GUI III

Mobilne aplikacije

Pregled sadržaja

Pregled sadržaja

Toasts

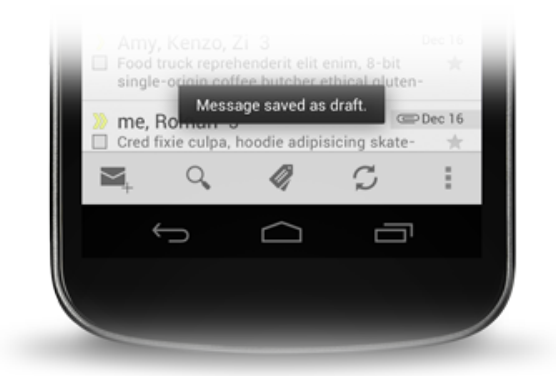


Figure 1: Toast.

- 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 jer dozvoljava interakciju sa korisnikom

MainActivity.java

- Primer za Toast

```
Toast toast = Toast.makeText(
                getApplicationContext(),
                "Hello World!",
                Toast.LENGTH_SHORT);
toast.show();
```

Snackbar

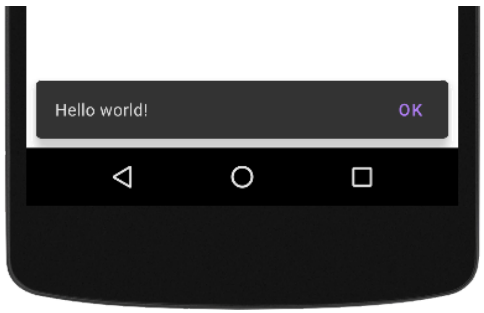


Figure 2: Snackbar.

- Snackbar može sadržati neku akciju (najviše jednu).
- Trajanje prikazivanja može biti `Snackbar.LENGTH_SHORT`, `Snackbar.LENGTH_LONG`, `Snackbar.LENGTH_INDEFINITE`.
- Snackbar se prikazuje preko korenskog rasporeda aktivnosti.
- Snackbar se uklanja protekom vremena, klikom na ponuđenu akciju ili pozivom `dismiss` metode.

MainActivity.java

- Primer za Snackbar

```
View parentView = findViewById(R.id.rootLayout);
Snackbar snackbar = Snackbar.make(
                    parentView,
                    "Hello world!",
                    Snackbar.LENGTH_INDEFINITE);
snackbar.setAction("Undo", new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        // do something
                    }
                });
snackbar.show();
```

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

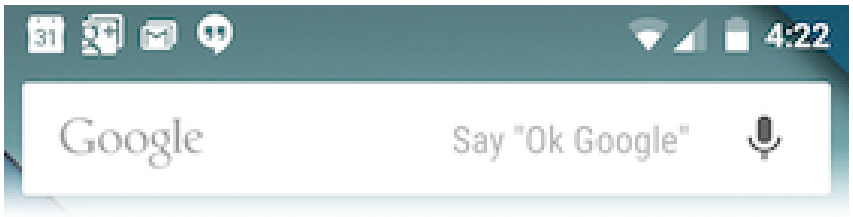


Figure 3: Površina za obaveštenja.

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

Obaveštenja

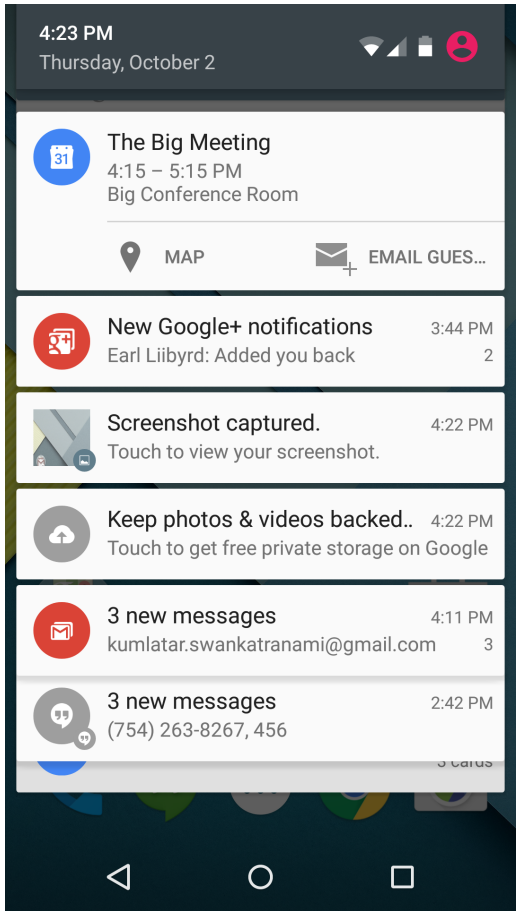
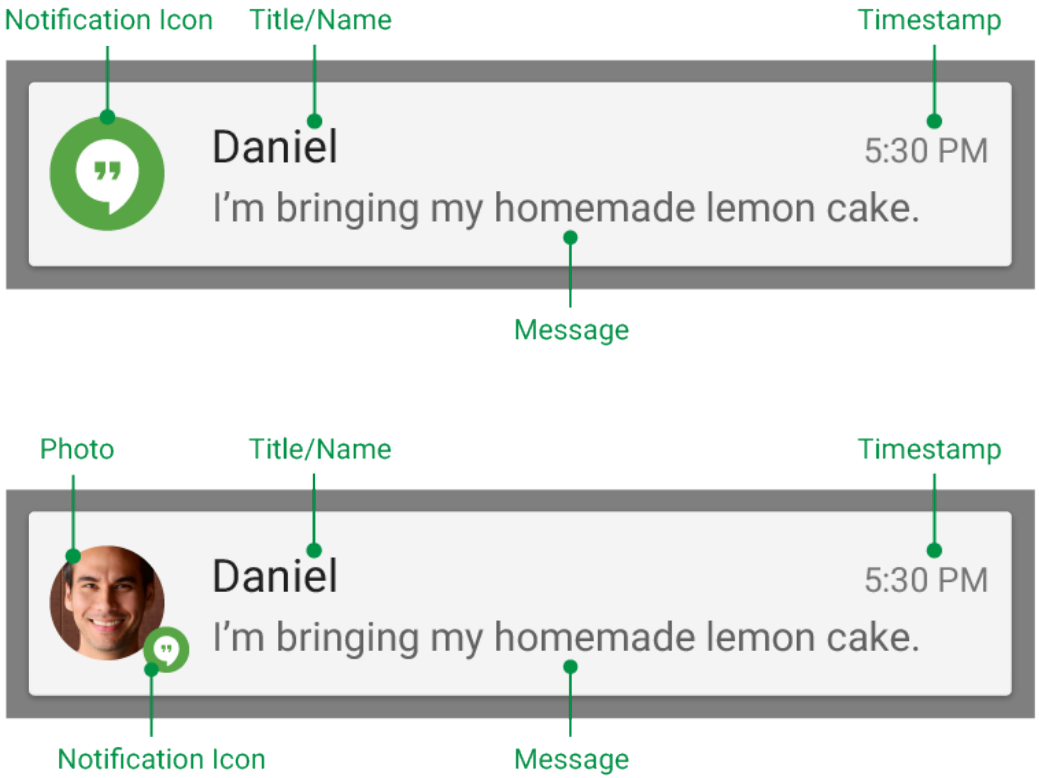


Figure 4: Fioka za obaveštenja.

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

Obaveštenja



- mala ikona
- naslov
- tekst

Figure 5: Osnovni raspored obaveštenja.

Obaveštenja - osnovni primer

- Od API level 26 notifikacije moraju biti pridružene nekom notifikacionom kanalu.
- Korisnik u podešavanjima može izmeniti ponašanje notifikacija koje pripadaju određenom notifikacionom kanalu.
- Pojedinačne notifikacije imaju svoj identifikator putem kojeg ih je moguće naknadno izmeniti.
- Od API level 33 za notifikacije je potrebna permisija `android.permission.POST_NOTIFICATIONS`.

```
String CH_ID = "my_notifications";
NotificationCompat.Builder builder =
        new NotificationCompat.Builder(getApplicationContext(), CH_ID);
builder.setContentTitle("Hello")
       .setContentText("Hello World!!!")
       .setSmallIcon(R.drawable.ic_launcher_foreground);

int NOTIFICATION_ID = 1;
NotificationManager manager =
  (NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
manager.notify(NOTIFICATION_ID, builder.build());

```

- Android Studio poseduje alat Asset Studio koji olakšava kreiranje ikonica: File -> New -> Image Asset -> Icon Type: Notification Icons

Obaveštenja - dodatna podešavanja

```
String CH_ID = "my_notifications";
Context context = getApplicationContext();
NotificationCompat.Builder builder = new NotificationCompat.Builder(
    context, CH_ID)
  .setContentTitle(title) // Mandatory property
  .setLargeIcon(R.drawable.large_icon)
  .setContentText(text) // Mandatory property
  .setContentInfo(info)
  .setSmallIcon(R.drawable.small_icon) // Mandatory property
  .setWhen(when)
  .setStyle(new Notification.BigPictureStyle().bigPicture(bigBitmap))
  .setSound(soundURI)
  .setLights(color, onDuration, offDuration)
  .setVibrate(pattern)
  .setPriority(priority);
```

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.

- Za korišćenje vibracije je potrebno u Manifest.xml navesti permisiju

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

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 - pridruživanje namere

- Ukoliko je potrebno, klikom na notifikaciju se može izvršiti neka akcija. Za to je potrebno da notifikaciji pridružimo odgovarajuću nameru.
- Pošto se ova namera pokreće izvan naše aplikacije, potrebno joj je dodeliti privilegije koje ima i aplikacija. To nam omogućava PendingIntent odn. namera na čekanju.

```
// First, we create an explicit intent
Intent intent = new Intent(this, ResultActivity.class);

// Then, we build a pending intent using the explicit intent
// requestCode differentiate our intent between other intents
int requestCode = (int) System.currentTimeMillis();
// this flag cancels an old intent and create new one
int flags = PendingIntent.FLAG_CANCEL_CURRENT;
PendingIntent pIntent = PendingIntent.getActivity(this, requestCode, intent, flags);

// Now, we assign the pending intent to the notification builder
builder.setContentIntent(pIntent);

// Finally, we retrieve notification manager and build notification
NotificationManager manager =
  (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
manager.notify(id, builder.build());

```

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

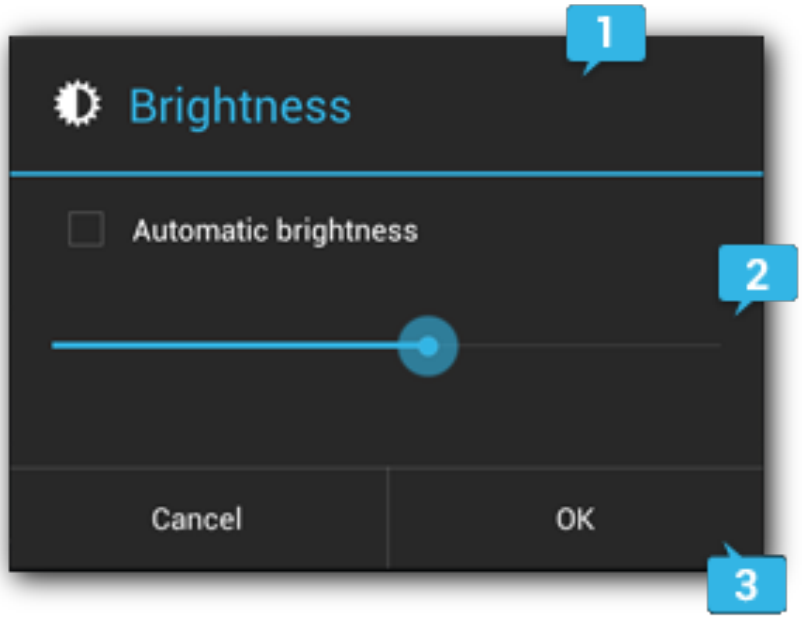


Figure 6: AlertDialog.

AlertDialog sadrži:

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

ExampleActivity.java

```
AlertDialog dialog = new AlertDialog.Builder(ExampleActivity.this)
  .setMessage(R.string.message_id)
  .setPositiveButton(
    R.string.btnOK,
    new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int id) {
        // ...
      }
    })
  .create();
dialog.show();

```

Dijalozi

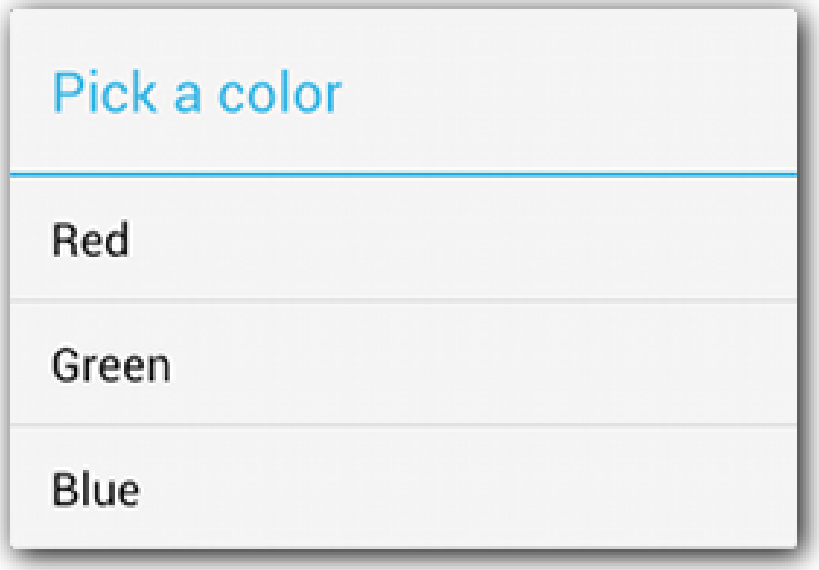


Figure 7: Dijalog sa listom.

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

Dijalog sa listom - ExampleDialogFragment.java

```
AlertDialog dialog = new AlertDialog.Builder(getActivity())
    .setTitle(R.string.pick_color)
    .setItems(
      R.array.colors_array,
      new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
          // ...
        }
      })
    .create();
dialog.show();

```

- U projektu se nizovi mogu definisati kao resursi: File -> New -> Android Resource File -> Resource Type: Values

```
<resources>
    <string-array name="colors_array">
        <item>Red</item>
        <item>Green</item>
        <item>Blue</item>
    </string-array>
</resources>

```

Dijalozi sa rasporedom

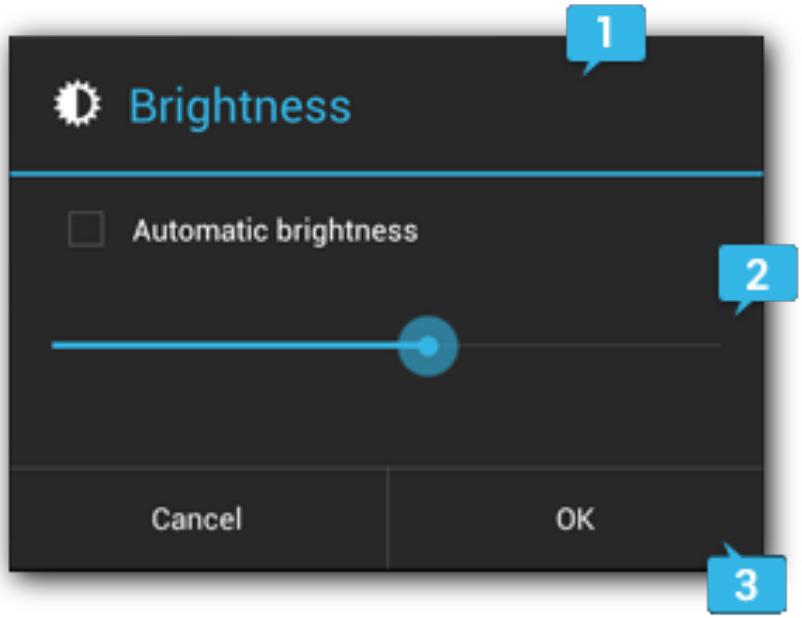


Figure 8: Dijalog sa rasporedom.

- Podrazumevano ponašanje je da raspored zauzme ceo dijalog.
- Ali je ipak moguće dodati i naslov i dugmad.

Dijalog sa rasporedom - ExampleDialogFragment.java

```
AlertDialog dialog = new AlertDialog.Builder(getActivity())
  .setView(
    getActivity().getLayoutInflater().inflate(R.layout.dialog_layout, null))
  .setPositiveButton(
    R.string.btnOK,
    new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int id) {
        // ...
      }
    })
  .create();
dialog.show();

```

TimePicker & DatePicker

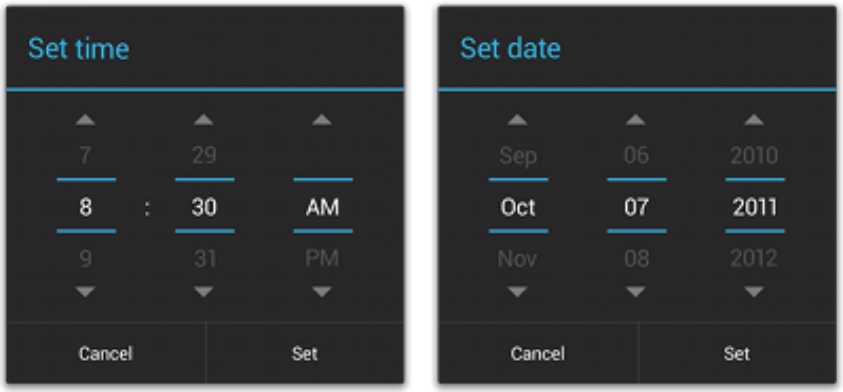


Figure 9: TimePicker & DatePicker.

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

DatePicker

```
public class DatePickerFragment
  extends DialogFragment
  implements DatePickerDialog.OnDateSetListener {

  @Override
  public Dialog onCreateDialog(Bundle state) {
    Calendar c = Calendar.getInstance();
    int year = c.get(Calendar.YEAR);
    int month = c.get(Calendar.MONTH);
    int day = c.get(Calendar.DAY_OF_MONTH);
    return new DatePickerDialog(getActivity(), this, year, month, day);
  }

  @Override
  public void onDateSet(DatePicker view, int year, int month, int day) {
    // ...
  }
}
// month value is zero-based (0,1,2,...)

```

TimePicker

```
public class TimePickerFragment
  extends DialogFragment
  implements TimePickerDialog.OnTimeSetListener {

  @Override
  public Dialog onCreateDialog(Bundle state) {
    Calendar c = Calendar.getInstance();
    int hourOfDay = c.get(Calendar.HOUR_OF_DAY);
    int minute = c.get(Calendar.MINUTE);
    return new TimePickerDialog(getActivity(), this, hourOfDay, minute, true);
  }

  @Override
  public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
    // ...
  }
}

```

Prikazivanje dijaloga

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

Pregled sadržaja

Podešavanja

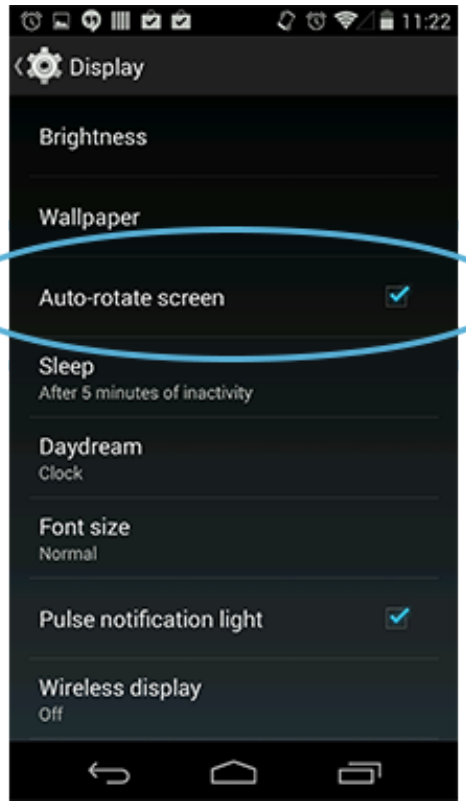


Figure 10: Podešavanja.

- 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

Podešavanja

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

Table 4: Tipovi kontrola.

res/xml/preferences.xml

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

    <CheckBoxPreference
        android:key="pref_sync"
        android:title="@string/pref_sync"
        android:summary="@string/pref_sync_summ"
        android:defaultValue="true" />

    <ListPreference
        android:dependency="pref_sync"
        android:key="pref_syncConnectionType"
        android:title="@string/pref_syncConnectionType"
        android:dialogTitle="@string/pref_syncConnectionType"
        android:entries="@array/pref_syncConnectionTypes_entries"
        android:entryValues="@array/
      pref_syncConnectionTypes_values"
        android:defaultValue="@string/
      pref_syncConnectionTypes_default" />

</PreferenceScreen>
```

- Kreiranje foldera za XML resurse: File -> New -> Folder -> XML Resources Folder
- Dodavanje XML resursa: File -> New -> XML Resource File

Podešavanja

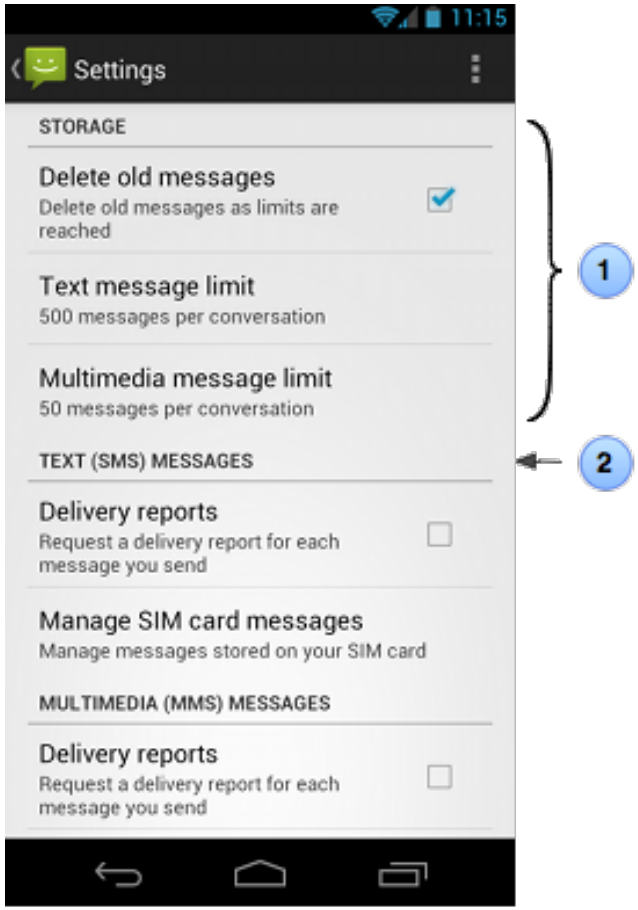


Figure 11: Podešavanja sa naslovima.

Grupisanje podešavanja u kategorije - preferences.xml (1/2)

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

    <PreferenceCategory
        android:title="@string/pref_sms_storage_title"
        android:key="pref_key_storage_settings">

        <CheckBoxPreference
            android:key="pref_key_auto_delete"
            android:summary="@string/pref_summary_auto_delete"
            android:title="@string/pref_title_auto_delete"
            android:defaultValue="false" />

```

Grupisanje podešavanja u kategorije - preferences.xml (2/2)

```
    <EditTextPreference
2       android:key="pref_key_sms_delete_limit"
        android:dependency="pref_key_auto_delete"
4       android:summary="@string/pref_summary_delete_limit"
        android:title="@string/pref_title_sms_delete" />
6
    <EditTextPreference
8       android:key="pref_key_mms_delete_limit"
        android:dependency="pref_key_auto_delete"
10      android:summary="@string/pref_summary_delete_limit"
        android:title="@string/pref_title_mms_delete" />
12
  </PreferenceCategory>
14
</PreferenceScreen>
16
```

Kreiranje fragmenta sa prikazom podešavanja

```
public class SettingsFragment extends PreferenceFragmentCompat {

    @Override
    public void onCreatePreferences(Bundle state, String rootKey) {
        // Load the preferences from an XML resource
        addPreferencesFromResource(R.xml.preferences);
    }

    // ...

}

// since API level 29, preferences use dependency:
// implementation 'androidx.preference:preference:1.2.0'

```

Aktivnost za prikaz fragmenta sa podešavanjima

```
public class SettingsActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle state) {
        super.onCreate(state);
        // Display the fragment as the main content
        getSupportFragmentManager()
            .beginTransaction()
            .replace(android.R.id.content, new SettingsFragment())
            .commit();
    }

    // ...

}
```

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

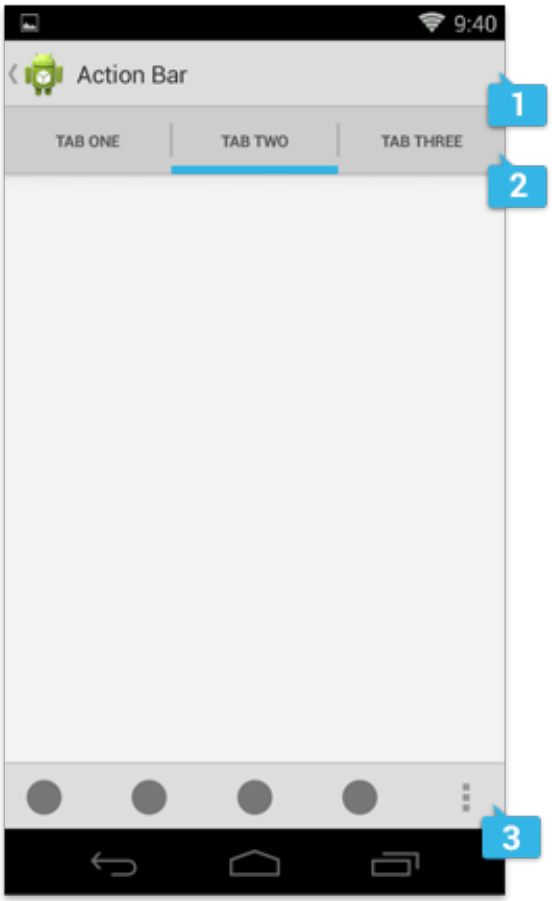


Figure 12: Toolbar.

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

Toolbar

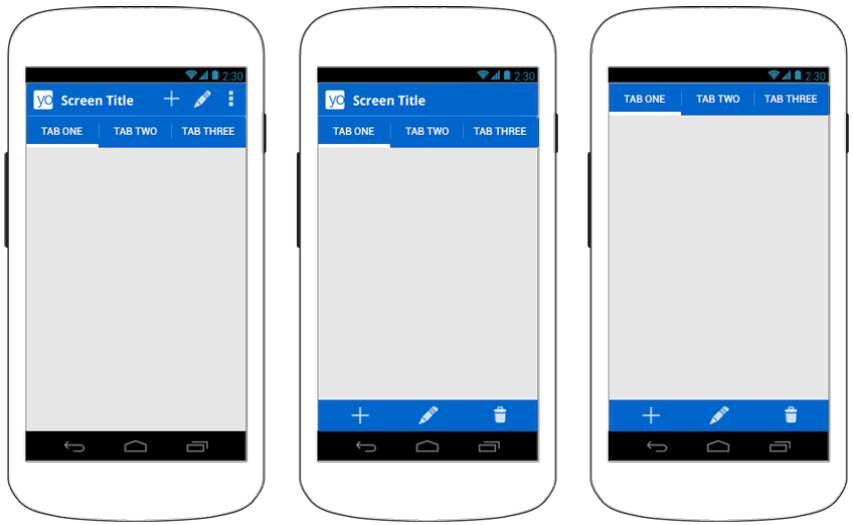


Figure 13: Toolbar.

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 appcompat i Material Design zavisnost 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

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
}
```

layout_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">

  <com.google.android.material.appbar.AppBarLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/ThemeOverlay.AppCompat.Dark">

      <androidx.appcompat.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                android:background="?attr/colorPrimary"
                app:layout_scrollFlags="scroll"/>

  </com.google.android.material.appbar.AppBarLayout>

</LinearLayout>

```

ExampleActivity.java

```
public class ExampleActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle state) {
        super.onCreate(state);
        setContentView(R.layout.layout_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.
        toolbar);
        setSupportActionBar(toolbar);
    }

}

```

AndroidManifest.java

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application android:theme="@style/Theme.AppCompat.
    Light.NoActionBar">
    <!-- ... -->
  </application>
</manifest>
```

ExampleActivity.java

```
ActionBar actionBar = getActionBar();
// Hides action bar
actionBar.hide();
// Shows action bar
actionBar.show();

```

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

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search"
        android:showAsAction="ifRoom|withText|always" />

    <item
        android:id="@+id/action_compose"
        android:icon="@drawable/ic_action_compose"
        android:title="@string/action_compose" />

</menu>
```

- Android Studio olakšava definisanje menija kao resursa:
  res -> New -> Android Resource File -> Resource Type: Menu

ExampleActivity.java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu items for use in the action bar
    getMenuInflater().inflate(R.menu.action_bar, menu);
    return super.onCreateOptionsMenu(menu);
}
```

ExampleActivity.java

```
public class ExampleActivity extends AppCompatActivity {

  // Menu icons are inflated just as they were with
    actionbar
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action
    bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
      case R.id.action_search:
        openSearch();
        return true;
      case R.id.action_compose:
        composeMessage();
        return true;
      default:
        return super.onOptionsItemSelected(item);
    }
  }
}
```

Up dugme

- Up dugme se prikazuje u toolbar-u kao strelica na levo i služi za povratak na prethodnu aktivnost
- Za prikazivanje Up dugmeta je potrebno:
  - U `AndroidManifest.xml` povezati (child) aktivnost sa drugom (parent) aktivnošću
  - U (child) aktivnosti pozvati `setDisplayHomeAsUpEnabled` metodu i proslediti joj argument `true`

AndroidManifest.xml

```
<manifest ... >
  <application ... >
    <!-- ... -->
    <!-- The main/home activity (has no parent activity) -->
    <activity android:name="com.example.MainActivity">
      <!-- ... -->
    </activity>

    <!-- A child of the main activity -->
    <activity
      android:name="com.example.ChildActivity"
      android:parentActivityName="com.example.MainActivity">
      <!-- ... -->
    </activity>
  </application>
</manifest>
```

ExampleActivity.java

```
public class ExampleActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle state) {
    super.onCreate(state);
    setContentView(R.layout.layout_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    ActionBar actionbar = getSupportActionBar();
    actionbar.setDisplayHomeAsUpEnabled(true);

    // instead left arrow as up button, an icon can be shown
    actionBar.setHomeAsUpIndicator(R.drawable.ic_menu);
  }

}

```

Pregled sadržaja

NavigationDrawer

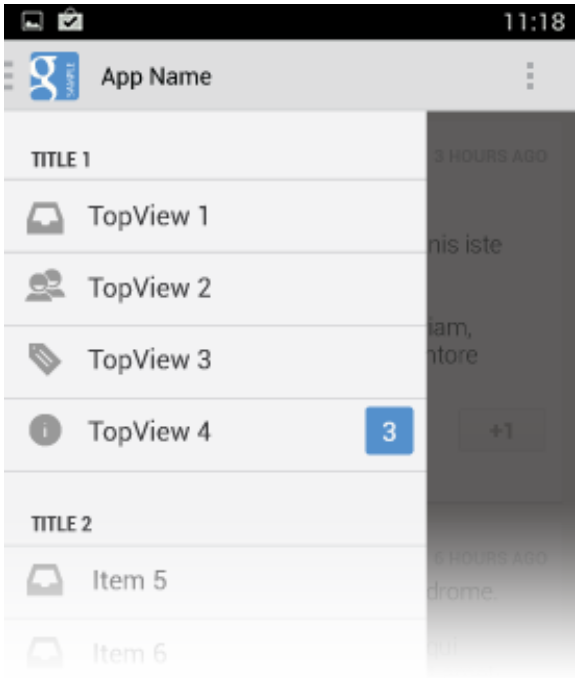


Figure 14: NavigationDrawer

- Deklarisati DrawerLayout raspored kao koreni raspored
- Dodati jedan pogled koji sadrži glavni sadržaj aktivnosti i drugi pogled (NavigationView) koji predstavlja sadržaj fioke za navigaciju
- Kreirati resurs (menu) sa stavkama fioke za navigaciju
- Reagovati na akcije korisnika

layout_main.xml

```
<androidx.drawerlayout.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:id="@+id/drawer_layout"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

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

  </LinearLayout>

  <!-- The navigation drawer view -->
  <com.google.android.material.navigation.NavigationView
        android:id="@+id/navigation_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:headerLayout="@layout/drawer_header"
        app:menu="@menu/drawer"/>

</androidx.drawerlayout.widget.DrawerLayout>

```

menu/drawer.xml

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

  <item
    android:icon="@drawable/icon_01"
    android:title="@string/menu_item01" />
  <item
    android:icon="@drawable/icon_02"
    android:title="@string/menu_item02" />

</menu>

```

ExampleActivity.java

```
public class ExampleActivity extends AppCompatActivity {

  DrawerLayout drawerLayout;

  @Override
  public void onCreate(Bundle state) {
    super.onCreate(state);
    setContentView(R.layout.layout_main);

    drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
    NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_view);
    navigationView.setNavigationItemSelectedListener(
      new NavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(MenuItem menuItem) {
          menuItem.setChecked(true);
          drawerLayout.closeDrawers();
          Toast.makeText(MainActivity.this, menuItem.getTitle(), Toast.LENGTH_LONG).show();
          return true;
        }
    });
  }
}

```