GUI II

Mobilne aplikacije

Stevan Gostojić

Fakultet tehničkih nauka, Novi Sad

1. novembar 2022.

Pregled sadržaja

Adapteri

- Adapteri povezuju poglede (naslednice `AdapterView` pogleda) i izvore podataka
- Postoje predefinisani adapteri koji povezuju različite poglede (ListView, GridView, Spinner, itd.) i različite izvore podataka (nizove, kolekcije, kursore, itd.)
- Moguće je napraviti adaptere koji povezuju proizvoljan pogled i proizvoljni izvor podataka

ArrayAdapter

- Povezuje `TextView` pogled (ili pogled koji sadrži `TextView` pogled) i niz ili kolekciju
- Automatski se poziva `toString()` metoda svakog objekta u nizu ili kolekciji i njena povrana vrednost se prikazuje u pogledu

CursorAdapter

- `CursorAdapter` je adapter koji koristi kursor kao izvor podataka
- Kursor sadrži rezultat upita nad bazom podataka (više o kurzorima na jednom od narednih časova)

Custom Adapters

- Moguće je definisati adapter koji koristi proizvoljan izvor podataka
- Potrebno je definisati klasu koja nasleđuje `Adapter` ili `BaseAdapter` i redefinisati njene metode

ExampleAdapter.java

```
1  public class ExampleAdapter extends BaseAdapter {
2
3    Activity activity;
4
5    public ExampleAdapter(Activity activity) {
6      this.activity = activity;
7    }
8
9    @Override
10   public int getCount() {
11     // return item count
12   }
13
14   @Override
15   public Object getItem(int position) {
16     // return item at position
17   }
18
19   @Override
20   public long getItemId(int position) {
21     // return item ID at position
22   }
23
24   @Override
25   public View getView(int position, View convertView, ViewGroup
       parent) {
26     // return view at position
27   }
28 }
29
```

ExampleAdapter.java

```
1  @Override
2  public View getView(int position, View view, ViewGroup parent) {
3
4    if (view == null) {
5      view = activity.getLayoutInflater().inflate(R.layout.example_adapter, null)
       ;
6    }
7
8    TextView tvName = (TextView) view.findViewById(R.id.tv_name);
9    tvName.setText(...);
10   TextView tvDescription = (TextView) view.findViewById(R.id.tv_description);
11   tvDescription.setText(..);
12   ImageView ivIcon = (ImageView) view.findViewById(R.id.iv_icon);
13   ivIcon.setImageResource(...);
14
15   return view;
16 }
17
```

example_adapter.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/
       android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent" >
5
6      <ImageView android:id="@+id/iv_icon" />
7
8      <TextView android:id="@+id/tv_name" />
9
10     <TextView android:id="@+id/tv_description" />
11
12 </LinearLayout>
13
14
```

ListView

Figure 1: ListView pogled.

- `ListView` pogled prikazuje listu stavki (koja može da se skroluje)
- Stavke se preuzimaju iz adaptera koji je pridružen pogledu

list_view.xml

```
1  <LinearLayout
2    xmlns:android="http://schemas.android.com/apk/res/android"
3    android:orientation="vertical"
4    android:layout_width="fill_parent"
5    android:layout_height="fill_parent">
6
7   <ListView
8      android:id="@+id/list_view"
9      android:layout_width="wrap_content"
10     android:layout_height="wrap_content" />
11
12 </LinearLayout>
13
```

ListViewActivity.java

```
1  public class ListViewActivity extends Activity {
2
3    @Override
4    protected void onCreate(Bundle state) {
5
6      // ...
7      List<String> list = populate();
8      ArrayAdapter adapter = new ArrayAdapter(
9        this,
10       android.R.layout.simple_list_item_1,
11       list);
12     ListView listView = (ListView) findViewById(R.id.list_view);
13     listView.setAdapter(adapter);
14   }
15 }
16
```

GridView

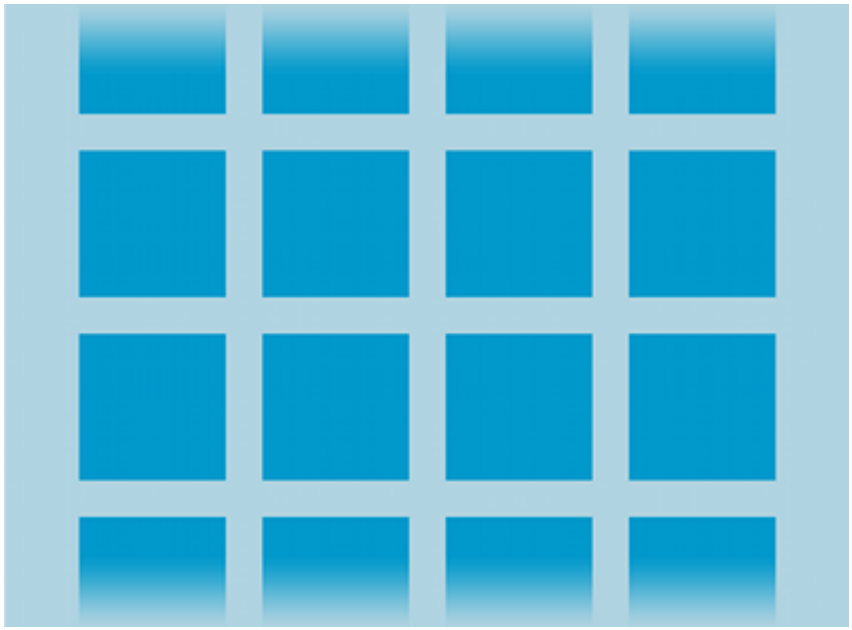


Figure 2: GridView pogled.

- `GridView` pogled prikazuje tabelu stavki (koja može da se skroluje)
- Stavke se preuzimaju iz adaptera koji je pridružen pogledu

grid_view.xml

```
1  <LinearLayout
2    xmlns:android="http://schemas.android.com/apk/res/android"
3    android:orientation="vertical"
4    android:layout_width="fill_parent"
5    android:layout_height="fill_parent">
6
7    <GridView
8      android:id="@+id/grid_view"
9      android:numColumns="auto_fit"
10     android:gravity="center"
11     android:columnWidth="50dp"
12     android:layout_width="fill_parent"
13     android:layout_height="fill_parent" />
14
15 </LinearLayout>
16
```

gridview_item.xml

```
1  <LinearLayout
2    xmlns:android="http://schemas.android.com/apk/res/android"
3    android:layout_width="fill_parent"
4    android:layout_height="wrap_content"
5    android:gravity="center_vertical">
6
7    <CheckedTextView
8      android:id="@android:id/checked_text"
9      android:layout_width="0px"
10     android:layout_height="fill_parent"
11     android:layout_weight="0.9"
12     android:gravity="center_vertical" />
13
14 </LinearLayout>
15
```

GridViewActivity.java

```
1  public class GridViewActivity extends Activity {
2
3    @Override
4    public void onCreate(Bundle state) {
5      // ...
6      List<String> list = populate() ;
7      ArrayAdapter adapter = new ArrayAdapter(
8        this ,
9        R.layout.gridview_item ,
10       list);
11     GridView gridView = (GridView) findViewById(R.id.grid_view);
12     gridView.setAdapter(adapter);
13   }
14 }
15
```

Spinner

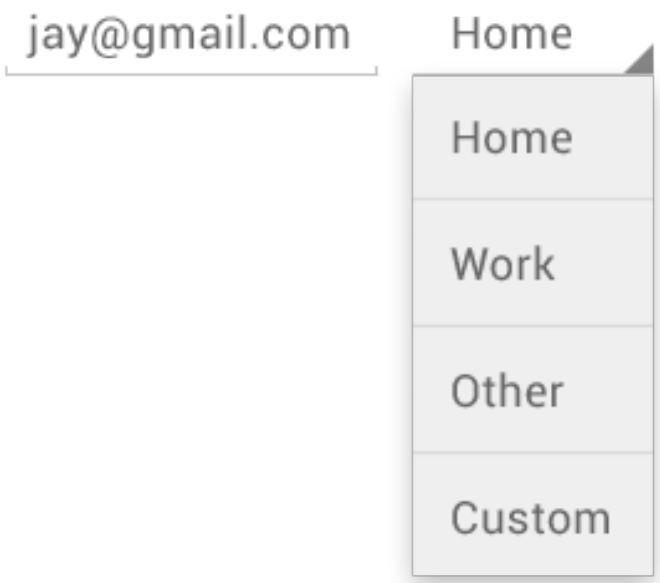


Figure 3: Spinner.

- Spinner pogled prikazuje stavke u meniju (korisnik može da izabere jednu stavku iz menija)
- Stavke se preuzimaju iz adaptera koji je pridružen pogledu

spinner.xml

```
1  <LinearLayout
2    xmlns:android="http://schemas.android.com/apk/res/android"
3    android:orientation="vertical"
4    android:layout_width="fill_parent"
5    android:layout_height="fill_parent">
6
7    <Spinner
8      android:id="@+id/spinner"
9      android:layout_width="fill_parent"
10     android:layout_height="wrap_content" />
11
12 </LinearLayout>
13
```

SpinnerActivity.java

```
1  public class SpinnerActivity extends Activity
2    implements OnItemSelectedListener {
3
4    @Override
5    public void onCreate(Bundle state) {
6      // ...
7      List<String> list = populate();
8      ArrayAdapter adapter = new ArrayAdapter(
9        this,
10       android.R.layout.simple_spinner_item,
11       list);
12     Spinner spinner = (Spinner) findViewById(R.id.spinner);
13     spinner.setAdapter(adapter);
14     spinner.setOnItemClickListener(new AdapterView.
       OnItemClickListener() {
15       public void onItemClick(AdapterView<?> parent, View view, int
       position, long id) {
16         Intent intent = new Intent(SpinnerActivity.this,
       SecondActivity.class);
17         intent.putExtra("position", position);
18         intent.putExtra("id", id);
19         startActivity(intent);
20       }
21     });
22
23   }
24 }
25
```

Pregled sadržaja

Toasts

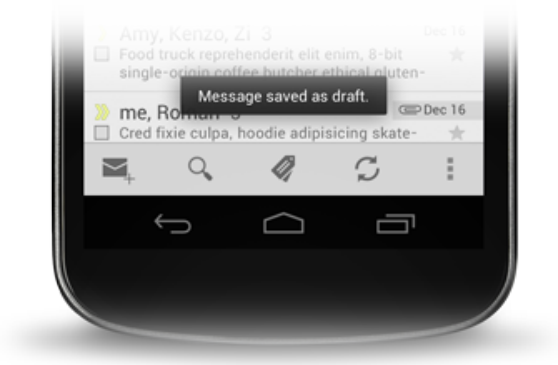


Figure 4: Toast.

- Toast je pop-up poruka koja automatski nestaje posle određenog vremena
- Korisniku daje povratnu informaciju da je akcija izvršena
- Aktivnost na vrhu povratnog steka ostaje vidljiva i u fokusu
- U novijim verzijama Android platforme preporučljivo je koristiti Snackbar

MainActivity.java

```
1 Context context = getApplicationContext();
2 CharSequence message = "Hello World!";
3 int duration = Toast.LENGTH_SHORT;
4 Toast toast = Toast.makeText(context, message, duration)
      ;
5 toast.show();
6
```

Pregled sadržaja

Obaveštenja

- Obaveštenje (notification) je poruka koja se prikazuje van korisničkog interfejsa aplikacije (u površini za obaveštenja ili fioci za obaveštenja)
- Ne prekida korisnika u izvršavanju tekućeg zadatka
- Obično se prikazuju obaveštenja o vremenski kritičnim događajima ili događajima u kojima učestvuju drugi ljudi
- Moguće je i izvršiti akciju iz obaveštenja

Obaveštenja

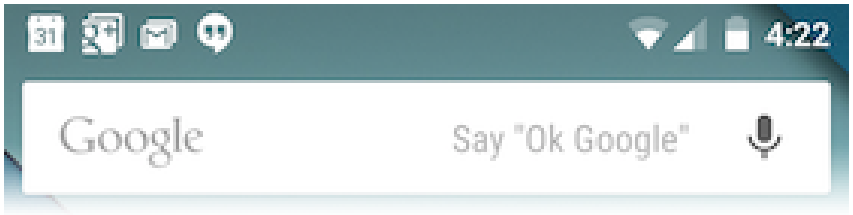


Figure 5: Površina za obaveštenja.

- Obaveštenje se prikazuje kao ikona u površini za obaveštenja (notofication area)

Obaveštenja

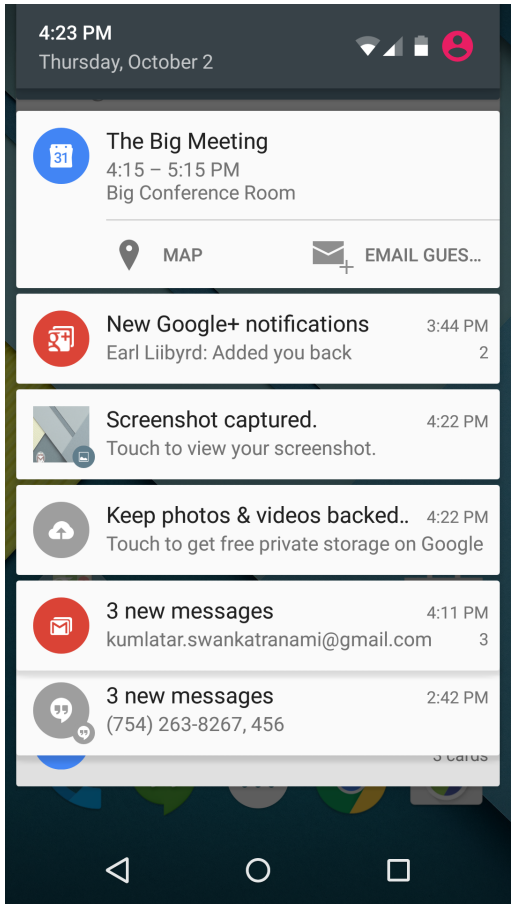
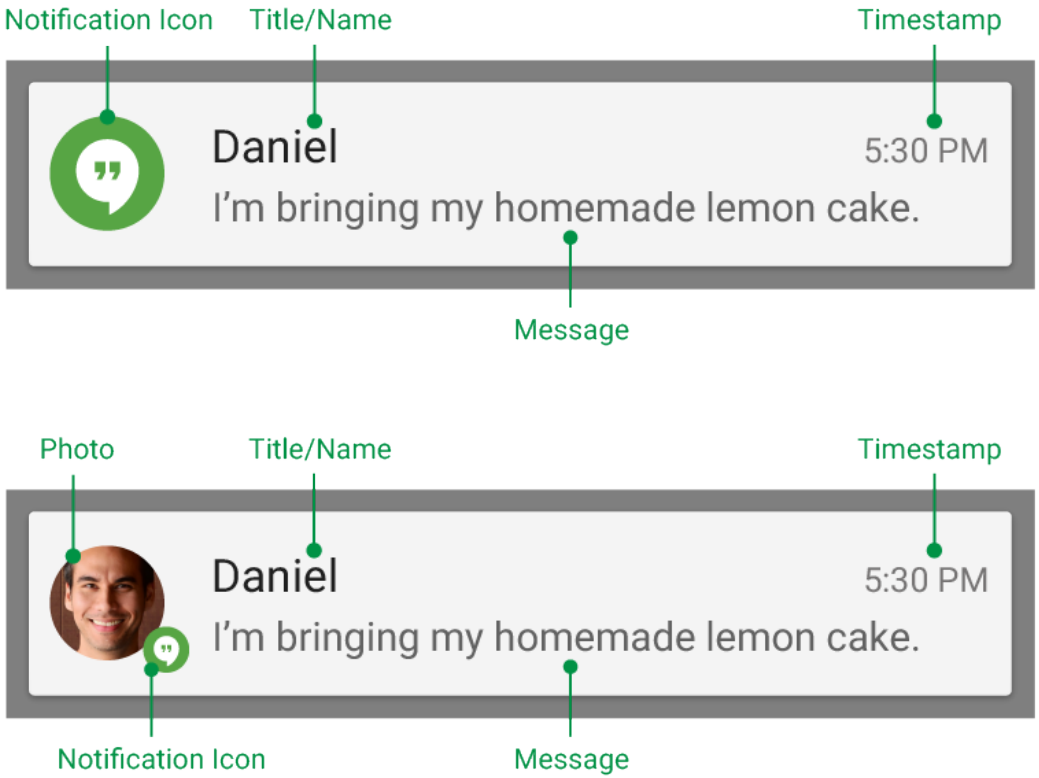


Figure 6: Fioka za obaveštenja.

- Više informacija o obaveštenju prikazuje se u fioci za obaveštenja (notification drawer)

Obaveštenja



- mala ikona
- naslov
- tekst

Figure 7: Osnovni raspored obaveštenja.

Obaveštenja

```
1  Context context = getApplicationContext();
2  NotificationCompat.Builder builder = new NotificationCompat.Builder(
       context)
3    .setContentTitle(title) // Mandatory property
4    .setLargeIcon(R.drawable.large_icon)
5    .setContentText(text) // Mandatory property
6    .setContentInfo(info)
7    .setSmallIcon(R.drawable.small_icon) // Mandatory property
8    .setWhen(when)
9    .setStyle(new Notification.BigPictureStyle().bigPicture(bigBitmap))
10   .setSound(soundURI)
11   .setLights(color, onDuration, offDuration)
12   .setVibrate(pattern)
13   .setPriority(priority)
14
```

Obaveštenja

Parametar	Opis
color	boja LED (RGB)
onDuration	interval u kome je LED uključena (ms)
offDuration	interval u kome je LED isključena (ms)

Table 1: Parametri setLights metode.

Obaveštenja

Parametar	Opis
pattern	interval u kome telefon vibrira (ms), interval u kome telefon ne vibrira (ms), itd.

Table 2: Parametri setVibrate metode.

Obaveštenja

Vrednost	Opis
priority	MAX (critical and urgent), HIGH (high priority), DEFAULT (default), LOW (low in urgency), MIN (contextual or background)

Table 3: Parametri setPriority metode.

Obaveštenja

```
1  // Creates an explicit intent
2  Intent intent = new Intent(this, ResultActivity.class);
3  // The stack builder contains an artificial back stack
4  TaskStackBuilder tsb = TaskStackBuilder.create(this);
5  // Adds the back stack for the Intent
6  tsb.addParentStack(ResultActivity.class);
7  // Adds the Intent to the top of the stack
8  tsb.addNextIntent(intent);
9  PendingIntent pe = tsb.getPendingIntent(0, pe.FLAG_UPDATE_CURRENT);
10 builder.setContentIntent(pe);
11 // Returns the handle to the notification manager
12 NotificationManager manager =
13   (NotificationManager).getSystemService(Context.NOTIFICATION_SERVICE
       );
14 // Notifies the user (id allows notification update later on)
15 manager.notify(id, builder.build());
16
```

Pregled sadržaja

Dijalozi

- Dijalog je modalni prozor koji prikazuje poruku i (opciono) omogućava unos podataka i potvrdu izvršavanja akcije
- Ne zauzima ceo ekran (aktivnost koja prikazuje dijalog se pauzira)
- Postoje predefinisani dijalozi kao što su: AlertDialog, DatePicker i TimePicker
- Moguće je definisati sopstvene dijaloge (preporučljivo je da se umesto klase `Dialog` koristi klasa `DialogFragment` zato što ona vodi računa o životnom ciklusu dijaloga i omogućava ponovno korišćenje dijaloga)

Dijalozi

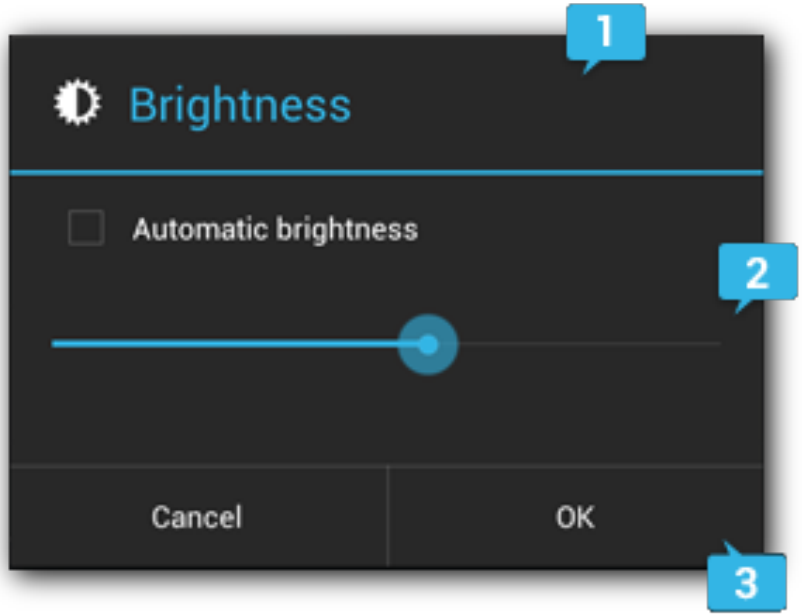


Figure 8: AlertDialog.

AlertDialog sadrži:

1. naslov
2. poruku, listu ili proizvoljan raspored
3. do tri dugmeta (negativno, neutralno i pozitivno)

ExampleDialogFragment.java

```
1  public class ExampleDialogFragment extends DialogFragment {
2
3    @Override
4    public Dialog onCreateDialog(Bundle state) {
5      return new AlertDialog.Builder(getActivity())
6        .setMessage(R.string.message_id)
7        .setPositiveButton(
8          R.string.btnOK,
9          new DialogInterface.OnClickListener() {
10           public void onClick(DialogInterface dialog, int id) {
11             // ...
12           }
13         })
14       .create();
15   }
16 }
17
```

Dijalozi

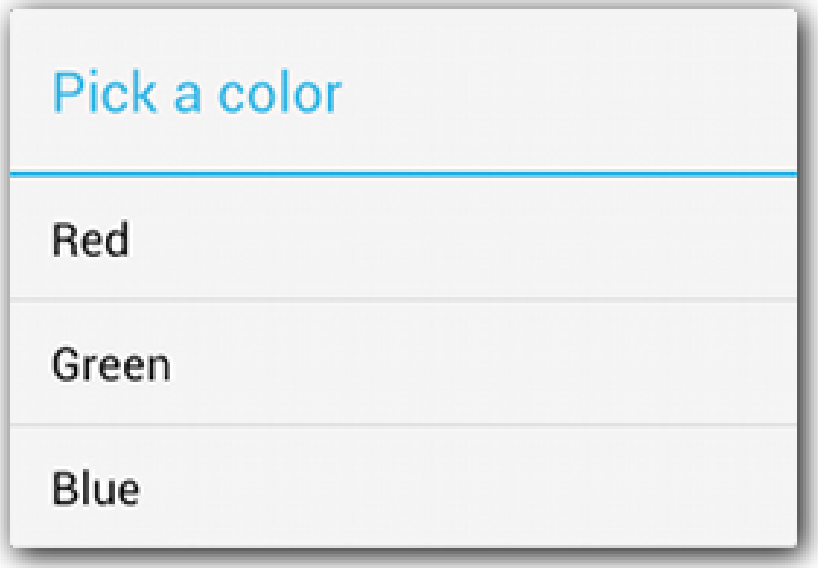


Figure 9: Dijalog sa listom.

- Alert dijalog može da sadrži:
  - listu
  - radio buttons
  - checkboxes
- Stavke se mogu definisati u statičkom nizu ili adapteru

ExampleDialogFragment.java

```
1  public class ExampleDialogFragment extends DialogFragment {
2
3    @Override
4    public Dialog onCreateDialog(Bundle state) {
5      return new AlertDialog.Builder(getActivity())
6        .setTitle(R.string.pick_color)
7        .setItems(
8          R.array.colors_array,
9          new DialogInterface.OnClickListener() {
10           public void onClick(DialogInterface dialog, int which) {
11             // ...
12           }
13         })
14       .create();
15   }
16 }
17
```

Dijalozi

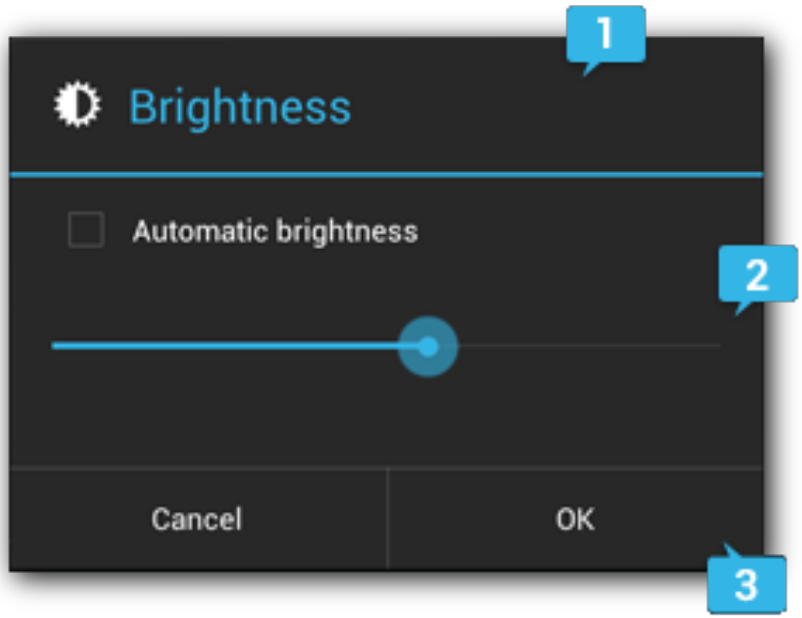


Figure 10: Dijalog sa rasporedom.

- Podrazumevano ponašanje je da raspored zauzme ceo ekran
- Moguće je dodati naslov i dugmad

ExampleDialogFragment.java

```
1  public class ExampleDialogFragment extends DialogFragment {
2
3    @Override
4    public Dialog onCreateDialog(Bundle state) {
5      return new AlertDialog.Builder(getActivity())
6        .setView(
7          getActivity().getLayoutInflater().inflate(R.layout.dialog_layout,
       null))
8        .setPositiveButton(
9          R.string.btnOK,
10         new DialogInterface.OnClickListener() {
11           @Override
12           public void onClick(DialogInterface dialog, int id) {
13             // ...
14           }
15         })
16       .create();
17   }
18 }
19
```

TimePicker & DatePicker

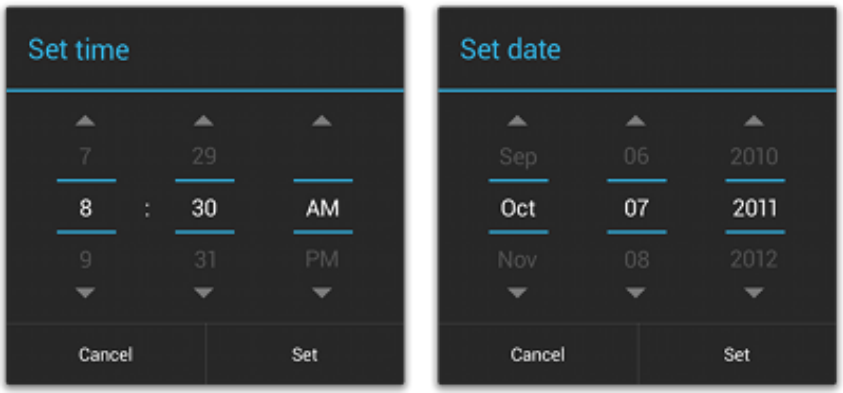


Figure 11: TimePicker & DatePicker.

1. Za unos vremena koristi se predefinisani dijalog TimePicker
2. Za unos datuma koristi se predefinisani dijalog DatePicker

DatePicker

```
1  public class DatePickerFragment
2    extends DialogFragment
3    implements DatePickerDialog.OnDateSetListener {
4
5    @Override
6    public Dialog onCreateDialog(Bundle state) {
7      Calendar c = Calendar.getInstance();
8      int year = c.get(Calendar.YEAR);
9      int month = c.get(Calendar.MONTH);
10     int day = c.get(Calendar.DAY_OF_MONTH);
11     return new DatePickerDialog(getActivity(), this, year, month, day
       );
12   }
13
14   @Override
15   public void onDateSet(DatePicker view, int year, int month, int day
       ) {
16     // ...
17   }
18 }
19
```

TimePicker

```
1  public class TimePickerFragment
2    extends DialogFragment
3    implements TimePickerDialog.OnTimeSetListener {
4
5    @Override
6    public Dialog onCreateDialog(Bundle state) {
7      Calendar c = Calendar.getInstance();
8      int hourOfDay = c.get(Calendar.HOUR_OF_DAY);
9      int minute = c.get(Calendar.MINUTE);
10     return new TimePickerDialog(getActivity(), this, hourOfDay, minute,
       true);
11   }
12
13   @Override
14   public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
15     // ...
16   }
17 }
18
```

Prikazivanje dijaloga

```
1 public void showDialog() {
2   DialogFragment dialog = new DatePickerFragment();
3   // Tag name is used to save and restore fragment state
4   // and to get a handle to the fragment
5   dialog.show(getSupportFragmentManager(), "tag_name");
6 }
7
```

Pregled sadržaja

Podešavanja

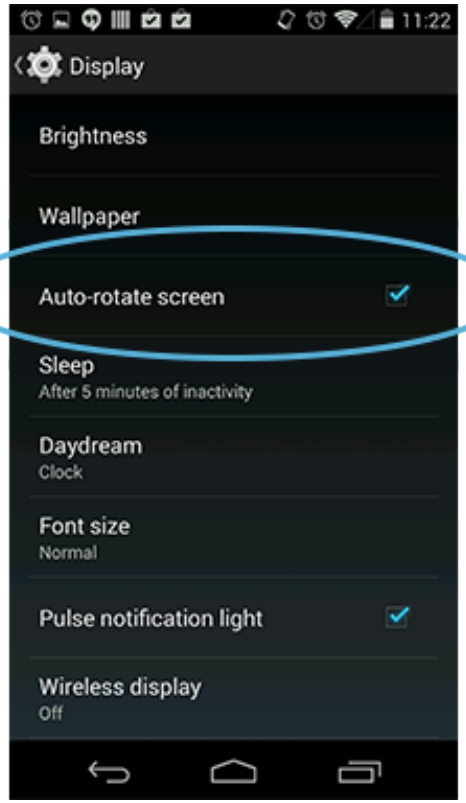


Figure 12: Podešavanja.

- Za podešavanje aplikacije koristi se Preferece API (da bi ponašanje aplikacija bilo konzistentno)
- Različitim tipovima parametrima odgovaraju različiti tipovi kontrola koje nasleđuju Preference klasu
- Kontrole se mogu grupisati u kategorije ili u podekrane
- Vrednosti parametara se automatski učitavaju i snimaju (više o tome na jednom od narednih časova)

Podešavanja

Tip parametra	Tip kontrole
Boolean	CheckBoxPreference
Float	EditTextPreference
Int	EditTextPreference
Long	EditTextPreference
String	EditTextPreference, ListPreference
Set<String>	MultiSelectListPreference

Table 4: Tipovi kontrola.

preferences.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen
3    xmlns:android="http://schemas.android.com/apk/res/android">
4
5    <CheckBoxPreference
6      android:key="pref_sync"
7      android:title="@string/pref_sync"
8      android:summary="@string/pref_sync_summ"
9      android:defaultValue="true" />
10
11   <ListPreference
12     android:dependency="pref_sync"
13     android:key="pref_syncConnectionType"
14     android:title="@string/pref_syncConnectionType"
15     android:dialogTitle="@string/pref_syncConnectionType"
16     android:entries="@array/pref_syncConnectionTypes_entries"
17     android:entryValues="@array/
       pref_syncConnectionTypes_values"
18     android:defaultValue="@string/
       pref_syncConnectionTypes_default" />
19
20 </PreferenceScreen>
21
```

Podešavanja

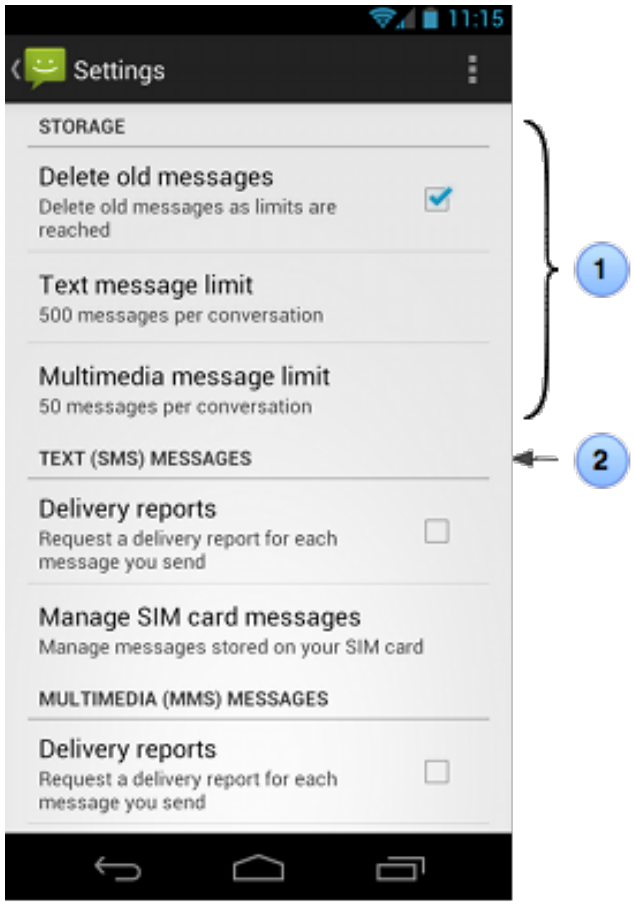


Figure 13: Podešavanja sa naslovima.

preferences.xml (1/2)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen
3    xmlns:android="http://schemas.android.com/apk/res/android">
4
5    <PreferenceCategory
6      android:title="@string/pref_sms_storage_title"
7      android:key="pref_key_storage_settings">
8
9      <CheckBoxPreference
10       android:key="pref_key_auto_delete"
11       android:summary="@string/pref_summary_auto_delete"
12       android:title="@string/pref_title_auto_delete"
13       android:defaultValue="false" />
14
```

preferences.xml (2/2)

```
1       <EditTextPreference
2           android:key="pref_key_sms_delete_limit"
3           android:dependency="pref_key_auto_delete"
4           android:summary="@string/pref_summary_delete_limit"
5           android:title="@string/pref_title_sms_delete" />
6
7       <EditTextPreference
8           android:key="pref_key_mms_delete_limit"
9           android:dependency="pref_key_auto_delete"
10          android:summary="@string/pref_summary_delete_limit"
11          android:title="@string/pref_title_mms_delete" />
12
13    </PreferenceCategory>
14
15  </PreferenceScreen>
16
```

Podešavanja

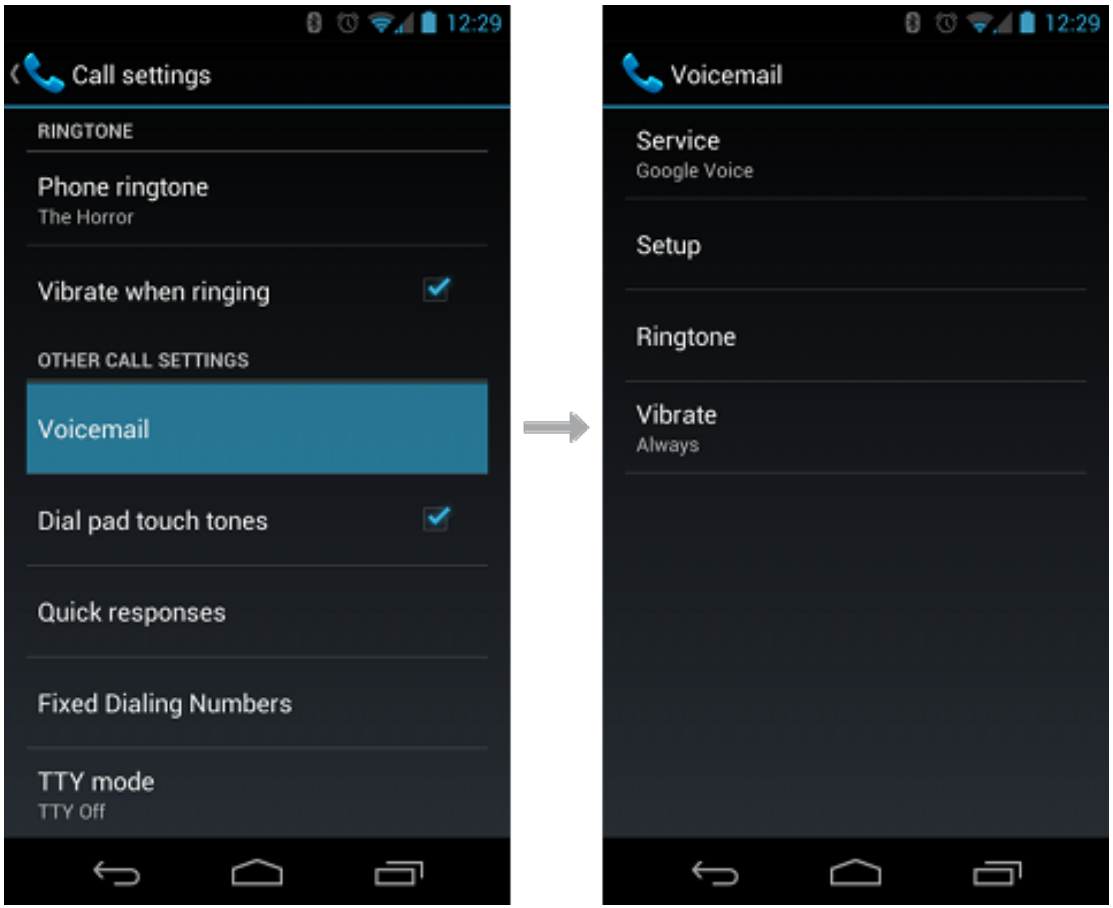


Figure 14: Podešavanja sa podekranima.

preferences.xml (1/2)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen
3    xmlns:android="http://schemas.android.com/apk/res/android">
4
5    <!-- ... -->
6
7    <!-- opens a subscreen of settings -->
8    <PreferenceScreen
9      android:key="button_voicemail_category_key"
10     android:title="@string/voicemail"
11     android:persistent="false">
12
13     <ListPreference
14       android:key="button_voicemail_provider_key"
15       android:title="@string/voicemail_provider" />
16
```

preferences.xml (2/2)

```
    <!-- opens another nested subscreen -->
    <PreferenceScreen
        android:key="button_voicemail_setting_key"
        android:title="@string/voicemail_settings"
        android:persistent="false">

        <!-- ... -->

    </PreferenceScreen>

    <RingtonePreference
        android:key="button_voicemail_ringtone_key"
        android:title="@string/voicemail_ringtone_title"
        android:ringtoneType="notification" />

    <!-- ... -->

  </PreferenceScreen>

  <!-- ... -->

</PreferenceScreen>

```

Prikaz ekrana za podešavanja

```
1  public class SettingsFragment extends PreferenceFragment {
2
3    @Override
4    public void onCreate(Bundle state) {
5      super.onCreate(state);
6      // Load the preferences from an XML resource
7      addPreferencesFromResource(R.xml.preferences);
8    }
9
10   // ...
11
12 }
13
```

Prikaz ekrana za podešavanja

```
1  public class SettingsActivity extends Activity {
2
3    @Override
4    protected void onCreate(Bundle state) {
5      super.onCreate(state);
6      // Display the fragment as the main content
7      getFragmentManager()
8        .beginTransaction()
9        .replace(android.R.id.content, new SettingsFragment())
10       .commit();
11   }
12
13   // ...
14
15 }
16
```

Pregled sadržaja

Toolbar

Toolbar je element GUI-a koji se (obično) nalazi na vrhu ekrana i obezbeđuje:

- branding aplikacije
- navigaciju
- promenu pogleda
- izvršavanje akcija

Toolbar

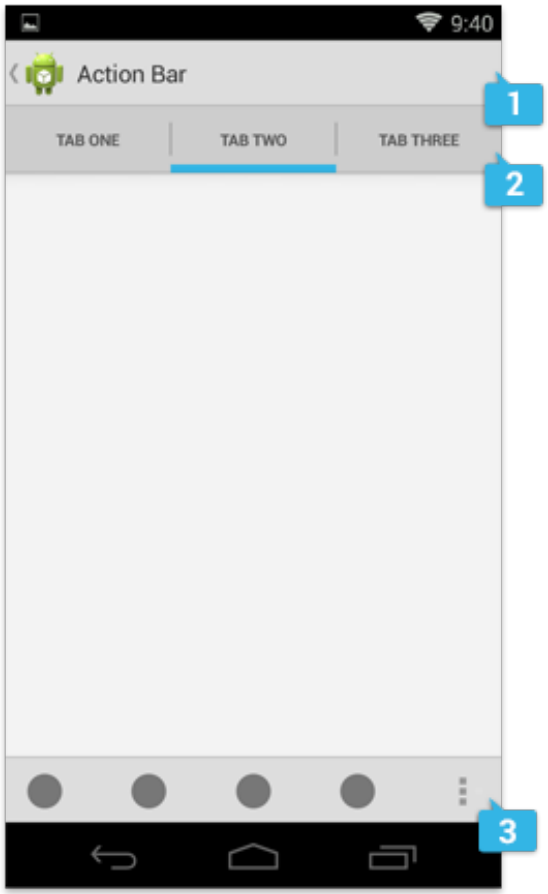


Figure 15: Toolbar.

1. title area (identifikuje aplikaciju)
2. navigation area (omogućava navigaciju)
3. action area (omogućava izvršavanje akcija, ređe korišćene akcije su "prelivene" u meni)

Toolbar

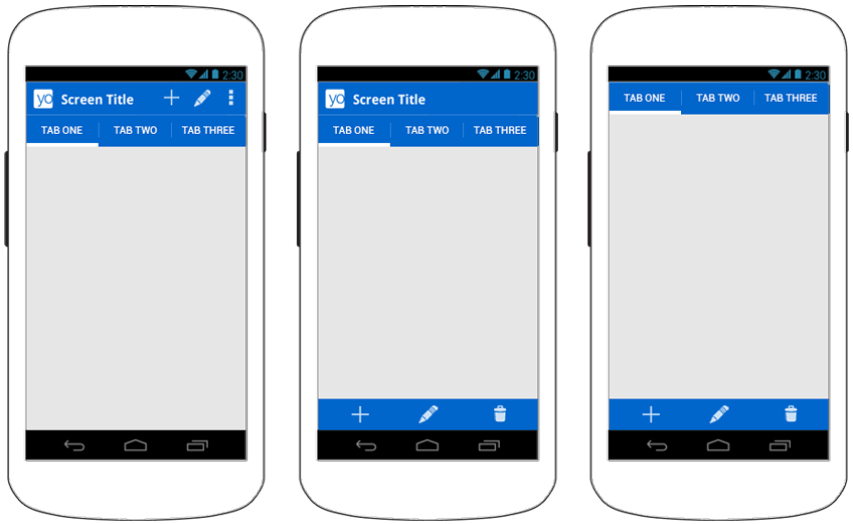


Figure 16: Toolbar.

Može se podeliti u više delova:

1. glavni deo (prikazuje ikonu i omogućava navigaciju)
2. gornji deo (omogućava promenu pogleda)
3. donji deo (omogućava izvršavanje akcija)

Pravljenje toolbar-a

- Dodati v7 appcompat support library u projekat
- Dodati toolbar u raspored
- Definisati klasu koja nasleđuje `AppCompatActivity` klasu i u `onCreate()` metodi pozvati `setSupportActionBar()`
- Koristiti jednu od `AppCompat.NoActionBar` tema (na taj način se sprečava korišćenje ugrađene `ActionBar` klase)

build.gradle

```
1 dependencies {
2     compile 'com.android.support:appcompat-v7:23.2.1'
3 }
4
```

layout_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/
       apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:orientation="vertical">
6
7    <android.support.v7.widget.Toolbar
8       android:id="@+id/toolbar"
9       android:layout_width="match_parent"
10      android:layout_height="?attr/actionBarSize"
11      android:background="?attr/colorPrimary"
12      android:elevation="4dp"
13      android:theme="@style/ThemeOverlay.AppCompat.
        ActionBar"
14      app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
         />
15
16 </LinearLayout>
17
```

ExampleActivity.java

```
1   public class ExampleActivity extends
    AppCompatActivity {
2
3   @Override
4   protected void onCreate(Bundle state) {
5     super.onCreate(state);
6     setContentView(R.layout.layout_main);
7     Toolbar toolbar = (Toolbar) findViewById(R.id.
    toolbar);
8     setSupportActionBar(toolbar);
9   }
10
11 }
12
```

AndroidManifest.java

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest ... >
3   <application android:theme="@style/Theme.AppCompat.
      Light.NoActionBar">
4     <!-- ... -->
5   </application>
6 </manifest>
7
```

ExampleActivity.java

```
1 ActionBar actionBar = getActionBar();
2 // Hides action bar
3 actionBar.hide();
4 // Shows action bar
5 actionBar.show();
6
```

Izvršavanje akcija

- Za implementaciju dugmadi se koristi mehanizam kontekstno zavisnog menija koji je nasleđen od starijih verzija Androida
- Deklarisati meni kao resurs
- Prikazati meni u `onCreateOptionsMenu` metodi i reagovati na akcije korisnika u `onOptionsItemSelected` metodi aktivnosti

action_bar.xml

```
1  <menu xmlns:android="http://schemas.android.com/apk/res/android">
2
3    <item
4       android:id="@+id/action_search"
5       android:icon="@drawable/ic_action_search"
6       android:title="@string/action_search"
7       android:showAsAction="ifRoom|withText|always" />
8
9    <item
10      android:id="@+id/action_compose"
11      android:icon="@drawable/ic_action_compose"
12      android:title="@string/action_compose" />
13
14 </menu>
15
```

ExampleActivity.java

```
1 @Override
2 public boolean onCreateOptionsMenu(Menu menu) {
3   // Inflate the menu items for use in the action bar
4   MenuInflater inflater = getMenuInflater();
5   inflater.inflate(R.menu.action_bar, menu);
6   return super.onCreateOptionsMenu(menu);
7 }
8
```

ExampleActivity.java

```
1  public class ExampleActivity extends AppCompatActivity {
2
3    // Menu icons are inflated just as they were with
       actionbar
4    @Override
5    public boolean onCreateOptionsMenu(Menu menu) {
6      // Inflate the menu; this adds items to the action
       bar if it is present.
7      getMenuInflater().inflate(R.menu.menu_main, menu);
8      return true;
9    }
10
11   @Override
12   public boolean onOptionsItemSelected(MenuItem item) {
13     switch (item.getItemId()) {
14       case R.id.action_search:
15         openSearch();
16         return true;
17       case R.id.action_compose:
18         composeMessage();
19         return true;
20       default:
21         return super.onOptionsItemSelected(item);
22     }
23   }
24 }
25
```

Up dugme

- U `AndroidManifest.xml` deklarisati aktivnost kao dete druge aktivnosti
- U njoj aktivnosti pozvati `setDisplayHomeAsUpEnabled` metodu i proslediti joj argument `true`

AndroidManifest.xml

```
1  <manifest ... >
2    <application ... >
3      <!-- ... -->
4      <!-- The main/home activity (has no parent activity)
        -->
5      <activity android:name="com.example.MainActivity">
6        <!-- ... -->
7      </activity>
8
9      <!-- A child of the main activity -->
10     <activity
11       android:name="com.example.ChildActivity"
12       android:parentActivityName="com.example.
       MainActivity">
13       <!-- ... -->
14     </activity>
15   </application>
16 </manifest>
17
```

ExampleActivity.java

```
1  public class ExampleActivity extends AppCompatActivity {
2
3    @Override
4    protected void onCreate(Bundle state) {
5      super.onCreate(state);
6      setContentView(R.layout.layout_main);
7      Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
8      setSupportActionBar(toolbar);
9      ActionBar actionbar = getSupportActionBar();
10     actionbar.setDisplayHomeAsUpEnabled(true);
11   }
12
13 }
14
```

Pregled sadržaja

NavigationDrawer

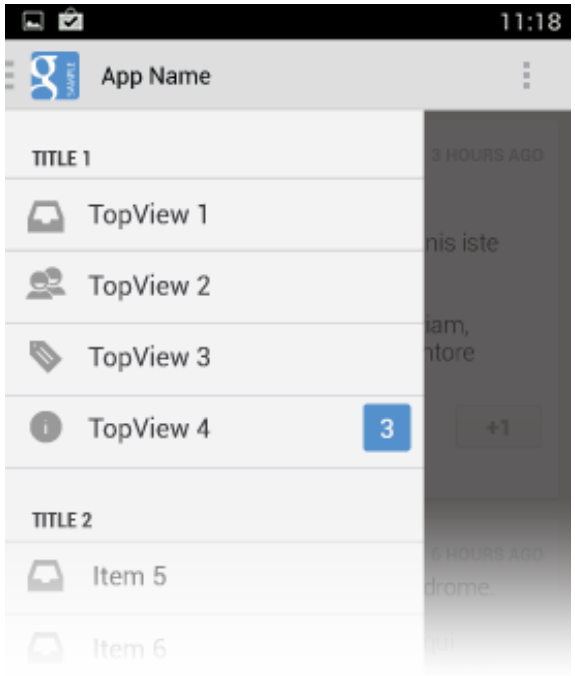


Figure 17: NavigationDrawer.

- Deklarisati DrawerLayout raspored kao koreni raspored
- Dodati jedan pogled koji sadrži glavni sadržaj aktivnosti i drugi pogled koji sadrži sadržaj fioke za navigaciju
- Inicijalizovati fioku za navigaciju
- Reagovati na akcije korisnika

layout_main.java

```
<android.support.v4.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/drawer_layout"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <!-- The main content view -->
  <FrameLayout
    android:id="@+id/content_frame"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

  <!-- The drawer view -->
  <FrameLayout
    android:id="@+id/drawer_frame"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- The navigation drawer menu -->
    <ListView android:id="@+id/list_view"
      android:layout_width="240dp"
      android:layout_height="match_parent"
      android:layout_gravity="start"
      android:choiceMode="singleChoice"
      android:divider="@android:color/transparent"
      android:dividerHeight="0dp"
      android:background="#111"/>

  </FrameLayout>

</android.support.v4.widget.DrawerLayout>

```

ExampleActivity.java

```
1  public class ExampleActivity extends Activity {
2
3    private String[] drawerItems;
4    private DrawerLayout drawerLayout;
5    private ListView listView;
6
7    @Override
8    public void onCreate(Bundle state) {
9      super.onCreate(state);
10     setContentView(R.layout.layout_main);
11
12     drawerItems = getResources().getStringArray(R.array.drawer_items)
       ;
13     drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
14     listView = (ListView) findViewById(R.id.list_view);
15
16     // Set the adapter for the list view
17     listView.setAdapter(new ArrayAdapter<String>(
18       this,
19       android.R.layout.simple_list_item_1,
20       listItems));
21     // Set the list's click listener
22     listView.setOnItemClickListener(new DrawerItemClickListener());
23   }
24 }
25
```

ExampleActivity.java

```
1  private class DrawerItemClickListener implements ListView.
       OnItemClickListener {
2
3    // Swaps fragments in the main content view
4    @Override
5    public void onItemClick(AdapterView parent, View view, int position
       , long id) {
6      // Create a new fragment and specify the planet to show based on
       position
7      Fragment fragment = new PlanetFragment();
8      Bundle args = new Bundle();
9      args.putInt(PlanetFragment.ARG_PLANET_NUMBER, position);
10     fragment.setArguments(args);
11
12     // Insert the fragment by replacing any existing fragment
13     FragmentManager fragmentManager = getFragmentManager();
14     fragmentManager
15       .beginTransaction()
16       .replace(R.id.content_frame, fragment)
17       .commit();
18
19     // Highlight the selected item, update the title, and close the
       drawer
20     listView.setItemChecked(position, true);
21     setTitle(drawerItems[position]);
22     FrameLayout drawerFrame = (FrameLayout)findViewById(R.id.
       drawer_frame);
23     drawerLayout.closeDrawer(drawerFrame);
24   }
25
26   @Override
27   public void setTitle(CharSequence title) {
28     this.title = title;
29     getActionBar().setTitle(title);
30   }
31 }
32
33
```

Pregled sadržaja

Material Design

- Material Design je skup principa za vizuelni dizajn, dizajn pokreta i dizajn interakcija
- Aplikacije dizajnirane po ovim principima pružaju korisnicima konzistentno iskustvo na različitim platformama (mobilnim, web i desktop) i u različitim aplikacijama
- Material Design koristi metafore da bi korisničko iskustvo bilo intuitivno

Material Design

Principi Material Design preporuka mogu se grupisati u tri kategorije:

- opipljive površine (tengable sufraces)
- smeo grafički dizajn (bold graphic design)
- smisleni pokreti (meaningful motion)

Opipljive površine

Senke simuliraju visinu listova papira koja određuje njihov međusobni odnos:

- seam (dva lista papira koji dele zajedničku ivicu se kreću zajedno
- step (dva lista papira koji se preklapaju se kreću nezavisno)
- floating action button (dugme odvojeno od toolbar-a)

Smeo grafički dizajn

Na listovima se prikazuje:

- tekst (Roboto i Noto)
- fotografije, ilustracije i ikonografija (predefinisane ikone za uobičajene akcije)
- boje (primarna, sekundarna i akcentovana)

Smisleni pokreti

- autentični pokreti (pokreti treba da budu usklađeni sa masom, zapreminom i fleksibilnošću objekta
- interakcija sa kraktim odzivom (aplikacije reaguju na akcije korisnika i obezbeđuju vizuelnu potvrtdu)
- smisleni prelazi (prelazi treba da usmere pažnju korisnika i da budu glatki)

Material Design za Android

Android podržava Material Design tako što pruža:

- nove teme
- nove poglede (npr. RecycleView, CardView, itd.)
- novi API za senke i animacije

AndroidManifest.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest ... >
3   <application android:theme="@android:style/Theme.Material" ... >
4     <!-- ... -->
5   </application>
6 </manifest>
7
```

Material Design teme

Konstanta	Opis
Material	tamna verzija
Material.Light	svetla verzija
Material.Light.DarkActionBar	svetla verzija sa toolbar-om

Table 5: Izabrane Material Design teme.

Reference

- Material Design, https://www.google.com/design/spec/material-design
- Material Design for Android, http://developer.android.com/design/material