MINISTERUL EDUCAŢIEI REPUBLICII MOLDOVA

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Facultatea „Calculatoare, Informatică şi Microelectronică”

FILIERA ANGLOFONĂ

RAPORT

Lucrare de laborator nr. 1

la Programarea Aplicațiilor Mobile

A efectuat:

st. gr. FAF-151 Bîzdîga Stanislav

A verificat:

prof.univ. Sergiu Ciudin

Chișinău-2017

Laboratory Work 1

Subject: UI View Model

Objectives: Develop an app on Android using Android Studio.

Purpose of work:
Present an app that runs on a device or emulator, that will contain on its interface the following elements:

- 4 Buttons (that will execute the tasks below)
- 1 TextBox (for input)
- 2 Radio butoane (for front and back camera)

Tasks:

Use UI components for the following tasks:

1. Create **push notification(toast)** on the device screen, that will appear in **10s**.
2. Use internal device browser for initializing a **Google Search** with the **key-word** introduced in the **TextBox**.
3. Run front or back **camera event** due the selection of either 2 radio buttons.
4. Treat the **photo capture event** executed with either of cameras, then show the picture in another **activity**.
5. It's a bonus point for any additional feature.

Introduction

The following laboratory work is extremely important due to the fact, that it is the first mobile application development experience. It is supposed to set a base for self initialization into the world of developing apps.

In this project, it is required to be researched upon the UI elements, layouts, activities, intents, notification toasts, camera handling, code execution delays, and possibly more.

The amount of tasks is small but each has its own special category of things to be learnt.

Anything that's challenging, is what helps improve the skill significantly.

Short Theory

Intent

The Android way of delegating actions to other applications is to invoke an Intent that describes what you want done.An intent is an abstract description of an operation to be performed.

It can be used with startActivity to launch an Activity, broadcastIntent to send it to any interested BroadcastReceiver components, and startService(Internt) or bindService(Intent, ServiceConnection, int) to communicate with a background Service. An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.

Example:

```
Intent i = new Intent(this, SecondActivity.class);
startActivity(i);
```

Activity

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View). While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with windowIsFloating set) or embedded inside of another activity (using ActivityGroup).

Example:

```
public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);
    protected void onStart();
    protected void onRestart();
    protected void onResume();
    protected void onPause();
    protected void onStop();
    protected void onDestroy();
}
```

Toast

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.

Example:

```
Toast.makeText(context, text, duration).show();
```

Alternative:

```
Toast toast= Toast.makeText(getApplicationContext(), "Example", Toast.LENGTH_SHORT);
toast.setGravity(Gravity.TOP|Gravity.CENTER_HORIZONTAL, 0, 0);
toast.show();
```

Task Implementations and results

1. Create **push notification(toast)** on the device screen, that will appear in **10s**.

The following snippet contains the declaration of the toast and its placement setting:

```
final Toast toast = Toast.makeText(context, "10s passed!", Toast.LENGTH_SHORT);
toast.setGravity(Gravity.CENTER,0,0);
```

The following snippet contains the main idea of the delay, using a timer:

```
        new Timer().schedule(new TimerTask() {
            @Override
            public void run() {
                toast.show(); } //run this code snippet
        }, 10000);              //after 10 seconds
```

Results:

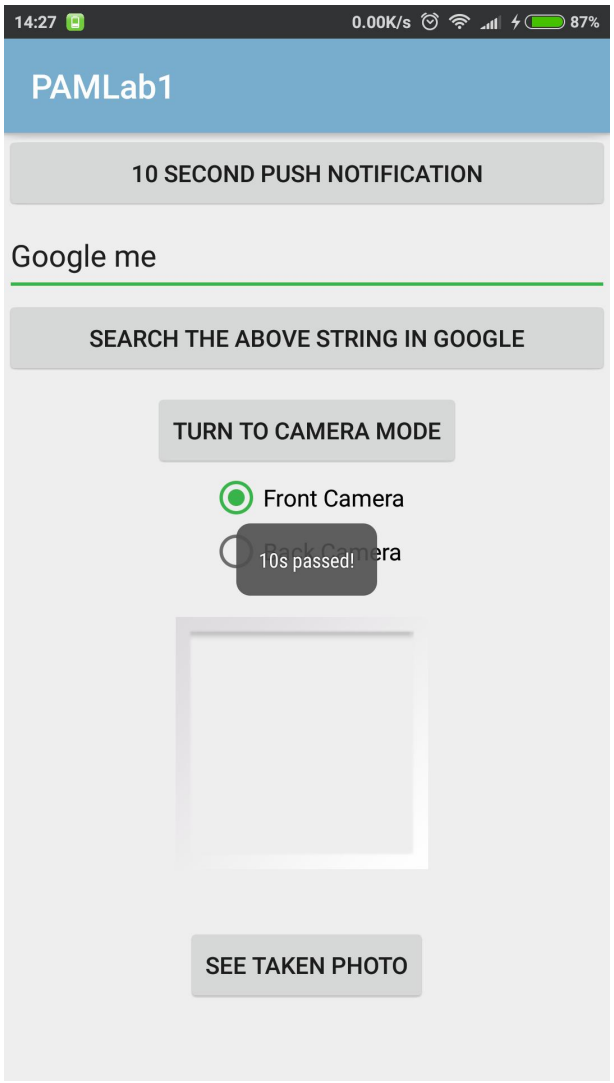

fig.1. Toast notification

Once clicked on the first button, after 10 seconds, the following toast pops up, notifying the user that the given amount of time has passed.

2. Use internal browser for initializing a **Google Search** with the **key-word** introduced in the **TextBox**.

The following snippet contains the implementation of the google search intent:

```
Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
String googledString = textToSearch.getText().toString();
intent.putExtra(SearchManager.QUERY, googledString);
startActivity(intent);
```

Note! It's required to have the editText UI element from the xml layout:

```
<EditText
    android:id="@+id/editTextSearch"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:ems="10"
    android:inputType="text"
    android:text="Google me"
    android:layout_below="@+id/button1"
    android:layout_alignParentStart="true" />
```

Stored in a variable (here: textToSearch):

```
final EditText textToSearch = (EditText) findViewById(R.id.editTextSearch);
```

Results:

First, the keyword is written in the editText area by the user. Then search button is clicked.

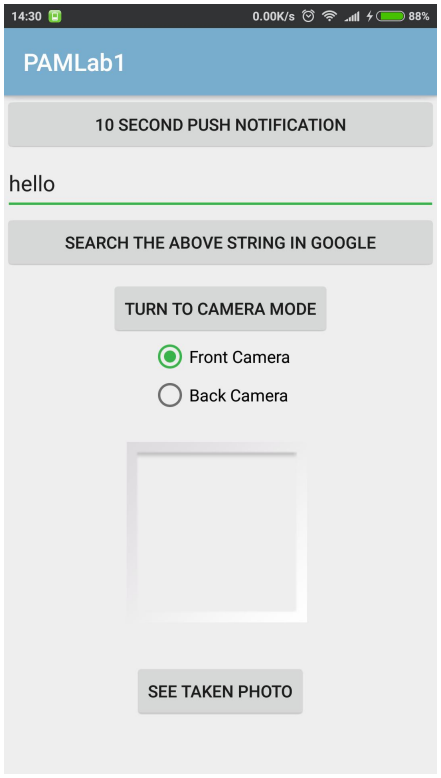

fig.2.1. Typing a new key-word

Next, the intent for a web search requests user select a browser to perform the search with.

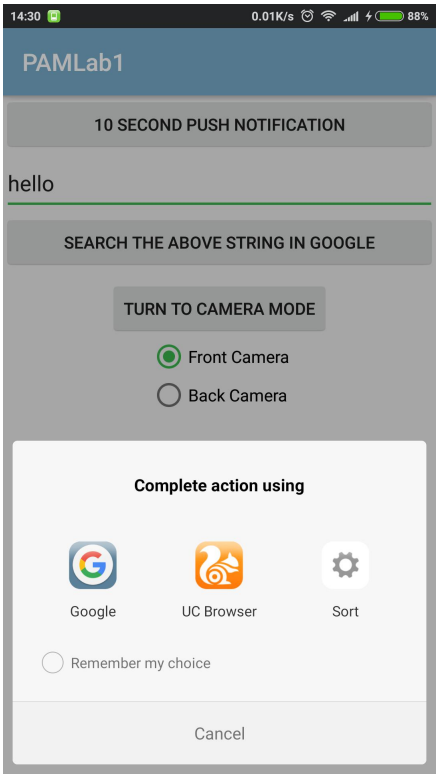

fig.2.2. The Intent

Lastly, the search is performed by the chosen web browser with the string from the app.

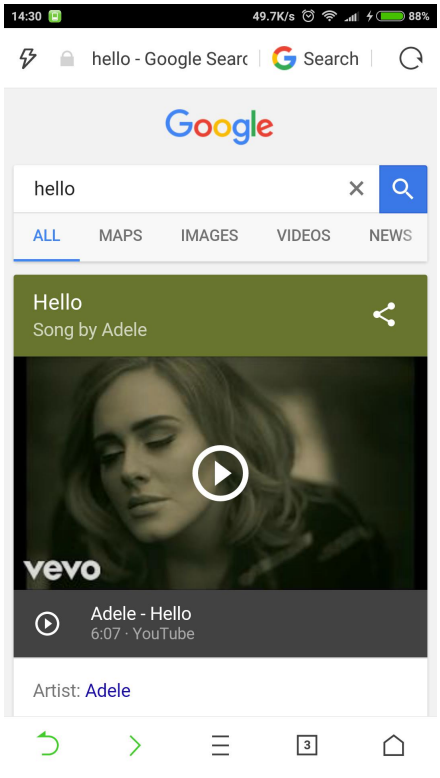

fig.2.3. Search performed

3. Run front or back **camera event** due the selection of either 2 radio buttons.

The following code snippet contains the key elements to the implementation of the camera front/back switching and the taking pictures intention:

```
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
takePictureIntent.putExtra("android.intent.extras.CAMERA_FACING", 1);  // or 0
startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
```

Results:

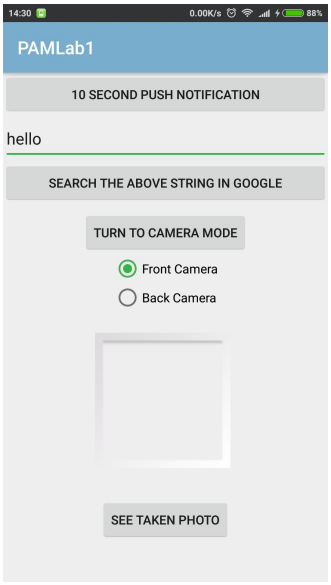

fig.3.1. Front Camera selected

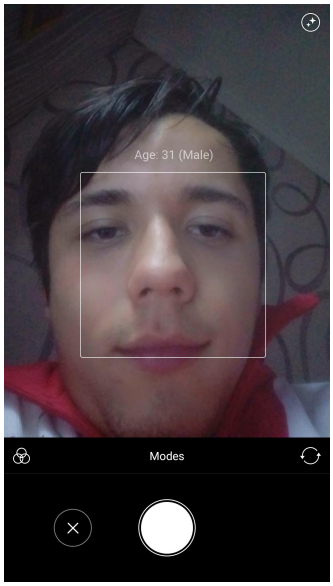

fig.3.2. Front Camera enabled

Front Camera:
Once front camera radio button is selected, anytime the button to enable camera mode is clicked, the camera app will run with the front camera by default.

It is done by putting 1 in CAMERA_FACING.

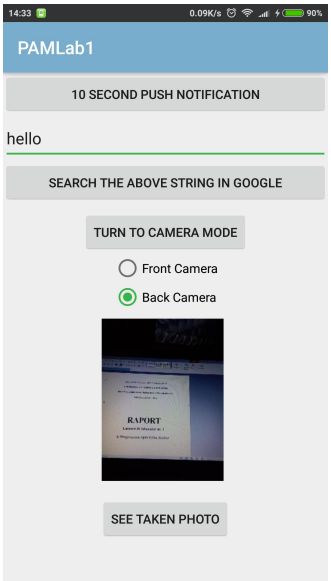

fig.3.3. Back Camera selected

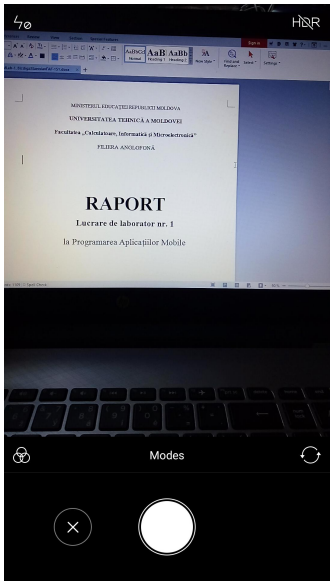

fig.3.4. Back Camera enabled

Back Camera:
Once back camera radio button is selected, anytime the button to enable camera mode is clicked, the camera app will run with the back camera by default.

It is done by putting 0 in CAMERA_FACING.

Note: The tiny thumbnail is an additional feature, that is explained in the item 5 of this chapter.

4. Treat the **photo capture event** executed with either of cameras, then show the picture in another **activity**.

The following snippet has the call to the second activity:

```
Intent intent = new Intent(MainActivity.this, Preview.class);
intent.putExtra("pic", imageBitmap);
startActivity(intent);
```

This is the second activity that shows the picture:

```
public class Preview extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_preview);

        ImageView imageFrame = (ImageView) findViewById(R.id.preview);
        Bitmap photo = (Bitmap) getIntent().getExtras().get("pic");

        imageFrame.setImageBitmap(photo);
    }
}
```

Results:

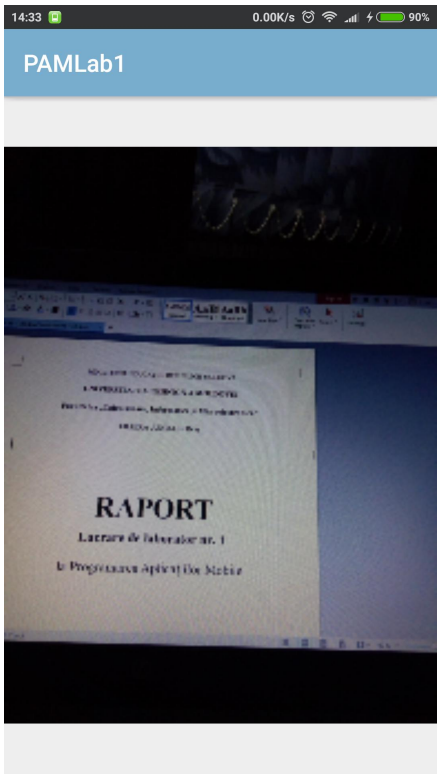

fig.4. Preview Photo Activity

Once a picture is taken (e.g. fig.3.4), the picture is stored in a bitmap variable, and once the user decides to click “see taken photo”, a new activity is started where the taken picture is shown.

5. Bonus point for any additional feature.

The code snippet below is performed once the activity for an image capture is successful:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        imageBitmap = (Bitmap) extras.get("data");
        photoTaken = true;
        ImageView photo = (ImageView) findViewById(R.id.photo);
        photo.setImageBitmap(imageBitmap);
    }
}
```

Results:

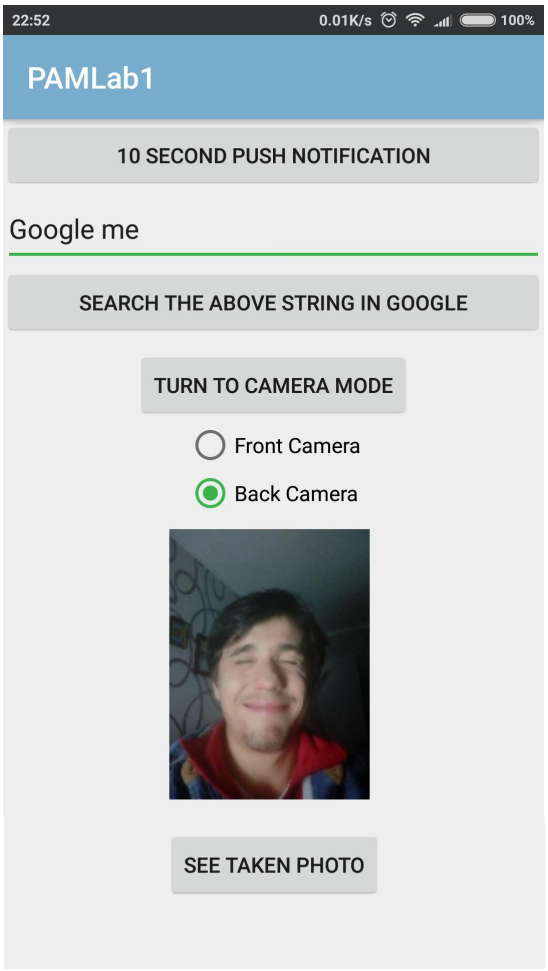

fig.5. Automatic thumbnail

Additional feature:

The additional feature here is that once a picture is taken, it is shown as a small thumbnail in the bottom area of the app. It is done automatically, thanks to the placement of the code in the onActivityResult( ) method, which is performed once the picture is taken, i.e. taking photos activity has beed completed successfully.

Also, if you decide clicking the “see taken photo” button before actually taking a photo, there’s a special case that pops a toast telling the user to take a photo first.

Conclusion

During this laboratory work, there was a lot of important moments of learning and problem solving. The whole project took a few days to complete - which is, personally, a stunning fact, as well as the fact that the progress was constant and linear. This shows, that the choice to develop on this very mobile platform was a successful one, and a well calculated decision.

A good experience gain was the one related to the layout file (.xml) - the file generally regarding the visual part of the application. The implementation of the look of the application through Android Studio drag-and-drop methods, was a huge help at the beginning, making it a intuitive, easy to grasp experience, leading to a further clear understanding of what each line of code in the layout file does.

Buttons, text views, and other UI elements are the building bricks to an application design, and it turns out that without them, the interaction between the user and the device is fairly impossible.

There are also other important concepts that were encountered in the current laboratory work.
An activity - which is a trivial element of the application, like an instance of a window that is meant to be interacted with, by the user, in a focused way - i.e. taking care of a single thing the user may do.
An intent - representing a operation execution intention, that is able to communicate with background services and send components.

The given laboratory work marks a new step towards the mastering of developing on a mobile platform.

Bibliography

- **Study Sources: Android Developers**
  https://developer.android.com/index.html

  Intent:
  https://developer.android.com/reference/android/content/Intent.html

  Activity:
  https://developer.android.com/reference/android/app/Activity.html

  Toast:
  https://developer.android.com/reference/android/widget/Toast.html

  Camera:
  https://developer.android.com/reference/android/hardware/camera2/package-summary.html

- **Handwritings from Mobile Applications Programming course** of Lector:
  prof.univ. I.Antohi.
  Chișinău: UTM
  2017