Mobile application for finding ATMs

Mario Lončar, Jasmin Abou Aldan, Marina Ivašić-Kos
Department of Informatics, University of Rijeka, Rijeka, Croatia
{mario.loncar, jasmin.abou.aldan, marinai}@uniri.hr

Abstract - This paper describes mobile application that gives Android users the ability to find ATMs quickly. Currently, the application offers locations of ATMs for various banks in Rijeka, Croatia. It also provides the possibility of finding the shortest path to the chosen ATM, as well as helpful tips where they can withdraw the money for free of charge. The idea for this application was born after realizing that many people were wasting time looking for a place to withdraw money. This application saves time to inhabitants as well as to visitors of the city.

Keywords: mobile application, Android, geo-locations, maps, navigation.

I. INTRODUCTION

Many people waste their time looking for places where they can withdraw the money. Positions of the closest ATM are sometimes unknown to the inhabitants, especially when they are in the part of the city that do not visit often. Finding the ATM becomes even more demanding task for tourists in unfamiliar city. They want to spend as much time as possible having fun and sightseeing instead of wandering to get money.

Touristic cities like Rijeka usually offer Wi-Fi hotspots at most frequently visited places so everyone can use internet connection. This opportunity can be used to download and use available applications on Google Play that facilitate the user to gather additional information about the destination. Below are applications that could be used for finding ATM together with their shortcomings.

AdriaGUIDE [13] Rijeka is a travel guide that provides most important information for the visitors, including ATM locator. However, after choosing a language on welcome screen, the application crashes so we could not see its functionality. Last update was on October 12, 2012.

Bankomati HR [14] offers ATM locations of all Croatian cities. On the other hand, it offers only eight banks and no navigation to chosen ATM. User interface looks out of date and there is no option for showing locations of all ATMs, instead you have to check each of them. On the bottom of the screen are ads that cannot be moved. Last update was on September 16, 2013.

Adria ATM [15] has intrusive ads on the top of action bar. When starting this application, map shows all ATMs of all banks on the list. If you want to check ATM locations of only one bank you have to uncheck banks on the list and then choose the bank you need which takes a lot of time. Application does not offer navigation, but only address of an ATM. It has implemented crowdsourcing, but crashes sometimes after adding a new ATM location. Last update was on August 2, 2013.

BankoMap [16] does not offer navigation nor banks to choose, every bank has the same default pin. It offers a small amount of ATMs. Last update was on October 12, 2010.

In order to help users finding an ATM quickly and easily, we developed Find My ATM application. It is a mobile application that offers ATM locations of all banks in Rijeka, useful advices on where the money can be withdrawn for free of charge and shortest path to each ATM.

II. APPLICATION OVERVIEW

Find My ATM is mobile application developed for Android platform using Android Studio [3]. Android Studio has recently replaced Eclipse with Android Development Tools (ADT) and become the official Java integrated development environment (IDE) for developing Android applications.

It is based on Java IDE developed by JetBrains and available as an IntelliJ IDEA open source community edition [8]. The IntelliJ IDE enables fast coding by giving smart and relevant suggestions, code completion, on-the-fly code analysis, project navigation and reliable refactoring tools. It has better layout previewing and is much easier to use and user friendly than Eclipse.

The key components of the Find My ATM application are working with maps, determining the geo-location of the user and computing the shortest path to the selected ATM. Therefore, the Android package that includes map libraries was added to the Android software development kit (SDK).

A. Android mapping libraries

We have considered several mapping libraries available for Android applications [1]: Google Maps API [2], ArcGIS Runtime API [9], OSMDroid API [10] and MapsForge [11].

Google Maps Android API is the default mapView for Android, and uses the data from Google Maps. It allows presentation of maps as satellite, terrain, and hybrid as well as Street View. Vector-based tiles of Google map can be used in offline mode. Additionally, map tiles can be customized adding information and markers onto the map to indicate specific points for an application. Customized maps can be shown in online mode.
ArcGIS runtime API shows data in offline mode, by storing tiles in the TIFF, PNG or JPG storage format, or by using tile packages. Users can access high-quality geospatial data and explore raster data and aerial imagery through Image GeoServices. OSMDroid is also library of raster maps and provides using tiles for off-line access.

MapsForge is library for rendering OSM (Open Street Maps) data on the fly. It has a special optimized tile format, and good rendering performance. Nutiteq library is open source if it is used with OpenStreetMaps. It has many unique features like support for 3D, custom Map APIs, offline routing and maps in any projection.

After testing different mapping APIs, we have chosen Google Maps API. The main reason was the familiarity of users to Google Map interface and functionalities since it is preinstalled on every Android device.

B. Set up of Google Maps in application

The Google Maps Android API uses OpenGL ES version 2 to render the map. If OpenGL ES version 2 is not installed, map will not appear. It is necessary to add the following in AndroidManifest.xml:

```xml
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true"/>
```

This notifies external services of the requirement. In particular, it has the effect of preventing Google Play Store from displaying your app on devices that does not support OpenGL ES version 2.

To use Google Maps in an application you need to create a valid Google Maps API key. The key can be used with any application that calls Google Maps API, and supports unlimited number of users. To obtain the key one should register to the Google APIs Console providing the application signature key given to application during development and the application package name. It has many unique features like support for 3D, custom Map APIs, offline routing and maps in any projection.

After testing different mapping APIs, we have chosen Google Maps API. The main reason was the familiarity of users to Google Map interface and functionalities since it is preinstalled on every Android device.

B. Set up of Google Maps in application

The Google Maps Android API uses OpenGL ES version 2 to render the map. If OpenGL ES version 2 is not installed, map will not appear. It is necessary to add the following in AndroidManifest.xml:

```xml
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true"/>
```

This notifies external services of the requirement. In particular, it has the effect of preventing Google Play Store from displaying your app on devices that does not support OpenGL ES version 2.

To use Google Maps in an application you need to create a valid Google Maps API key. The key can be used with any application that calls Google Maps API, and supports unlimited number of users. To obtain the key one should register to the Google APIs Console providing the application signature key given to application during development and the application package name. Then, a valid API key must be entered in AndroidManifest.xml file. Later it will be necessary to register application via its package in this Play Store console together with the SHA-1 fingerprint of signature key.

C. User interface

Android Studio offers advanced layout editor that allows you to drag-and-drop widgets into your layout and preview layout while editing the XML. In general, the XML vocabulary for declaring user interface elements closely follows the structure and naming of the classes and methods, where elements names correspond to class names and attribute names correspond to methods. This application has two XML layouts. First is for the main activity, which includes Google Maps fragment and listview for banks. Second XML layout is consisted of one textview that presents one row in listview.

An Android application typically has several activities. Each activity displays a user interface that allows the user to perform a specific task such as to select an item or view a map. To perform interactions between application components or different applications, an Intent should be used. Intent is abstract description of an operation to be performed. It is most often used for launching of activities.

Our application is launched by passing a method `startActivity()` to the system that opens a single screen with user interface showing all banks that have ATMs in Rijeka, Fig. 1. To define the list of all banks in Rijeka, data available on the website [12] was used. For locations of each ATM, geo-coordinates posted on the official documents of each of the banks were used.

![Figure 1. Navigation drawer displays available banks in Rijeka](image)

Data about banks and their ATM coordinates and addresses is stored in JSON format database on Parse.com [6]. Parse.com offers backend services for the developers as well as SDK for a multitude of platforms. It allows accessing data through objects. When you create a new App in Parse, you can create a new class and then import data to that class. Imports can be from CSV, XLS, TXT and JSON files, or you can manually create a table in each class by defining the columns and types of the stored data.

The important feature of Parse.com is the subsequent implementation of crowdsourcing. In that way, users can enter position and the address of new ATM on their own, as well as report possible wrong coordinates or address of an ATM. In order to use Parse classes and methods, a developer has to include Parse SDK files to Android Studio project within the folder libs. The following code directs `build.gradle` to look for appropriate libraries in the folder libs:

```
dependencies: {
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

To connect an application with Parse Core, in `onCreate()` method, a `Parse.initialize()` method is embedded that takes context, application ID and client key as parameters. If `ParseAnalytics().trackAppOpened()` and
ParseAnonymousUtils.logIn() methods are implemented in application, developer can keep track and get daily reports about usage of the application.

The main advantage of using Parse.com is that the developer does not have to think about the infrastructure, because the team responsible for Parse takes care of it. By using Parse, one can easily upgrade a database that is immediately available for use without entering the code of the application.

List of 20 banks is stored in Activity object in string array called banks_array:

```java
mBankNames = getResources()
    .getStringArray(com.mario.findmyatm.R.array.banks_array);
```

To provide access to the data items that should be presented and to enable view for each item, an Adapter object is used. An Adapter object acts as a bridge between types of available displays of data defined in AdapterView object and the data that will be used for a selected view. In this application ArrayAdapter object is used and backed by an array of arbitrary mBankNames objects. The ArrayAdapter class expects that provided resource id reference a single TextView in the larger layout resource. When TextView is referenced, it is filled with each object in the mBankNames array.

```java
mDrawerList = (ListView) findViewById
    (com.mario.findmyatm.R.id.left_drawer);

mDrawerList.setAdapter(new ArrayAdapter<String>
    (this, com.mario.findmyatm.R.layout.drawer_list_item,
    mBankNames));
```

D. Implementation and Customization of Google Maps

If the Internet connection is available, user can choose one bank among 20 of them on the list for which wants to see the ATM locations. In addition, an option that displays ATMs of all banks on the list is available too. To perform these operations the system uses the Intent that identifies and starts the appropriate service with method SetUpMapIfNeeded. The Intent can be also used to start an activity that is contained in a separate application or to communicate with a background service using method bindService(Intent, ServiceConnection, int).

A method SetUpMapIfNeeded() sets up the map if the Google Play services APK is correctly installed and the map has not already been instantiated. If the map is not installed, a prompt for the user is shown to install/update the Google Play services APK on his device. A user can return to this fragment activity after following the prompt and correctly installing/updating the Google Play services. Since the fragment activity may not have been completely destroyed during this process, onCreate() method may not be called again so we should call this method in onResume() to guarantee that it will be called.

If the map is installed properly, data taken from Parse Core [7] is displayed at map in the form of pins (Fig. 2), each time the user selects a bank item in the list view.

```java
for (String s : myArray) {
    String[] s2 = s.split(",");
    String address = s2[0];
    Float lat = Float.parseFloat(s2[1]);
    Float lng = Float.parseFloat(s2[2]);
    LatLng coo = new LatLng(lat, lng);
}
```

Instead of standard Google Maps pins, we used custom pins, so that each bank uses specific pin as its logo (Fig. 2). Pins are 32 x 32 .png format images. After selecting a bank, action bar changes title with setTitle() method according to the bank we have chosen:

```java
mTitle = title;
getSupportActionBar().setTitle(mTitle);
```

E. Implementation of Navigation Directions

To be able to provide information to the user about the closest ATM, the current location of the user should be determined.

One has to allow access to the location information in the settings in order to use My Location. Otherwise, an Alert, which leads to Location settings, will pop up as shown on the Fig. 3.
To check if Location access is enabled we have used the code snippet:

```java
if(!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) &&
!locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER))
    //alert shows up
```

Also an Internet connection is required for using the application. If a user does not have Internet connection, an Alert which leads to Internet settings will appear (Fig. 4). We used methods from Connectivity Manager class to check whether Network Connection is available or not:

```java
connectivity(){
    if(isNetworkAvailable()){
        //set up map
    }else{
        //alert shows up
    }
}
```

When the location information is enabled and network connection is available, the button with label My Location appears in the upper right corner and shows the location of the user (Fig. 5).

Navigation options appear when one of the ATMs is selected. The navigation to a specific ATM depends on the chosen means of transport. All functions from Google Maps API are implemented so the user can select the navigated to a specific ATM by car or by public transport or on foot (Fig. 6). For the selected navigation type, several different routes are offered and time necessary to reach the selected ATM is calculated. In addition, voice control can be used to control Navigation.

Furthermore, an application offers all sorts of information available for the selected ATM and advices on where you can withdraw your money free of charge (Fig. 7). Advices appear in a Dialog box that can display one, two or three buttons. Which button to implement depends on dialog’s action. Method `setPositiveButton()` is used when you should accept and continue with the action. For canceling the action method `setNegativeButton()` is used. Method `setNeutralButton()` is used when the user may not want to proceed with the action, but does not necessarily want to cancel. It is possible to add only one of each button type to an
AlertDialog. To display information to the user in a dialog box we have used the `setMessage()` method.

Find My ATM runs on android 4.0 or higher which covers 87% of Android devices [4]. Using lower Android API versions may affect in losing core feature set. Application requires multiple permissions defined in `AndroidManifest.xml` (Fig. 8).

We have used Genymotion [5] emulator for testing the application as well as physical devices. Genymotion is a fast third-party emulator that can be used instead of the default Android emulator that turned to be very slow even after installing Intel Hardware Accelerated Execution Manager.

III. CONCLUSION

We have developed the Android mobile application Find My ATM that enables finding of ATM locations quickly. The application also provides the possibility of finding the shortest path to the chosen ATM, as well as helpful tips where money can be withdrawn free of charge. The application is currently available only for various banks in Rijeka. However, an expansion on other Croatian cities is expected.

Inhabitants find this application very useful. Foreign tourists does not have an internet connection in Croatia, but major Croatian cities offer Wi-Fi hotspots on frequently visited places, so in future we will try to make an arrangement with tourist office for a free promotion of this application.

Currently, Find My ATM is in version 1.1.1 available for Android. It will be available for iOS users soon. In addition, voice search on Croatian language will be implemented so ATMs will be even easier to find. Parse.com offers an easy way to implement local data store, so we are planning to add this feature, as well as offline maps, in version 1.2. Other upgrading of the application will be made according to the user’s needs and suggestions.

REFERENCES

[1] gis.stackexchange.com; questions; Geographic Information Systems; http://gis.stackexchange.com/questions; (Accessed 2014-12)