Mobile Applications for Business

Master SIA/SDBIS

Octavian Dospinescu

2021

General topic

- GSM capabilities in Android
- Sending/receiving SMSs in a programmatic manner
- Class **android.telephony.SmsManager** – general anatomy
- Specific rights in AndroidManifest.xml
- Specific concepts: broadcast, intent etc.
- Future directions

Final goal (for the current course☺)

A mini-application which will be able to do the following operations:

- Getting a text message from the mobile device's user;
- Sending a SMS message (Short Message Service) to a mobile recipient;
- Informing the user about the message that was sent;
- Informing the user about any error that occurs during the transmission.

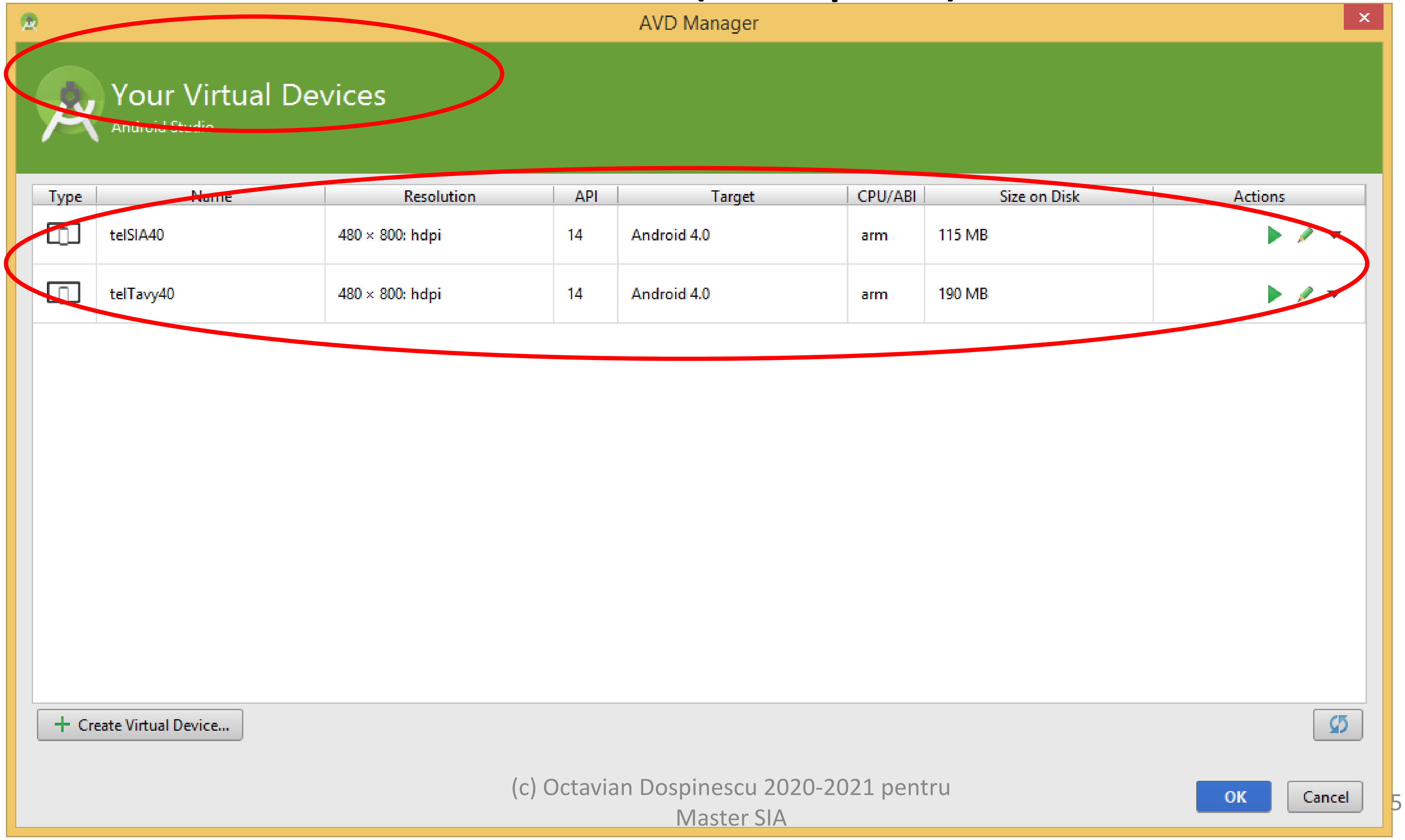
Additional goal

A module able to “intercept” the messages received on the mobile phone.

This operation aims explicit processing of the data contained in an SMS message that we receive on the mobile phone: the number of the sender and the message itself.

Preamble

Firtst of all, it is necessary to define 2 emulators: a sender and a receiver (receptor) for SMS.



AVD Manager

Your Virtual Devices

Android Studio

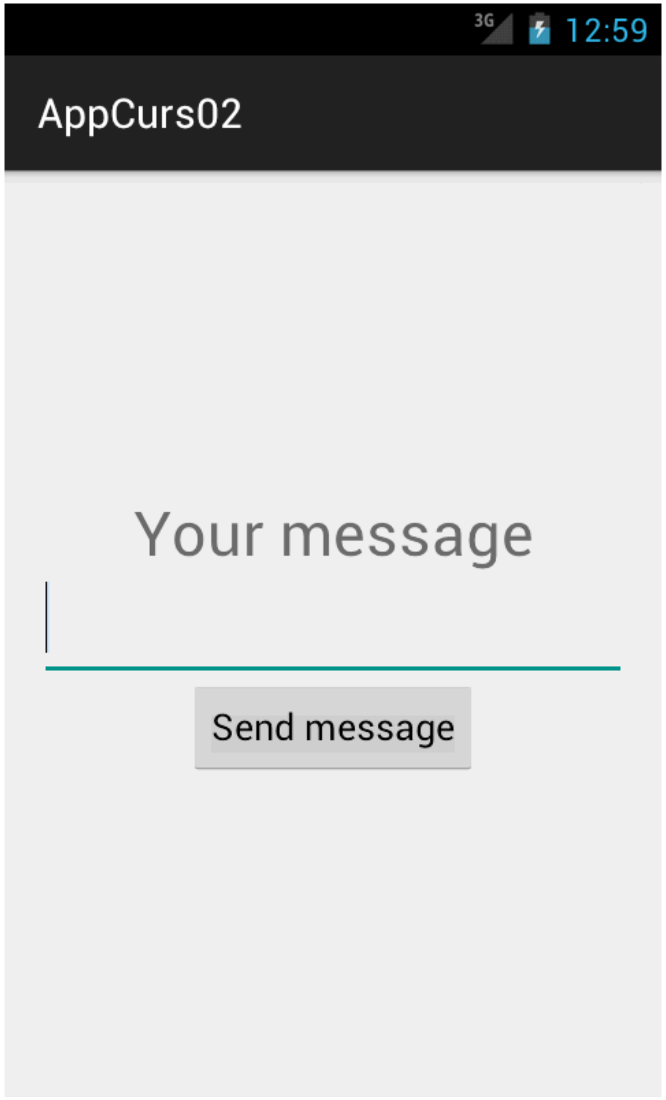
Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	telSIA40	480 × 800: hdpi	14	Android 4.0	arm	115 MB	
	telTavy40	480 × 800: hdpi	14	Android 4.0	arm	190 MB	

Simplified architecture of the activity (form)

A graphical interface having 3 simple elements:

- **an informative text** (*TextView* – Specify your message);
- **a text-box** where the user will be able to type his own message (*EditText*);
- **a button** by which the operation of sending an SMS will be started (*Button*).

Simplified architecture of the form



Class **android.telephony.SmSManager**

import android.telephony.SmsManager;

The class **SmsManager** is to allow the management of the sms messages in terms of sending them to the recipient.

We obtain objects from the SmsManager class by calling the static method **SmsManager.getDefault()**.

Class **android.telephony.SmSManager**

The used method is **sendTextMessage**, with the following list of parameters:

•**destinationAddress**: it represents the phone number of the receipient, expressed as String format;

•**scAddress**: it represents the number of the **s**ervice **c**enter, in String format. By default, this number is already known by the mobile device and as result we could use the *null* value in order to get the default number.

Class **android.telephony.SmSManager**

The used method is **sendTextMessage**, with the following list of parameters:

- **text**: it represents the message that will be sent to the destinationAddress, using the service center specified by scAddress. It is very important to mention that if this text has more than 160 chars (the maximum length of a standard SMS message), then it will be generated an exception that should be treated.

- **sentIntent** and **deliveryIntent** have the role of catching codes resulted after the sending and delivering of the messages. For the beginning, we recommend the using of *null* values in the case of these 2 last parameters.

Class **android.telephony.SmSManager**

VERY IMPORTANT!!!

- The method **sendTextMessage** generates an IllegalArgumentException exception if *destinationAddress* or *text* are empty.

Class **android.telephony.SmSManager**

A model of using (just a suggestion☺)

```
SmsManager sms;
sms = SmsManager.getDefault();
sms.sendTextMessage(numarDestinatar,  null,  mesajSMS,
null, null);
```

A model of implementation

```
@Override
        public void onClick(View v) {
                // TODO Auto-generated method stub
                if(v==btnSMS) {  //the button has been pressed by the user
                        //we are generating a new manager for sending SMSs
                        SmsManager sms;
                        sms = SmsManager.getDefault();
                        //we define the recipient's phone’s number
                        String numarDestinatar;
                        numarDestinatar="5556";
                        //we take the message typed by the user on the form
                        String mesajSMS;
                        mesajSMS=this.txtMesaj.getText().toString();
                        try {
                                //we send the SMS message
                        sms.sendTextMessage(numarDestinatar, null, mesajSMS, null, null);
                                //we prepare a message to inform the user
                        notificare = Toast.makeText(getApplicationContext(), "Mesajul a 
fost trimis",Toast.LENGTH_LONG);
                                //we are showing the notification
                                notificare.show();
                }
                        catch (Exception eroare) {
                                //if troubles, we show the error message
                                notificare = Toast.makeText(getApplicationContext(), 
"Eroare:" + eroare.getMessage(), Toast.LENGTH_LONG);
                                notificare.show();
                        }
                }
```

Permissions and rights

To send an SMS, it is necessary that the application has the permission from the Android operating system.

The permission is written in the file AndroidManifest.xml, in the section **uses-permission**.

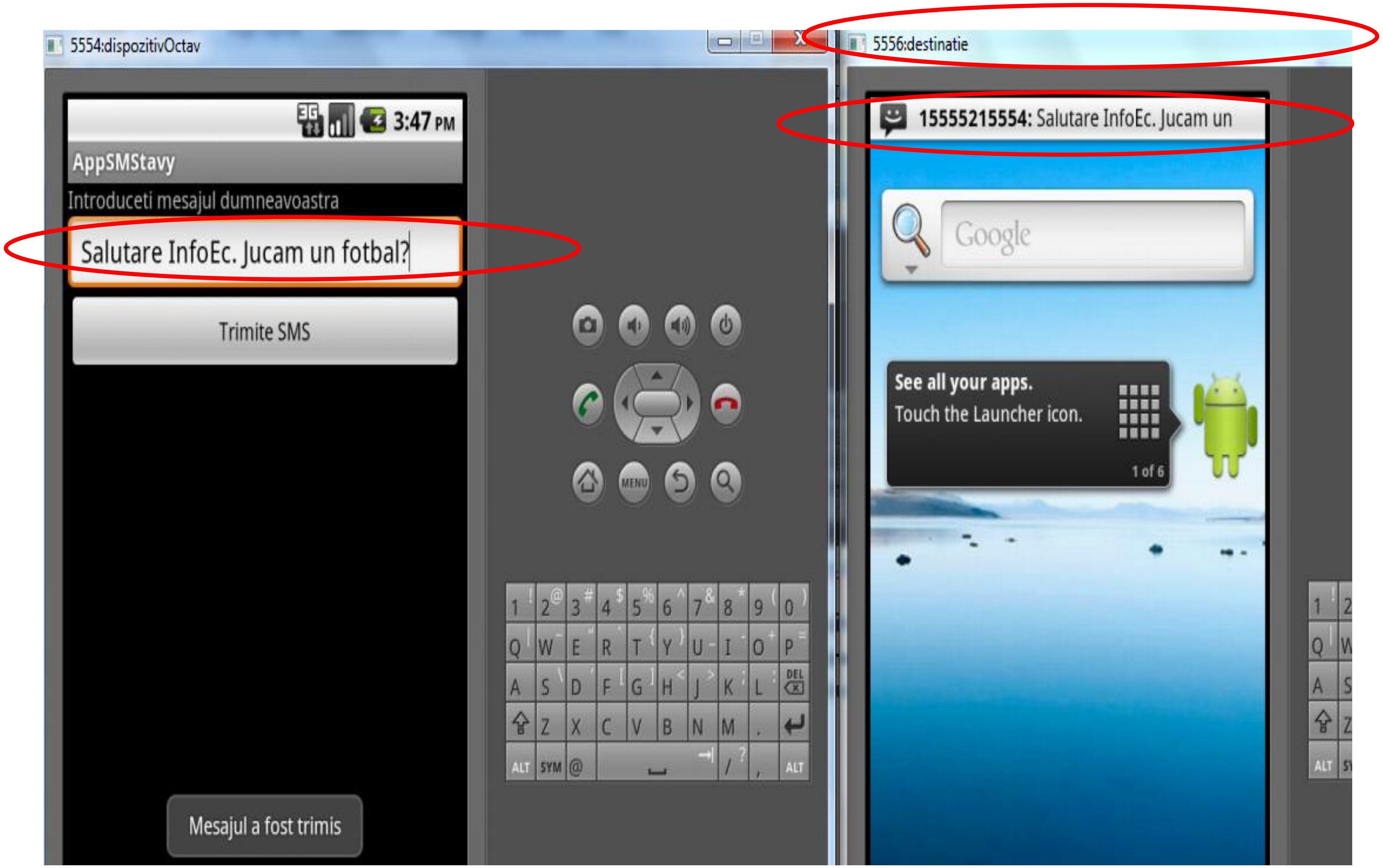
In our case, the name of the permission is **android.permissions.SEND_SMS**.

Permissions and rights

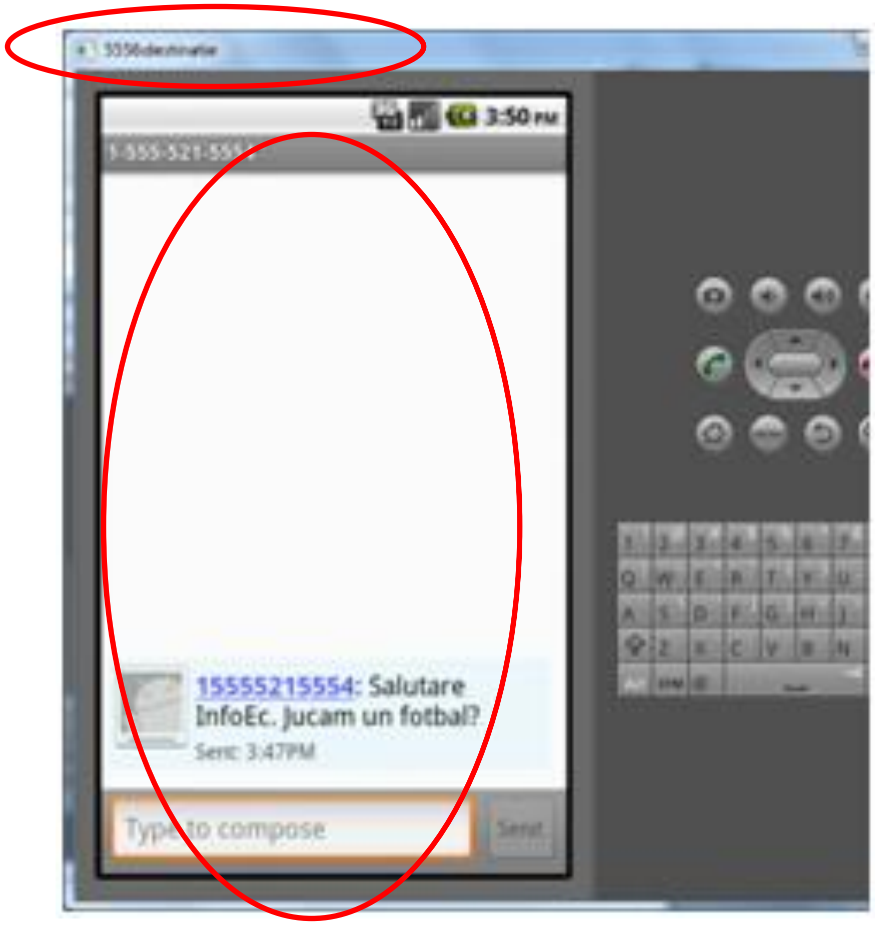
After adding this permission for our application, the AndroidManifest.xml file stored this aspect in a specific line (uses-permission tag).

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.adminlocal.appcurs02" >

       <uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".AppMessages"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Running the application



The list of messages from the **destination** emulator



15555215554: Salutare InfoEc. Jucam un fotbal?
Sent: 3:47PM

Future directions☺

- The possibility of sending “flows” of SMSs

Implement a new application which sends customized SMSs to a list of recipients. We recommend that the list is stored in a text file, in a format like:

recipientPhoneNumber:CustomMessage

Future directions☺

- The possibility of sending “flows” of SMSs

That way, the user has the possibility to send a whole set of SMS messages to a list of people, by loading that file.

Future directions☺

- The possibility of sending “flows” of SMSs

The immediate applicability of this application can be found in mass information campaigns carried out by large retail chains via SMS.

Future directions ☺

- The possibility of sending “flows” of SMSs…☺
- What if the recipient’s number is a “hot” one?

Full implementation (for the lab☺)

```
package com.example.adminlocal.appcurs02;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;


public class AppMessages extends ActionBarActivity implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_app_messages);
        Button btnSend;
        btnSend = (Button) findViewById(R.id.btnSendMessage);
        btnSend.setOnClickListener(this);
ActivityCompat.requestPermissions(this,new String[]{Manifest.permission.SEND_SMS},1);
    }
@Override
   public void onClick(View v) {
      SmsManager sms;
      sms = SmsManager.getDefault();
      String destinationNumber = "5556";  //for the second  emulator

      EditText txtMesaj = (EditText) findViewById(R.id.txtMesaj);
      String mesajSms;
      mesajSms =txtMesaj.getText().toString();
      try
      {
         sms.sendTextMessage(destinationNumber,null,mesajSms,null,null);
         Toast notificare = Toast.makeText(getApplicationContext(),"Message sent",Toast.LENGTH_LONG);
         notificare.show();
      }
      catch (Exception eroare)
      {
         Toast notificare = Toast.makeText(getApplicationContext(),"Trouble: " +
eroare.getMessage(),Toast.LENGTH_LONG);
         notificare.show();
      }
   }
}
```

More details about Android permissions

https://developer.android.com/training/permissions/requesting.html