Ministry of Education of Republic of Moldova

Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Computer Science

Object Oriented Programming

Course Work

TaxiShmaxi APP

Authors:
Daniel Surdu

Supervisor:
Anastasia Șerșun

Chișinău - 2018

1 Introduction

1.1 About Android Studio

Android Studio is Android's official IDE. It is purpose-built for Android to accelerate your development and help you build the highest-quality apps for every Android device.

Instant Run

Android Studio's Instant Run feature pushes code and resource changes to your running app. It intelligently understands the changes and often delivers them without restarting your app or rebuilding your APK, so you can see the effects immediately.

Intelligent code editor

The code editor helps you write better code, work faster, and be more productive by offering advanced code completion, refactoring, and code analysis. As you type, Android Studio provides suggestions in a dropdown list. Simply press Tab to insert the code.

Fast and feature-rich emulator

The Android Emulator installs and starts your apps faster than a real device and allows you to prototype and test your app on various Android device configurations: phones, tablets, Android Wear, and Android TV devices. You can also simulate a variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input.

Code templates and sample apps

Android Studio includes project and code templates that make it easy to add well-established patterns such as a navigation drawer and view pager. You can start with a code template or even right-click an API in the editor and select Find Sample Code to search for examples. Moreover, you can import fully functional apps from GitHub, right from the Create Project screen.

Lintelligence

Android Studio provides a robust static analysis framework and includes over 280 different lint checks across the entirety of your app. Additionally, it provides several quick fixes that help you address issues in various categories, such as performance, security, and correctness, with a single click.

Testing tools and frameworks

Android Studio provides extensive tools to help you test your Android apps with JUnit 4 and functional UI test frameworks. With Espresso Test Recorder, you can generate UI test code by recording your interactions with the app on a device or emulator. You can run your tests on a device, an emulator, a continuous integration environment, or in Firebase Test Lab.

Robust and flexible build system

Android Studio offers build automation, dependency management, and customizable build configurations. You can configure your project to include local and hosted libraries, and define build variants that include different code and resources, and apply different code shrinking and app signing configurations.

Designed for teams

Android Studio integrates with version control tools, such as GitHub and Subversion, so you can keep your team in sync with project and build changes. The open source Gradle build system allows you to tailor the build to your environment and run on a continuous integration server such as Jenkins.

Optimized for all Android devices

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug.

C++ and NDK support

Android Studio fully supports editing C/C++ project files so you can quickly build JNI components in your app. The IDE provides syntax highlighting and refactoring for C/C++, and an LLDB-based debugger that allows you to simultaneously debug your Java and C/C++ code. The build tools can also execute your CMake and ndk-build scripts without any modification and then add the shared objects to your APK.

Firebase and Cloud integration

The Firebase Assistant helps you connect your app to Firebase and add services such as Analytics, Authentication, Notifications and more with step-by-step procedures right inside Android Studio. Built-in tools for Google Cloud Platform also help you integrate your Android app with services such as Google Cloud Endpoints and project modules specially-designed for Google App Engine.

Layout Editor

When working with XML layout files, Android Studio provides a drag-and-drop visual editor that makes it easier than ever to create a new layout. The Layout Editor was built in unison with the ConstraintLayout API, so you can quickly build a layout that adapts to different screen sizes by dragging views into place and then adding layout constraints with just a few clicks.

APK Analyzer

You can use the APK Analyzer to easily inspect the contents of your APK. It reveals the size of each component so you can identify ways to reduce the overall APK size. It also allows you preview packaged assets, inspect the DEX files to troubleshoot multidex issues, and compare the differences between two APKs.

Vector Asset Studio

Android Studio makes it easy to create a new image asset for every density size. With Vector Asset Studio, you can select from Google-provided material design icons or import an SVG or PSD file. Vector Asset Studio can also generate bitmap files for each screen density to support older versions of Android that don't support the Android vector drawable format.

Translations Editor

The Translations Editor gives you a single view of all of your translated resources, making it easy to change or add translations, and to find missing translations without opening each version of the strings.xml file. It even provides a link to order translation services.

2 Tools

2.1 HTTP requests

Request types that Volley supports:

a) StringRequest. Specify a URL and receive a raw string in response. See Setting Up a Request Queue for an example.

b) JsonObjectRequest and JsonArrayRequest (both subclasses of JsonRequest). Specify a URL and get a JSON object or array (respectively) in response.

Request JSON

Volley provides the following classes for JSON requests:

a) JsonArrayRequest—A request for retrieving a JSONArray response body at a given URL.

b) JsonObjectRequest—A request for retrieving a JSONObject response body at a given URL, allowing for an optional JSONObject to be passed in as part of the request body.

```
String url = "http://my-json-feed";

JsonObjectRequest jsonObjectRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {

    @Override
    public void onResponse(JSONObject response) {
        mTextView.setText("Response: " + response.toString());
    }
}, new Response.ErrorListener() {

    @Override
    public void onErrorResponse(VolleyError error) {
        // TODO: Handle error

    }
});

// Access the RequestQueue through your singleton class.
MySingleton.getInstance(this).addToRequestQueue(jsonObjectRequest);
```

2.2 MapView

public class MapView extends FrameLayout A View which displays a map (with data obtained from the Google Maps service). When focused, it will capture keypresses and touch gestures to move the map.

Users of this class must forward all the life cycle methods from the Activity or Fragment containing this view to the corresponding ones in this class. In particular, you must forward on the following methods:

a) onCreate(Bundle)

b) onStart()

c) onResume()

d) onPause()

e) onStop()

f) onDestroy()

g) onSaveInstanceState()

h) onLowMemory()

A GoogleMap must be acquired using getMapAsync(OnMapReadyCallback). The MapView automatically initializes the maps system and the view.

For a simpler method of displaying a Map use MapFragment (or SupportMapFragment) if you are looking to target earlier platforms.

Note: You are advised not to add children to this view.

Public Constructors public MapView (Context context)

public MapView (Context context, AttributeSet attrs)

public MapView (Context context, AttributeSet attrs, int defStyle)

public MapView (Context context, GoogleMapOptions options)

Public Methods

```
public void getMapAsync (OnMapReadyCallback callback)
    Returns a non-null instance of the GoogleMap, ready to be used.

public final void onCreate (Bundle savedInstanceState)
    You must call this method from the parent Activity/Fragment's
    corresponding method.

public final void onDestroy ()
    You must call this method from the parent Activity/Fragment's
```

```
    corresponding method.

public final void onEnterAmbient (Bundle ambientDetails)
    You must call this method from the parent WearableActivity's
    corresponding method.

public final void onExitAmbient ()
    You must call this method from the parent WearableActivity's
    corresponding method.

public final void onLowMemory ()
    You must call this method from the parent Activity/Fragment's
    corresponding method.

public final void onPause ()
    You must call this method from the parent Activity/Fragment's
    corresponding method.

public final void onResume ()
    You must call this method from the parent Activity/Fragment's
    corresponding method.

public final void onSaveInstanceState (Bundle outState)
    You must call this method from the parent Activity/Fragment's
    corresponding method.

    Provides a Bundle to store the state of the View before it
    gets destroyed. It can later be retrieved when
    onCreate(Bundle) is called again.

public final void onStart ()
    You must call this method from the parent Activity/Fragment's
    corresponding method.

public final void onStop ()
    You must call this method from the parent Activity/Fragment's
    corresponding method.
```

3 Implementation

3.1 Diagrams

The use-case diagram for the application looks like this:

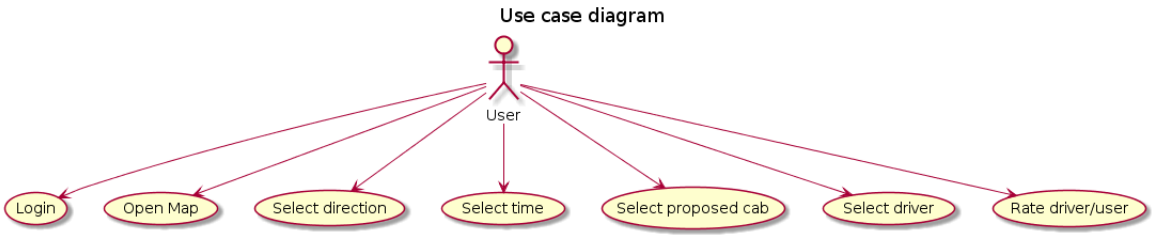

Figure 3.1– Use-case Diagram

Some Activity diagrams for some ideas of working:
Here is shown how select cab use-case is executed, and what alternatives you have:

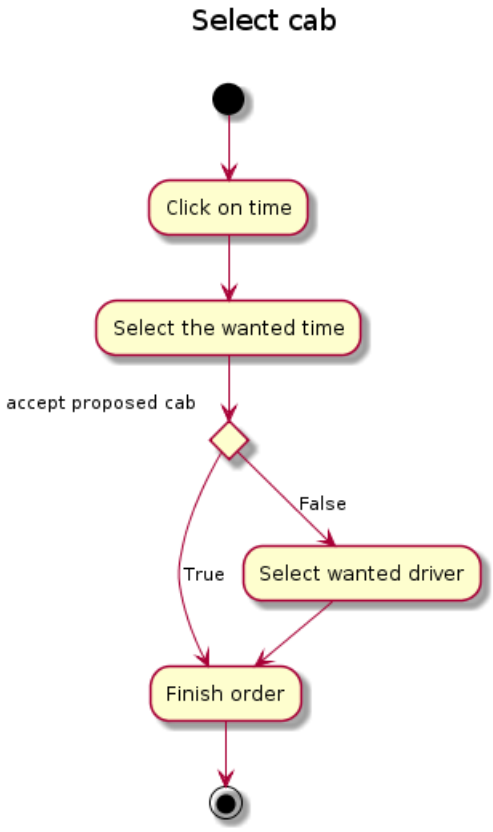

Figure 3.2– Select Cab

In this diagram is shown haw is chosen the direction:

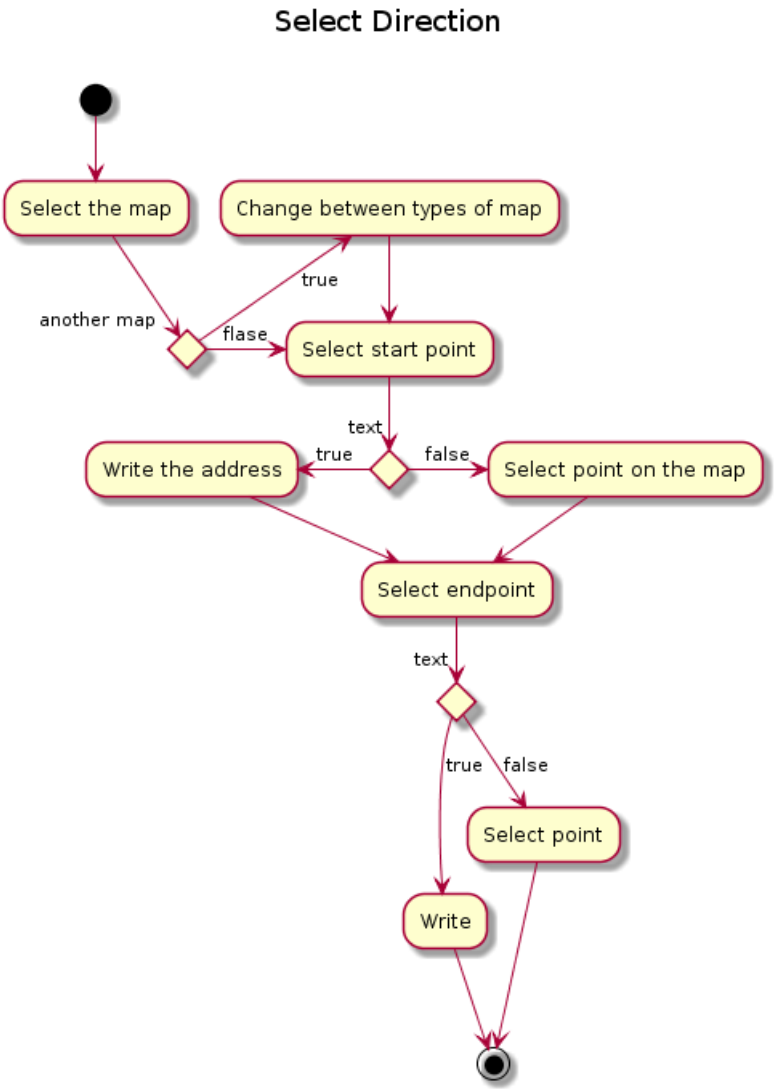

Figure 3.3– Select direction

To have a better idea of the application flow here is shown the sequence diagram:

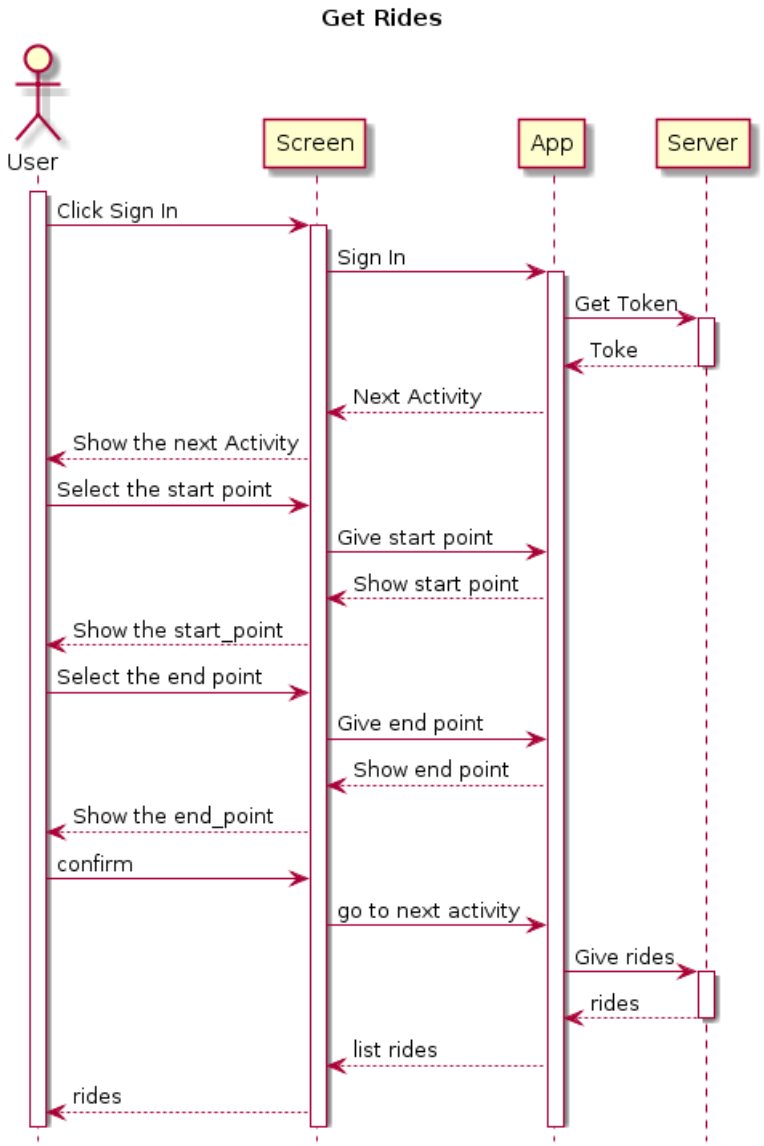

Figure 3.4– Sequence direction

Component diagram show the file locations in the application:

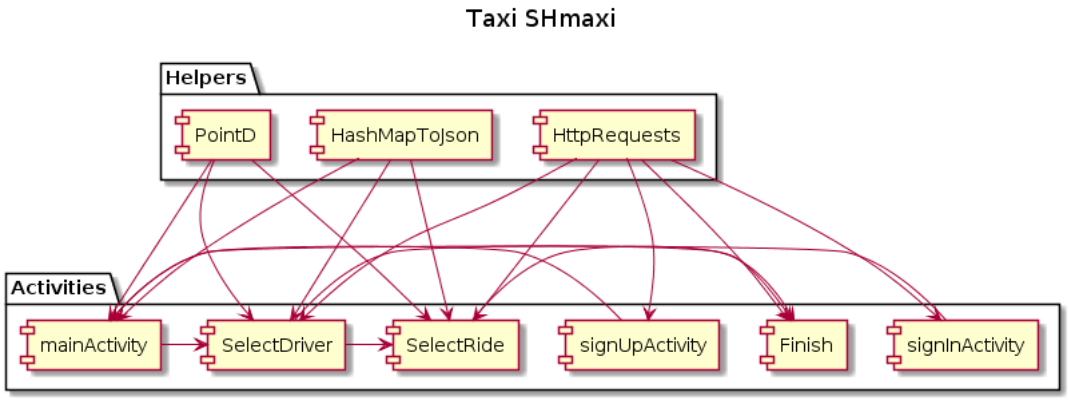

Figure 3.5– Activity Diagram

Deployment diagram shows us how are connected the component on the hard level:

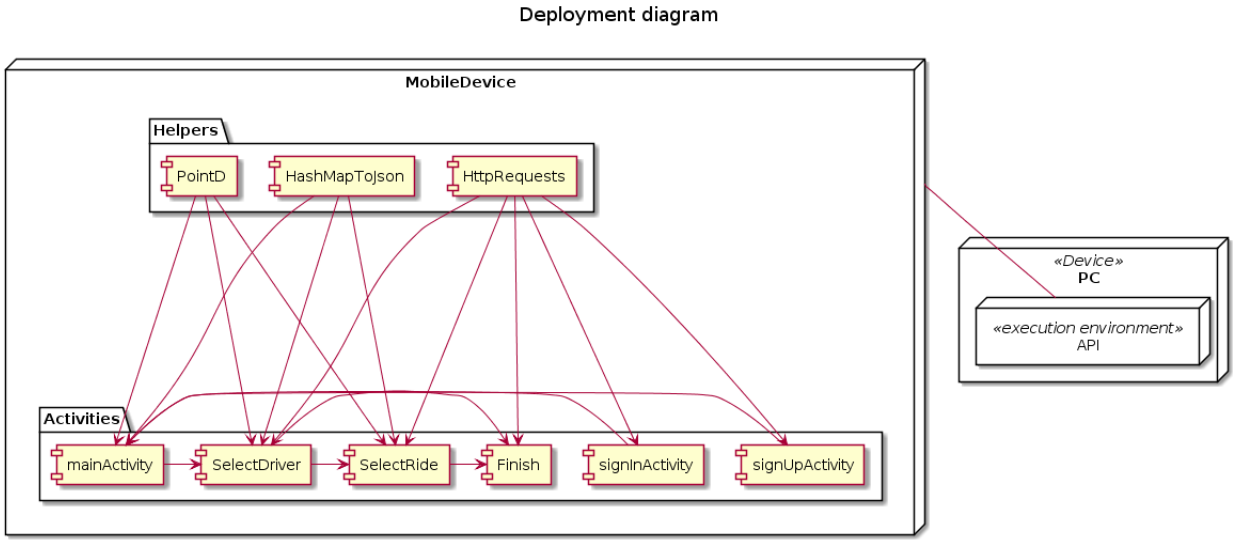

Figure 3.6– Deployment direction

3.2 Sign Up

The first page that you get when you enter in the application is Sign in

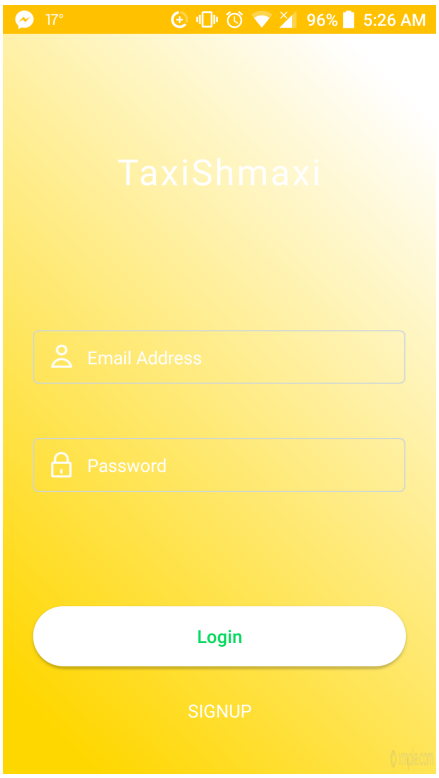

Figure 3.7– Sign IN

If you are first time in the application you will have to create a user, for this you will need an email and a password longer that 5 characters.

At the moment when SIGNUP button is clicked the program gets the data from the email, password and repeated password, first of all is checked if email is valid then if password is no less that 6 characters and after that is created a JSON post request with the body that contains user data.

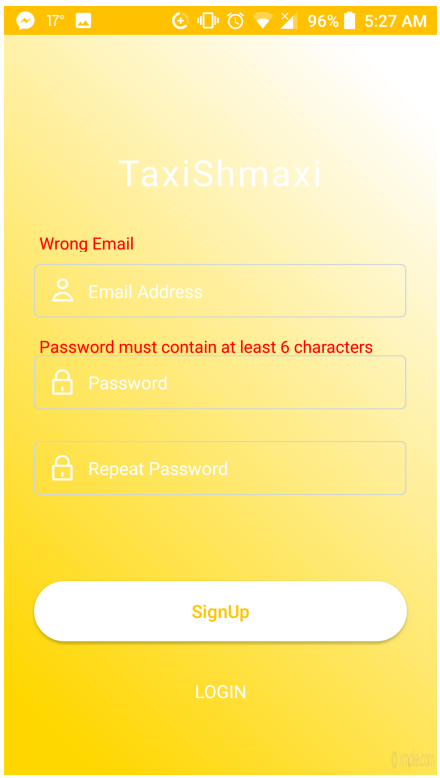

Figure 3.8– Sign UP errors

The same errors you get on the Sign In page:

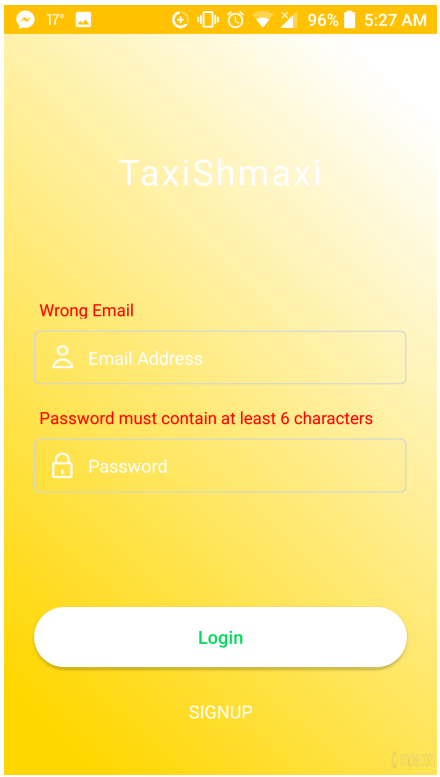

Figure 3.9– Sign In errors

The request is asynchronous event so we have to wait to get the results. We cannot wait in UI activity thus I created a new thread where I wait for data. When I get the data I open the next Activity where I will select the start and end points

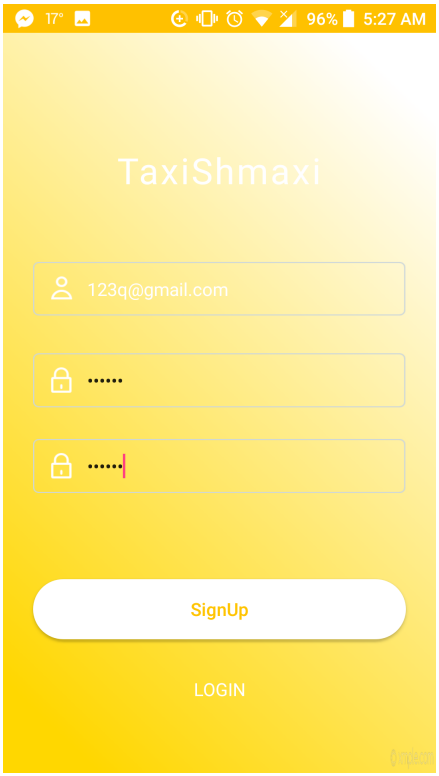

Figure 3.10– Sign UP entered data

After login or signup appears the map and 3 buttons:

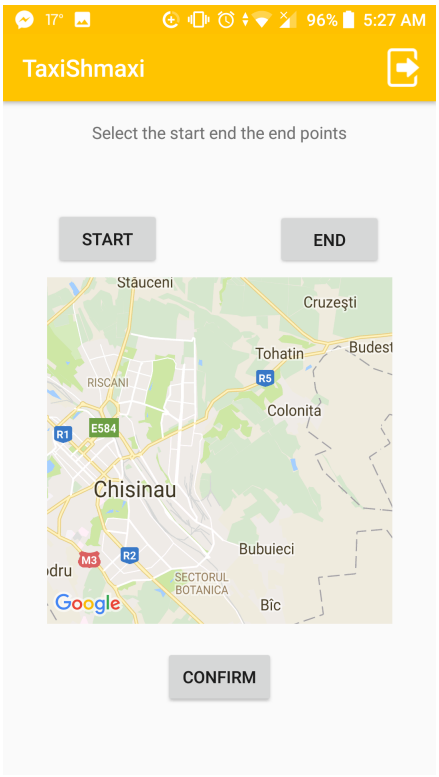

Figure 3.11– Map

To select the start and end points you have to click on the map for the first point and after clicking the button end select the end point on the map:

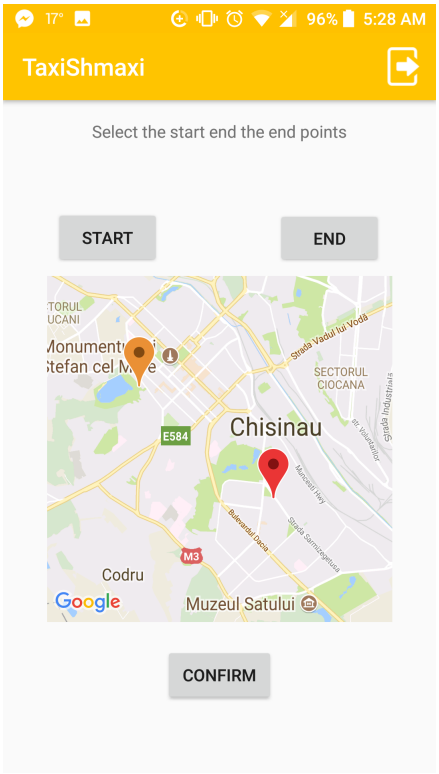

Figure 3.12– Map selected points

In case you didn't selected any points you will get errors like this:

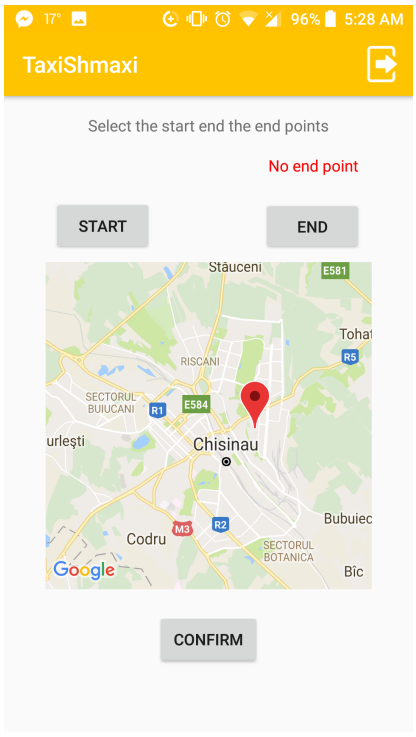

Figure 3.13– Error no end point

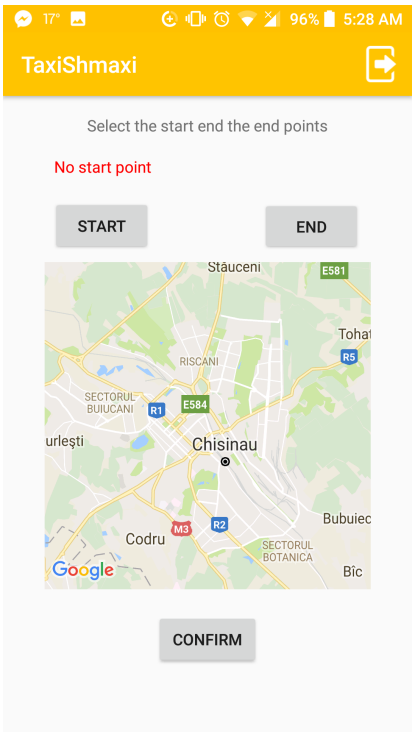

Figure 3.14– Error no start point

After confirmation you get on the next screen where you can ore cannot get the list of rides depending if there exist ore not:

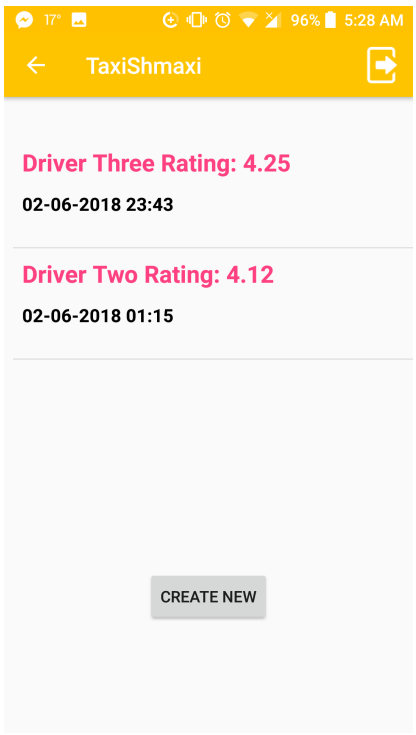

Figure 3.15– Proposed

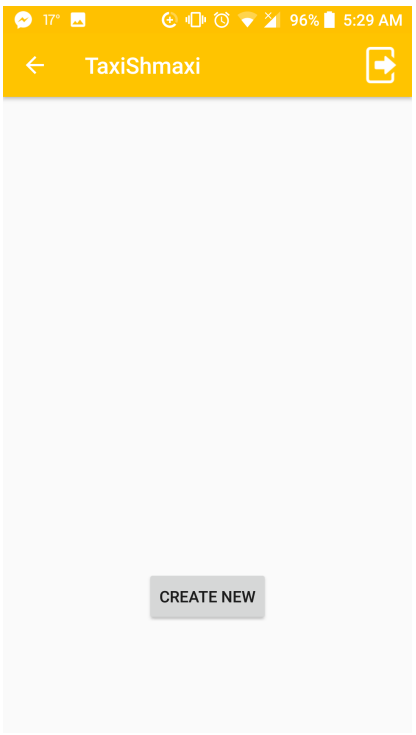

Figure 3.16– No proposed

If you select on of the rides you get to the end page that says that your order was confirmed:

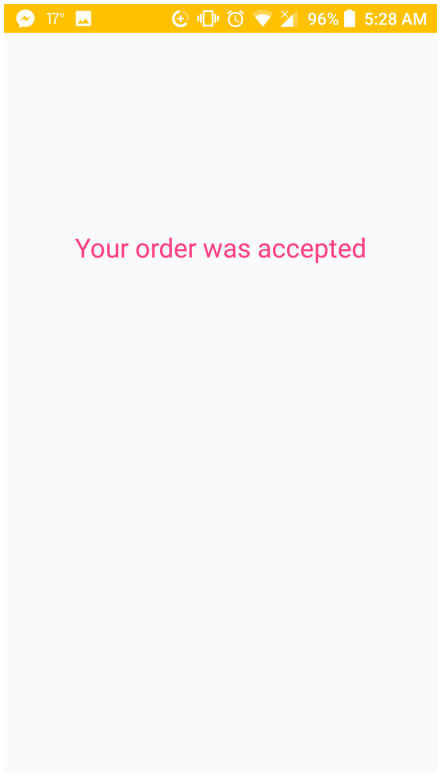

Figure 3.17– Accepted Order

If not you can create you oun ride and in this case will appear a pop-up time selector:

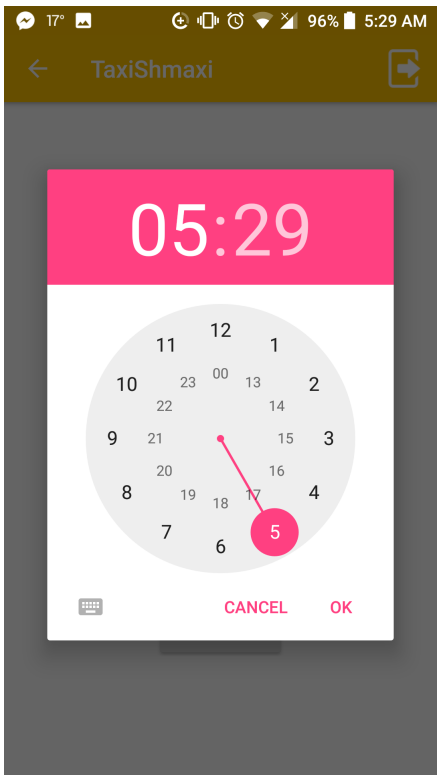

Figure 3.18– Select time

After you select a time you automatically go to the page where you can select a free driver.

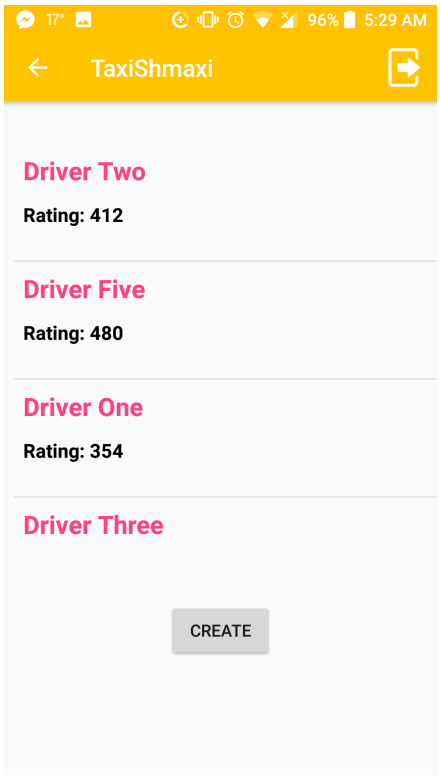

Figure 3.19– Free cabs

And finally if you select a free driver your order will be confirmed:

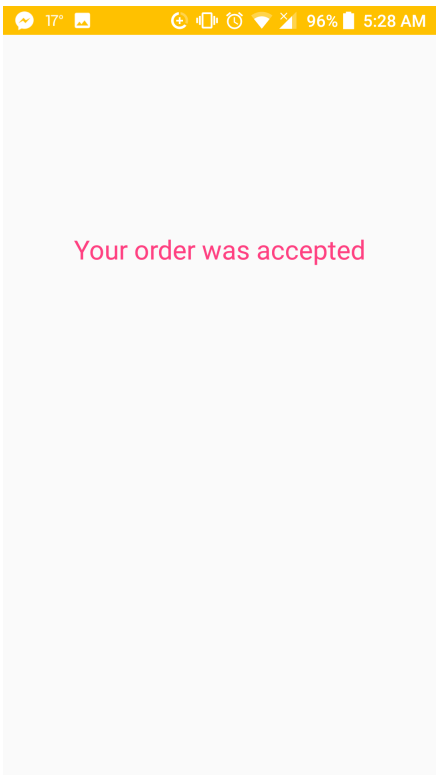

Figure 3.20– Accepted Order

Conclusion

During this course work, I had to create an mobile application that will improve the working of taxi. My application provides the possibility of sharing a cab. When you want to create a ride application offers all rides that are commanded in radios of 500m from your selected start_point. End point doesn't matter very much because app is created only for Chisinau and it's not such a big city. If aren't any created requests in 500m area, then you have the possibility to create a new request, when you create a new request application gives you all free drivers at the selected time and after selecting one you will get a message that your request was approved.

References

1 Android Studio Documentation *https://developer.android.com/studio/*

2 MapView
https://developers.google.com/android/reference/com/google/android/gms/maps/MapView