Rad sa porukama u Android operativnom sistemu

Doc. dr Vladimir Milićević

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 kodom;
- Šalu SMS pozivajući ugrađenu Android aplikaciju.

Prvi zadatak će biti savladavanje prosleđivanja i prijema SMS poruka kreiranim programskim kodom. XML datoteka korisničkog interfejsa primera, prikazana je sledećim kodom.

```
<?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>
```

Slika-1 main.xml datoteka primera

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 AndroidManifest.xml datoteci projekta, ugraditi odgovarajuće privilegije. Privilegija se pakuje u XML element <*uses-permission …*> i ima zadatak da dozvoli aplikaciji slanje i prijem SMS poruka.

Kod AndroidManifest.xml datoteke prikazan je sledećom slikom, a SMS privilegija je istaknuta posebno.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.SMS"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" />
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".SMSActivity"
            android:launchMode="singleTask" >
            <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>
```

Slika-2 AndroidManifest.xml sa SMS privilegijama

KLASA AKTIVNOSTI ZA SLANJE SMS PORUKE

Instrukcije za slanje SMS poruka ugrađene su klasu aktivnosti aplikacije.

Nakon kreiranja korisničkog interfejsa i davanja privilegija aplikaciji za rad sa SMS porukama, pristupa se kreiranju klase aktivnosti aplikacije. Klasa aktivnosti će sadržati izvesne funkcionalnosti koje omogućavaju rad sa porukama. U početku, to će biti jednostavno prosleđivanje SMS poruke. Poruka je definisana i prosleđena na unapred određeni broj telefona u okviru programskog koda. Programski kod, klase aktivnosti sa navedenim funkcionalnostima, priložen je sledećom slikom.

Posebno je istaknut deo koda koji se odnosi na slanje SMS poruke.

```
package net.learn2develop.SMS;

import android.app.Activity;
import android.os.Bundle;

import android.telephony.SmsManager;
import android.view.View;

public class SMSActivity extends Activity {
   /* Poziva se kada se aktivnost kreira. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.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 );
    }
}
```

Slika-3 Klasa aktivnosti i SMS funkcionalnost

PROGRAMSKO SLANJE SMS PORUKA - FUNKCIONISANJE

Android operativni sistem primenjuje politiku zasnovanu na privilegijama.

Za prevođenje i pokretanje aplikacije, podrazumevanim emulatorom (5554) neophodno je kliknuti na F11. 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.

Slika-4 Emulator prima SMS poruku

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.

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()*.

```
package net.learn2develop.SMS;
import android.app.Activity;
public class SMSActivity extends Activity {
    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"));
        }
    };
    /* Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.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 Activity.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 Activity.RESULT_OK:
            Toast.makeText(getBaseContext(), "SMS dostavljen",
                    Toast.LENGTH_SHORT).show();
            break;
        case Activity.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);
}
}
```

```
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;

import android.telephony.SmsManager;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
```

Slika-5 Proširena klasa aktivnosti

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).

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

PendingIntent i *BroadcastReceiver* objekti, izdvojeni su sledećim kodom.

```
    //PendingIntent objekti
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));
```

Slika-6 PendingIntent i BroadcastReceiver objekti

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

Kreirani *PendingIntent* objekti se pakuju u metodu *sendTextMessage()* kao poslednja dva argumenta (izdvojeno sledećom slikom), obezbeđujući na taj način informaciju da li je poruka ispravno prosleđena ili ne.

Na kraju, metodom *onPause()* prekida se korišćenje kreiranih *BroadcastReceiver objekata* (izdvojeno sledećom slikom).

```
   //PendingIntent objekti kao argumenti metode
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(phoneNumber, null, message,
    sentPI, deliveredPI);

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

Slika-7 Delovi metoda sendTextMessage() i onPause()

SLANJE PORUKA POMOĆU NAMERA

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

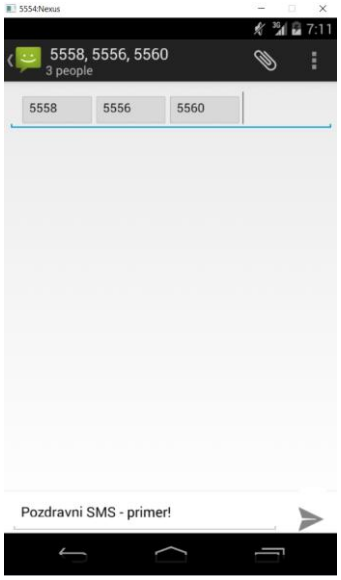
SmsManager klasa omogućava slanje SMS poruka iz aplikacije bez oslanjanja na sistemski ugrađenu Android aplikaciju. Međutim, ponekad je jednostavnije dozvoliti ugrađenoj aplikaciji da obavi kompletan posao slanja (ili prijema) SMS poruka, pri čemu kreirana aplikacija vrši kontrolu nad celokupnim poslom.

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);
}
```

Slika-8 Metoda onSMSIntentClick()

U navedenoj metodi smešten je i poziv metode *putExtra()* 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-9 Poziv ugrađene SMS aplikacije

UVOD U PRIJEM SMS PORUKA

Primenom BroadcastReceiver objekta moguće je primati SMS poruke kreiranom aplikacijom.

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.

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.

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

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

Slika-1 Dozvola i klasa prijemnik

PRIJEM SMS PORUKA – KLASA PRIJEMNIK

Klasa prijemnik je članica src foldera zajedno sa klasom aktivnosti.

U folder *src* dodaje se nova klasa pod imenom *SMSReceiver* koja nasleđuje klasu *BroadcastReceiver*. *Kod klase SMSReceiver* priložen je sledećom slikom.

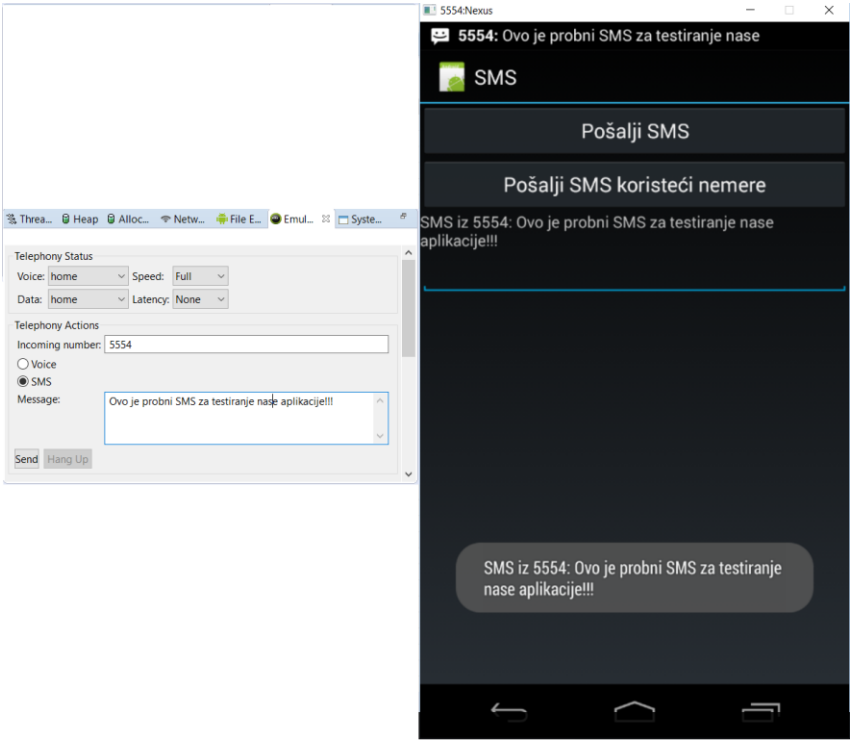
```
package net.learn2develop.SMS;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;
import android.widget.Toast;
public class SMSReceiver extends BroadcastReceiver{
    @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);

              }
          }
      }
```

Slika-2 Klasa prijemnik

KLASA PRIJEMNIK - FUNKCIONISANJE

Svaka primljena poruka poziva ponovo onReceive() metodu.

Klikom na F11, aplikacija je pokrenuta emulatorom, a DDMS režim je iskorišćen da se emulatoru pošalje SMS poruka (sledeća slika).



Slika-3 Prijem SMS emulatorom

BroadcastReceiver omogućava aplikaciji da prima sadržaje koje šalju druge aplikacija pomoću *sendBroadcast()* metode. Kada je SMS poruka primljena, pokreće se *onReceive()* metoda koja je pre toga predefinisana. SMS poruka nosi *Intent* objekat, koji odgovara drugom parametru metode *onReceive(),* koristeći *Bundle* objekat. 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.

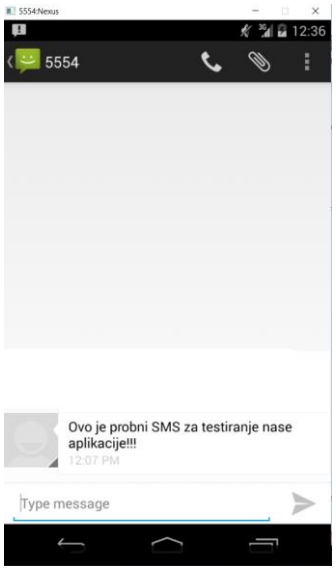
Metodom *createFromPdu()* klase *SmsMessage* preuzima se kompletan sadržaj SMS poruke. Telefonski broj pošiljaoca daje metoda *getOriginatingAdress()*, a telo poruke obezbeđuje metoda *getMessageBody()*.

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-4 Prijem ugrađenom 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()*.

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

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

Slika-5 Sprečavanje prijema SMS drugim aplikacijama

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 *TextView* polje korisničkog interfejsa. Sledećim primerom će to biti demonstrirano, a polazi se od main.xml datoteke.

```
<?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" />

</LinearLayout>
```

Slika-6 UI sa TextView za prikaz SMS poruke

U postojeću klasu prijemnik, pod nazivom *SMSReceiver.java*, na sam kraj metode *onReceive()*, neophodno je dodati obeleženi kod sa sledeće slike.

```
@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);

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

Slika-7 Intent objekat za ažuriranje SMS iz aktivnosti

KLASA AKTIVNOSTI – METODE ONRECEIVE() I ONCREATE()

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

Sledeći zadatak predstavlja kreiranje klase aktivnosti koja će omogućiti ažuriranje aktivnosti iz *BroadcastReceiver* objekta. Sledeći kod prikazuje pakete neophodne za korišćenje relevantih objekata i metoda u klasi aktivnosti i metodu *onReceive()* zaduženu za prikazivanje SMS poruke u *TextView* pogledu.

```
package net.learn2develop.SMS;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;

import android.telephony.SmsManager;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
public class SMSActivity extends Activity {
    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"));
        }
    };
```

Slika-8 Prikazivanje SMS u TextView pogledu

Za učitavanje korisničkog interfejsa i namere za filtriranje SMS poruka, zadužena je metoda *onCreate()* čiji je kod dat sledećom slikom.

```
/* Poziva se kada se aktivnost kreira. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.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");



}
```

Slika-9 Učitavanje UI i filtriranje SMS

PREDEFINISANJE 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 kreiranje *BroatcastReceiver* objetaka prilikom slanja i prijema SMS poruka respektivno. Kod metode *onResume()* prikazan je sledećom slikom.

```
@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 Activity.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 Activity.RESULT_OK:
                Toast.makeText(getBaseContext(), "SMS dostavljen",
                        Toast.LENGTH_SHORT).show();
                break;
            case Activity.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));
}
```

Slika-10 Implementacija onResume() sa onReceive()

UKLANJANJE PRIJEMNIKA I METODE ZA DEFINISANJE PORUKA

Metoda onPause() je zadužena za uklanjanje prijemnika.

Uklanjanje registrovanog prijemnika regulisano je predefinisanjem metode *onPause()* na način prikazan sledećom slikom.

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

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

}
```

Slika-11 Uklanjanje prijemnika

Za kraj klase aktivnosti neophodno je dodati još nekoliko metoda koje omogućavaju direktnu manipulaciju SMS porukama. Kod metoda prikazan je sledećom slikom.

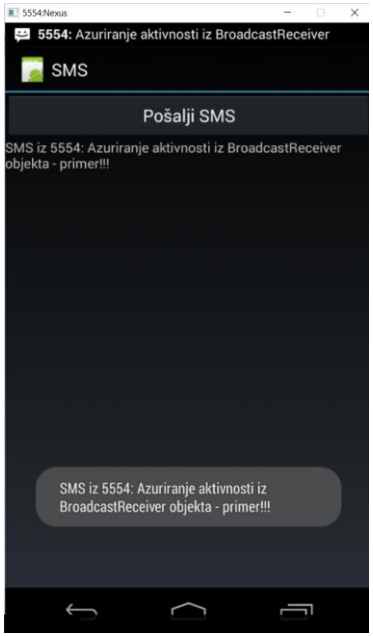
```
public void onClick(View v) {
    sendSMS("5556", "Pozdravni SMS - primer!");
}

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);
}
}
```

Slika-12 Manipulisanje porukama

AŽURIRANJE AKTIVNOSTI IZ BROADCASTRECEIVER OBJEKTA - FUNKCIONISANJE

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

U prvom koraku kreirano je, u datoteci main.xml, *TextView* polje za prikazivanje sadržaja SMS poruke. Klikom na F11 i pokretanjem aplikacije emulatorom moguće je videti rezultate prijema poruke poslate iz DDMS ili dugog emulatora (sledeća slika).



Slika-13 Ažurirana aktivnost - demonstracija

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

U nastavku, kreiran je *BroadcastReceiver* objekat 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 *IntentFilter* objekta 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.

```
/* Poziva se kada se aktivnost kreira. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.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();

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

Slika-14 Migracija registrovanja prijemnika

KLASA AKTIVNOSTI - UKLANJANJE PRIJEMNIKA

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

Za kompletiranje definicije klase aktivnosti, aplikacije koja omogućava ažuriranje aktivnosti primenom *BroadcastReceiver* objekta, neophodno je predefinisati metode za uklanjanje prijemnika. Metodom *onPause()* vrši se uklanjanje kreiranih *BroadcastReceiver* objekata, a metodom *onDestoy()* vrši se uklanjanje prijemnika. Kod navedenih metoda, prikazan je sledećom slikom.

```
@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);
}
```

Slika-15 Uklanjanje prijemnika

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.

Posebnu ulogu u iniciranju aktivnosti iz *BroadcastReceiver* objekta, obavlja klasa prijemnik, koja se u konkretnom slučaju naziva *SMSReceiver.java*. Posebni zadaci ove klase su predefinisanje *onReceive()* metode, koja koristi *Intent* objekat 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.

Modifikovan kod klase prijemnika, prikazan je sledećom slikom.

```
package net.learn2develop.SMS;
import android.content.BroadcastReceiver;
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);
            //---pokretanje SMSActivity---
            Intent mainActivityIntent = new Intent(context, SMSActivity.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);
        }
    }
}
```

Slika-16 Modifikovana klasa prijemnik

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 *BroadcastReceiver* objekta 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 *Intent* objekat (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.

```
<activity
    android:label="@string/app_name"
    android:name=".SMSActivity"
    android:launchMode="singleTask" >
```

Slika-17 Modifikacija u AndroidManifest.xml

SLANJE E-MAIL PORUKA – OSNOVNA RAZMATRANJA

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

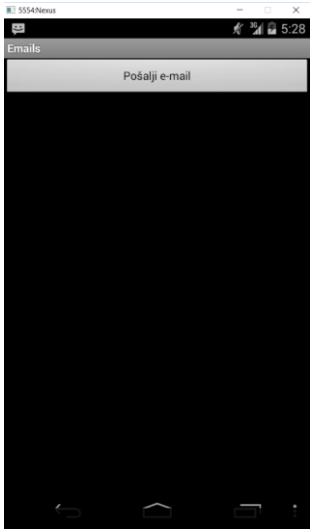
Pod e-mail porukom podrazumeva se poruka koja je elektronskim putem razmenjena izmeću jednog pošiljaoca, sa jedne, i jednog ili više primalaca, sa druge strane. Pre pokretanja e-mail aktivnosti neophodno je razumeti e-mail funkcionalnost i namere. Namera predstavlja prenos podataka sa jedne komponente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

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 main.xml, pored podataka o rasporedu komponenata na ekranu, nosi podatak o kontroli *dugme* čijim klikom će započeti proces slanja e-mail poruke.



Slika-1 UI primera

Sledećim kodom prikazana je 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>
```

Slika-2 XML datoteka korisničkog interfejsa

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.

Sledećom slikom su prikazani paketi koji su neophodni za funkcionisanje aplikacije.

```
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
```

Slika-3 Uključeni paketi aplikacije

Sledećim kodom data je klasa aktivnosti.

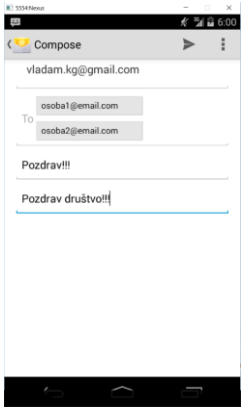
```
package net.learn2develop.Emails;
import android.app.Activity;
public class EmailsActivity extends Activity {
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    //priložene adrese, prilikom testiranja, zameniti pravim
    public void onClick(View v) {
        String[] to =
            {"osoba1@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"));
    }
}
```

Slika-4 Klasa aktivnosti primera

SLANJE E-MAIL PORUKA – TESTIRANJE I DEMONSTRACIJA

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

Klikom na F11 pokreće se aplikacija u 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.



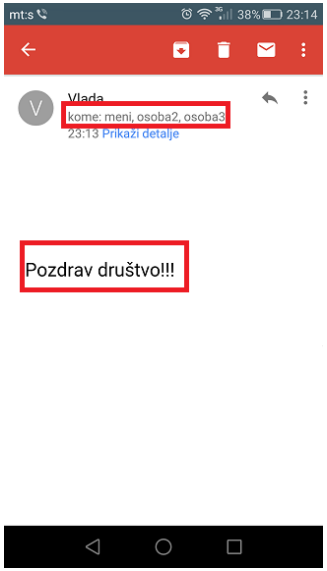
Slika-5 Ugrađena aplikacija

Ugrađena aplikacija je pokrenuta *Intent* objektom uz definisanje različitih parametara pomoću metoda: *setData()*, *putExtra()* i *setType()*. Poziv metoda i kreiranje *Intent* objekta izolovano je sledećim kodom.

```
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"));
```

Slika-6 Intent objekat

Ugrađena aplikacija, kreiranog emulatora, ne poseduje opciju *CC* već dodatne primaoce e-mail poruke postavlja u red iza primarnog primaoca. Međutim, efekat je isti kao da su upisani u zasebno *CC* polje, a to je prikazano sledećom slikom telefona koji je primio prosleđenu poruku.



Slika-7 Primalac poruke