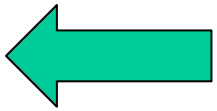
VII – Korisnički interfejs

S A D R Ž A J

7.1 - Korisnički interfejs putem pogleda

- **TextView** pogled se koristi za prikazivanje teksta korisnicima aplikacije i na drugim platformama on se najčešće naziva **labela**.
- To je osnovni pogled koji se skoro uvek koristi u Android aplikaciji.
- Prilikom kreiranja svakog novog Android projekta kreira se **main.xml** datoteka koja obavezno sadrži **<TextView>** element.
- Ako je neophodno da se omogući korisnicima aplikacije da menjaju tekst, biće korišćena **EditText** klasa, izvedena iz TextView klase.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.metropolitan.hello.MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```



7.1 – Osnovni pogledi

- Osnovni pogledi se najčešće kreiraju **main.xml** datotekom.
- Pored ***TextView***, veoma često se koriste i sledeći osnovni pogledi:

1. **EditText** – omogućava ažuriranje teksta;
2. **Button** – predstavlja dugme koje je moguće pritisnuti;
3. **ImageButton** – dugme sa slikom;
4. **CheckBox** – specijalni tip tastera sa dva stanja: selektovan (čekiran) i neselektovan (nečekiran);
5. **ToggleButton** – dugme sa svetlosnim indikatorom za prikazivanje stanja selektovan/neselektovan;
6. **RadioButton** – ima dva stanja selektovan i neselektovan;
7. **RadioGroup** – Grupa radio dugmadi u kojoj samo jedan, u datom trenutku, može biti selektovan.

7.1 – Osnovni pogledi

```
<?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/btnSave"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/save"
    android:onClick="btnSaved_clicked"/>

<Button android:id="@+id/btnOpen"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Open" />

<ImageButton android:id="@+id/btnImg1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher" />

<EditText android:id="@+id/txtName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<CheckBox android:id="@+id/chkAutosave"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Autosave" />
```

```
<CheckBox android:id="@+id/star"
    style="?android:attr/starStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />


<RadioGroup android:id="@+id/rdbGp1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >
    <RadioButton android:id="@+id/rdb1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Option 1" />
    <RadioButton android:id="@+id/rdb2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Option 2" />
</RadioGroup>

<ToggleButton android:id="@+id/toggle1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</LinearLayout>
```

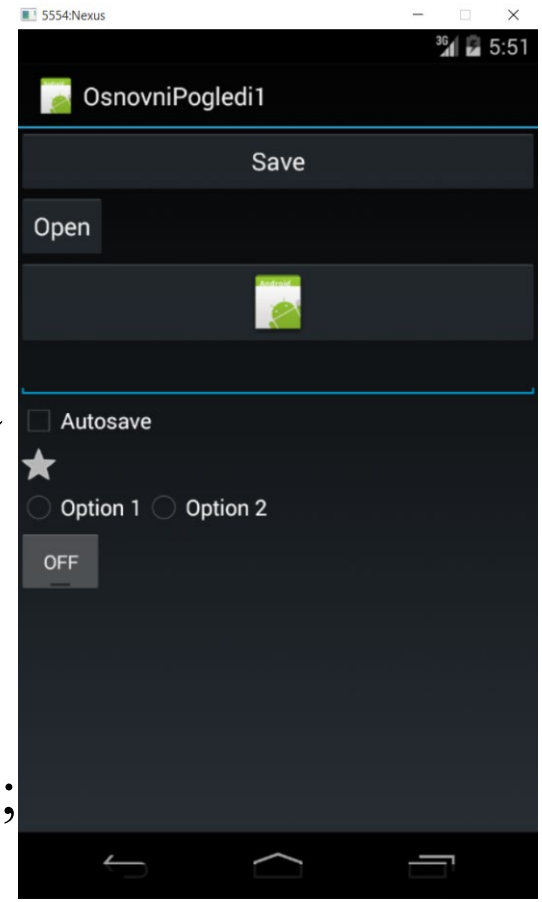
7.2-Iterakcija korisnika sa osnovnim pogledima

- Akcije koje su rezultat interakcije korisnika sa osnovnim pogledima definisane su **JAVA klasom** aktivnosti.
- Da bi prikazali upotrebu osnovnih pogleda, potrebno je modifikovati klasu aktivnosti tako da kreirane kontrole imaju konkretne zadatke.

```
package com.metropolitan.BasicViews1;
import android.app.Activity;
public class BasicViews1Activity extends Activity {

    public void btnSaved_clicked (View view) {
        DisplayToast("Kliknuli ste na dugme Save");
    }

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


        //---Button view---
        Button btnOpen = (Button) findViewById(R.id.btnOpen);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                DisplayToast("Kliknuli ste na dugme Open");
            }
        });


        //---CheckBox---
        CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
        checkBox.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v) {
                if (((CheckBox)v).isChecked())
                    DisplayToast("CheckBox je čekiran");
                else
                    DisplayToast("CheckBox nije čekiran");
            }
        });

        //---RadioButton---
        RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
        radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()
        {
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                RadioButton rb1 = (RadioButton) findViewById(R.id.rdb1);
                if (rb1.isChecked()) {
                    DisplayToast("Option 1 je čekiran!");
                } else {
                    DisplayToast("Option 2 je čekiran!");
                }
            }
        });

        //---ToggleButton---
        ToggleButton toggleButton =
                (ToggleButton) findViewById(R.id.toggle1);
        toggleButton.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v) {
                if (((ToggleButton)v).isChecked())
                    DisplayToast("Toggle button je uključen");
                else
                    DisplayToast("Toggle button je isključen");
            }
        });
    }

    private void DisplayToast(String msg)
    {
        Toast.makeText(getBaseContext(), msg,
                Toast.LENGTH_SHORT).show();
    }

}
```

7.2-Iterakcija korisnika sa osnovnim pogledima

- Klikom na različite poglede moguće je primetiti kako se menjaju izgled i funkcija kontrole.
- Koristeći vertikalni **LinearLayout** raspored, pogledi su postavljeni jedan iznad drugog:



- Prvo dugme je definisano tako da je njegov element ***layout_width*** postavljen na ***fill_parent***, a to znači da zauzima celu širinu ekrana; (**Save**)
- Drugo dugme ima vrednost ***wrap_content*** za naznačeni atribut, a to znači da je njegova širina definisana prostorom za prikazivanje sadržaja;
- ***ImageButton*** pomoću **src** atributa postavlja sliku;
- **EditText** pokazuje tekst polje za unos teksta;
- ***CheckBox*** definiše polje koje je moguće selektovati ili deselektovati. Upotrebom atributa ***style*** (videti **main.xml**) moguće je promeniti izgled polja (u zvezdicu u ovom primeru);
- **Radio grupa** sadrži dva dugmeta koji se međusobno isključuju;
- ***ToggleButton*** prikazuje taster koji ima dva stanja on/off.

7.2-Iterakcija korisnika sa osnovnim pogledima

- Svaki pogled ima vlastiti identifikator definisan ***id*** atributom, koji je moguće koristiti pomoću metoda ***View.findViewById()*** i ***Activity.findViewById()***.
- Za svaki pogled, kreiran tokom ***onCreate()*** događaja, upravljanje je moguće ukoliko je pogled programski lociran.
- Navedeno je omogućeno prosleđivanjem ***id*** atributa metodi ***findViewById()*** koja pripada osnovnoj klasi ***Activity***.
- Za dugme korišćeno u primeru, lociranje je rešeno na sledeći način:

```
Button btnOpen = (Button) findViewById(R.id.btnOpen);
```

- Metodom ***setOnClickListener()*** omogućeno je dobijanje povratne informacije koja će biti prezentovana prilikom obraćanja pogledu:

```
btnOpen.setOnClickListener(new View.OnClickListener() {
   public void onClick(View v) {
   DisplayToast("Kliknuli ste na dugme Open");
   }
```

7.2-Iterakcija korisnika sa osnovnim pogledima

- Stanje ***CheckBox***-a proverava se pomoću argumenta za metodu ***onClick()*** kojim se, potom, poziva metoda ***isChecked()***
- Metoda ***setOnCheckedChangeListener()*** se koristi kada u radio grupi treba da se dobije informacija o promeni stanja radio dugmeta.
- Za svako dugme postoji metoda ***isChecked()*** kojom se proverava stanje kontrole
- ***ToggleButton*** funkcioniše potpuno isto kao ***CheckBox.***

```
CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
checkBox.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v) {
        if (((CheckBox)v).isChecked())
            DisplayToast("CheckBox je čekiran");
        else
            DisplayToast("CheckBox nije čekiran");
    }
});
```

```
RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()
{
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        RadioButton rb1 = (RadioButton) findViewById(R.id.rdb1);
        if (rb1.isChecked()) {
            DisplayToast("Option 1 je čekiran!");
        } else {
            DisplayToast("Option 2 je čekiran!");
        }
    }
});
```

7.3 – ProgressBar kontrola

- Kontrola ***ProgessBar*** je veoma korisna za zadatke koji nemaju specifično vreme izvršavanja, gde treba vremenski prikazati koji deo zadatka je izvršen kao ***send*** i ***response*** komunikacija sa web serverom
- Postavljanjem **xml** taga **<ProgressBar>** u **main.xml** datoteku, biće obezbeđeno prikazivanje ikone koja se okreće.
- Obaveza programera je da obezbedi zaustavljanje kontrole kada se završi obavljanje datog zadatka.
- ***ProgressBar*** kontrola, u ovom primeru, prati pozadinsku nit za simuliranje zadatka koji dugo traje.
- Zato je neophodno imlementiranje klase ***Thread*** i ***Runnable*** objekata.
- Metodom ***run()*** započinje izvršavanje niti, koja u konkretnom primeru izvršava metodu, simbolično nazvanu, ***doSomeWork()***.
- Po završetku zadatka, upotrebljen je ***Handler*** objekat za slanje poruke do niti za prekidanje ***ProgressBar***-a.
- Inicijalno, ***ProgressBar*** je realizovan kao ciklična animacija.
- Instrukcija: **style="@android:style/Widget.ProgressBar.Horizontal„** ga pretvara u horizontalnu animaciju

7.4 – TimePicker pogled

```
<?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/btnSet"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="I am all set!"
    android:onClick="onClick" />

<DatePicker android:id="@+id/datePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TimePicker android:id="@+id/timePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</LinearLayout>
```

- **TimePicker** pogled omogućava korisnicima selektovanje vremena
- Izbor datuma i vremena su akcije koje se veoma često izvode u Android aplikacijama.
- Android obezbeđuje dva alata kojima su ove funkcionalnosti dostupne: ***TimePicker i DatePicker***
- ***TimePicker*** pogled omogućava korisnicima da izaberu vreme, i to, u dva režima 24h ili AM/PM.
- Glavnu datoteku aktivnosti treba modifikovati na sledeći način, pri čemu će biti uključen kod i za kontolu ***DatePicker***.

7.4 - TimePicker pogled

```
package com.metropolita.Picker;
import java.text.SimpleDateFormat;
public class PickerActivity extends Activity {
    TimePicker timePicker;
    DatePicker datePicker;
    int hour, minute;
    int yr, month, day;
    static final int TIME_DIALOG_ID = 0;
    static final int DATE_DIALOG_ID = 1;
    /** Poziva se kada se kreira aktivnost */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);

        // showDialog(TIME_DIALOG_ID);
        datePicker = (DatePicker) findViewById(R.id.datePicker);

        //---uzima tekući datum---
        Calendar today = Calendar.getInstance();
        yr = today.get(Calendar.YEAR);
        month = today.get(Calendar.MONTH);
        day = today.get(Calendar.DAY_OF_MONTH);

        showDialog(DATE_DIALOG_ID);
    }
    @Override
    protected Dialog onCreateDialog(int id)
    {
        switch (id) {
        case TIME_DIALOG_ID:
            return new TimePickerDialog(
                    this, mTimeSetListener, hour, minute, false);
        case DATE_DIALOG_ID:
            return new DatePickerDialog(
                    this, mDateSetListener, yr, month, day);

        }
        return null;
    }
```

```
    private DatePickerDialog.OnDateSetListener mDateSetListener =
            new DatePickerDialog.OnDateSetListener()
    {
        public void onDateSet(
                DatePicker view, int year, int monthOfYear, int dayOfMonth)
        {
            yr = year;
            month = monthOfYear;
            day = dayOfMonth;
            Toast.makeText(getBaseContext(),
                    "Izabrali ste : " + (month + 1) +
                    "/" + day + "/" + year,
                    Toast.LENGTH_SHORT).show();
        }
    };   private TimePickerDialog.OnTimeSetListener mTimeSetListener =
                new TimePickerDialog.OnTimeSetListener()
         {
             public void onTimeSet(
                     TimePicker view, int hourOfDay, int minuteOfHour)
             {
                 hour = hourOfDay;
                 minute = minuteOfHour;

                 SimpleDateFormat timeFormat = new SimpleDateFormat("hh:mm aa");
                 Date date = new Date(0,0,0, hour, minute);
                 String strDate = timeFormat.format(date);

                 Toast.makeText(getBaseContext(),
                         "Izabrali ste " + strDate,
                         Toast.LENGTH_SHORT).show();
             }
         };
         public void onClick(View view) {
             Toast.makeText(getBaseContext(),
                     "Izabrani datum:" + (datePicker.getMonth() + 1) +
                     "/" + datePicker.getDayOfMonth() +
                     "/" + datePicker.getYear() + "\n" +
                     "Izabrano vreme:" + timePicker.getCurrentHour() +
                     ":" + timePicker.getCurrentMinute(),
                     Toast.LENGTH_SHORT).show();
         }

    }
```

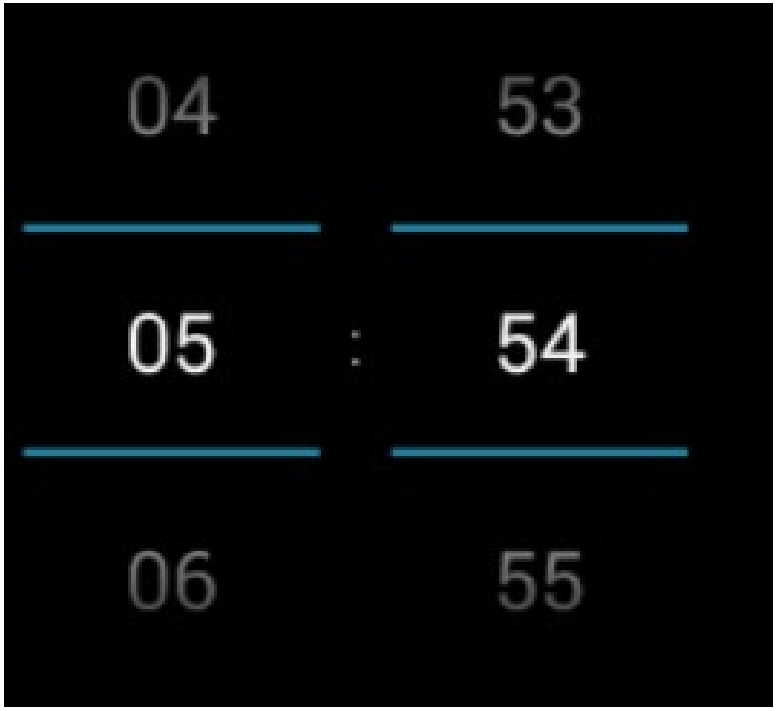
7.4 – TimePicker pogled

- ***TimePicker*** koristi standardni korisnički interfejs u kome je korisniku aplikacije omogućeno da podesi vreme.
- Po osnovnim podešavanjima, vreme se prikazuje u formatu AM/FM.
- Ukoliko se želi prikaz vremena u 24h formatu, neophodno je koristiti metodu ***setIs24HourView()*** (videti metodu ***onCreate() kod*** priloženog koda).
- Da bi vreme bilo programski prikazano, neophodno je upotrebiti metode ***getCurrentHour()*** i ***getCurrentMinute()***, na sledeći način:

```
Toast.makeText(getBaseContext(),
"Izabrano vreme:" + timePicker.getCurrentHour() +
":" + timePicker.getCurrentMinute(),
```

7.4 – TimePicker pogled

- Prosleđivanjem odgovarajućeg indikatora metodi ***showDialog***, ***TimePicker*** pogled se smešta u okvir dijaloga
- Do sada je poznato da poziv dijaloga okvira inicira izvršavanje metode ***showDialog()***.
- U konkretnom slučaju ovoj metodi je prosleđen atribut ***TIME_DIALOG_ID*** kojim je identifikovan izvor dijaloga okvira.
- Izvršavanjem metode ***showDialog()***, za prikazivanje okvira, izvršava se i metoda ***onCreateDialog()*** u okviru koje se kreira instanca klase ***TimePickerDialog***.
- Navedena instanca će preuzeti informacije od značaja, a to su u konkretnom slučaju sati i minuti.
- Na kraju, podaci će biti prezentovani u 24h formatu.



7.5 - DatePicker pogled

- ***DatePicker*** pogled omogućava korisnicima selekciju datuma.
- Priloženim kodom definisan je ***DatePicker*** pogled koji je po načinu korišćenja veoma sličan ***TimePicker*** pogledu.
- Primenom ovog pogleda, u određenoj aktivnosti, korisnicima je omogućeno da izaberu određeni datum.
- Takođe, **main.xml** datotekom definiše se ***DatePicker*** pogled odgovarajućim XML tagom:

```
<DatePicker android:id="@+id/datePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

7.5 – DatePicker pogled

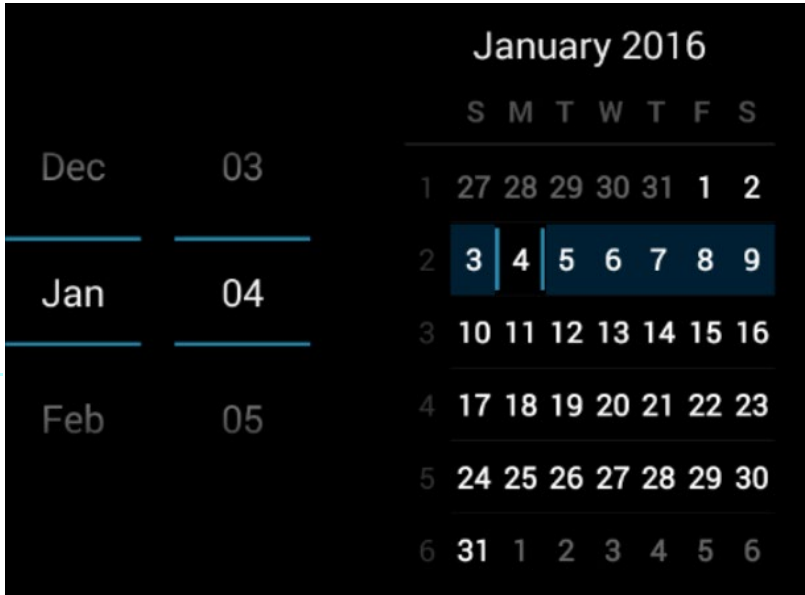
- ***DatePicker*** pogled koristi specifične metode za prikazivanje podataka o datumu.
- Na sličan način kao ***TimePicker***, ***DatePicker*** pogled koristi izvesne metode za prikazivanje podataka od značaja: dan, mesec i godina.
- Za prikazivanje ovih podataka odgovorne su metode ***getDayOfMonth()***, ***getMonth()*** i ***getYear()***, respektivno.
- Trebalo bi napomenuti da metoda ***getMonth()*** vraća vrednost 0 za mesec januar.
- Prilikom upotrebe ove metode potrebno je izvršiti inkrementiranje rezultata da bi on imao valjanu vrednost.
- Primena ovih metoda je ugrađena u priloženi kod, a ovde će biti prezentovan samo konkretan deo poziva metoda.

```
"Izabrani datum:" + datePicker.getDayOfMonth() +
"/" + (datePicker.getMonth() + 1) + "/" + datePicker.getYear()
```

7.5 – DatePicker pogled

- Kao i ***TimePicker*** i ***DatePicker*** može biti prikazan u okviru za dijalog.
- Identifikator za ***DatePicker*** definisan je kao DATE_DIALOG_ID koji se predaje metodi ***showDialog()***.
- Po ovome, ***DatePicker*** i ***TimePicker*** funkcionišu na isti način.
- Nakon definisanja datuma, poziva se metoda ***onSetDate()*** koja kao rezultat vraća datum koji je korisnik izabrao.
- Trebalo bi napomenuti da je neophodno inicijalizovati promenljive za dan, mesec i godinu pre prikazivanja okvira za dijalog.
- U suprotnom javiće se izuzetak, tokom izvršavanja programa, zbog upotrebe ilegalnog argumenta.

```
yr = year;
month = monthOfYear;
day = dayOfMonth;
Toast.makeText(getBaseContext(),
        "Izabrali ste : " + (month + 1) +
        "/" + day + "/" + year,
        Toast.LENGTH_SHORT).show();
```



7.6 – ListView pogled

```
<?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/btn"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Prikaži izabrane stavke"
    android:onClick="onClick"/>
<ListView
    android:id="@+id/android:list"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>
```

- Prikazivanje liste stavki u vertikalnoj skrol listi omogućeno je sa ***ListView*** pogledom.
- List pogledi omogućavaju prikazivanje dugačkih listi.
- U Android OS podrška prikazivanju dugih listi realizovana je primenom pogleda ***ListView*** i ***SpinnerView***.
- Prikazivanje liste stavki u vertikalnoj skrol listi omogućeno je ***ListView*** pogledom.
- Primerom na sledećem slajdu biće prikazana upotreba ***ListView*** pogleda

7.6 – ListView pogled

```
package com.metropolitan.ListView;
import android.app.ListActivity;
public class ListViewActivity extends ListActivity  {
        String[] predmeti;
        /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ListView lstView = getListView();
        lstView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
        lstView.setTextFilterEnabled(true);

        predmeti =
                getResources().getStringArray(R.array.predmeti_array);

        setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_checked, predmeti));
    }
    public void onListItemClick(
    ListView parent, View v, int position, long id)
    {
        Toast.makeText(this,
            "Izabrali ste " + predmeti[position],
            Toast.LENGTH_SHORT).show();
    }
    public void onClick(View view) {
        ListView lstView = getListView();
        String itemsSelected = "Izabrana stavka: \n";
        for (int i=0; i<lstView.getCount(); i++) {
            if (lstView.isItemChecked(i)) {
                itemsSelected += lstView.getItemAtPosition(i) + "\n";
            }
        }
        Toast.makeText(this, itemsSelected, Toast.LENGTH_LONG).show();
    }

}
```

7.6 – ListView pogled

- Klasa aktivnosti, koja je kreirana za prikaz liste stavki, ***ListView***, nasleđuje klasu *ListActivity* koja je potklasa bazne klase *Activity*
- Datoteku main.xml nije potrebno modifikovati da bi uključili ***ListView***
- Klasa koja nasleđuje *ListActivity* već sadrži ***ListView***.
- U *onCreate()* metodi nije potrebno pozvati metodu *setContentView()* sa ciljem učitavanja korisničkog interfejsa iz datoteke main.xml.
- U *onCreate()* metodi se koristi *setListAdapter()* za popunjavanje ekrana aktivnošću koja odgovara ***ListView*** pogledu
- Komponenta *ArrayAdapter* upravlja nizom stringova prezentovanih ***ListView*** pogledom.
- Ako se pogleda priloženi kod, ***ListView*** pogled je definisan da bude prikazan u jednostavnom *simple_list_item_checked* režimu.
- Na kraju, metoda *onListItemClick()* se inicira uvek kada korisnik klikne na neku stavku iz liste.
- Na slici je prikazan izgled ekrana uređaja

7.6 – ListView pogled

- **ListView** pogled podržava različite prikaze koji se mogu posebno podešavati.
- U prikazanom primeru definisano je da se iz liste, u svakom trenutku izvršavanja programa, može izabrati više stavki
- Moguće je prilagoditi i druge opcije:
  1. nije dozvoljeno selektovanje iz liste
  2. moguće je selektovati samo jednu stavku liste
- Navedeno je prikazano kodom na slici.

```
public class BasicViews5Activity extends ListActivity  {
    String[] predmeti;
    /** Poziva se kada se aktivnost kreira. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ListView lstView = getListView();
    lstView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
    lstView.setTextFilterEnabled(true);

    predmeti =
            getResources().getStringArray(R.array.predmeti_array);

    setListAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_checked, predmeti));
}
```

7.6 – ListView pogled

- Lista ne dozvoljava izbor ukoliko se u metodi ***setChoiceMode()*** kao argument javi konstanta ***CHOICE_MODE_NONE***
- Ako je ovoj metodi pridružena konstanta ***CHOICE_MODE_SINGLE*** iz liste je moguće izabrati samo jednu stavku.
- Da bi navedenu metodu bilo moguće koristiti, neophodno je kreirati objekat klase ***ListView***, za čije je učitavanje neophodno pozvati metodu ***getListView().***
- Ovde je korišćen još jedan bitan element a to je podrška za filtriranje.
- Nakon omogućavanja metode ***setTextFilterEnabled()***, otvara se mogućnost za unost teksta pomoću tastature.
- ***ListView*** prikaz će automatski biti modifikovan da se filtriraju samo one stavke koje odgovaraju unetim podacima zahvaljujući stavki:

lstView.setTextFilterEnabled(true);

7.6 – ListView pogled

- Stavke liste moguće je čuvati izvan klase aktivnosti aplikacije.
- U jednostavnim aplikacijama moguće je čuvati podatke kao niz u JAVA klasi aktivnosti aplikacije.
- Međutim, u realnim uslovima, poželjnije je čuvati stavke u bazama podataka ili u datotekama.
  - U prikazanom primeru, odabrana je datoteka **strings.xml** da sačuva članove liste do njihovog poziva.
  - Budući da su nazivi predmeta sačuvani u **string.xml** datoteci, njih je moguće učitati, na veoma jednostavan način, upotrebom metode ***getResources()*** u klasi aktivnosti aplikacije.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">ListView demonstracija!</string>
    <string name="app_name">ListView</string>
    <string-array name="predmeti_array">
        <item>Razvoj mobilnih apliacija</item>
        <item>Web sistemi 1</item>
        <item>Web sistemi 2</item>
        <item>Distribuirani sistemi</item>
        <item>Skripting jezici</item>
        <item>Razvoj igara</item>
        <item>Java programiranje 1</item>
        <item>Java programiranje 2</item>
        <item>Programiranje u C#</item>
        <item>Operativni sistemi</item>
        <item>Matematika</item>
    </string-array>
</resources>
```

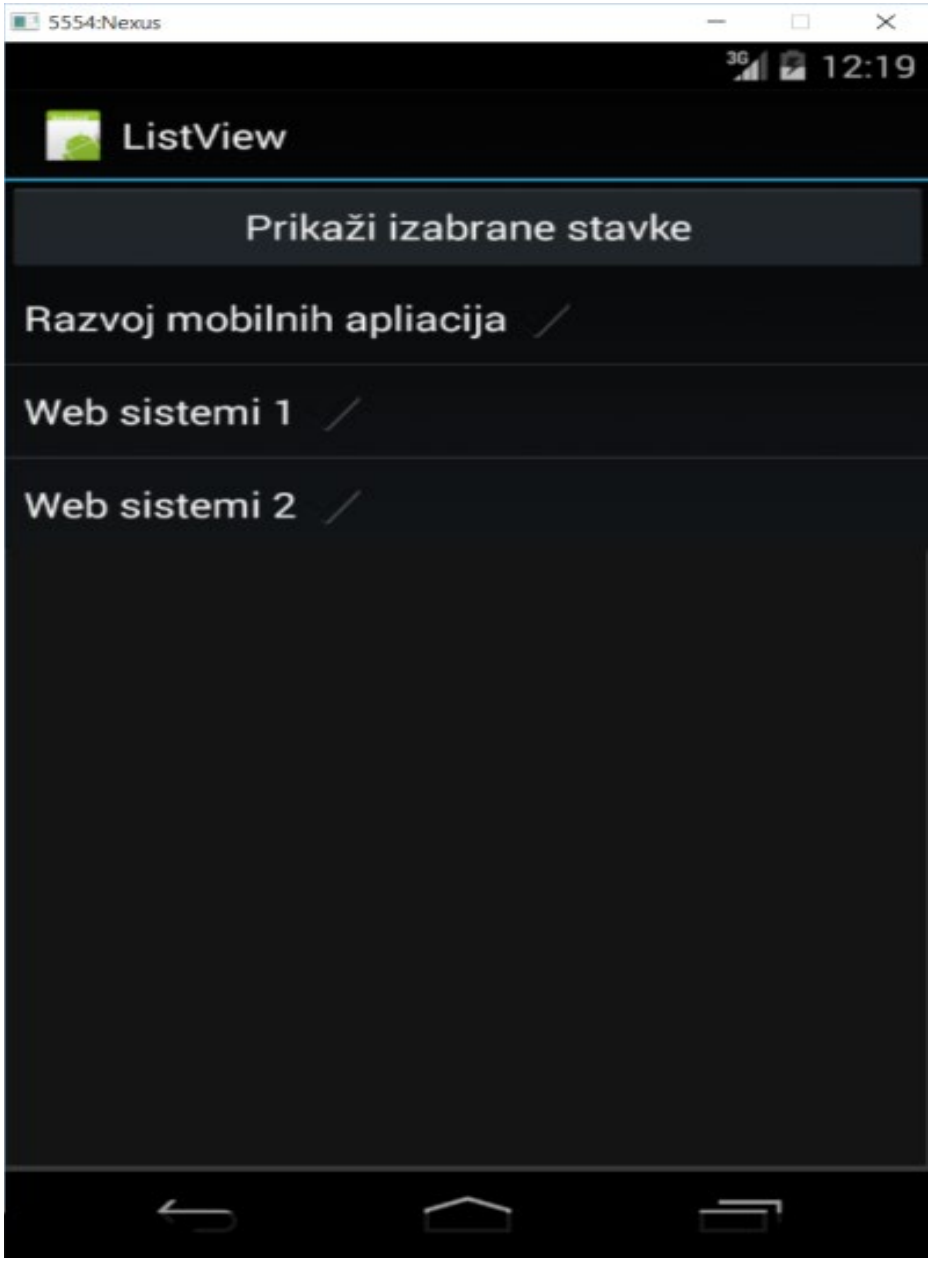
7.6 – ListView pogled

- Do sada smo videli kako se prikazuje ***ListView*** pogled koji zauzima celu površinu za prikazivanje određene aktivnosti i nije bilo potrebe za dodavanjem **<ListView>** elementa u **main.xml**.
- Da bi prostor predviđen za celokupnu aktivnost bio delimično ispunjen, neophodno je dodati **<ListView>** element u **main.xml** datoteku na način prikazan priloženim programskim kodom na slici.
- Ovaj element mora da bude snabdeven i ***id*** atributom čija je vrednost: ***@+id/android:list.***
- To znači da je u ovom slučaju neophodno učitati metodu ***setContentView()*** da bi interfejs iz datoteke **main.xml** bio učitan.
- Metoda ***isItemChecked()*** proverava da li je stavka u listi izabrana ili ne

```
ListView lstView = getListView();
String itemsSelected = "Izabrana stavka: \n";
for (int i=0; i<lstView.getCount(); i++) {
    if (lstView.isItemChecked(i)) {
        itemsSelected += lstView.getItemAtPosition(i) + "\n";
    }
}
Toast.makeText(this, itemsSelected, Toast.LENGTH_LONG).show();
```

7.6 – ListView pogled

- U slučaju učitavanja liste iz niza i ***ListView*** pogleda, definisanih klasom aktivnosti aplikacije, bez primene metode ***setContentView()***, pogled bi zauzeo celokupnu površinu namenjenu za aktivnost i UI aplikacije bi izgledao kao na slici.



7.7 - SpinnerView pogled

- Ako je neophodno da se pored liste stavki, u odgovarajućoj aktivnosti, prikažu i drugi pogledi, a da se ne zauzme cela površina ekrana kao kod ***ListView*** pogleda, trebalo bi koristiti ***SpinnerView*** prikaz.
- Ovaj prikaz omogućava prikazivanje jedne po jedne stavke iz liste.
- Prethodni primer je potrebno modifikovati na sledeći način.
- Datoteka **main.xml** imaće sledeći kod:

```
<?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" >

<Spinner
    android:id="@+id/spinner1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawSelectorOnTop="true" />

</LinearLayout>
```

7.7 - SpinnerView pogled

➢Datoteka **SpinnerActivity.java** imaće sledeći kod:

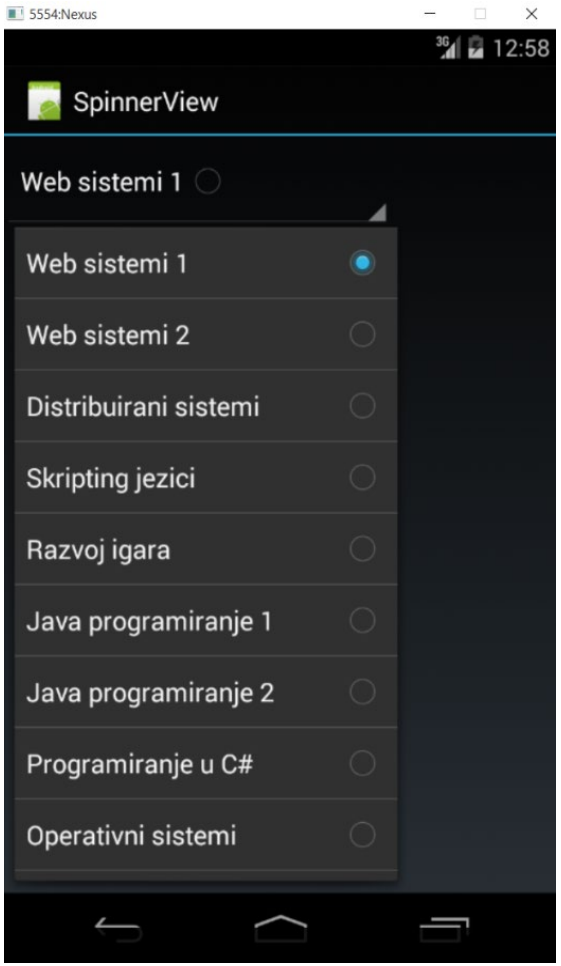
```
package com.metropolita.Spinner;

import android.app.Activity;

public class SpinnerActivity extends Activity {
    String[] predmeti;

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

        predmeti =
                getResources().getStringArray(R.array.predmeti_array);
        Spinner s1 = (Spinner) findViewById(R.id.spinner1);

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_single_choice, predmeti);

        s1.setAdapter(adapter);
        s1.setOnItemSelectedListener(new OnItemSelectedListener()
        {
            @Override
            public void onItemSelected(AdapterView<?> arg0,
            View arg1, int arg2, long arg3)
            {
                int index = arg0.getSelectedItemPosition();
                Toast.makeText(getBaseContext(),
                        "Izabrali ste stavku : " + predmeti[index],
                        Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) { }
        });
    }
}
```

7.7 - SpinnerView pogled

- Tekući primer funckioniše veoma slično prethodnom.
- Za početak je neohodno implementirati metodu ***onNothingSelected().***
- Ova metoda se izvršava kada korisnik pritisne taster ***Back*** na mobilnom uređaju i na taj način prekida dalje prikazivanje liste.
- Umesto stavki za ***ArrayAdapter***, kojima se prikazuje jednostavna lista, moguće je prikazati elemente korišćenjem radio tastera.
- Da bi to uradili neophodno je modifikovati drugi parametar u konstruktoru ***ArrayAdapter*** klase, kao što je to urađeno u priloženom kodu primera:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_single_choice, predmeti);
```

- Klikom na **F11**, program se prevodi i pokreće emulatorom i daje korisnički interfejs kao na sledećoj slici.



7.8 – ListFragment klasa

- ListFragment klasa definiše fragment koji sadrži ***ListView*** pogled.
- Fragmenti su mini-aktivnosti koje imaju sopstvene životne cikluse.
- Da bi fragment bio kreiran, on mora da bude podržan klasom koja nasleđuje baznu klasu ***Fragment***.
- Pored osnovne klase omogućeno je i nasleđivanje izvesnih njenih potklasa sa ciljem kreiranja specijalizovanih fragmenata.
- Potklase klase ***Fragment*** su: ***ListFragment***, ***DialogFragment*** i ***PreferenceFragment***.
- ListFragment klasa definiše fragment koji sadrži **ListView** pogled.
- ***ListFragment*** je veoma koristan alat jer omogućava prikazivanje liste stavki na ekranu zajedno sa ostalim elementima korisničkog interfejsa, a to može biti i još neka lista stavki.
- Da bi fragment za prikazivanje liste bio kreiran, neophodno je da ključna klasa nasledi klasu ***ListFragment***.

7.8 – ListFragment klasa

- Za svaki fragment je neophodno kreirati odgovarajuću XML datoteku koja definiše njegov sadržaj.
- Kao i svaka Android aplikacija, aplikacija koja obrađuje fragmente mora da sadrži **main.xml** datoteku koja je prikazana:

```
<?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="horizontal" >

<fragment
    android:name="net.learn2develop.ListFragmentExample.Fragment1"
    android:id="@+id/fragment1"
    android:layout_weight="0.5"
    android:layout_width="0dp"
    android:layout_height="200dp" />

<fragment
    android:name="net.learn2develop.ListFragmentExample.Fragment1"
    android:id="@+id/fragment2"
    android:layout_weight="0.5"
    android:layout_width="0dp"
    android:layout_height="300dp" />

</LinearLayout>
```

7.8 - ListFragment klasa

- Za fragment, koji je dobio naziv ***Fragment1***,kreirana je XML datoteka naziva ***Fragment1.xml*** u kojoj se čuva definicija njegovog sadržaja:

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

    <ListView
        android:id="@id/android:list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:drawSelectorOnTop="false"/>

</LinearLayout>
```

7.8 – ListFragment klasa

- Pored glavne JAVA klase, koja je zadužena za aktivnosti, neophodno je kreirati i JAVA klasu koja nasleđuje klasu **ListFragment**.
- Svaka Android aplikacija mora da sadrži glavnu JAVA klasu aktivnosti.
- Zadatak ove klase je, u velikom broju slučajeva, da obezbedi izvršavanje metode ***setContentView()*** kojom se učitava korisnički interfejs definisan u **main.xml** datoteci.
- U ovom primeru, klasa aktivnosti će imati isključivo navedeni zadatak, a to je prikazano sledećim kodom.

```
package net.learn2develop.ListFragmentExample;

import android.app.Activity;

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

7.8 - ListFragment klasa

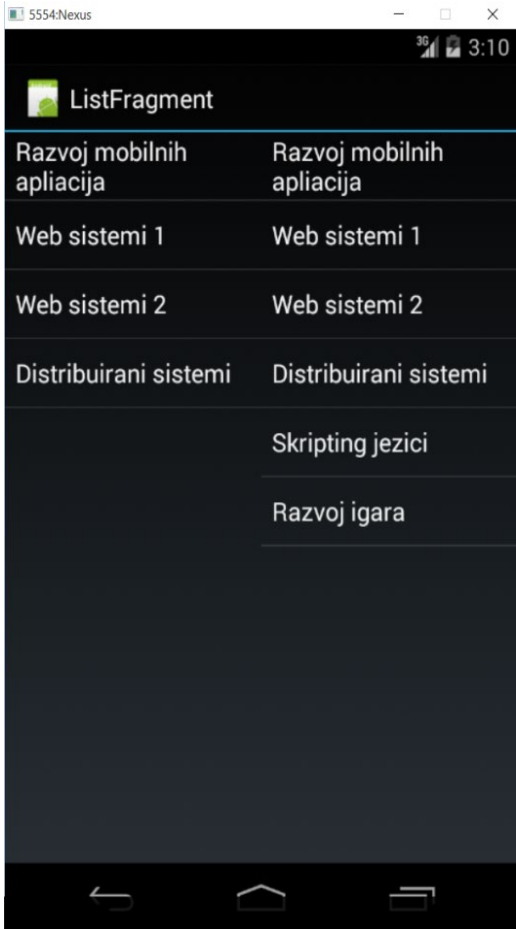
- Da bi fragment za prikazivanje listi bio implementiran, neophodno je kreirati njegovu klasu.
- Kao što je napomenuto, on mora da nasledi osnovnu ***ListFragment*** klasu, a to je pokazano sledećim kodom.

```
package net.learn2develop.ListFragmentExample;
import android.app.ListFragment;
public class Fragment1 extends ListFragment {
    String[] predmeti = {
            "Razvoj mobilnih apliacija",
            "Web sistemi 1",
            "Web sistemi 2",
            "Distribuirani sistemi",
            "Skripting jezici",
            "Razvoj igara",
            "Java programiranje 1",
            "Java programiranje 2",
            "Programiranje u C#",
            "Operativni sistemi",
            "Matematika"
    };
    @Override
    public View onCreateView(LayoutInflater inflater,
    ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment1, container, false);
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setListAdapter(new ArrayAdapter<String>(getActivity(),
            android.R.layout.simple_list_item_1, predmeti));
    }
        public void onListItemClick(ListView parent, View v,
    int position, long id)
    {
        Toast.makeText(getActivity(),
            "Izabrali ste " + predmeti[position],
            Toast.LENGTH_SHORT).show();
    }

}
```

7.8 – ListFragment klasa

- Aplikacija pokazuje dva fragmenta za prikaz listi, jedan pored drugog.
- Kreiranje aplikacije, čiji je kod priložen, teče u sledećim koracima:

1. Kreirana je XML datoteka za fragment sa **<ListView>** elementom;
2. Kreirana je JAVA klasa koja nasleđuje klasu ***ListFragment***;
3. U ovoj klasi je kreiran niz predmeta (polje) koji će biti upisan u listu;
4. Metoda ***onCreate()*** koristi metodu ***setListAdapter()*** čiji je zadatak punjenje liste sadržajem polja; ***ArrayAdapter*** upravlja nizom stringova koji će biti prikazani u ***ListView*** rasporedu jednostavnim režimom ***simple_list_item_1***
5. Metoda ***onListItemClick()*** poziva se prilikom svakog klika na stavku u ***ListView*** prikazu;
6. Datotekom **main.xml** je definisan interfejs koji poseduje dva fragmenta za prikazivanje listi.
7. Za svaki fragment određena je drugačija visina prostora koji će zauzeti
8. Klikom na **F11**, primer se prevodi i pokreće.



7.8 - ListFragment klasa

- Fragmenti za dijalog su veoma korisni kada bi trebalo dobiti odgovor korisnika pre nego što se nastavi izvršavanje Android aplikacije.
- Postoji još jedan tip fragmenata čiji je zadatak da od korisnika dobije odgovor pre nego što se nastavi izvršavanje aplikacije
- Ovakvi fragmeti su izvedeni iz klase ***DialogFragment*** i nazivaju se fragmentima za prikazivanje dijaloga.
- Takođe, uvodi se primer za demonstraciju primene fragmenata za prikazivanje dijaloga. Sledeća slika daje kod datoteke **main.xml**.

```
<?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" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

7.8 – ListFragment klasa

- Prozor sa porukom i tasterima za reakciju - aplikacija čeka na odgovor
- Funkcionisanje aplikacije, koja implementira fragment sa dijalogom, podrazumeva da su ispunjeni sledeći uslovi:

✓ Kreirana fragment klasa koja nasleđuje klasu ***Fragment***;

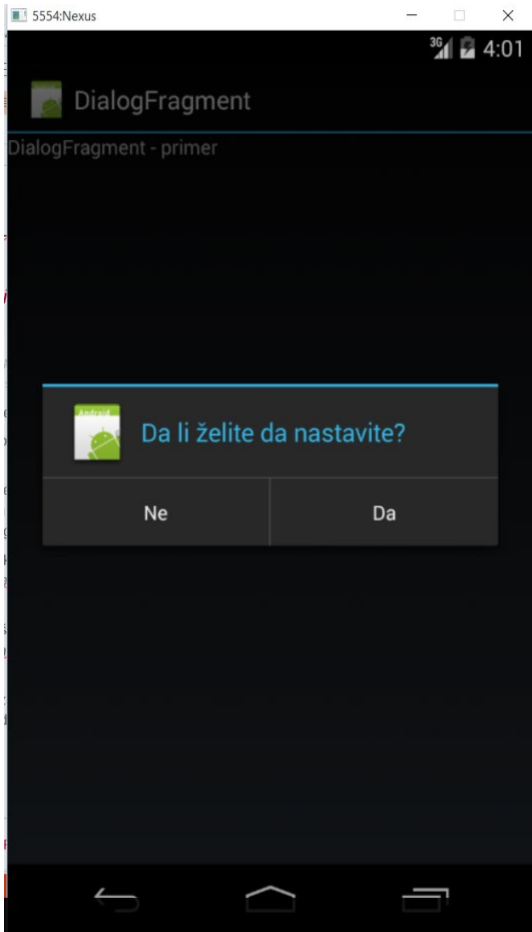
✓ Kreiran okvir za dijalog koji prikazuje poruku i odgovarajuće tastere;

✓ Metodom ***newInstance()*** kreira se nova instanca fragmenta koja prihvata ***string*** argument koji će biti prikazan kao poruka u okviru za dijalog

✓ Postojanje ***onCreateDialog()*** metode, koja se izvršava nakon ***onCreate()*** metode, a pre ***onCreateView()*** metode

✓ Kreirana instanca fragmenta koja izvršava metodu ***show():dialogFragment.show(getFragmentManager(), "dialog");***

✓ Treba implementirati metode ***doPositiveClick()*** i ***doNegativeClick()*** za rukovanje aktivnostima koje su rezultat klika na dugmad ***Da*** i ***Ne***, respektivno



Hvala na pažnji !!!

Pitanja

? ? ?