Mobile Applications for Business

Master SIA/SDBIS

Octavian Dospinescu

General topic

- photo camera ☺…smile, please…☺
- the possibility to “open” the mobile device’s photo camera and to preview images from the environment;
- the possibility to make an instant photo;
- the possibility to automatically save the picture taken by the camera;
- the possibility to “close” the device’s photo camera.

Main concepts

- AndroidManifest, rights and permissions;
- CameraManager;
- CameraCharacteristics;
- Output targets;
- CameraDevice;
- onOpened() callback;
- CaptureRequest;
- CaptureRequestSession.

Main concepts

- AndroidManifest, rights and permissions;
- CameraManager;
- CameraCharacteristics;
- Output targets;
- CameraDevice;
- onOpened() callback;
- CaptureRequest;
- CaptureRequestSession.

AndroidManifest, rights and permissions

```
<uses-permission android:name="android.permission.CAMERA"></uses-permission>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>

<uses-feature android:name="android.hardware.camera2.full"></uses-feature>
```

Main concepts

- CameraManager;
- CameraCharacteristics;
- Output targets;
- CameraDevice;
- onOpened() callback;
- CaptureRequest;
- CaptureRequestSession.

CameraManager

- Through it, we can iterate all the cameras that are available in the system.
- Every camera in the system has its own **cameraId**. We'll use this cameraId in order to obtain an instance of CameraDevice.

CameraManager – useful code

```
//cer permisiunea in mod explicit pentru Camera
ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);
```

```
CameraManager manager;
String cameraId;

manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);

//iau prima camera disponibila din dispozitiv
cameraId = manager.getCameraIdList()[0];
```

Main concepts

- CameraManager;
- CameraCharacteristics;
- Output targets;
- CameraDevice;
- onOpened() callback;
- CaptureRequest;
- CaptureRequestSession.

CameraCharacteristics

- Using the **cameraId**, we get can the properties of the specified camera device.
- These characteristics are specific to the camera (ex.: “is front or back camera”, “output resolutions supported”)

CameraCharacteristics – useful code

```
CameraCharacteristics caracteristiCamera = manager.getCameraCharacteristics(cameraId);

StreamConfigurationMap map = caracteristiCamera.get(caracteristiCamera.SCALER_STREAM_CONFIGURATION_MAP);
```

Main concepts

- CameraManager;
- CameraCharacteristics;
- Output targets;
- CameraDevice;
- onOpened() callback;
- CaptureRequest;
- CaptureRequestSession.

Output targets

- The camera image data should always go to somewhere.
- Use SurfaceView or **SurfaceTexture** for preview;
- Use ImageReader for picture;
- Use MediaRecorder for video recording.
- Also those classes have one common element behind: a **Surface**. The Surface is the physical place where the image is shown.

Main concepts

- CameraManager;
- CameraCharacteristics;
- Output targets;
- CameraDevice;
- onOpened() callback;
- CaptureRequest;
- CaptureRequestSession.

Camera Device

- The camereDevice is the physical camera.
- Get a CameraDevice by calling CameraManager.open(cameraId)
- Very important: The **open** call is asynchronized and we will get de CameraDevice in the **onOpened()** callback.

Main concepts

- CameraManager;
- CameraCharacteristics;
- Output targets;
- CameraDevice;
- onOpened() callback;
- CaptureRequest;
- CaptureRequestSession.

onOpened() callback

- The **onOpened()** callback is called when the camera device is opened.
- Here we can get the reference to the camera device and starting from this point, we can really use the camera.
- It’s also possible to be generated an error when we try to open the camera. For these case we have **onError()** callback.

onOpened() – useful code

```
//definesc callback-urile personalizate pentru camera
private final CameraDevice.StateCallback stateCallBackPropriu = new CameraDevice.StateCallback() {
    @Override
    public void onOpened(@NonNull CameraDevice camera) {
        //se apeleaza cand se deschide camera cu succes
        Log.i("TAVY","S-a apelat onOpened pentru camera.");
        cameraDevice = camera;
        if(cameraDevice==null) {
            Log.i("TAVY", "cameraDevice este null in metoda onOpened.");
        }
        else
        {
            Log.i("TAVY","cameraDevice s-a obtinut fizic si pornesc preview-ul.");
            //incep preview-ul imediat dupa ce am deschis camera
            startPreview();
        }
    }

    @Override
    public void onDisconnected(@NonNull CameraDevice camera) {
        Log.i("TAVY","onDisconnected. Camera deconectata.");
    }

    @Override
    public void onError(@NonNull CameraDevice camera, int error) {
        Log.i("TAVY","Eroare la deschiderea camerei in onError. " + error);
    }
};
```

In fact, we define how the camera device will behave.

Main concepts

- CameraManager;
- CameraCharacteristics;
- Output targets;
- CameraDevice;
- onOpened() callback;
- CaptureRequest;
- CaptureRequestSession.

CaptureRequest

- A CaptureRequest is a package of settings and outputs needed to capture **a single image** from the camera device.
- Builder pattern is applied here.
- A CaptureRequest.Builder is created from CameraDevice with a predefined template and we need to set the output target.

createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW)

CaptureRequest – useful code

```
private void startPreview()
{
    Log.i("TAVY","Start preview start.");
    try {
        CameraCharacteristics caracteristiCamera = manager.getCameraCharacteristics(cameraId);
        StreamConfigurationMap map = caracteristiCamera.get(caracteristiCamera.SCALER_STREAM_CONFIGURATION_MAP);
        dimensiuneImagine =map.getOutputSizes(SurfaceTexture.class)[0];

        //setez formatul output-ului pe suprafata (Surface) aferenta TextureView-ului de pe layout
        SurfaceTexture suprafata = viewImagine.getSurfaceTexture();
        suprafata.setDefaultBufferSize(dimensiuneImagine.getWidth(), dimensiuneImagine.getHeight());
        Surface suprafataDeAfisare = new Surface(suprafata);

        //creez un request de captura
        captureRequestBuilder = cameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
        captureRequestBuilder.addTarget(suprafataDeAfisare);


        //definesc sesiunea de captura
        cameraDevice.createCaptureSession(Arrays.asList(suprafataDeAfisare), new CameraCaptureSession.StateCallback() {
            @Override
            public void onConfigured(@NonNull CameraCaptureSession session) {
                //daca camera este deja inchisa, ies
                if(cameraDevice==null)
                {
                    return;
                }
                //daca sesiunea este pregatita, incepem afisarea preview-ului
                sesiuneCamera=session;
                updatePreview();
            }

            @Override
            public void onConfigureFailed(@NonNull CameraCaptureSession session) {
                Log.i("TAVY","onConfiguredFailed");
            }
        }, null);

    } catch (CameraAccessException e) {
        Log.i("TAVY","Eroare la captarea caracteristicilor camerei sau la startPreview: " + e.getMessage().toString());
    }
}
```

In fact, we set the parameters of the capture.

Main concepts

- CameraManager;
- CameraCharacteristics;
- Output targets;
- CameraDevice;
- onOpened() callback;
- CaptureRequest;
- CaptureRequestSession.

CaptureRequestSession

- A **CaptureRequestSession** is a context in which CaptureRequest can be submitted or executed.
- Important: the creation of CaptureRequestSession is also asynchronized.
- setRepeatingRequest() is used to issue a repeating request for the preview showing.

CaptureRequestSession – useful code

```
private void startPreview()
{
    Log.i("TAVY","Start preview start.");
    try {
        CameraCharacteristics caracteristiCamera = manager.getCameraCharacteristics(cameraId);
        StreamConfigurationMap map = caracteristiCamera.get(caracteristiCamera.SCALER_STREAM_CONFIGURATION_MAP);
        dimensiuneImagine =map.getOutputSizes(SurfaceTexture.class)[0];

        //setez formatul output-ului pe suprafata (Surface) aferenta TextureView-ului de pe layout
        SurfaceTexture suprafata = viewImagine.getSurfaceTexture();
        suprafata.setDefaultBufferSize(dimensiuneImagine.getWidth(), dimensiuneImagine.getHeight());
        Surface suprafataDeAfisare = new Surface(suprafata);

        //creez un request de captura
        captureRequestBuilder = cameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
        captureRequestBuilder.addTarget(suprafataDeAfisare);


        //definesc sesiunea de captura
        cameraDevice.createCaptureSession(Arrays.asList(suprafataDeAfisare), new CameraCaptureSession.StateCallback() {
            @Override
            public void onConfigured(@NonNull CameraCaptureSession session) {
                //daca camera este deja inchisa, ies
                if(cameraDevice==null)
                {
                    return;
                }
                //daca sesiunea este pregatita, incepem afisarea preview-ului
                sesiuneCamera=session;
                updatePreview();
            }

            @Override
            public void onConfigureFailed(@NonNull CameraCaptureSession session) {
                Log.i("TAVY","onConfiguredFailed");
            }
        }, null);

    } catch (CameraAccessException e) {
        Log.i("TAVY","Eroare la captarea caracteristicilor camerei sau la startPreview: " + e.getMessage().toString());
    }
}
```

In fact, we get the session of the capture process.

CaptureRequestSession – useful code

We set the “style” of the session: ***setRepeatingRequest*** using the previously defined captureRequestBuilder

```
//definesc updatePreview
private void updatePreview()
{
    Log.i("TAVY","Am intrat in updatePreview.");
    captureRequestBuilder.set(CaptureRequest.CONTROL_MODE, CameraMetadata.CONTROL_MODE_AUTO);

    try {
        sesiuneCamera.setRepeatingRequest(captureRequestBuilder.build(),null,null);
    } catch (CameraAccessException e) {
        Log.i("TAVY","Eroare la updatePreview: " + e.getMessage().toString());
    }
}
```

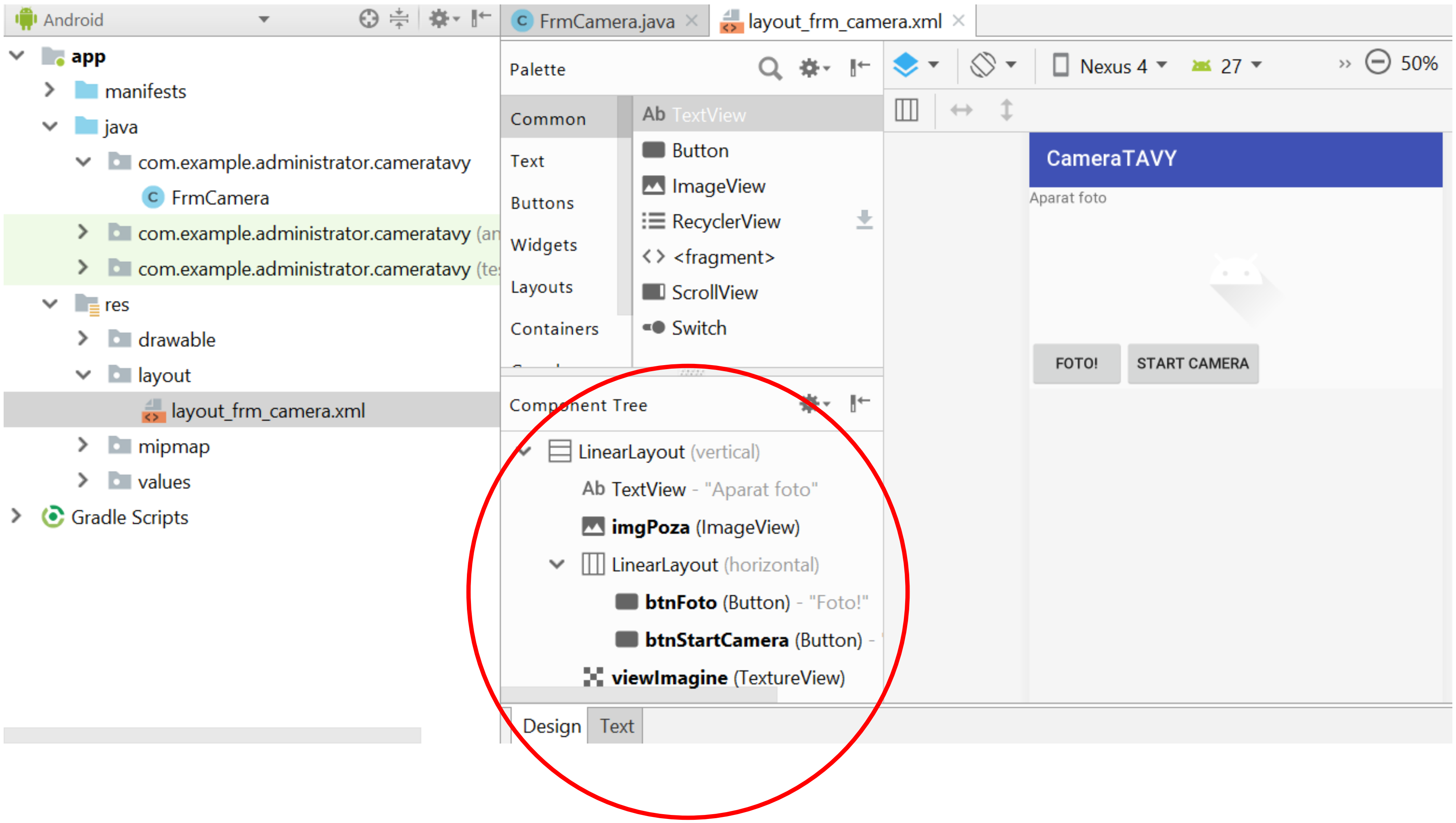
Important!!!

- Almost all the calls, requests and callbacks regardind the Camera in Android are **asynchronized**.
- So, it is necessary to understand all the concepts and to be very careful during the implementation.

Detailed code implementation

- When reading the following code sequences, please keep in mind that the calls are asyncronized.

Visual layout



Component Tree

```
LinearLayout (vertical)
    Ab TextView - "Aparat foto"
    imgPoza (ImageView)
    LinearLayout (horizontal)
        btnFoto (Button) - "Foto!"
        btnStartCamera (Button) -
    viewImagine (TextureView)
```

Layout – xml definition

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".FrmCamera">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Aparat foto" />

    <ImageView
        android:id="@+id/imgPoza"
        android:layout_width="match_parent"
        android:layout_height="121dp"
        app:srcCompat="@drawable/ic_launcher_foreground" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/btnFoto"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Foto!" />
        <Button
            android:id="@+id/btnStartCamera"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Start camera" />


    </LinearLayout>

    <TextureView
        android:id="@+id/viewImagine"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.administrator.cameratavy">

    <uses-permission android:name="android.permission.CAMERA"></uses-permission>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
    <uses-feature android:name="android.hardware.camera2.full"></uses-feature>


    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".FrmCamera">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Imports

```java
package com.example.administrator.cameratavy;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.SurfaceTexture;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraCaptureSession;
import android.hardware.camera2.CameraCharacteristics;
import android.hardware.camera2.CameraDevice;
import android.hardware.camera2.CameraManager;
import android.hardware.camera2.CameraMetadata;
import android.hardware.camera2.CaptureRequest;
import android.hardware.camera2.params.StreamConfigurationMap;
import android.os.Environment;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.util.Size;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FilterOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Arrays;
import java.util.List;
```

Activity’s implementation

```
public class FrmCamera extends AppCompatActivity {

    Button btnFoto, btnStartCamera;
    TextureView viewImagine;
    ImageView imgPoza;

    String cameraId;
    CameraManager manager;
    protected CameraDevice cameraDevice;

    Size dimensiuneImagine;

    CaptureRequest.Builder captureRequestBuilder;
    CameraCaptureSession sesiuneCamera;
```

Activity’s implementation

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_frm_camera);

        //instantiez controalele grafice de pe layout
        btnFoto = findViewById(R.id.btnFoto);
        viewImagine = findViewById(R.id.viewImagine);
        imgPoza = findViewById(R.id.imgPoza);

                //stabilesc listenerul pentru TextureView
        viewImagine.setSurfaceTextureListener(textureListener);
        Log.i("TAVY","Am stabilit listenerul pentru textureView cu succes.");
```

Activity's implementation

```
//setez listenerul pentru btnFoto
        btnFoto.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.i("TAVY","S-a actionat butonul Foto pentru salvare imagine");
                Bitmap imagineCurenta;
                imagineCurenta=viewImagine.getBitmap();
                Log.i("TAVY","Am obtinut imaginea curenta din textureView-ul viewImagine.");
                imgPoza.setImageBitmap(imagineCurenta);
                Log.i("TAVY","Am setat imaginea pentru ImageView-ul imgPoza.");
                //incerc salvarea imaginii curente de tip Bitmap
               File radacina = Environment.getExternalStorageDirectory();
               File dir = new File(radacina.getAbsolutePath() + "/SDCARD");
                dir.mkdirs();
                File fisier = new File(dir,"imagineTavy.png");
                OutputStream output;
                try
                {
                    output = new FileOutputStream(fisier);
                    imagineCurenta.compress(Bitmap.CompressFormat.PNG,100,output);
                    output.flush();  output.close();
                    Log.i("TAVY","Imagine salvata cu succes!!!  " + fisier.getAbsolutePath());
                    Toast.makeText(getApplicationContext(),"Imagine salvata cu succes!!!  " + fisier.getAbsolutePath(),
Toast.LENGTH_LONG).show();
                }
                catch (Exception e)
                {
                    Log.i("TAVY","Eroare la salvare fisier: " + e.getMessage().toString());
                }
            }
        });

        //cer permisiunea in mod explicit pentru Camera
ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);

    }
```

Activity's implementation

```
//definesc listener-ul pentru TextureView (atunci cand suprafata este pregatita, incep preview)
  TextureView.SurfaceTextureListener textureListener = new TextureView.SurfaceTextureListener() {
      @Override
      public void onSurfaceTextureAvailable(SurfaceTexture surface, int width, int height) {
          //atunci cand suprafata este disponibila pentru a fi utilizata,
          //deschidem camera proprie
          Log.i("TAVY", "Suprafata este pregatita si deschid camera proprie");
          openCamera();
      }

      @Override
      public void onSurfaceTextureSizeChanged(SurfaceTexture surface, int width, int height) {

      }

      @Override
      public boolean onSurfaceTextureDestroyed(SurfaceTexture surface) {
          return false;
      }

      @Override
      public void onSurfaceTextureUpdated(SurfaceTexture surface) {

      }
  };
```

Activity's implementation

```
@Override
  protected void onResume() {
     super.onResume();
     Log.i("TAVY","Sunt in onResume.");
     if(viewImagine.isAvailable())
     {
        Log.i("TAVY","Sunt in onResume si apelez openCamera.");
        openCamera();

     }
     else
     {
        Log.i("TAVY","Sunt in onResume si setez listenerul pentru textureViee");
        viewImagine.setSurfaceTextureListener(textureListener);
     }
  }
```

Activity's implementation

```
//definesc metoda proprie de deschidere a camerei
   private void openCamera() {
      manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
      try {
         //iau prima camera disponibila din dispozitiv
         cameraId = manager.getCameraIdList()[0];
         //!!!
         ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA}, 1);

     if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED)
{
            Log.i("TAVY","Permisiunea nu a fost acordata cu succes!");
            return;
         }
         Log.i("TAVY","Permisiunea a fost acordata ok!");
         manager.openCamera(cameraId, stateCallBackPropriu, null);
         Log.i("TAVY","Am deschis cu succes camera cu numarul " + cameraId);
      } catch (CameraAccessException e) {
         Log.i("TAVY","Eroare la gasirea unei camere: " + e.getMessage().toString());
      }

   }
```

Activity's implementation

```
private void startPreview()
  {
    Log.i("TAVY","Start preview start.");
    try {
      CameraCharacteristics caracteristiCamera = manager.getCameraCharacteristics(cameraId);
      StreamConfigurationMap map = caracteristiCamera.get(caracteristiCamera.SCALER_STREAM_CONFIGURATION_MAP);
      dimensiuneImagine =map.getOutputSizes(SurfaceTexture.class)[0];

      //setez formatul output-ului pe suprafata (Surface) aferenta TextureView-ului de pe layout
      SurfaceTexture suprafata = viewImagine.getSurfaceTexture();
      suprafata.setDefaultBufferSize(dimensiuneImagine.getWidth(), dimensiuneImagine.getHeight());
      Surface suprafataDeAfisare = new Surface(suprafata);

      //creez un request de captura
      captureRequestBuilder = cameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
      captureRequestBuilder.addTarget(suprafataDeAfisare);


      //definesc sesiunea de captura
      cameraDevice.createCaptureSession(Arrays.asList(suprafataDeAfisare), new CameraCaptureSession.StateCallback() {
        @Override
        public void onConfigured(@NonNull CameraCaptureSession session) {
          //daca camera este deja inchisa, ies
          if(cameraDevice==null)
          {
            return;
          }
          //daca sesiunea este pregatita, incepem afisarea preview-ului
          sesiuneCamera=session;
          updatePreview();
        }

        @Override
        public void onConfigureFailed(@NonNull CameraCaptureSession session) {
          Log.i("TAVY","onConfiguredFailed");
        }
      }, null);

    } catch (CameraAccessException e) {
      Log.i("TAVY","Eroare la captarea caracteristicilor camerei sau la startPreview: " + e.getMessage().toString());
    }
  }
```

Activity’s implementation

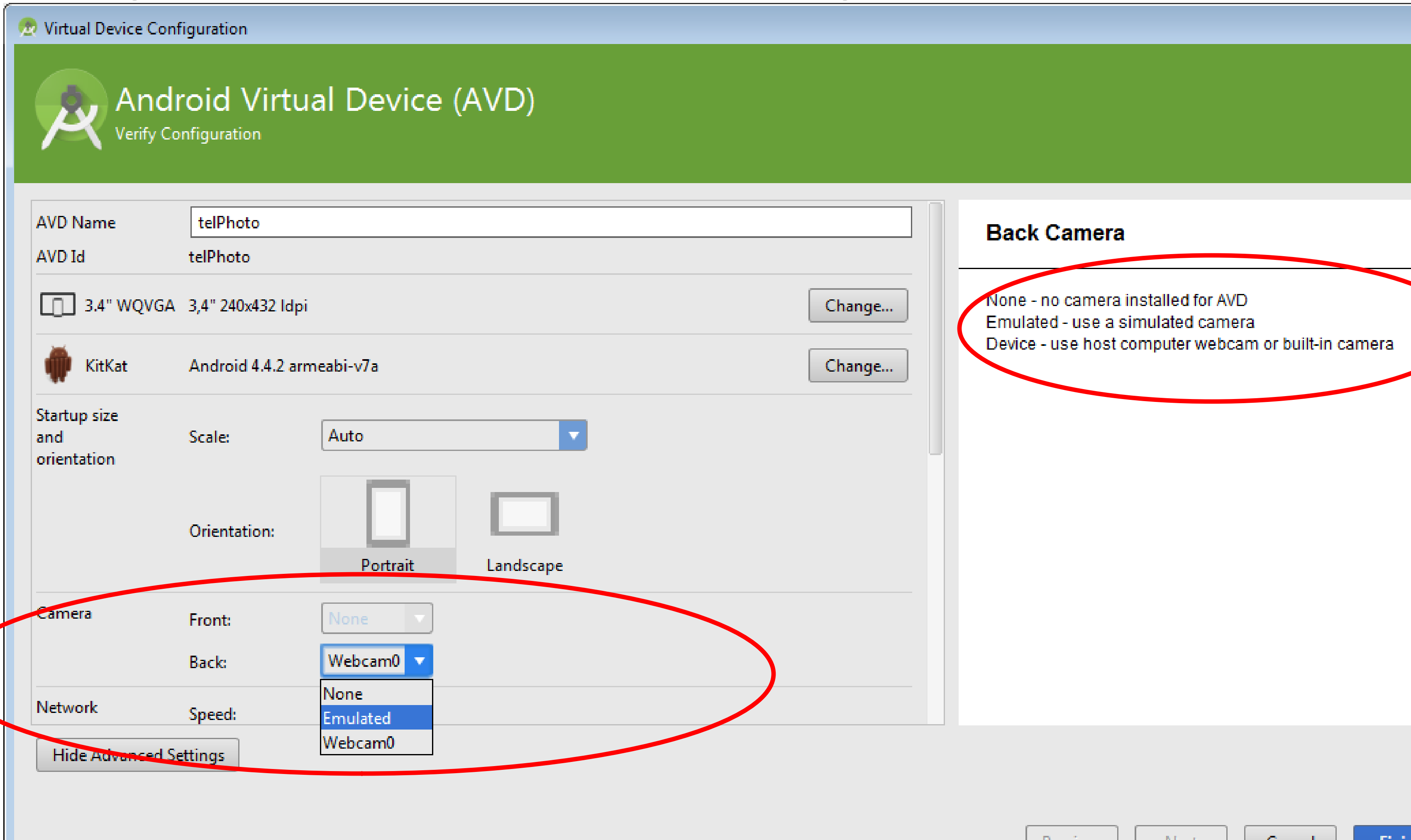
```
//definesc updatePreview
  private void updatePreview()
  {
     Log.i("TAVY","Am intrat in updatePreview.");
     captureRequestBuilder.set(CaptureRequest.CONTROL_MODE, CameraMetadata.CONTROL_MODE_AUTO);

     try {
        sesiuneCamera.setRepeatingRequest(captureRequestBuilder.build(),null,null);
     } catch (CameraAccessException e) {
        Log.i("TAVY","Eroare la updatePreview: " + e.getMessage().toString());
     }
  }
```

Activity's implementation

```
//definesc callback-urile personalizate pentru camera
 private final CameraDevice.StateCallback stateCallBackPropriu = new CameraDevice.StateCallback() {
    @Override
    public void onOpened(@NonNull CameraDevice camera) {
       //se apeleaza cand se deschide camera cu succes
       Log.i("TAVY","S-a apelat onOpened pentru camera.");
       cameraDevice = camera;
       if(cameraDevice==null) {
          Log.i("TAVY", "cameraDevice este null in metoda onOpened.");
       }
       else
       {
          Log.i("TAVY","cameraDevice s-a obtinut fizic si pornesc preview-ul.");
          //incep preview-ul imediat dupa ce am deschis camera
          startPreview();
       }
    }

    @Override
    public void onDisconnected(@NonNull CameraDevice camera) {
       Log.i("TAVY","onDisconnected. Camera deconectata.");
    }

    @Override
    public void onError(@NonNull CameraDevice camera, int error) {
       Log.i("TAVY","Eroare la deschiderea camerei in onError. " + error);
    }
 };



}
```

Emulator conditions for the camera

- set parameter **Configure Camera Facing Front/Back** to value **Emulated/Webcam0.**



Virtual Device Configuration

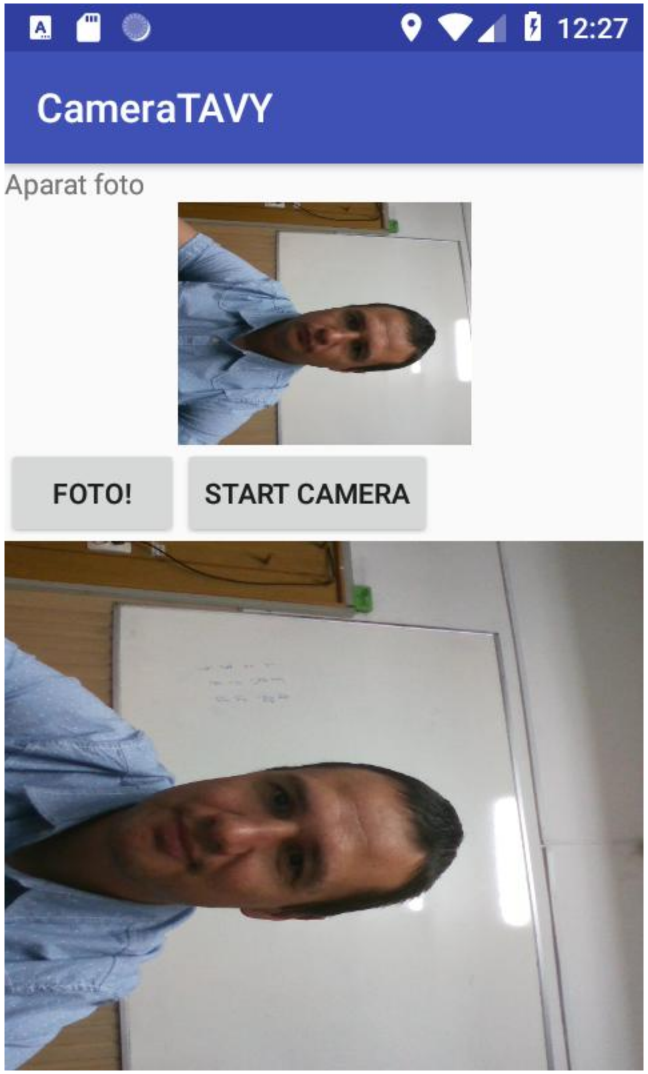
Android Virtual Device (AVD)
Verify Configuration

AVD Name: telPhoto
AVD Id: telPhoto
3.4" WQVGA 3,4" 240x432 ldpi
KitKat Android 4.4.2 armeabi-v7a
Startup size and orientation — Scale: Auto
Orientation: Portrait / Landscape
Camera — Front: None
Back: Webcam0
None
Emulated
Webcam0
Network — Speed:

Back Camera

None - no camera installed for AVD
Emulated - use a simulated camera
Device - use host computer webcam or built-in camera

Emulator Results ☺



Future directions…

- the ability to organize photos in an album by saving them with different names;
- the ability to take "spy" photos, without the user's agreement; these photos will be periodically uploaded to a server (the server of the husband/wife/girlfriend/the boyfriend of the girlfriend☺);
- the ability to control the camera's flash in a programmatic manner;
- and not least, the opportunity to take a picture of ourselves „prettier and smarter" ☺.

See you at the lab… ☺