PT&SCHE

Introduction of part-time and short cycle studies in Serbia

Funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

UNIVERZITET METROPOLITAN BEOGRAD

KI205 - JAVA 8: PROGRAMIRANJE U JAVI NA ANDROID PLATFORMI

Rad sa porukama u Android operativnom sistemu

Lekcija 07

PRIRUČNIK ZA STUDENTE

KI205 - JAVA 8: PROGRAMIRANJE U JAVI NA ANDROID PLATFORMI

Lekcija 07

RAD SA PORUKAMA U ANDROID OPERATIVNOM SISTEMU

- Rad sa porukama u Android operativnom sistemu
- Poglavlje 1: Slanje SMS poruka
- Poglavlje 2: Prijem SMS poruka
- Poglavlje 3: Slanje e-mail poruka iz aplikacije
- Poglavlje 4: Domaći zadatak 7
- Pregled Lekcije08

UVOD

Često se sreću specifične Android aplikacije koje korisnicima moraju da obezbede slanje SMS poruka.

Svi Android mobilni telefoni imaju već ugrađenu aplikaciju za slanje SMS poruka. Međutim, veoma često sreću se specifični Android programi koji nakon realizovanja izvesnog događa, na primer, otvaranje liste poslovnih kontakata, pristup GPS lokaciji i tako dalje, moraju korisnicima da obezbede mogućnost slanja SMS poruke na neki broj.

Ova lekcija ima zadatak da predstavi mogućnosti programskog slanja i primanja SMS poruka u kreiranoj Android aplikaciji. Takođe, biće govora i o načinima pozivanja *Mail* aplikacije sa ciljem razmenjivanja elektronske pošte primenom kreirane Android aplikacije. Takođe, od ove lekcije će imati koristi i oni studenti koji žele da razviju vlastitu aplikaciju za razmenjivanje SMS poruka ili elektronske pošte.

Poseban akcenat će lekcija da stavi na sledeće teme:

- Kako se iz kreirane aplikacije programski šalju SMS poruke?
- Kako se šalju poruke primenom ugrađene aplikacije?
- Kako je moguće primiti dolazne SMS poruke?
- Kako je moguće poslati e-mail iz kreirane vlastite aplikacije?

Savladavanjem ove lekcije, studenti će biti osposobljeni da kreiraju složenije Android aplikacije koje imaju ugrađene funkcionalnosti razmenjivanja poruka, bilo da su to sms ili e-mail poruke.

Poglavlje 1

Slanje SMS poruka

SLANJE SMS PORUKA KREIRANIM PROGRAMOM

Svaki mobilni uređaj podržava slanje i primanje SMS poruka.

Svaki mobilni telefon, bilo da pripada generaciji pametnih telefona ili nekoj starijoj, podržava slanje i prijem *SMS* poruka. Tako, da ova usluga je vremenom postala standardna i neizostavna u industriji mobilni telefona i pratećeg softvera. Android operativni sistem ima ugrađenu SMS aplikaciju koja omogućava prijem i slanje poruka. Međutim, velika sloboda u izradi aplikacija i primeni funkcionalnosti Android operativnog sistema, omogućava programerima da kreiraju vlastite SMS aplikacije, sa dodatnim funkcionalnostima i mogućnosti personalizacije SMS aplikacije različitim korisničkim potrebama i navikama. Takođe, uslugu prijema i slanja SMS poruka moguće je ugraditi i u neke druge Android aplikacije. Na primer, kreirana je aplikacija *Planer* koja ima zadatak da periodično šalje cirkularne SMS poruke na više kontakata. Takođe, postoje Android aplikacije namenjene roditeljskoj kontroli dece koje, u određenim intervalima sa dečjeg telefona, šalju poruke na roditeljske telefone u vezi sa informacijama o lokaciji na kojoj se deca trenutno nalaze.

Aplikacije koje manipulišu SMS porukama mogu da se ponašaju da dva načina:

- Šalju SMS vlastitim programskim kodom;
- Šalu SMS pozivajući ugrađenu Android aplikaciju pomoću Intent objekata.

Prvi zadatak će biti savladavanje i demonstracija prosleđivanja i prijema SMS poruka kreiranim programskim kodom. U tu svrhu je neophodno razviti novi Android projekat sa vlastitim korisničkim interfejsom iz kojeg se prosleđuju i primaju sms poruke uz pomoć i podršku odgovarajućih JAVA klasa projekta.

Pa, kao što je napomenuto, u kreiranom projektu, pod nazivom SmsDemo, prvo će biti kreirana XML datoteka korisničkog interfejsa. Datoteka će zadržati podrazumevani naziv activity_main.xml i njen kod može biti prezentovan sledećim listingom:

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

<Button
    android:id="@+id/btnSendSMS"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
    android:text="Pošalji SMS"
    android:onClick="onClick" />

</LinearLayout>
```

Trebalo bi napomenuti, da će ova datoteka biti proširivana novim GUI elementima kako se budu dodavale nove funkcionalnosti u projekat.

SMS PORUKE - PRIVILEGIJE

Aplikacija koja uključuje rad sa SMS porukama u AndroidManifest.xml datoteci mora da obezbedi izvesne privilegije.

Da bi kreirana aplikacija imala mogućnost upravljanja procesima slanja i prijema SMS poruka, neophodno je u okviru AndroidManifest.xml datoteke projekta, ugraditi odgovarajuće privilegije. Privilegija se pakuje u XML element *<uses-permission ...>* i ima zadatak da dozvoli aplikaciji slanje ili prijem SMS poruka. Za svaku od navedenih privilegija postoji odgovarajuća konstanta koja je realizuje. Tako, za slanje SMS poruka neophodno je, kao privilegiju istaći ***SEND_SMS,*** dok privilegija, koja odgovara prijemu SMS poruka, glasi ***RECEIVE_SMS***.

Kod AndroidManifest.xml datoteke, u koji su ugrađene navedene dozvole slanja i prijema SMS poruka, prikazan je sledećim listingom.

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

    <uses-permission android:name="android.permission.SEND_SMS"/>

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

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

</manifest>
```

PRIMER 1 - KLASA AKTIVNOSTI ZA SLANJE SMS PORUKE

Instrukcije za slanje SMS poruka ugrađene su u klasu aktivnosti aplikacije.

Nakon kreiranja korisničkog interfejsa i davanja privilegija aplikaciji za rad sa SMS porukama, pristupa se kreiranju klase aktivnosti aplikacije koja će u ovom slučaju zadržati podrazumevani naziv MainActivity.java. Klasa aktivnosti će sadržati izvesne funkcionalnosti koje omogućavaju rad sa porukama. U početku, to će biti jednostavno prosleđivanje SMS poruke. Poruka je definisana unapred zadatim stringom i prosleđena na unapred određeni broj telefona u okviru programskog koda klase aktivnosti. Programski kod, klase aktivnosti sa navedenim funkcionalnostima, priložen je sledećim listingom.

Takođe, neophodno je istaći da će u narednim izlaganjima, definicija i logika koju klasa implementira biti proširivana kako dopunske funkcionalnosti budu uvođene u projekat.

```
package com.metropolitan.smsdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View v) {
        sendSMS("5556", "Pozdravni SMS - primer!");
    }
    //Šalje poruku drugom uređaju”-
    private void sendSMS(String phoneNumber,
                         String message){
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message,
                null, null);
    }

}
```

PROGRAMSKO SLANJE SMS PORUKA - FUNKCIONISANJE

Android operativni sistem primenjuje politiku zasnovanu na privilegijama.

Za prevođenje i pokretanje aplikacije, podrazumevanim emulatorom (5554) neophodno je kliknuti, U razvojnom alatu Android Studio IDE, na Run App ili Shift + F10. Biće kreiran i novi emulator (u kodu je obeležen sa 5556 – sledeća slika) čiji će zadatak biti da slimulira telefon koji prima poruku. Da bi novi emulator mogao da primi poruku, a podrazumevani da je pošalje, bilo je neophodno ugraditi već pomenute privilegije u AndroidManifest.xml datoteku. Ovo je važno budući da slanje SMS poruka podrazumeva troškove mobilnog saobraćaja pa privilegija omogućava korisnicima izbor da li da instaliraju aplikaciju ili ne.

> Ukoliko koristite Genymotion emulator visokih performansi, ova funkcionalnost nije dostupna u besplatnoj verziji emulatora. Za korišćenje pune funkcionalnosti ovog emulatora neophodno je izdvojiti 25 EUR na mesečnom nivou za pro licencu.

Za slanje SMS poruke, u okviru vlastitog programskog koda, neophodno je koristiti klasu SmsManager. Ova klasa koristi statičku metodu getDefault() za kreiranje objekta koji poziva metodu sendTextMessage() za slanje SMS poruke (videti kod metode sendSMS() klase aktivnosti).

Metoda sendTextMessage() sadrži sledeće argumente navedene po redosledu pojavljivanja:

- Telefonski broj primaoca SMS poruke;
- Adresa servisnog centra (null u slučaju podrazumevanog);
- Sadržaj SMS poruke;
- Iniciranje akcije koja se događa prilikom slanja poruke;
- Iniciranje akcije koja će se desiti nakon pristizanja poruke na odredište.

Slika 1.1 Prijem SMS poslate SMS poruke emulatorom

SLANJE SMS PORUKA – POVRATNE INFORMACIJE

Praćenje statusa procesa slanja SMS poruka moguće je podržati kreiranjem objekata tipa PendingIntent.

U prethodnim izlaganjima je naučeno kako se iz kreiranog programskog koda šalje SMS poruka. Sledeći zadatak je savladavanje mehanizama provere da li je poruka ispravno poslata. Za upravljanje ovim procesom moguće je kreirati dva *PendingIntent* objekta u klasi aktivnosti koji će pratiti status procesa slanja SMS poruka kao dva poslednja argumenta metode sendTextMessage().

Kao što je napomenuto u prethodnom izlaganju, klasa aktivnosti će biti proširivana kako projekat bude nailazio na sve veće zahteve. U svetlu navedenog, u polaznu klasu aktivnosti biće ugrađene nove funkcionalnosti. Novi kod klase aktivnosti projekta predstavljen je sledećim listingom.

Slika 1.2 Prijem povratne informacije o prosleđenom SMS - u

```
package com.metropolitan.smsdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    String SENT = "SMS_SENT";
    String DELIVERED = "SMS_DELIVERED";
    PendingIntent sentPI, deliveredPI;
    BroadcastReceiver smsSentReceiver, smsDeliveredReceiver;
    IntentFilter intentFilter;

    private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            //-prikazuje primljeni SMS u TextView-
            TextView SMSes = (TextView) findViewById(R.id.textView1);
            SMSes.setText(intent.getExtras().getString("sms"));
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sentPI = PendingIntent.getBroadcast(this, 0,
                new Intent(SENT), 0);

        deliveredPI = PendingIntent.getBroadcast(this, 0,
                new Intent(DELIVERED), 0);

    }

    @Override
    public void onResume() {
```

```
        super.onResume();

        //---registrovanje primaoca---
       // registerReceiver(intentReceiver, intentFilter);

        //---kreira BroadcastReceiver kada je SMS poslat---
        smsSentReceiver = new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case AppCompatActivity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS prosleđen",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                        Toast.makeText(getBaseContext(), "Generička greška",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NO_SERVICE:
                        Toast.makeText(getBaseContext(), "Nema usluge",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NULL_PDU:
                        Toast.makeText(getBaseContext(), "Null PDU",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_RADIO_OFF:
                        Toast.makeText(getBaseContext(), "Radio isključen",
                                Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        };

        //---kreira BroadcastReceiver kada SMS dostavljen---
        smsDeliveredReceiver = new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case AppCompatActivity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS dostavljen",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case AppCompatActivity.RESULT_CANCELED:
                        Toast.makeText(getBaseContext(), "SMS nije dostavljen",
                                Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        };
```

```
        //---registruje dva BroadcastReceiver - a---
        registerReceiver(smsDeliveredReceiver, new IntentFilter(DELIVERED));
        registerReceiver(smsSentReceiver, new IntentFilter(SENT));
    }

    @Override
    public void onPause() {
        super.onPause();

        //---odjavljuje primaoca---
        //unregisterReceiver(intentReceiver);

        //---odjavljuje dva BroadcastReceiver-a---
        unregisterReceiver(smsSentReceiver);
        unregisterReceiver(smsDeliveredReceiver);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        //---odjavljivanje primaoca---
        unregisterReceiver(intentReceiver);
    }



    public void onClick(View v) {
        sendSMS("5556", "Pozdravni SMS - primer!");
    }

    //Šalje poruku drugom uređaju”-
    private void sendSMS(String phoneNumber,
                         String message){
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message,
                sentPI, deliveredPI);
    }

}
```

POVRATNE INFORMACIJE - FUNKCIONISANJE

Za upravljanje povratnim informacijama kreiraju se i BroadcastReciever objekti.

Iz priloženog kosa se vidi da su kreirana dva PendingIntent objekta u onCreate() metodi klase aktivnosti. Ovim objektima je obezbeđeno slanje potvrda nakon slanja poruke (***SMS_SENT***) i njenog isporučivanja na odredište (***SMS_DELIVERED***). Upravo na ovaj način je objašnjena

pojava koju svakodnevno srećemo koristeći mobilne telefone, a to je informacija u formi Toast objekta na ekranu o statusu slanja / prijema SMS poruke.

U kreiranoj metodi onResume() kreirana su dva *BroadcastReciever* objekta koji prate poruke ***SMS_SENT*** i ***SMS_DELIVERED*** inicirane od strane ***SmsManager***.

PendingIntenti BroadcastReceiverobjekti, izdvojeni su sledećim kodom.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sentPI = PendingIntent.getBroadcast(this, 0,
                new Intent(SENT), 0);

        deliveredPI = PendingIntent.getBroadcast(this, 0,
                new Intent(DELIVERED), 0);

    }

**********************************************

//---registruje dva BroadcastReceiver - a---
        registerReceiver(smsDeliveredReceiver, new IntentFilter(DELIVERED));
        registerReceiver(smsSentReceiver, new IntentFilter(SENT));
```

Za svaki od kreiranih BroadcastReceiver objekata je redefinisana metoda OnReceive() koja kroz svoju ***switch-case*** strukturu izvodi određenu akciju u vezi sa praćenjem statusa poruke.

Kreirani PendingIntentobjekti se pakuju u metodu sendTextMessage() kao poslednja dva argumenta (izdvojeno sledećim listingom), obezbeđujući na taj način informaciju da li je poruka ispravno prosleđena ili ne.

Na kraju, metodom onPause() prekida se korišćenje kreiranih BroadcastReceiver *objekata* (izdvojeno sledećim listingom kao delovi metoda sendSMS() i onPause()).

```
/Šalje poruku drugom uređaju”-
    private void sendSMS(String phoneNumber,
                         String message){
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message,
                sentPI, deliveredPI);
    }

**********************************************************************

@Override
    public void onPause() {
        super.onPause();

        //---odjavljuje primaoca---
        //unregisterReceiver(intentReceiver);
```

```
        //---odjavljuje dva BroadcastReceiver-a---
        unregisterReceiver(smsSentReceiver);
        unregisterReceiver(smsDeliveredReceiver);
    }
```

SLANJE PORUKA POMOĆU NAMERA

Moguće je koristiti i ugrađenu aplikaciju za obavljanje slanja SMS poruke.

SmsManager klasa omogućava slanje SMS poruka iz aplikacije bez oslanjanja na sistemski ugrađenu Android aplikaciju. Međutim, ponekad je jednostavnije dozvoliti ugrađenoj aplikaciji da obavi kompletan posao slanja (ili prijema) SMS poruka, pri čemu kreirana aplikacija vrši kontrolu nad celokupnim poslom.

Da bi ugrađena SMS aplikacija bila dostupna za slanje (ili prijem) poruka, na gore navedeni način, neophodno je primeniti Intent objekat kojem se pridružuje MIME tip vnd.android-dir/mms-sms. Navedeno je ugrađeno u metodu onSMSIntentClick() koja, ako za to postoji potreba, može biti ugrađena u klasu aktivnosti kao i prethodne metode ove klase o kojima je u lekciji bilo govora.

```
public void onSMSIntentClick (View v) {
        Intent i = new
                Intent(android.content.Intent.ACTION_VIEW);
        i.putExtra("address", "5556; 5558; 5560");

        i.putExtra("sms_body", "Pozdravni SMS - primer!");
        i.setType("vnd.android-dir/mms-sms");
        startActivity(i);
    }
```

U navedenoj metodi smešten je i poziv metodeputExtra() objektom klase *Intent.* Na ovaj način jedna poruka je prosleđena na više brojeva telefona (5556, 5558, 5560 – brojevi koji odgovaraju različitim emulatorima za simuliranje primalaca poruka). Navedeni brojevi su odvojeni zarezima prilikom izvršavanja ugrađene SMS aplikacije (sledeća slika).

Slika 1.3 Slanje poruka ugrađenom SMS aplikacijom

ZADATAK 1 - ANALIZIRAJTE SLEDEĆI KOD

Pokušajte sami!!!

Pogledajte pažljivo priloženi listing i dajte njegovo objašnjenje.

```
private void sendSMS(String phoneNumber,
                        String message){
        SmsManager sms = SmsManager.getDefault();
```

```
        sms.sendTextMessage(phoneNumber, null, message,
                null, null);
    }
```

Poglavlje 2

Prijem SMS poruka

UVOD U PRIJEM SMS PORUKA

Primenom BroadcastReceiver objekta moguće je primati SMS poruke kreiranom aplikacijom.

U nastavku izlaganja biće obrađeni i prezentovani mehanizmi pomoću kojih kreirana Android aplikacija može da preuzima SMS poruke poslate sa drugom mobilnog uređaja. Takođe, ovde će biti opisane privilegije koje ovakva aplikacija mora da dobije, kao i svi bitni elementi (klase, objekti i ponašanja) koja ovakva klasa mora da poseduje da bi imala mogućnost prijema SMS poruka.

Kao što je moguće poslati, SMS poruke je moguće i primiti kreiranom Android aplikacijom koja koristi BroadcastReceiver objekat. Ovde je reč o veoma korisnoj funkcionalnosti aplikacije posebno u slučajevima kada prijem SMS poruke inicira neku određenu akciju. Primer za ovakav tip aplikacije može biti aplikacija koja automatskom SMS porukom vraća lokaciju mobilnog uređaja u slučaju da je on izgubljen ili ukraden. U slučaju da se pokuša pristup takvom telefonu, on automatski šalje poruku na predefinisani broj telefona, sa lokacijom telefona. Telefon koji je primio poruku, može da ima instaliranu aplikaciju koja prosleđuje predefinisanu poruku, kao odgovor na primljenu, u koju su ugrađene informacije o primljenoj lokaciji telefona, osobi od koje se očekuje da vrati telefon vlasniku, policiji i slično.

PRIJEM SMS PORUKA – DOZVOLE I REGISTROVANJE KLASE PRIJEMNIKA

Da bi aplikacija mogla da prima SMS poruke, neophodno joj je dodeliti privilegiju prijema SMS poruka.

Prethodni primer, za slanje SMS poruka, može biti proširen novim funkcionalnostima koje se odnose na mogućnost prijema poruka. U tu svrhu je neophodno dodati nove linije koda u AndroidManifest.xml datoteku.

Prvi korak je davanje dozvola aplikaciji da prima poruke i registrovanje klase prijemnika. Kreiranje klase prijemnika sada postaje novi zadatak programera.

Dozvola i registrovanje klase prijemnika prikazani su sledećim kodom izdvojenim iz AndroidManifest.xml datoteke.

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

    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>

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

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

        <receiver android:name=".SMSReceiver">
            <intent-filter android:priority="100">
                <action android:name=
                    "android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>

    </application>

</manifest>
```

PRIJEM SMS PORUKA – KLASA PRIJEMNIK - PRIMER

Klasa prijemnik je članica src foldera zajedno sa klasom aktivnosti.

Pošto je u AndroidManifest.xml datoteci, pored dozvole prijema SMS poruka kreiranom aplikacijom, registrovana nova klasa - klasa SMSReciever, neophodno je pristupiti njenom kodiranju. U folder *src* dodaje se nova klasa pod imenom SMSReceiver koja nasleđuje klasu *BroadcastReceiver*. *Kod klase SMSReceiver* priložen je sledećim listingom.

Zadatak kreirane klase je složen:

- preuzimanje prosleđene SMS poruke;
- učitavanje primljene SMS poruke;
- preuzimanje podataka o pošiljaocu;
- preuzimanje tela (sadržaja) poruke;
- prikazivanje nove SMS poruke;
- zaustavljanje procesa slanja i primanja SMS poruka;
- komuniciranje preko Intent objekta sa glavnom aktivnošću aplikacije;

- ažuriranje SMS - a iz aktivnost.

```
package com.metropolitan.smsdemo;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;
import android.widget.Toast;

/**
 * Created by Vladimir Milicevic on 11.11.2016..
 */

public class SMSReceiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent){
        //---preuzimanje prosleđene SMS poruke---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "SMS iz ";

        if (bundle != null){
            //---učitavanje primljene SMS poruke---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i<msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                if (i==0) {
                    //---preuzimanje podataka o pošiljaocu---
                    str += msgs[i].getOriginatingAddress();
                    str += ": ";
                }
                //---preuzimanje tela poruke---
                str += msgs[i].getMessageBody().toString();
            }

            //---prikazivanje nove SMS poruke---
            Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
            Log.d("SMSReceiver", str);

           /* //---zaustavljanje slanja/primanja---
            this.abortBroadcast();*/

            //---pokretanje MainActivity---
            Intent mainActivityIntent = new Intent(context, MainActivity.class);
            mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(mainActivityIntent);

            //---Slanje namere BroadcastIntent za ažuriranje SMS iz aktivnosti---
            Intent broadcastIntent = new Intent();
```

```
            broadcastIntent.setAction("SMS_RECEIVED_ACTION");
            broadcastIntent.putExtra("sms", str);
            context.sendBroadcast(broadcastIntent);
        }
    }
}
```

KLASA PRIJEMNIK - FUNKCIONISANJE

Svaka primljena poruka poziva ponovo onReceive() metodu.

Iz Android Studio IDE razvojnog okruženja, ili Eclipse IDE ukoliko je neko od studenata izabrao ovo razvojno okruženje, neophodno je pristupiti prevođenju i testiranju kreirane aplikacije koja ima mogućnost, pored prethodno definisanog slanja SMS poruka, sada i prijem poslatih SMS poruka. Klikom na Run App (ili Shift + F10) u Android Studio IDE razvojnom okruženju, na emulatoru se pojavljuje sledeću ekran.

Slika 2.1 Simuliranje slanja SMS poruke aplikaciji na emulatoru

BroadcastReceiveromogućava aplikaciji da prima sadržaje koje šalju druge aplikacija pomoću sendBroadcast() metode. Kada je SMS poruka primljena, pokreće seonReceive() metoda koja je pre toga predefinisana. SMS poruka nosi Intent objekat, koji odgovara drugom parametru metode onReceive(),koristeći Bundleobjekat. Svaka primljena poruka poziva ponovo onReceive() metodu.

Nakon prijema, SMS poruke se smeštaju u Object polju sa formatom PDU. Polje će imati jedan element ako je poruka kraća od 160 karaktera. U suprotnom, poruka će biti podeljena na više manjih koje su smeštene u odgovarajući broj polja.

Metodom createFromPdu() klase SmsMessagepreuzima se kompletan sadržaj SMS poruke. Telefonski broj pošiljaoca daje metoda getOriginatingAdress(), a telo poruke obezbeđuje metoda getMessageBody().

Veoma važna osobina objekta tipa BroadcastReceiver jeste da on omogućava da aplikacija osluškuje dolazak SMS poruka čak i kada se ne izvršava. To, praktično znači, da će aplikacija primati SMS poruke sve vreme dok je instalirana na Android mobilnom telefonu.

SPREČAVANJE PRIJEMA SMS PORUKA UGRAĐENOM APLIKACIJOM

Prilikom prijema SMS poruke aktiviraju se sve aplikacije koje mogu da je prime.

Prilikom testiranja aktuelne aplikacije, bilo je moguće primetiti da je poslata poruka registrovana kreiranom aplikacijom, ali i ugrađenom aplikacijom u telefon (sledeća slika). Prijemom SMS poruke aktiviraju se sve aplikacije u telefonu koje mogu da je prime. Ukoliko

se želi da neka poruka ostane nedostupna ostalim aplikacijama, moraju se tražiti pogodni mehanizmi.

Slika 2.2 SMS poruka je primljena i ugrađenom SMS aplikacijom

Rešenje je veoma jednostavno. Kreirana aplikacija mora da obradi poruku pre bilo koje druge, a to se postiže tako što joj se dodeli viši prioritet primenom atributa <intent-filter> u AndroidManifest.xml datoteci. Što je veća vrednost ovog atributa (u ovom slučaju 100) to će Android ranije izvršiti ovu aplikaciju. Da bi ostale aplikacije bile onemogućene da preuzmu poruku, na kraju priložene onReceive() metode, klase prijemnika, neophodno je izvršiti metodu abortBroadcast().

Slika 2.3 Sprečavanje da druge SMS aplikacije prime željenu poruku

AŽURIRANJE AKTIVNOSTI IZ BROADCASTRECEIVER OBJEKTA

U brojnim aplikacijama se često dešava potreba da se primljena SMS poruka prosledi u glavnu aktivnost aplikacije.

U dosadašnjem izlaganju pokazano je kako objekat klase BroadcastReceiver osluškuje dolazeće SMS poruke i kako je iskorišćena Toast klasa za prikazivanje primljene SMS poruke na ekranu mobilnog uređaja. U brojnim aplikacijama se često dešava potreba da se primljena SMS poruka prosledi u glavnu aktivnost, na primer u TextViewpolje korisničkog interfejsa ili bilo koji drugi pogled koji može da prikazuje tekst. Upravo, demonstriranje navedenih funkcionalnosti predstavlja zadatak narednog izlaganja u lekciji. U prvom koraku će biti izvedene modifikacije nad datotekom korisničkog interfejsa activity_main.xml. Definisani pogled TextView, odgovarajućim XML elementom će biti rezervisan za prikazivanje sadržaja poruka.

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

    <Button
        android:id="@+id/btnSendSMS"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Pošalji SMS"
        android:onClick="onClick" />
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

U postojeću klasu prijemnik, pod nazivom SMSReceiver.java, na sam kraj metode onReceive(), neophodno je dodati sledeći kod:

```
//---Slanje namere BroadcastIntent za ažuriranje SMS iz aktivnosti---
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("SMS_RECEIVED_ACTION");
            broadcastIntent.putExtra("sms", str);
            context.sendBroadcast(broadcastIntent);
```

Zaokružena definicija klase SMSReceiver.java ima sada oblik koji je istaknut sledećim listingom, a nakon izvršene modifikacije vlastite metode onReceive().

```
package com.metropolitan.smsdemo;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;
import android.widget.Toast;

/**
 * Created by Vladimir Milicevic on 11.11.2016..
 */

public class SMSReceiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent){
        //---preuzimanje prosleđene SMS poruke---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "SMS iz ";

        if (bundle != null){
            //---učitavanje primljene SMS poruke---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i<msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                if (i==0) {
                    //---preuzimanje podataka o pošiljaocu---
                    str += msgs[i].getOriginatingAddress();
                    str += ": ";
                }
                //---preuzimanje tela poruke---
                str += msgs[i].getMessageBody().toString();
            }

            //---prikazivanje nove SMS poruke---
            Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
            Log.d("SMSReceiver", str);
```

```
           /* //---zaustavljanje slanja/primanja---
            this.abortBroadcast();*/

            //---pokretanje SMSActivity---
            Intent mainActivityIntent = new Intent(context, MainActivity.class);
            mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(mainActivityIntent);

            //---Slanje namere BroadcastIntent za ažuriranje SMS iz aktivnosti---
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("SMS_RECEIVED_ACTION");
            broadcastIntent.putExtra("sms", str);
            context.sendBroadcast(broadcastIntent);
        }
    }
}
```

KLASA AKTIVNOSTI – METODE ONRECEIVE() I ONCREATE()

Prikazivanje SMS poruke u TextView pogledu realizovano je metodom onReceive().

Sledeći zadatak predstavlja kreiranje klase aktivnosti koja će omogućiti ažuriranje aktivnosti iz BroadcastReceiver objekta. U tu svrhu biće modifikovana postojeća klasa MainActivity.java i u narednom izlaganju biće, odgovarajućim listinzima, prezentovane dopune ove klase. Sledeći kod prikazuje implementaciju metode onReceive(), u glavnoj klasi aktivnosti, zaduženu za prikazivanje SMS poruke u TextViewpogledu.

Umesto postojeće implementacije objekta intentReceiver i klase onReceive() biće postavljen sledeći kod:

```
private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            //-prikazuje primljeni SMS u TextView pogledu -
            TextView SMSes = (TextView) findViewById(R.id.textView1);
            SMSes.setText(intent.getExtras().getString("sms"));
        }
    };
```

Za učitavanje korisničkog interfejsa i namere za filtriranje SMS poruka, zadužena je metoda onCreate() koja će takođe pretrpeti izvesne modifikacije u odnosu na dosadašni kod. U onCreate() datoteku, glavne klase aktivnosti biće ugrađeni filteri za prijem SMS poruka. Navedena modifikacija prikazana je sledećim listingom.

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

```
        setContentView(R.layout.activity_main);
        sentPI = PendingIntent.getBroadcast(this, 0,
                new Intent(SENT), 0);

        deliveredPI = PendingIntent.getBroadcast(this, 0,
                new Intent(DELIVERED), 0);

        //-filter prijema SMS poruka-
        intentFilter = new IntentFilter();
        intentFilter.addAction("SMS_RECEIVED_ACTION");

    }
```

U daljem izlaganju, neophodno je redefinisati i ostale metode glavne klase aktivnosti sa ciljem davanja mogućnosti aplikaciji da rukuje ažuriranjem aktivnosti iz BroadcastReceiver objekta.

REDEFINISANJE METODE ON RECEIVE() U METODI ONRESUME()

Registrovanje prijemnika obavljeno je metodom onResume()

Sledeći zadatak jeste proširenje definicije klase aktivnosti metodom onResume() koja prvo vrši registraciju prijemnika. Nakon obavljenog posla ova metoda daje dve verzije metode onReceive() za kreiranjeBroatcastReceiver objetaka prilikom slanja i prijema SMS poruka respektivno. Kod metode onResume() prikazan je sledećim listingom.

Posebno u kodu obratiti pažnju na:

- registrovanje primaoca;
- kreiranje BroadcastReceiver - a kada je SMS poslat;kreiranje BroadcastReceiver - a kada je SMS dostavljen;
- registrovanje dva odgovarajuća BroadcastReceiver - a.

```
@Override
    public void onResume() {
        super.onResume();

        //---registrovanje primaoca---
        registerReceiver(intentReceiver, intentFilter);

        //---kreira BroadcastReceiver kada je SMS poslat---
        smsSentReceiver = new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case AppCompatActivity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS prosleđen",
                                Toast.LENGTH_SHORT).show();
                        break;
```

```
                case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                    Toast.makeText(getBaseContext(), "Generička greška",
                            Toast.LENGTH_SHORT).show();
                    break;
                case SmsManager.RESULT_ERROR_NO_SERVICE:
                    Toast.makeText(getBaseContext(), "Nema usluge",
                            Toast.LENGTH_SHORT).show();
                    break;
                case SmsManager.RESULT_ERROR_NULL_PDU:
                    Toast.makeText(getBaseContext(), "Null PDU",
                            Toast.LENGTH_SHORT).show();
                    break;
                case SmsManager.RESULT_ERROR_RADIO_OFF:
                    Toast.makeText(getBaseContext(), "Radio isključen",
                            Toast.LENGTH_SHORT).show();
                    break;
            }


        //---kreira BroadcastReceiver kada SMS dostavljen---
        smsDeliveredReceiver = new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case AppCompatActivity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS dostavljen",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case AppCompatActivity.RESULT_CANCELED:
                        Toast.makeText(getBaseContext(), "SMS nije dostavljen",
                                Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        };

        //---registruje dva BroadcastReceiver - a---
        registerReceiver(smsDeliveredReceiver, new IntentFilter(DELIVERED));
        registerReceiver(smsSentReceiver, new IntentFilter(SENT));
    }
```

UKLANJANJE PRIJEMNIKA I METODE ZA DEFINISANJE PORUKA

Metoda onPause() je zadužena za uklanjanje prijemnika.

Uklanjanje registrovanog prijemnika SMS sadržaja je u domenu metode onPause(). Sam postupak uklanjanja registrovanog prijemnika regulisan je, na veoma jednostavan način,

redefinisanjem metode onPause() i onDestroy(). Da bi metoda aktivnosti omogućila izvođenje ove funkcionalnosti, mora da poseduje kod koji je priložen sledećim listingom.

```
@Override
    public void onPause() {
        super.onPause();

        //---odjavljuje primaoca---
        unregisterReceiver(intentReceiver);

        //---odjavljuje dva BroadcastReceiver-a---
        unregisterReceiver(smsSentReceiver);
        unregisterReceiver(smsDeliveredReceiver);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        //---odjavljivanje primaoca---
        unregisterReceiver(intentReceiver);
    }
```

Za kraj klase aktivnosti neophodno je dodati još nekoliko metoda koje omogućavaju direktnu manipulaciju SMS porukama. Metode koje su, za ovu svrhu ugrađene u glavnu klasu aktivnosti, prezentovane su sledećim listingom.

```
public void onClick(View v) {
        sendSMS("5556", "Pozdravni SMS - primer!");
    }

   public void onSMSIntentClick (View v) {
        Intent i = new
                Intent(android.content.Intent.ACTION_VIEW);
        i.putExtra("address", "5556; 5558; 5560");

        i.putExtra("sms_body", "Pozdravni SMS - primer!");
        i.setType("vnd.android-dir/mms-sms");
        startActivity(i);
    }


    //Šalje poruku drugom uređaju”-
    private void sendSMS(String phoneNumber,
                         String message){
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message,
                sentPI, deliveredPI);
    }
```

AŽURIRANJE AKTIVNOSTI IZ BROADCASTRECEIVER OBJEKTA - FUNKCIONISANJE

Slanjem poruke iz DDMS ili drugog emulatora dolazi do definisanog ažuriranja glavne aktivnosti.

Najvažniji deo ovog izlaganja jeste demonstracija uvedenih funkcionalnosti rada sa SMS porukama u okviru kreirane mobilne aplikacije. Nakon izvedenih modifikacija, nad klasama i XML datotekama projekta, pristupa se ponovnom prevođenju i testiranju urađene aplikacije.

U prvom koraku kreirano je, u datoteci activity_main.xml, *TextView* polje za prikazivanje sadržaja SMS poruke. Klikom na Run app ili Shift + F10, u Nadroid Studio IDE razvojnom okruženju i pokretanjem aplikacije emulatorom moguće je videti rezultate prijema poruke poslate iz DDMS ili dugog emulatora (sledeća slika).

Slika 2.4 Primer ažurirane aktivnosti

U nastavku, modifikovana je SMSReceiver klasa koja nakon prijema poruke emituje novi Intentobjekat da bi aplikacije koje *osluškuju* pojavu ovih objekata mogle da budu obaveštene, a to će tek biti implementirano u aktivnosti.

Dalje, kreiran je BroadcastReceiverobjekat koji *osluškuje* pojavu *Intent* sadržaja (videti metodu *onReceive* i naredbu pre nje u klasi aktivnosti). Nakon primljenog *Intent* objekta ažurira se *TextView* sadržaj primljenom SMS porukom.

Takođe, neophodno je posedovanje IntentFilterobjekta koji prati pojavu konkretnog sadržaja (u ovom slučaju ***SMS_RECEIVED_ACTION*** – videti kod metode onCreate()).

U nastavku, metodom aktivnosti onResume() registrovan je BroadcastReceiver objekat koji je, nakon iskorišćenja, uklonjen metodom onPause()(videti priloženi kod metoda).

Kreirani primer ukazuje na to da će primljena SMS poruka biti dostupna u kreiranom TextView pogledu isključivo dok je glavna aktivnost prikazana na ekranu emulatora ili mobilnog telefona. Ukoliko je poruka primljena, a aktivnost nije u prvom planu, sadržaj *TextView* kontrole korisničkog interfejsa aplikacije ostaće nepromenjen.

INICIRANJE AKTIVNOSTI IZ BROADCASTRECEIVER OBJEKTA

Prijemom SMS poruke moguće je inicirati da se aktivnost iz pozadine stavi u prvi plan.

U prethodnom izlaganju je pokazano kako je moguće primljenu SMS poruku proslediti u aktivnost koja se nalazi u prvom planu. Ipak, u brojnim slučajevima, prilikom prijema SMS poruka, aktivnost može da se nalazi u pozadini i neophodno je nakon prijema poruke izvršiti njeno aktiviranje i pojavljivanje na ekranu mobilnog uređaja. Ukoliko se izvrše izvesne

modifikacije na prethodnom primeru to je moguće i ostvariti. Registrovanje prijemnika biće pomereno iz onResume() metode u onCreate() metodu klase aktivnosti.

Novi kod metode onCreate() je priložen sledećim listingom:

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sentPI = PendingIntent.getBroadcast(this, 0,
                new Intent(SENT), 0);

        deliveredPI = PendingIntent.getBroadcast(this, 0,
                new Intent(DELIVERED), 0);

        //-filter prijema SMS poruka-
        intentFilter = new IntentFilter();
        intentFilter.addAction("SMS_RECEIVED_ACTION");

        //---registrovanje primaoca---
        registerReceiver(intentReceiver, intentFilter);
    }
```

Kod modifikovane onResume() metode priložen je sledećim listingom:

```
@Override
    public void onResume() {
        super.onResume();

        /*//---registrovanje primaoca---
        registerReceiver(intentReceiver, intentFilter);*/

        //---kreira BroadcastReceiver kada je SMS poslat---
        smsSentReceiver = new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case AppCompatActivity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS prosleđen",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                        Toast.makeText(getBaseContext(), "Generička greška",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NO_SERVICE:
                        Toast.makeText(getBaseContext(), "Nema usluge",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NULL_PDU:
                        Toast.makeText(getBaseContext(), "Null PDU",
                                Toast.LENGTH_SHORT).show();
```

```
                    break;
                case SmsManager.RESULT_ERROR_RADIO_OFF:
                    Toast.makeText(getBaseContext(), "Radio isključen",
                            Toast.LENGTH_SHORT).show();
                    break;
            }


        //---kreira BroadcastReceiver kada SMS dostavljen---
        smsDeliveredReceiver = new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case AppCompatActivity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS dostavljen",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case AppCompatActivity.RESULT_CANCELED:
                        Toast.makeText(getBaseContext(), "SMS nije dostavljen",
                                Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        };

        //---registruje dva BroadcastReceiver - a---
        registerReceiver(smsDeliveredReceiver, new IntentFilter(DELIVERED));
        registerReceiver(smsSentReceiver, new IntentFilter(SENT));
    }
```

KLASA AKTIVNOSTI - UKLANJANJE PRIJEMNIKA

Metoda on receive() može biti redefinisana za kreiranje prijemnika i prilikom prijema SMS poruke.

Za kompletiranje definicije klase aktivnosti, aplikacije koja omogućava ažuriranje aktivnosti primenom BroadcastReceiver objekta, neophodno je redefinisati metode za uklanjanje prijemnika. Metodom onPause() vrši se uklanjanje kreiranih *BroadcastReceiver* objekata, a metodom onDestoy() vrši se uklanjanje prijemnika. Pre testiranja same aplikacije na nove funkcionalnosti, u klasu aktivnosti je ugrađen sledeći kod koji je u vezi sa pomenutim metodama onPause() i onDestroy().

Prvo sledi kod metode onPause():

```
@Override
    public void onPause() {
        super.onPause();

        //---odjavljuje primaoca---
```

```
        unregisterReceiver(intentReceiver);

        //---odjavljuje dva BroadcastReceiver-a---
        unregisterReceiver(smsSentReceiver);
        unregisterReceiver(smsDeliveredReceiver);
    }
```

Na kraju sledi i kod metode onDestroy():

```
@Override
    protected void onDestroy() {
        super.onDestroy();

        //---odjavljivanje primaoca---
        unregisterReceiver(intentReceiver);
    }
```

INICIRANJE AKTIVNOSTI IZ BROADCASTRECEIVER OBJEKTA - KLASA PRIJEMNIK

Metoda onReceive(), klase SMSReceiver.java, je modifikovana tako da koristi Intent objekat za postavljanje aktivnosti u prvi plan.

U novom kontekstu, posebno je bitno osvrnuti se i na zadatak kreirane klase prijemnika u folderu src projekta. Prvo što bi trebalo napomenuti je da posebnu ulogu u iniciranju aktivnosti iz *BroadcastReceiver* objekta, obavlja klasa prijemnik, koja se u konkretnom slučaju naziva SMSReceiver.java. Posebni zadaci ove klase su redefinisanje onReceive() metode, koja koristi Intentobjekat za vraćanje aktivnosti u prvi plan nakon prijema SMS poruke, i omogućavanje da se izvrši metoda startActivity() koja inicira izvršavanje aktivnosti i dovodi je u prvi plan. Intent objekat se obraća glavnoj klasi aktivnosti sa ciljem realizovanja navedene funkcionalnosti.

Ovako definisana klasa prijemnik, pod nazivom SMSReceiver.java, realizovana je sledećim konačnim kodom u koji su ugrađene sve navedene modifikacije:

```
package com.metropolitan.smsdemo;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;
import android.widget.Toast;

/**
 * Created by Vladimir Milicevic on 11.11.2016..
 */
```

```
public class SMSReceiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent){
        //---preuzimanje prosleđene SMS poruke---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "SMS iz ";

        if (bundle != null){
            //---učitavanje primljene SMS poruke---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i<msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                if (i==0) {
                    //---preuzimanje podataka o pošiljaocu---
                    str += msgs[i].getOriginatingAddress();
                    str += ": ";
                }
                //---preuzimanje tela poruke---
                str += msgs[i].getMessageBody().toString();
            }

            //---prikazivanje nove SMS poruke---
            Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
            Log.d("SMSReceiver", str);

           /* //---zaustavljanje slanja/primanja---
            this.abortBroadcast();*/

            //---pokretanje MainActivity---
            Intent mainActivityIntent = new Intent(context, MainActivity.class);
            mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(mainActivityIntent);

            //---Slanje namere BroadcastIntent za ažuriranje SMS iz aktivnosti---
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("SMS_RECEIVED_ACTION");
            broadcastIntent.putExtra("sms", str);
            context.sendBroadcast(broadcastIntent);
        }
    }
}
```

Posebnu pažnju obratiti na kod koji sledi ispod komentara //---pokretanje MainActivity--- . Ovim kodom je realizovano iniciranje aktivnosti i njeno dovođenje u prvi plan.

INICIRANJE AKTIVNOSTI IZ BROADCASTRECEIVER OBJEKTA - FUNKCIONISANJE

Za iniciranje aktivnosti iz BroadcastReceiver objekta neophodno je modifikovati i AndroidManifest.xml datoteku.

Od ključnog značaja za dovođenje aktivnosti iz pozadine u prvi plan, prilikom prijema SMS poruke, jeste registrovanje BroadcastReceiverobjekta u onCreate() metodi klase aktivnosti aplikacije umesto u metodi onResume()iste klase. Takođe, umesto metodi onPause(), uklanjanje objekta prepušteno je metodi onDestroy().Na ovaj način je obezbeđeno da, i kada se aktivnost izvršava u pozadini, ona prati pojavu Intent objektaka.

Dalje, metoda onReceive(), klase SMSReceiver.java, je modifikovana tako da koristi Intent objekat za postavljanje aktivnosti u prvi plan pre nego što se emituje neki drugi Intentobjekat (videti priloženi kod metode). Metoda startActivity() inicira izvršavanje aktivnosti i dovodi je u prvi plan. Takođe, moguće je primetiti da izvršavanje navedene metode, izvan konteksta aktivnosti, zahteva korišćenje indikatora ***FLAG_ACTIVITY_NEW_TASK***.

Takođe, neophodno je izvršiti i modifikaciju u AndroidManifet.xml datoteci. Da bi iniciranje aktivnosti bilo moguće, potrebno je u <activity> elementu postaviti ***launchMode*** atribut čija je vrednost ***singleTask***. Ukoliko se ovo izostavi, sa svakim prijemom SMS poruke iniciraće se više instanci aktivnosti.

Sledećom slikom prikazana je modifikacija u AndroidManifest.xml datoteci.

Slika 2.5 Modifikacija singleTask u AndroidManifest.xml datoteci

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 2 - VEŽBANJE PRIJEMA I SLANJA SMS PORUKA KREIRANOM APLIKACIJOM.

Kreira se aplikacija sa funkcionalnostima prijema i slanja SMS poruka

Kreirati Android aplikaciju po sledećim zahtevima:

1. Aplikacija ima mogućnost slanja i prijema SMS poruka;
2. Realizovati odgovarajuće privilegije;
3. Po potrebi isključiti ostale SMS klijente za prijem poruka;
4. Po potrebi SMS se može poslati ugrađenom aplikacijom;
5. Realizovati klasu naslednicu osnovne klase BroadcastReceiver za podršku prijemu SMS poruka;
6. Realizovati mehanizam za prikazivanje primljene porke.

U Prvom koraku u AndroidManifest.xml datoteci je neophodno dodati sledeće privilegije:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
```

U nastavku je neophodno dodati datoteku kojom je određen korisnički interfejs:

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

    <Button
        android:id="@+id/btnSendSMS"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Pošalji SMS"
        android:onClick="onClick" />
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <!--
    <Button
        android:id="@+id/btnSendIntentSMS"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Pošalji SMS koristeći nemere"
        android:onClick="onSMSIntentClick" />



    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <requestFocus />
    </EditText>
    - -->
</LinearLayout>
```

Sledi klasa koja nasleđuje BroadcastReceiver klasu za podršku prijemu SMS poruka:

```
package com.metropolitan.smsdemo;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;
import android.widget.Toast;
```

```
/**
 * Created by Vladimir Milicevic on 11.11.2016..
 */

public class SMSReceiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent){
        //---preuzimanje prosleđene SMS poruke---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "SMS iz ";

        if (bundle != null){
            //---učitavanje primljene SMS poruke---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i<msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                if (i==0) {
                    //---preuzimanje podataka o pošiljaocu---
                    str += msgs[i].getOriginatingAddress();
                    str += ": ";
                }
                //---preuzimanje tela poruke---
                str += msgs[i].getMessageBody().toString();
            }

            //---prikazivanje nove SMS poruke---
            Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
            Log.d("SMSReceiver", str);

           /* //---zaustavljanje slanja/primanja---
            this.abortBroadcast();*/

            //---pokretanje SMSActivity---
            Intent mainActivityIntent = new Intent(context, MainActivity.class);
            mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(mainActivityIntent);

            //---Slanje namere BroadcastIntent za ažuriranje SMS iz aktivnosti---
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("SMS_RECEIVED_ACTION");
            broadcastIntent.putExtra("sms", str);
            context.sendBroadcast(broadcastIntent);
        }
    }
}
```

PRIMER 2 - KLASA AKTIVNOSTI

Kreiranje klase aktivnosti

Konačno, projekat će biti zaokružen klasom aktivnosti koja po potrebi može da uključuje različite funkcionalnosti. One koje se trenutno ne koriste, biće prikazane u formi komentara. Sledi kod klase:

```
package com.metropolitan.smsdemo;

import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.TextView;
import android.content.Intent;
import android.content.Context;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    String SENT = "SMS_SENT";
    String DELIVERED = "SMS_DELIVERED";
    PendingIntent sentPI, deliveredPI;
    BroadcastReceiver smsSentReceiver, smsDeliveredReceiver;
    IntentFilter intentFilter;

    /*private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            //-prikazuje primljeni SMS u TextView-
            TextView SMSes = (TextView) findViewById(R.id.textView1);
            SMSes.setText(intent.getExtras().getString("sms"));
        }
    };*/

    private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            //-prikazuje primljeni SMS u TextView pogledu -
            TextView SMSes = (TextView) findViewById(R.id.textView1);
            SMSes.setText(intent.getExtras().getString("sms"));
        }
    };


    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sentPI = PendingIntent.getBroadcast(this, 0,
                new Intent(SENT), 0);

        deliveredPI = PendingIntent.getBroadcast(this, 0,
                new Intent(DELIVERED), 0);

        //-filter prijema SMS poruka-
        intentFilter = new IntentFilter();
        intentFilter.addAction("SMS_RECEIVED_ACTION");

        //---registrovanje primaoca---
        registerReceiver(intentReceiver, intentFilter);
    }

    @Override
    public void onResume() {
        super.onResume();

        /*//---registrovanje primaoca---
        registerReceiver(intentReceiver, intentFilter);*/

        //---kreira BroadcastReceiver kada je SMS poslat---
        smsSentReceiver = new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case AppCompatActivity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS prosleđen",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                        Toast.makeText(getBaseContext(), "Generička greška",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NO_SERVICE:
                        Toast.makeText(getBaseContext(), "Nema usluge",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NULL_PDU:
                        Toast.makeText(getBaseContext(), "Null PDU",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_RADIO_OFF:
                        Toast.makeText(getBaseContext(), "Radio isključen",
                                Toast.LENGTH_SHORT).show();
                        break;
                }


        //---kreira BroadcastReceiver kada SMS dostavljen---
```

```
        smsDeliveredReceiver = new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case AppCompatActivity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS dostavljen",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case AppCompatActivity.RESULT_CANCELED:
                        Toast.makeText(getBaseContext(), "SMS nije dostavljen",
                                Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        };

        //---registruje dva BroadcastReceiver - a---
        registerReceiver(smsDeliveredReceiver, new IntentFilter(DELIVERED));
        registerReceiver(smsSentReceiver, new IntentFilter(SENT));
    }

    @Override
    public void onPause() {
        super.onPause();

        //---odjavljuje primaoca---
        unregisterReceiver(intentReceiver);

        //---odjavljuje dva BroadcastReceiver-a---
        unregisterReceiver(smsSentReceiver);
        unregisterReceiver(smsDeliveredReceiver);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        //---odjavljivanje primaoca---
        unregisterReceiver(intentReceiver);
    }



    public void onClick(View v) {
        sendSMS("5556", "Pozdravni SMS - primer!");
    }

   public void onSMSIntentClick (View v) {
        Intent i = new
                Intent(android.content.Intent.ACTION_VIEW);
        i.putExtra("address", "5556; 5558; 5560");
```

```
        i.putExtra("sms_body", "Pozdravni SMS - primer!");
        i.setType("vnd.android-dir/mms-sms");
        startActivity(i);
    }


    //Šalje poruku drugom uređaju”-
    private void sendSMS(String phoneNumber,
                         String message){
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message,
                sentPI, deliveredPI);
    }

}
```

ZADATAK 2 - ANALIZIRAJTE DOZVOLE ZA SLANJE I PRIJEM SMS PORUKA

Pokušajte sami!!!

1. Pogledajte pažljivo priloženi listing i dajte njegovo objašnjenje.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
```

2. Pogledajte pažljivo priloženi listing i dajte njegovo objašnjenje.

```
<receiver android:name=".SMSReceiver">
            <intent-filter android:priority="100">
                <action android:name=
                    "android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>
```

Poglavlje 3

Slanje e-mail poruka iz aplikacije

SLANJE E-MAIL PORUKA – OSNOVNA RAZMATRANJA

Da bi e-mail poruka bila poslata iz vlastite aplikacije, nije potrebno kreiranje e-mail klijenta ispočetka.

U narednom izlaganju, neophodno je obraditi mehanizme i postupke kojima je moguće kreiranoj Android aplikaciji dati mogućnost manipulisanja elektronskom poštom iz vlastite aktivnosti.

Pod *e-mail* porukom podrazumeva se poruka koja je elektronskim putem razmenjena izmeću jednog pošiljaoca, sa jedne, i jednog ili više primalaca, sa druge strane. Pre pokretanja e-mail aktivnosti neophodno je razumeti e-mail funkcionalnost i namere. Namera predstavlja prenos podataka sa jedne komponente na drugu, ugrađenom ili vlastitom aplikacijom.

Da bi e-mail poruka bila poslata iz vlastite aplikacije, nije potrebno kreiranje *e-mail klijenta* ispočetka već je moguće osloniti se na već postojeće aplikacije instalirane na Android, poput: *gmail, outlook, K-9* itd. U svrhu demonstracije ove funkcionalnosti biće kreirana aktivnost koja će pokrenuti ugrađeni e-mail klijent odgovarajućom akcijom i podacima. Drugim rečima, iz aplikacije će biti prosleđena e-mail poruka primenom *Intent* objekta za pokretanje ugrađenog e-mail klijenta.

Realizacija navedenih zahteva aplikacije sledi u narednom izlaganju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

SLANJE E-MAIL PORUKA – DATOTEKA KORISNIČKOG INTERFEJSA

Datotekom main.xml biće definisan UI primera za slanje e-mail poruka iz aplikacije.

Prvi korak u kreiranju vlastite aplikacije za slanje e-mail poruke jeste kreiranje datoteke koja će nostiti podatke o korisničkom interfejsu. Datoteka activity_main.xml, pored podataka o rasporedu komponenata na ekranu, nosi podatak o kontroli dugme čijim klikom će započeti proces slanja e-mail poruke.

Sledećom slikom je prikazan izgled glavne aktivnosti sa učitanim korisničkim interfejsom spremnim na reakciju korisnika tj, klik na dugme koje će inicirati slanje elektronske pošte.

Slika 3.1 Učitana glavna aktivnost aplikacije

U narednom izlaganju je neophodno priložiti programski kod kojim je realizovan prikazani korisnički interfejs glavne aktivnosti aplikacije-

Sledećim kodom prikazana je activity_main.xml datoteka primera slanja e-mail poruka iz aplikacije.

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

    <Button
        android:id="@+id/btnSendEmail"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Pošalji e-mail"
        android:onClick="onClick" />

</LinearLayout>
```

SLANJE E-MAIL PORUKA – KLASA AKTIVNOSTI APLIKACIJE

Klasa aktivnosti implementira metodu sendEmail().

Sledeći zadatak predstavlja kreiranje klase aktivnosti projekta. Klasa će implementirati dve metode onClick() za upravljanje kontrolom korisničkog interfejsa i sendEmail() za slanje e-mail poruke na drugi uređaj. Klasa će koristiti Intent objekat za pokretanje ugrađene aplikacije za razmenu e-mail poruka. Iz navedenog razloga je neophodno pre testiranja programa podesiti vlastiti e-mail nalog u ugrađenoj aplikaciji uređaja. Bez navedene aktivnosti, ugrađena aplikacije neće biti u mogućnosti da prosledi e - mail poruku, a samim tim, na indirektan način, i kreirana aplikacija će biti onemogućena da šalje poruke

Sledećom listingom su prikazani paket sa odgovarajućim klasama koje su neophodne za funkcionisanje aplikacije.

```
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
```

Navedene zahteve aplikacije nije moguće realizovati bez glavne klase aktivnosti. Klasa aktivnosti sa metodom onCreate(), za učitavanje korisničkog interfejsa, kao i metodama onClick(), reakcija na klik dugmeta, i sendEmail(), kreiranje Intent objekta i obraćanje

ugrađenom email klijentu za slanje definisane e-mail poruke, odgovara sledećem priloženom listingu:

```
package com.metropolitan.emaildemo;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    //priložene adrese, prilikom testiranja, zameniti pravim
    public void onClick(View v) {
        String[] to =
                {"vladam.kg@email.com",
                        "osoba2@email.com"};
        String[] cc = {"osoba3@email.com"};
        sendEmail(to, cc, "Pozdrav!!!", "Pozdrav društvo!!!");
    }
    //-slanje e-mail poruke na drugi uređaja”-
    private void sendEmail(String[] emailAddresses, String[] carbonCopies,
                           String subject, String message)
    {
        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.setData(Uri.parse("mailto:"));
        String[] to = emailAddresses;
        String[] cc = carbonCopies;
        emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
        emailIntent.putExtra(Intent.EXTRA_CC, cc);
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
        emailIntent.putExtra(Intent.EXTRA_TEXT, message);
        emailIntent.setType("message/rfc822");
        startActivity(Intent.createChooser(emailIntent, "Email"));
    }
}
```

SLANJE E-MAIL PORUKA – TESTIRANJE I DEMONSTRACIJA

Primer startuje ugrađenu e-mail aplikaciju da bi realizovao slanje poruke.

U nastavku je neophodno izvršiti testiranje kreirane aplikacije. U Android Studio IDE razvojnom okruženju klikom na Shift + F10 ili Run app pokreće se aplikacija u izabranom emulatoru i spremna je za testiranje. Nakon podešavanja vlastitog e-mail naloga, klikom na dugme *Pošalji e-mail, u emulatoru* (ili mobilnom uređaju) pokreće se ugrađena e-mail aplikacija sa podacima o pošiljaocu (sledeća slika), primaocima, temi i tekstu poruke definisanim u metodi onClick()klase aktivnosti.

Slika 3.2 Angažovanje e-mail aplikacije

U nastavku je neophodno analizirati kako je omogućeno da ugrađena aplikacija pošalje predefinisanu e-mail poruku iz koda klase aktivnosti.

Ugrađena aplikacija je pokrenuta Intent objektom uz definisanje različitih parametara pomoću metoda: setData(), putExtra() i setType(). Poziv metoda i kreiranjeIntent objekta izolovano je sledećim kodom metode sendEmail() glavne klase aktivnosti.

```
//-slanje e-mail poruke na drugi uređaj”-
    private void sendEmail(String[] emailAddresses, String[] carbonCopies,
                           String subject, String message)
    {
        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.setData(Uri.parse("mailto:"));
        String[] to = emailAddresses;
        String[] cc = carbonCopies;
        emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
        emailIntent.putExtra(Intent.EXTRA_CC, cc);
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
        emailIntent.putExtra(Intent.EXTRA_TEXT, message);
        emailIntent.setType("message/rfc822");
        startActivity(Intent.createChooser(emailIntent, "Email"));
    }
```

Ugrađena aplikacija, kreiranog emulatora, priložene adrese dodaje u polje To kao listu adresa na koje se poruka šalje. Ukoliko je navedeno, neke adrese mogu biti spakovane u polje Cc sa koga će biti označene kao lokacije na koje se takođe prosleđuje poruka predefinisana u programskom kodu aplikacije, odnosno, njene klase aktivnosti. Sada je sve spremno za kompletirane postupka slanja e-mail poruke.

Sledećom slikom je prikazan prijem poruke koja je prosleđena korišćenjem kreirane aplikacije.

Slika 3.3 Primljena e-mail poruka

ZADATAK 2 - VEŽBANJE SLANJA E-MAIL PORUKE IZ APLIKACIJE

Kreira se apliakcija koja ima mogućnost slanja e-mail poruke iz aplikacije

Zadatak:

Simulirati malu kancelarijsku aplikaciju koja ima mogućnost prikazivanja časovnika i slanja e-mail poruke.

Kod korisničkog interfejsa priložen je sledećom datotekom:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="@drawable/metpozadina">

    <Button
        android:id="@+id/btnSendEmail"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Pošalji e-mail"
        android:onClick="onClick" />

    <AnalogClock
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

Glavna klasa aktivnosti je priložena sledećim kodom:

```
package com.metropolitan.emaildemo;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    //priložene adrese, prilikom testiranja, zameniti pravim
    public void onClick(View v) {
        String[] to =
                {"vladam.kg@email.com",
                        "osoba2@email.com"};
        String[] cc = {"osoba3@email.com"};
        sendEmail(to, cc, "Pozdrav!!!", "Pozdrav društvo!!!");
    }
    //-slanje e-mail poruke na drugi uređaj”-
    private void sendEmail(String[] emailAddresses, String[] carbonCopies,
```

```
                            String subject, String message)
    {
        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.setData(Uri.parse("mailto:"));
        String[] to = emailAddresses;
        String[] cc = carbonCopies;
        emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
        emailIntent.putExtra(Intent.EXTRA_CC, cc);
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
        emailIntent.putExtra(Intent.EXTRA_TEXT, message);
        emailIntent.setType("message/rfc822");
        startActivity(Intent.createChooser(emailIntent, "Email"));
    }
}
```

Glavna aktivnost aplikacije može bit prikazana sledećom slikom:

Slika 3.4 Predloženi izgled aplikacije

ZADATAK 3 - PRERADITE ZADATAK IZ PRETHODNE LEKCIJE

Pokušajte sami!!!

Zadatak za samostalni rad iz prethodne lekcije dopunite mogućnošću slanja sms poruka na broj izabran iz kreirane liste kontakata.

Poglavlje 4

Domaći zadatak 7

ZADATAK 1

Vežbanje slanja i primanja sms poruka, kao i slanja e-mail poruka kreiranim aplikacijama

Kreirati Android aplikaciju na osnovu sledećih zahteva:

1. Kreirati korisnički interfejs, sa slobodnim rasporedom, koji sadrži kontrole: *EditText, ImageButton1, ImageButton2*.
2. U *ImageBatton1* je smeštena slika sa oznakom *SMS*, u *ImageButton2* je smeštena slika sa oznakom *e-mail*;
3. U AndroidManifest.xml datoteku ugraditi odgovarajuće privilegije za prijem i slanje SMS poruka;
4. Realizovati klasu prijemnik SMS poruka;
5. Klasu aktivnosti *osposobiti* da šalje SMS i e-mail poruke;
6. Sadržaj poruka koje se šalju dodaje se u kontrolu *EditText*;
7. Klikom na odgovarajuće dugme šalje se kao SMS ili e-mail na unapred određene adrese;
8. Klikom na dugme *ImageButton2* otvara se ugrađena aplikacija koja preuzima posao slanja poruke;
9. Primljena poruka zamenjuje sadržaj polja *EditText* i zahvaljujući definisanom prioritetu prikazuje se samo u kreiranoj aplikaciji;
10. Na ekranu uređaja, *Toast* porukom naglasiti da je primljena poruka i od koga.

Pregled Lekcije08

PREGLED LEKCIJE 07

Lekcija je prikazala dva ključna načina koja omogućavaju da aplikacija komunicira sa okolinom.

Lekcijom 07 obrađene su tri celine:

- *Slanje SMS poruka iz kreirane aplikacije;*
- *Prijem SMS poruka kreiranom aplikacijom;*
- *Slanje e-mail poruka kreiranom aplikacijom.*

Lekcija je na taj način prikazala dva ključna načina koja omogućavaju da aplikacija komunicira sa okolinom. Prvo je naučeno kako se šalju i primaju SMS poruke i na taj način su date smernice za kreiranje različitih aplikacija koje mogu da koriste ove funkcionalnosti na brojne načine. Takođe, pokazano je da aplikacija zahteva izvesne privilegije da bi joj bilo dozvoljeno slanje i primanje SMS poruka. Konkretne privilegije su ugrađene u AndroidManifest.xml datoteku.

U nastavku, naučeno je kako se šalju elektronske poruke iz kreirane aplikacije. Primenom *Intent* objekta, kreirana aplikacija prosleđuje podatke ugrađenoj e-mail aplikaciji, neophodne za realizaciju slanja e-mail poruka ka krajnjem odredištu. Navedeni podaci odnose se na: pošiljaoca, primaoce, temu i sadržaj poruke.

Nakon savladavanja ove lekcije, studenti su potpuno ovladali primenom mehanizama slanja i prijema SMS poruka u Android aplikacijama. Takođe, pokazani način slanja e - mail poruka daje studentima odličnu osnovu za kreiranja širokog spektra Android aplikacija. Ugrađivanjem kalendara i časovnika, zajedno sa navedenim funkcionalnostima, moguće je kreirati planer aplikaciju ili bilo koju drugu aplikaciju za primenu, na primer, u kancelarijskom poslovanju.

LITERATURA

Za pripremanje lekcije korišćena je sledeća literatura

1. Lee W. M. 2012. *Android 4 – razvoj aplikacija*, Wiley Publishing, INC
2. http://developer.android.com/training/index.html
3. http://www.tutorialspoint.com/android/
4. http://www.vogella.com/tutorials/android.html