.equals("passenger")) {
            ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.createTripPasse
nger);
        }
    }

    // internet connection state check
    public boolean isOnline() {
        ConnectivityManager cm = (ConnectivityManager)
getActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if (netInfo != null && netInfo.isConnectedOrConnecting()) {
            return true;
        }
        return false;
    }

    // hide keyboard on item chosen from atv
    private void hideKeyboard() {
        // Check if no view has focus
        View view = getActivity().getCurrentFocus();
        if (view != null) {
            InputMethodManager inputManager = (InputMethodManager)
getActivity().getSystemService(Context.INPUT_METHOD_SERVICE);

inputManager.hideSoftInputFromWindow(view.getWindowToken(),
InputMethodManager.HIDE_NOT_ALWAYS);
        }
    }

    // open FragmentUser
    public void tripcreateBWPE() {
        FragmentUser fuser = new FragmentUser();
        FragmentManager fm = getFragmentManager();
        fm.popBackStack(null,
FragmentManager.POP_BACK_STACK_INCLUSIVE);
        fm.beginTransaction().replace(R.id.container, fuser).commit();
    }
```

```
    // Self-created Geocoder post execute method
    public void myGeocoderPE(String json) {
        try {
            JSONObject jObj = new JSONObject(MyGeocoder.jsonResult);
            String Status = jObj.getString("status");

            if (Status.equalsIgnoreCase("OK")) {
                // getting latitude, longtitude and city name from
JSON reply
                Double lat =
Double.valueOf(jObj.getJSONArray("results").getJSONObject(0).getJSONOb
ject("geometry").getJSONObject("location").getString("lat"));
                Double lng =
Double.valueOf(jObj.getJSONArray("results").getJSONObject(0).getJSONOb
ject("geometry").getJSONObject("location").getString("lng"));
                String city =
jObj.getJSONArray("results").getJSONObject(0).getJSONArray("address_co
mponents").getJSONObject(0).getString("long_name");

                // create marker and polyline(optional) for place
chosen from AutoCompleteTextView "From"
                if (atvFromMarker == 1) {
                    tripFrom = city;
                    latLngFrom[0] = new LatLng(lat, lng);
                    markerFrom[0] = googleMap.addMarker(new
MarkerOptions().position(latLngFrom[0]).title("From"));

googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLngFrom[0]));

                    if (markerTo[0] != null) {
                        if (line[0] != null) {
                            line[0].remove();
                        }
                        line[0] = googleMap.addPolyline(new
PolylineOptions().add(latLngFrom[0],
latLngTo[0]).width(5).color(Color.BLUE));
                    }
                }

                // create marker and polyline(optional) for place
chosen from AutoCompleteTextView "To"
                if (atvToMarker == 1) {
                    tripTo = city;
                    latLngTo[0] = new LatLng(lat, lng);
                    markerTo[0] = googleMap.addMarker(new
MarkerOptions().position(latLngTo[0]).title("To").icon(BitmapDescripto
rFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));

googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLngTo[0]));

                    if (markerFrom[0] != null) {
                        if (line[0] != null) {
                            line[0].remove();
                        }
                        line[0] = googleMap.addPolyline(new
PolylineOptions().add(latLngFrom[0],
latLngTo[0]).width(5).color(Color.BLUE));
                    }
                }

                // create marker and polyline for place given in
```

```
TextViews
                if (atvToMarker == 2) {
                    tripTo = city;
                    latLngTo[0] = new LatLng(lat, lng);

                    // if all fields filled correctly, send data to
database
                    if (tripDay != null && tripMonth != null &&
tripYear != null && tripHour != null && tripMinute != null && tripFrom
!= null && tripTo != null) {
                        googleMap.addMarker(new
MarkerOptions().position(latLngFrom[0]).title("From"));
                        googleMap.addMarker(new
MarkerOptions().position(latLngTo[0]).title("To").icon(BitmapDescripto
rFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));

googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLngFrom[0]));
                        Polyline line = googleMap.addPolyline(new
PolylineOptions().add(latLngFrom[0],
latLngTo[0]).width(5).color(Color.BLUE));

                        TripCreateBW tripCreateBW = new
TripCreateBW(FragmentMap.this, getActivity());
                        tripCreateBW.execute(FragmentUser.userStatus,
FragmentUser.username, tripFrom, tripTo, tripDay, tripMonth, tripYear,
tripHour, tripMinute);

                    } else {
                        // if some field is empty or contain wrong
value
                        Toast.makeText(getActivity(),
R.string.fillFields, Toast.LENGTH_SHORT).show();
                    }
                }

                // get data for "From marker", call for self-created
Geocoder using Google Maps Geocoding API
                if (atvFromMarker == 2) {
                    tripFrom = city;
                    latLngFrom[0] = new LatLng(lat, lng);
                    atvFromMarker = 0;
                    atvToMarker = 2;

                    MyGeocoder myGeocoderTo = new
MyGeocoder(FragmentMap.this, getActivity());
                    myGeocoderTo.execute(locationTo);
                }
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    // set value to tripYear, tripMonth, tripDay
    private DatePickerDialog.OnDateSetListener datePickerListener =
new DatePickerDialog.OnDateSetListener() {

        // when dialog box is closed, below method will be called.
        public void onDateSet(DatePicker view, int selectedYear, int
selectedMonth, int selectedDay) {

            tripYear = String.valueOf(selectedYear);
```

```
            if (selectedMonth < 10) {
                tripMonth = 0 + String.valueOf(selectedMonth + 1);
            } else {
                tripMonth = String.valueOf(selectedMonth + 1);
            }

            if (selectedDay < 10) {
                tripDay = 0 + String.valueOf(selectedDay);
            } else {
                tripDay = String.valueOf(selectedDay);
            }

            etDate.setText(tripDay + "." + tripMonth + "." +
tripYear);
        }
    };

    // set value to tripHour, tripMinute
    private TimePickerDialog.OnTimeSetListener timePickerListener =
new TimePickerDialog.OnTimeSetListener() {

        // when dialog box is closed, below method will be called.
        public void onTimeSet(TimePicker view, int selectedHour, int
selectedMinute) {
            if (selectedHour < 10) {
                tripHour = 0 + String.valueOf(selectedHour);
            } else {
                tripHour = String.valueOf(selectedHour);
            }

            if (selectedMinute < 10) {
                tripMinute = 0 + String.valueOf(selectedMinute);
            } else {
                tripMinute = String.valueOf(selectedMinute);
            }

            etTime.setText(tripHour + ":" + tripMinute);
        }
    };

    // method to attach map to variable
    private void createMapView() {
        try {

            if (null == googleMap) {
                googleMap = ((MapFragment)
getChildFragmentManager().findFragmentById(R.id.mapView)).getMap();

                if (null == googleMap) {

Toast.makeText(getActivity().getApplicationContext(), "Error creating
map", Toast.LENGTH_SHORT).show();
                }

                // enabling current location
                if (ActivityCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                    return;
```

```
                }
                googleMap.setMyLocationEnabled(true);

                // zoom map on Rovaniemi
                LatLng userLocation = new LatLng(66.50394779999999,
25.7293905);

googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(userLocation
, 6));
            }

        } catch (NullPointerException exception) {
            Log.e("mapApp", exception.toString());
        }
    }

    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    public interface OnFragmentInteractionListener {
        void onFragmentInteraction(Uri uri);
    }
}
```

FragmentTripDetails

```
package ru.travellingtogether.travellingtogether.fragments;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;

import ru.travellingtogether.travellingtogether.R;
import ru.travellingtogether.travellingtogether.parsers.AddCommentBW;
import
ru.travellingtogether.travellingtogether.parsers.TripCommentsJSONParse
r;

public class FragmentTripDetails extends android.app.Fragment {

    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    // arrayList to show list to show list of trips
    ArrayList<HashMap<String, String>> mlist = new
ArrayList<HashMap<String,String>>();

    // declaration of variable for listView
    ListView list;

    // declaration of variable for tvs and et
    TextView detailsFrom, detailsTo, detailsDate, detailsStatus,
detailsName, detailsPhone;
    EditText etAddComment;

    // string for a given comment
    String commenttoadd;

    // strings for extractJson()
    private JSONArray comments = null;
```

```
    private static final String JSON_ARRAY ="result";
    private static final String JSON_USERNAME ="username";
    private static final String JSON_COMMENT ="comment";

    public FragmentTripDetails() {
        // Required empty public constructor
    }

    public static FragmentTripDetails newInstance(String param1,
String param2) {
        FragmentTripDetails fragment = new FragmentTripDetails();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_trip_details,
container, false);
        list = (ListView) v.findViewById(R.id.listviewComments);
        detailsFrom = (TextView) v.findViewById(R.id.detailsFrom);
        detailsTo = (TextView) v.findViewById(R.id.detailsTo);
        detailsDate = (TextView) v.findViewById(R.id.detailsDate);
        detailsStatus = (TextView) v.findViewById(R.id.detailsStatus);
        detailsName = (TextView) v.findViewById(R.id.detailsName);
        detailsPhone = (TextView) v.findViewById(R.id.detailsPhone);
        etAddComment = (EditText) v.findViewById(R.id.etAddComment);

        // Add comment button
        Button btnAddComment = (Button)
v.findViewById(R.id.btnAddComment);
        btnAddComment.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                commenttoadd = etAddComment.getText().toString();

                if (!commenttoadd.equals("")) {
                    if (isOnline()) {
                        // send commend to database
                        AddCommentBW addCommentBW = new
AddCommentBW(FragmentTripDetails.this, getActivity());
                        addCommentBW.execute(FragmentTripList.tripid,
FragmentUser.username, commenttoadd);
                    } else {
                        Toast.makeText(getActivity(),
R.string.noInternet, Toast.LENGTH_SHORT).show();
                    }
                    hideKeyboard();
```

```
                } else {
                    Toast.makeText(getActivity(),
R.string.emptyComment, Toast.LENGTH_SHORT).show();
                }
            }
        });

        return v;
    }

    public void onResume() {
        super.onResume();
        if (FragmentUser.userStatus.equals("driver")) {
            ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.requestedTrip);
        }
        if (FragmentUser.userStatus.equals("passenger")) {
            ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.forthcomingTrip
);
        }

        // attaching values from FragmentTripList chosen item
        String date =
FragmentTripList.day+"."+FragmentTripList.month+"."+FragmentTripList.y
ear+", "+FragmentTripList.hour+":"+FragmentTripList.minute;
        String name = FragmentTripList.name+"
"+FragmentTripList.surname+" ("+FragmentTripList.username+")";
        detailsFrom.setText(FragmentTripList.from);
        detailsTo.setText(FragmentTripList.to);
        detailsDate.setText(date);
        detailsName.setText(name);
        detailsPhone.setText(FragmentTripList.phone);

        if (FragmentUser.userStatus.equals("driver")){
            detailsStatus.setText(R.string.tripStatusPassenger);
        } else {
            detailsStatus.setText(R.string.tripStatusDriver);
        }

        // clear comments list and extractJson each time fragment
resumed
        mlist.clear();
        extractCommentJSON();
    }

    // internet connection state check
    public boolean isOnline() {
        ConnectivityManager cm = (ConnectivityManager)
getActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if (netInfo != null && netInfo.isConnectedOrConnecting()) {
            return true;
        }
        return false;
    }

    // hide keyboard on comment added
    private void hideKeyboard() {
        // Check if no view has focus
        View view = getActivity().getCurrentFocus();
```

```
        if (view != null) {
            InputMethodManager inputManager = (InputMethodManager)
getActivity().getSystemService(Context.INPUT_METHOD_SERVICE);

inputManager.hideSoftInputFromWindow(view.getWindowToken(),
InputMethodManager.HIDE_NOT_ALWAYS);
        }
    }

    // add new comment to a listView
    public void addcommentBWPE(){
        HashMap<String, String> map = new HashMap<String, String>();
        map.put(JSON_USERNAME, FragmentUser.username);
        map.put(JSON_COMMENT, commenttoadd);
        mlist.add(map);

        ListAdapter adapter = new SimpleAdapter(getActivity(), mlist,
R.layout.fragment_trip_detail_row,
                new String[] {JSON_USERNAME, JSON_COMMENT},
                new int[]{R.id.tvCommentUsername,
R.id.tvCommentComment,});
        list.setAdapter(adapter);
        list.setSelection(comments.length());

        etAddComment.setText("");
    }

    // extracting CommentsList json string to array
    private void extractCommentJSON(){
        JSONObject jsonObject;
        try {
            jsonObject = new
JSONObject(TripCommentsJSONParser.jsonResult);
            comments = jsonObject.getJSONArray(JSON_ARRAY);

            for(int i=0; i<comments.length(); i++) {
                JSONObject comment = comments.getJSONObject(i);
                String username = comment.getString(JSON_USERNAME);
                String comm = comment.getString(JSON_COMMENT);

                HashMap<String, String> map = new HashMap<String,
String>();
                map.put(JSON_USERNAME, username);
                map.put(JSON_COMMENT, comm);
                mlist.add(map);

                // attach comment to a list
                ListAdapter adapter = new SimpleAdapter(getActivity(),
mlist, R.layout.fragment_trip_detail_row,
                        new String[] {JSON_USERNAME, JSON_COMMENT},
                        new int[]{R.id.tvCommentUsername,
R.id.tvCommentComment,});
                list.setAdapter(adapter);
            }

        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
```

```
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    public interface OnFragmentInteractionListener {
        void onFragmentInteraction(Uri uri);
    }
}
```