7th International Scientific Conference
Technics and Informatics in Education
Faculty of Technical Sciences, Čačak, Serbia, 25-27th May 2018

Session 3: Engineering Education and Practice

UDC: 004.9

Creating an Android Weather Forecast Application in the Android Studio

Slobodan Aleksandrov[1*], Saša Vulović [2]
[1] College of Applied Mechanical and Technical Engineering Trstenik, Serbia
[2] Webelinx, Niš, Srbija
[*] slobodan.aleksandrov@vtmsts.edu.rs

Abstract: *A large number of requests that can be made on a computer can now be realized on smartphones or tablets. Because of their high hardware performance, mobility and low cost, smart mobile devices take on the primacy of classical computers in many spheres of life. The smart mobile software market is rapidly increasing, and the need for experts in this field is enormous. In this paper an analysis of operating systems of mobile devices was performed, the most commonly used development environment for application programming, and the process of development of weather forecast application on the Android platform is shown.*

Keywords: *Android; Java; smartphones; application*

1. INTRODUCTION

The development and application of information and communication technologies (ICT) brings major changes in all segments of the society. Expansion of the development of smart mobile devices at the beginning of the 21st century has opened a new large software market, ranging from the development of operating systems to the development of applications for various applications. The leader in the mobile operating system market is the Android platform with 74.23%, followed by the iOS platform with 20.84%, while all other mobile operating systems have a negligible small market share ([1], March 2018). Various development environments are used for programming mobile applications: Firebase, iOS SDK, Visual Studio, OutSystems, Xcode, Fabric, Android Studio and others. It is very important to define basic knowledge and modern software packages necessary for the development of software in the field of mobile devices. As the changes in Information Technology (IT) are very fast, it is necessary that the school system at all levels is flexible and modular, so it can quickly respond to the needs of the economy and society.

2. DEVELOPING A MOBILE ANDROID APPLICATION FOR WEATHER FORECAST

This paper presents the process of developing a mobile android application for the weather forecast. To create this application, the development environment Android Studio, as well as the Java programming language, was used. In addition to Java, starting with Android Studio 3.0.0, the Kotlin programming language is officially supported. Within this work, Android Studio 3.0.1 and Java programming language was used. When creating each application, it is necessary to create a new project first. Creating a new project takes place in a few simple steps. When launching Android Studio, we should select the "Start a new Android Studio project" option and assign the name for project, in this case it will be "WeatherApp" (Figure 1).

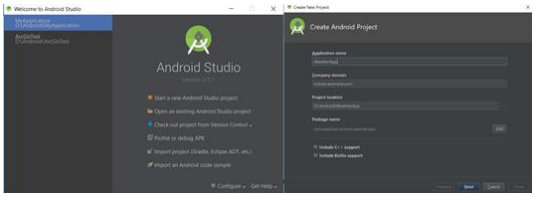

Figure 1. *Development enviroment Android Studio 3.0.1*

In the next step, we choose for which type of devices the application is going to be made: mobile smartphones and tablets, smart watches or televisions, as well as versions of the android operating system (Figure 2). In the final step, there is a possibility that the new project has pre-defined dedicated screens, such as the Google Maps screen, the Login screen, the Settings screen, and etc.

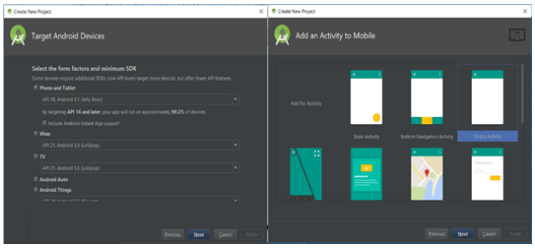

Figure 2. *Device and android version selection*

If no pre-defined screens are required, we select "Empty Activity" option and assign name for the activity, after which the process of creating a new project is completed. Before the development of application, it is necessary to get acquainted with the basic components that are used when creating android applications. These are "Activity", "Service", "Broadcast Receiver" and "Content Provider" [2]. Activity is one screen of the application. It consists of two part, one is the xml file, which represents the user interface, a screen that is visible to the user. The second part is a java class that responds to events when a user interacts with the screen. Each multi-screen application generally has different activity for each screen. The service is similar to activity, with the difference that it does not have an xml file and serves to perform tasks in the background so that the performance of the application would be better. The selected service is executed on the main thread, but with certain classes such as "Handler" or "AsyncTask" the code executed in the service can be transferred to the background thread. The service can be used to play music in the background, download data from the server, etc. Broadcast receivers are used to reply to broadcast messages that are sent from other activities or the android system itself. For example, if an application needs to know if the phone's screen is turned on or off, it can register and listen to the messages sent by the system, in this particular case, the messages for "SCREEN_ON" and "SCREEN_OFF" in order to get the appropriate information. Content providers serve to obtain certain data and are mainly used to communicate with the database. For example, using the content provider, we can get SMS messages that are stored on the device or missed calls.

Another important element in creating android applications is knowledge of the life cycle of the activity. Each activity has its own life cycle and phases in that cycle. There are several stages in which an activity can be found and through the predefined methods it informs the programmer in which phase it is located. Figure 3 shows the life cycle of the activity.

The "onCreate()" method is called when activity is first created. In this method, the initialization of the objects being used is mainly carried out. The "onStart()" method is called when activity is first visible to the user. When the "onResume()" method is invoked, the user can interact with the application. When the application switches to the background, the method "onPaused()" is called. The "onStop()" method is called when the application is no longer visible to the user after which either "onDestoy()" or "onRestart()" is called. If the application is "killed" by the user or system, the "onDestoy()" method is called, and if

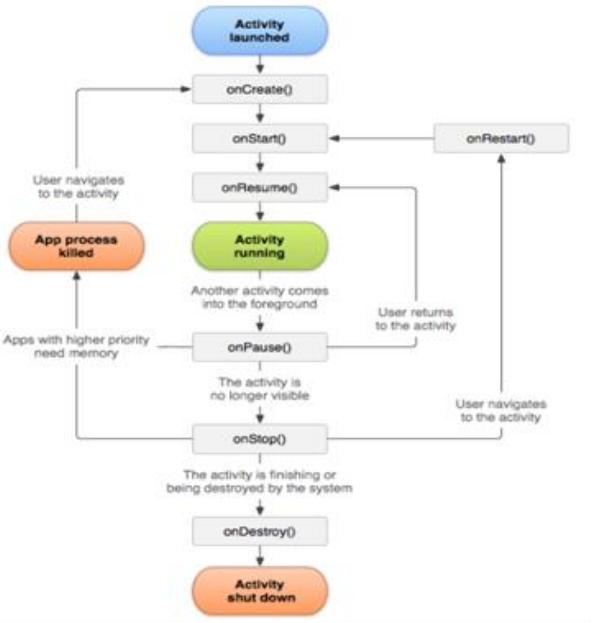

Figure 3. *The Activity Lifecycle [3]*

it is restarted, the "onRestart()" method is called after which the "onStart()" method is immediately invoked. To create a weather app, we need one of the services that provide information about the current weather forecast for a specific location. There are various services that provide this information, some are free, some are commercial, and there are also those that represent a combination of these two types, there are parts of the information that are the free, and also the parts that is paid. During the implementation of this application, the service "Dark Sky" was used, which can be found at https://darksky.net/forecast/40.7127,-74.0059/us12/en. The first step is to create an account on the site using email addresses and passwords. After creating an account, the key that is necessary for obtaining information about the weather is obtained. The site provides details on the information that can be obtained and how this service works. Data is obtained as a json file. Figure 4 shows the process of obtaining a key.

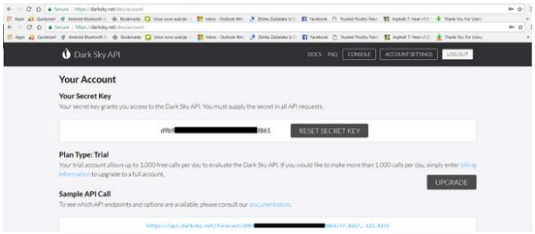

Figure 4. *Creating account for „Dark Sky" service*

When creating a new project, the activity for application is automatically generated and it is called "Main Activity". As already mentioned, it consists of a java class called "Main Activity" which inherits the "Activity" class and the xml file that will represent the user interface called

"activity_main.xml". The next step in developing this application is to create a user interface using the xml file and certain components such as "TextView", "ImageView", and components that allow the layout of components on the screen such as "RelativeLayout" and "LinearLayout". These components can be added via xml file or via java code in the java class. In this case, the components are added directly to the xml file. As the names themselves suggest, "TextView" is used to display text, and "ImageView" to display images. Each component serves to display certain information on the screen. From the above, it is necessary to add more "TextView" components that will be used to display the text and also weather information for the location for which we search weather forecasts, the time of the last update, a brief description of the weather forecast, the current temperature, as well as the humidity, pressure, wind speed and UV indexes. "ImageView" are used to display graphic elements on the interface, such as background images, graphics that represent the current weather situation and etc. The size and color of the text as well as the size of the images are also defined in the xml file. After adjusting and positioning the components on the screen, we need to connect the elements from the xml file with the java class, in order to be able to display the data that is obtained, and also we need to implement the logic for collecting data from the server. It is important to note that each component added to the xml file must have its unique "id" so that it can be connected to the java class. Figure 5 shows the created xml file.

Figure 5. *Creating xml file*

After creating the interface, it is necessary to connect the xml file with the java class, as well as the components that are used in the xml file. When creating a new activity, the connection process of java classes and xml files takes place automatically by calling the "setContentView (R.layout.activity_main)" method that is called in the "onCreate()" method. The next step is to connect a component from an xml file to a java class by creating an object of the appropriate type and calling the "findViewById (int id)" method. Connecting a component for displaying text can take the following form: TextView txtTemperature = findViewById (R.id.txtTemperature). It is important that each component in the xml file has its own unique id so the connection would be successful. The same principle applies to connecting other components, such as ImageView. After this step, it is possible to dynamically display text on the screen by simply calling the "setText (String s)" method using the TextView object. The "setImageResource (Resourse id)" or "setImageBitmap (Bitmap bm)" can be used to set up images using the ImageView object.

In order to display the desired textual and graphical data on the user interface, it is necessary to load these data from the service on which the user account is created. Information about current temperature, brief description of current weather conditions, pressure, humidity, wind speed and direction data are required. To obtain these data, an auxiliary class is created under the name "MyWeather". This class will contain all the information that is needed for display on the screen. The object of this class is initialized when the application receives a response from server. Before communicating with the server, it is necessary to determine the geographical length and width of the place where the user is located. To obtain these data, it is necessary to grant the appropriate permissions: "android.permission.ACCESS_FINE_LOCATION" and "android.permission.ACCESS_COARSE_LOCATION. Also, for the communication with the server, the" android.permission.INTERNET" permission is required. After the granting this permission, using the "LocationManager" we can detect the current geographic position of the user. The current location information is obtained in the pre-defined method "public void onLocationChanged (Location location)" that passes the "Location" object. Using "Location" object, we can get location information and then the request is sent to the server to get weather information. The request is sent via the auxiliary AsyncTaskHelper class that inherits the AsyncTask class. The reason for using the class that inherits the AsyncTask class is that in this way, the request will be sent to the server in a background thread, so the main thread of the application is not affected. This class has its own method "String doInBackground (Void ... voids)" within which the request is defined, and depending on the performance of this method, it returns the corresponding string as a return parameter. Sending requests is done by writing the following code:

```
URL url;
HttpURLConnection urlConnection = null;
try {
url = new URL(query);
urlConnection = (HttpURLConnection)
url.openConnection();
```

```
BufferedReader r = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()
));
StringBuilder total = new StringBuilder();
String line;
while ((line = r.readLine()) != null) {
total.append(line);
}
ParseJSON(total.toString(),myWeather);
return "OK";
}
catch (Exception e)
{
return "Error";
}
```

Using "HttpURLConnection" and "URL", a request is sent to a specific address, in this case it is the address provided by the Dark Sky service, after which desired data is received. Data is obtained in the form of an json file and using the method "ParseJSON(String data, MyWeather myWeather)" we preform data parsing that is received from server and at this point we initialize the "MyWeather" object. The following code is created for data parsing:

```
private void ParseJSON(String data,MyWeather
myWeather)
   {
      try {
         JSONObject jsonObject = new
JSONObject(data);
         JSONObject currentWeather =
jsonObject.getJSONObject("currently");
myWeather.setLastUpdate(currentWeather.getInt("t
ime"));
myWeather.setDescription(currentWeather.getStrin
g("summary"));
myWeather.setIcon(currentWeather.getString("icon"
));
myWeather.setTemperature(currentWeather.getDou
ble("temperature"));
myWeather.setHumidity(currentWeather.getDouble(
"humidity"));
myWeather.setPressure(currentWeather.getDouble(
"pressure"));
myWeather.setWindSpeed(currentWeather.getDoubl
e("windSpeed"));
myWeather.setUvIndex(currentWeather.getDouble("
uvIndex"));
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
```

If there is no error in communicating with the server or during data parsing, the return parameter is string "OK", and if an error has occurred, "Error" string is returned. AsyncTaskHelper has another important method, "onPostExecute (String s)" that is called when the "doInBackground" method is executed. This method is important because it reads the return parameter that is sent from "doInBackground" method. If there were no errors, the main activity is notified that new weather information has been received and the user interface is refreshed. In order to communicate with the activity, an auxiliary interface has been created, which has two methods, "onResultSuccess (MyWeather myWeather)" which is called if "OK" is received from the "doInBackground" as a return parameter. In the event that the string "Error" is returned as the return parameter, the "onResultFailed ()" method is used to inform the main activity that an error has occurred. In the end, using "Populate(MyWeather myWeather)" method main activity displays the data on the user interface. The appearance of the user interface after successful acceptance of data is shown in Figure 6.

Figure 6. *User interface for weather application*

Below is a presentation of the "Populate" method, which displays the data:

```
private void Populate(MyWeather myWeather)
   {
txtCityName.setText(myWeather.getCityName());
txtLasUpdate.setText(myWeather.getLastUpdate());

txtDescription.setText(myWeather.getDescription())
;
txtTemperature.setText(myWeather.getTemperatur
e()                                              );
txtHumidity.setText(myWeather.getHumidity()      );
txtPressure.setText(myWeather.getPressure());
txtWindSpeed.setText(myWeather.getWindSpeed())
;
txtUVIndex.setText(myWeather.getUVIndex());
imgWeatherIcon.setImageResource(getResources().
getIdentifier(myWeather.getIcon(),"drawable",getPa
ckageName()));   }
```

3. CONCLUSION

The development of a modern information society must be based on the application of new ICT technologies. The use of smart mobile devices in all segments of society requires the development of new mobile applications. The global software market in this area is growing at a tremendous pace, so the need for education of IT specialists is very high. This trend of the development of modern technologies enables the rapid development of the economy in the IT sector. Of great importance is the advancement of the educational system, which must be modular and dynamic, so that it can quickly implement new technologies into plans and programs in all of the levels of education.

REFERENCES

[1] http://gs.statcounter.com/os-market-share/mobile/worldwide (15.03.2018.)

[2] https://www.tutorialspoint.com/android/android_application_components.htm (16.03.2018.)

[3] https://developer.android.com/guide/components/activities/activity-lifecycle.html (16.03.2018.)