



# MOBILNE APLIKACIJE

Vežbe 3

GUI I

2024/2025

## Sadržaj

1. Resursi	3
1.1 Povezivanje resursa i Java klasa	4
2. Tipovi pogleda	6
2.1 Button	6
2.2 RadioButton	9
2.3 ToggleButton	9
2.4 CheckBox	9
2.5 TextView	10
2.6 ImageView	10
2.7 EditText	10
3. Rasporedi	11
3.1 Linearni raspored (Linear Layout)	12
3.2 Relativni raspored (Relative Layout)	12
3.3 Ograničavajući raspored (Constraint Layout)	13
3.4 Raspored okvira (FrameLayout)	14
3.5 Koordinirajući raspored (CoordinatorLayout)	14
3.6 Tabelačni raspored (TableLayout)	15
4. Domaći	17

## 1. Resursi

Struktura Android projekta može da se podeli na dve celine:

1. Funkcionalnosti
2. Resursi

**Funkcionalnosti** čine naše Java klase.

**Resursi** su tekstovi, slike, audio, video, ikonice itd. Svi resursi su smešteni u pakete koji se nalaze unutar direktorijuma *res*. Resursi unutar direktorijuma *res* su organizovani po tipovima u sledeće poddirektorijume:

- *drawable*
- *layout*
- *mipmap*
- *values*
- ...

Prilikom davanja naziva datotekama resursa potrebno je ispoštovati nekoliko pravila:

- Naziv datoteke treba da bude jedinstven.
- Koristiti samo mala slova.
- Dozvoljena je još upotreba i brojeva, donje crte i tačke.

### **drawable**

*Drawable* čine sve sličice i ikonice. U ovom direktorijumu se nalaze dodatni poddirektorijumi u kojima su slike razvrstane prema veličini (hdpi, mdpi, xhdpi, xxhdpi itd). Kada se pokrene aplikacija, Android će, u zavisnosti od rezolucije uređaja, znati da izabere odgovarajuću sliku/ikonu. Treba obezbediti različite dimenzije slika/ikonica, ali to ne znači da treba sami ručno da ih pravite. Pogledati: <https://developer.android.com/studio/write/resource-manager>

### **layout**

U ovom direktorijumu se čuvaju svi izgledi naših ekrana.

### **mipmap**

Za razliku od direktorijuma *drawable*, u ovom direktorijumu se čuvaju *app/launcher* ikonice, koje se prikazuju na *homescreen*-u uređaja.

### **values**

U *values* direktorijumu se navode boje, teme, stilovi, jezici itd. Postoji nekoliko xml datoteka u ovom direktorijumu:

- *string.xml* sadrži sve tekstove aplikacije. Ako je potrebno da se aplikacija lokalizuje, to znači da treba da se kreira novi direktorijum *res/values-kod\_drzave* (npr. *values-fr*, *values-ja*), koji će sadržati sav tekst preveden na taj drugi jezik. U

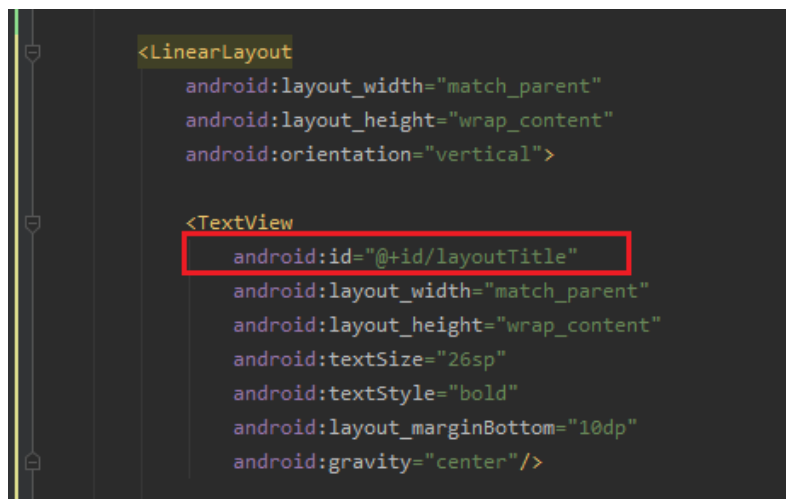
onom trenutku kada korisnik promeni jezik na uređaju, Android će promeniti jezik i u aplikaciji, tj. čitaće odgovarajući *values* direktorijum.

- *styles.xml* čine stilovi. Kada kreirate GUI Vaše aplikacije, vodite računa da aplikacija prati odgovarajuću temu.
- *colors.xml* su boje koje možete da koristite unutar komponenti.
- *arrays.xml* su statični nizovi unutar aplikacije.
- *themes.xml* predstavljaju teme unutar aplikacije. moguće je kreirati dark i light temu.

## 1.1 Povezivanje resursa i Java klasa

Resursi aplikacije mogu da se koriste u Java klasama uz pomoć *R.java* klase. Android održava posebnu datoteku *R.java*, koja se generiše svaki put kada se projekat izmeni. Ova datoteka čuva reference do svakog resursa unutar projekta.

Da bismo komponentu iz *layout*-a dobavili u *Java* klasi treba da je jedinstveno identifikujemo koristeći atribut *android:id* (**slika 1**).



Slika 1. *activity\_main.xml* postavljenje atributa *id* na komponentu

Kada se ID navodi u formatu: **@+id/vrednost** kreira se nova komponenta.

Kada je format: **@id/vrednost** referenciramo se na već postojeću komponentu.

Uz pomoć metode *findViewById* dobavljamo referencu komponente u java kodu (**slika 2**). Metoda *setText*, koju pozivamo nad komponentom, prima novi resurs, string iz *strings.xml* datoteke. String smo dobavili koristeći *dot* sintaksu. *R.string.mainActivityTitle* znači da unutar *res* foldera postoji string čiji *name* atribut ima vrednost *mainActivityTitle*.

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        /* ... */  
        TextView textView = findViewById(R.id.layoutTitle);  
  
        textView.setText(R.string.mainActivityTitle);  
  
        // U ovim fragmentima mozemo da vidimo da koristeci bilo k
```

Slika 2. Dobavljanje resursa u Java klasi

## 2. Tipovi pogleda

Tipovi pogleda koji postoje su:

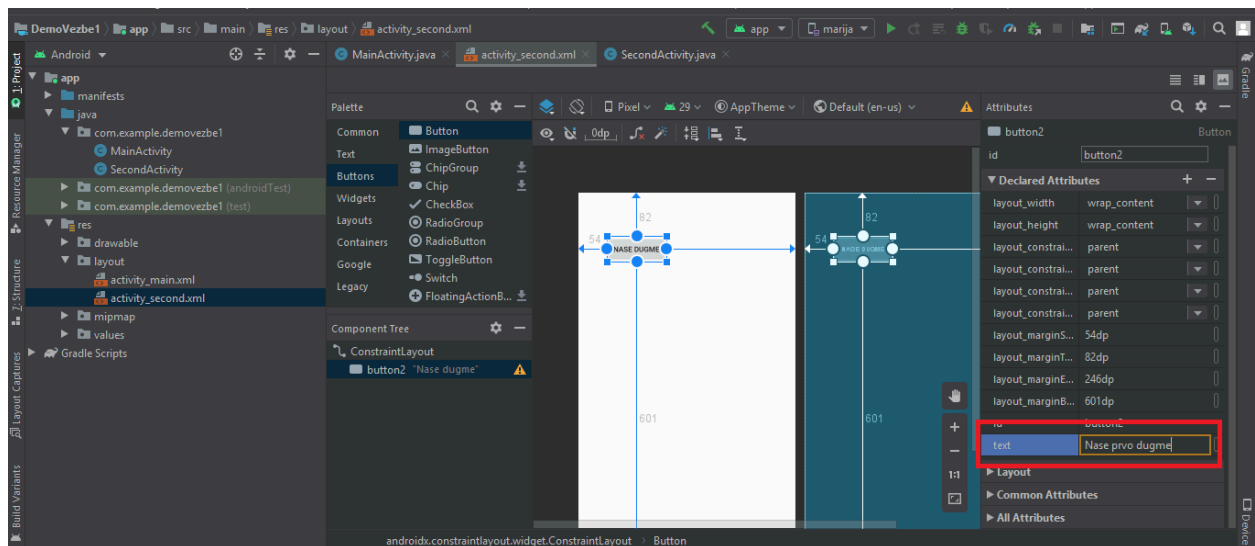
- *Button*
- *RadioButton*
- *ToggleButton*
- *Checkbox*
- *TextView*
- *ImageView*
- *EditText*

...

### 2.1 Button

Dugme (*button*) prikazuje tekst ili sliku koja simbolizuje određenu akciju. Kada korisnik pritisne dugme generiše se *click* događaj, koji obrađuje *onClick* metoda.

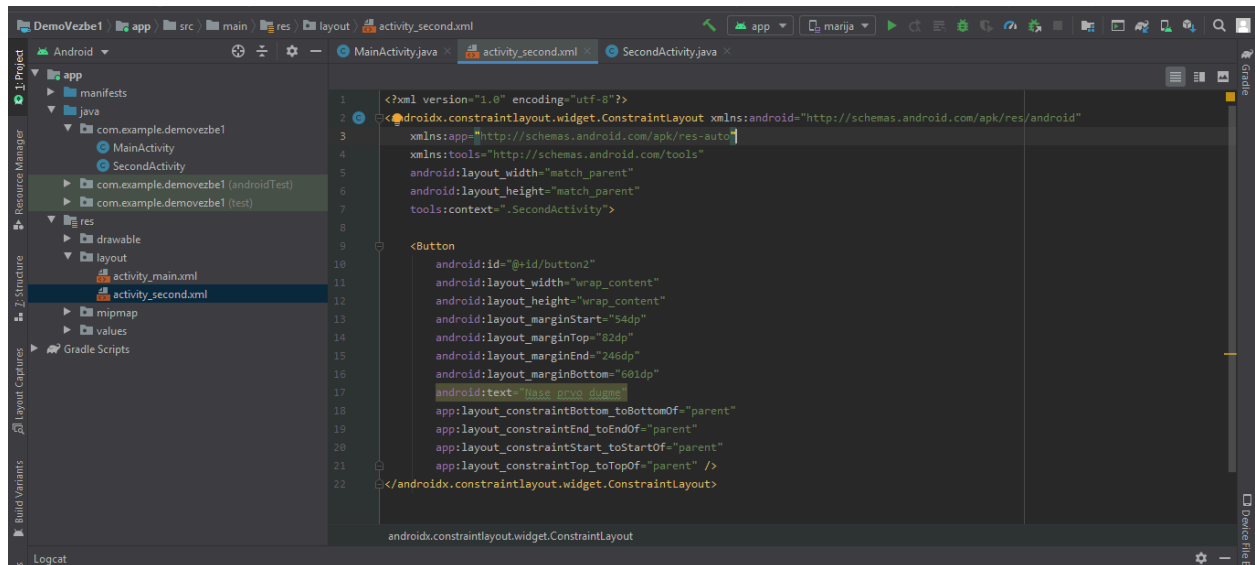
Kada otvorimo jedan od layout-a u *Design* režimu, možemo iz palete da prevlačimo i iscrtavamo različite elemente. Odabrali smo iz palete *Button* i prevukli ga na *canvas*, kao što se vidi na **slici 3**. Sa desne strane se nalazi lista atributa *Declared Attributes* i u polje *text* unosimo tekst: *Nase prvo dugme*. U istom trenutku taj tekst se prikazao i na dugmetu.



Slika 3. Kreiranje novog dugmeta u *Design* režimu

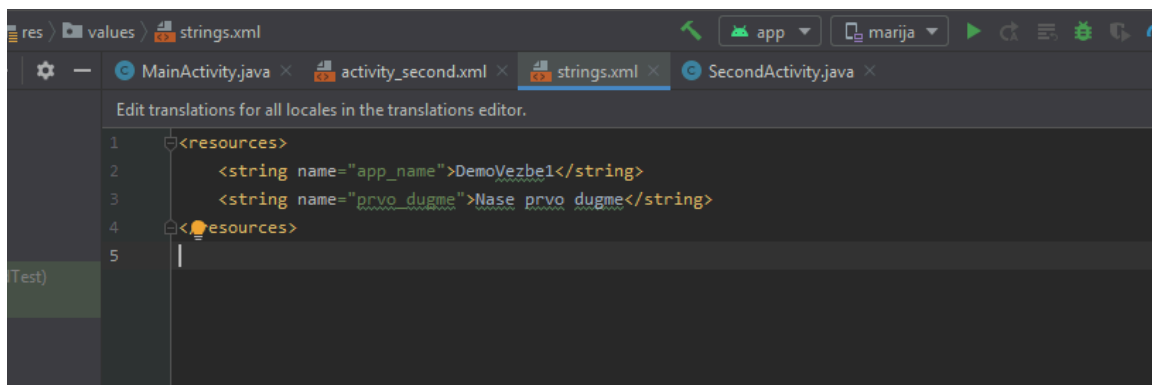
Prelaskom na *Code* režim (**slika 4**) vidimo da se datoteka proširila sa novim elementom `<Button>` i to je upravo isto dugme koje smo prevukli iz palete. Na slici se vidi da je atribut `android:text` označen žutom bojom. Kada pređemo kursom preko tog atributa iskače poruka sa tekstom: „Hardcoded string „Nase prvo dugme“, should use `@string` resource“. Naš trenutni kod će raditi, ali nam AS preporučuje da string „Nase prvo

*dugme*“ prebacimo u datoteku *strings.xml*, koja se nalazi unutar direktorijuma *res/values*.



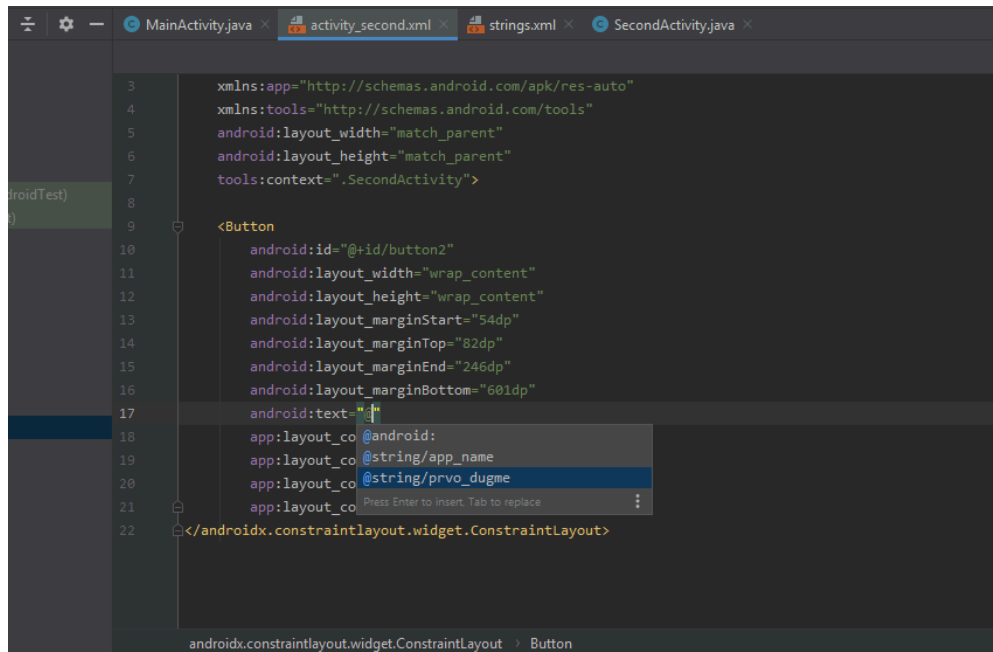
Slika 4. *activity\_second.xml* sa novim elementom

Novi string se navodi u datoteci *strings.xml* tako što se kreira element `<string>` i dodeli mu se atribut *name* (**slika 5**). Atribut *name* predstavlja identifikator za svaki string i njegova vrednost treba da bude jedinstvena. Sada kada smo kreirali novi string, treba da ga na neki način povežemo sa dugmetom u *activity\_second.xml* datoteci.

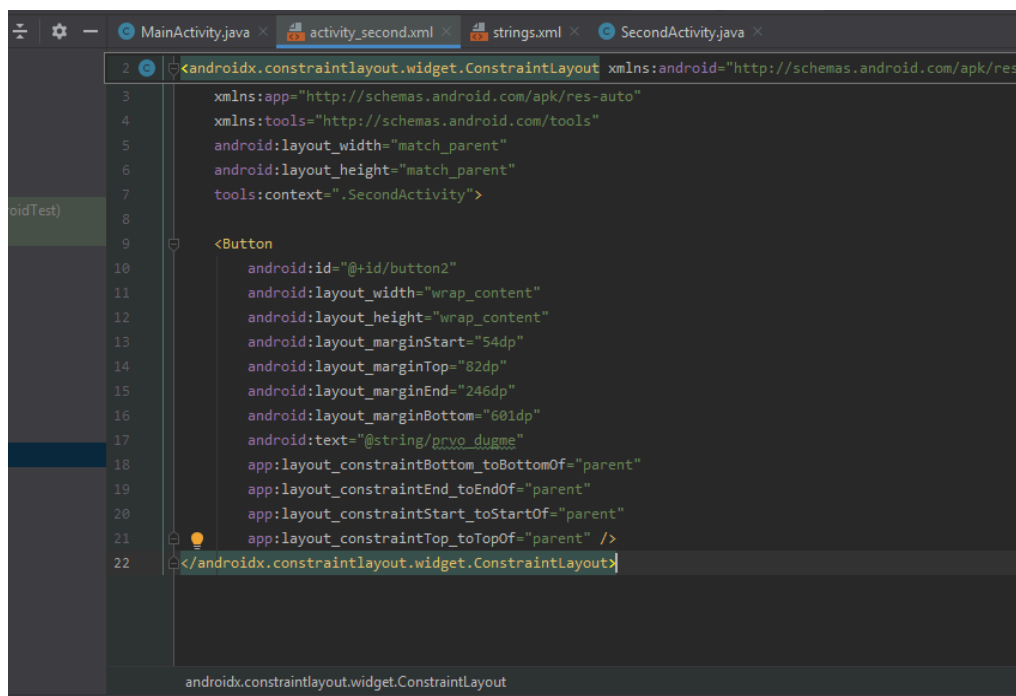


Slika 5. *strings.xml* proširen sa novim stringom

Pozicioniranjem kursora na *android:text* atribut i unosom karaktera „@“ AS nam nudi koje sve stringove možemo da dobavimo iz *strings.xml* datoteke (**slika 6**). Konačno, ovaj atribut postavljamo na `@string/prvo_dugme` (**slika 7**), jer je *prvo\_dugme* upravo vrednost *name* atributa našeg stringa.



Slika 6. Autocomplete AS-a



Slika 7. Izmenjen *android:text* atribut

## 2.2 RadioButton

Uz pomoć *RadioButton*-a korisnik može da izabere jednu opciju iz skupa više opcija (slika 8).



```
<RadioButton
    android:id="@+id/radioButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/radiobutton" />
```

Slika 8. *RadioButton*

## 2.3 ToggleButton

*ToggleButton* omogućava korisniku da promeni podešavanje između 2 stanja (**slika 9**).

```
<ToggleButton
    android:id="@+id/toggleButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/togglebutton" />
```

Slika 9. *ToggleButton*

## 2.4 CheckBox

Korisnik može da izabere jednu ili više opcija, iz skupa opcija, tako što označi *checkbox* (**slika 10**)

```
<CheckBox
    android:id="@+id/checkbox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/checkbox" />
```

Slika 10. *CheckBox*

## 2.5 TextView

*TextView* prikazuje tekst (**slika 11**).

```
<TextView
    android:id="@+id/layoutTitle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="26sp"
    android:textStyle="bold"
    android:layout_marginBottom="10dp"
    android:gravity="center"/>
```

Slika 11. *TextView*

## 2.6 ImageView

*ImageView* prikazuje sliku (**slika 12**).

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/about"
    android:padding="5dp"
    android:layout_margin="5dp"
    android:layout_below="@id/tag_text"
    android:contentDescription="@string/opis" />
```

Slika 12. *ImageView*

## 2.7 EditText

*EditText* omogućava unos teksta (**slika 13**). *InputType* atribut ima više vrednosti koje određuju tip unosa, kao što su tekst, broj, email, lozinka ili telefon, i utiču na tastaturu koja se prikazuje korisniku.

```
<EditText
    android:id="@+id/editText"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:autofillHints=""
    android:inputType="text" />
```

Slika 13. *EditText*

## 3. Rasporedi

**Raspored (layout)** određuje način na koji se korisnički interfejs (UI) elementi prikazuju i organizuju na ekranu. Raspored možemo definisati u XML datoteci ili programatski u Java (ili Kotlin) kodu.

Android pruža više različitih tipova rasporeda koji se koriste za postavljanje i pozicioniranje UI elemenata unutar aplikacije. Svaki od njih ima svoje specifične karakteristike i primenu.

Evo nekoliko najčešće korišćenih upravljača rasporedom u Androidu:

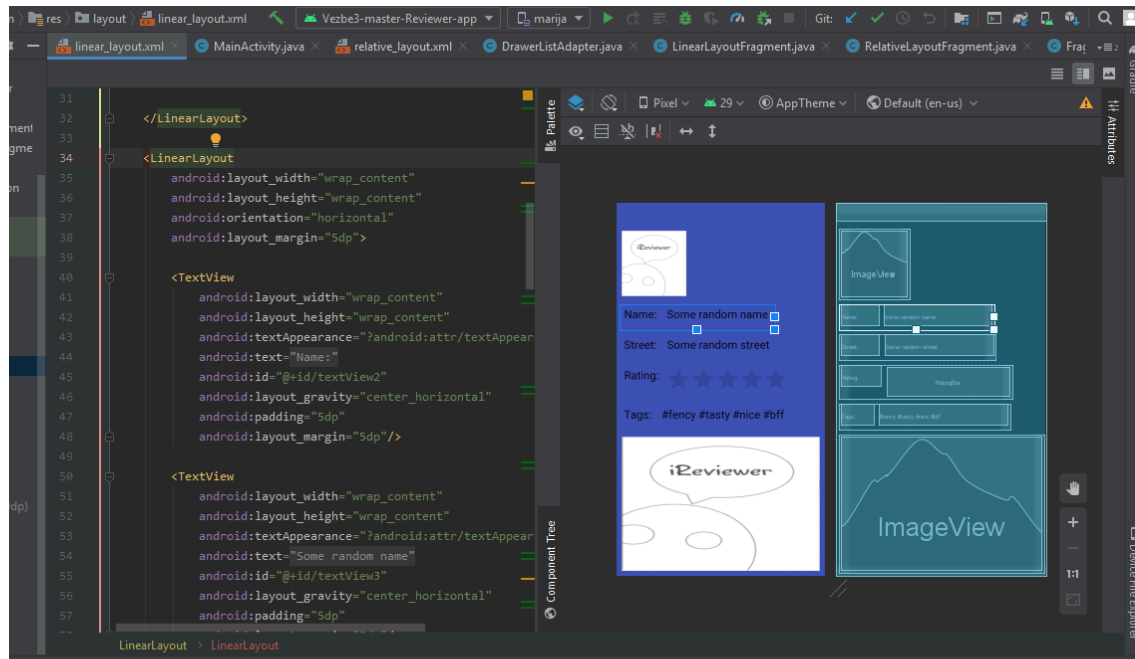
Postoji nekoliko vrsta rasporeda:

- Linearni raspored (*LinearLayout*)
  - Relativni raspored (*RelativeLayout*)
  - Ograničavajući raspored (*ConstraintLayout*)
  - Raspored okvira (*FrameLayout*)
  - Koordinirajući raspored (*CoordinatorLayout*)
  - Tabela raspored (*TableLayout*)
- 
- **LinearLayout** – raspoređuje elemente linearno, horizontalno ili vertikalno
  - **RelativeLayout** – omogućava pozicioniranje elemenata relativno u odnosu na druge elemente ili roditeljski raspored
  - **ConstraintLayout** – moćan raspored koji omogućava fleksibilno i efikasno pozicioniranje elemenata pomoću ograničenja
  - **FrameLayout** – koristi se za prikaz jednog elementa, dok se ostali preklapaju
  - **CoordinatorLayout** – napredni raspored koji omogućava koordinaciju ponašanja između više UI elemenata (npr. scroll efekti)
  - **TableLayout** – raspoređuje elemente u formi tabele, po redovima i kolonama

### 3.1 Linearni raspored (Linear Layout)

Pogledi (views) unutar linearnog rasporeda organizuju se u jednom pravcu – vertikalno ili horizontalno. Na **slici 14** prikazan je vertikalni **LinearLayout** (jedan pogled po redu), unutar kojeg se nalaze horizontalni **LinearLayout**-ovi (jedan pogled po koloni).

Atribut *android:orientation* određuje smer reda (horizontalno ili vertikalno).



Slika 14. Linearni raspored

### 3.2 Relativni raspored (Relative Layout)

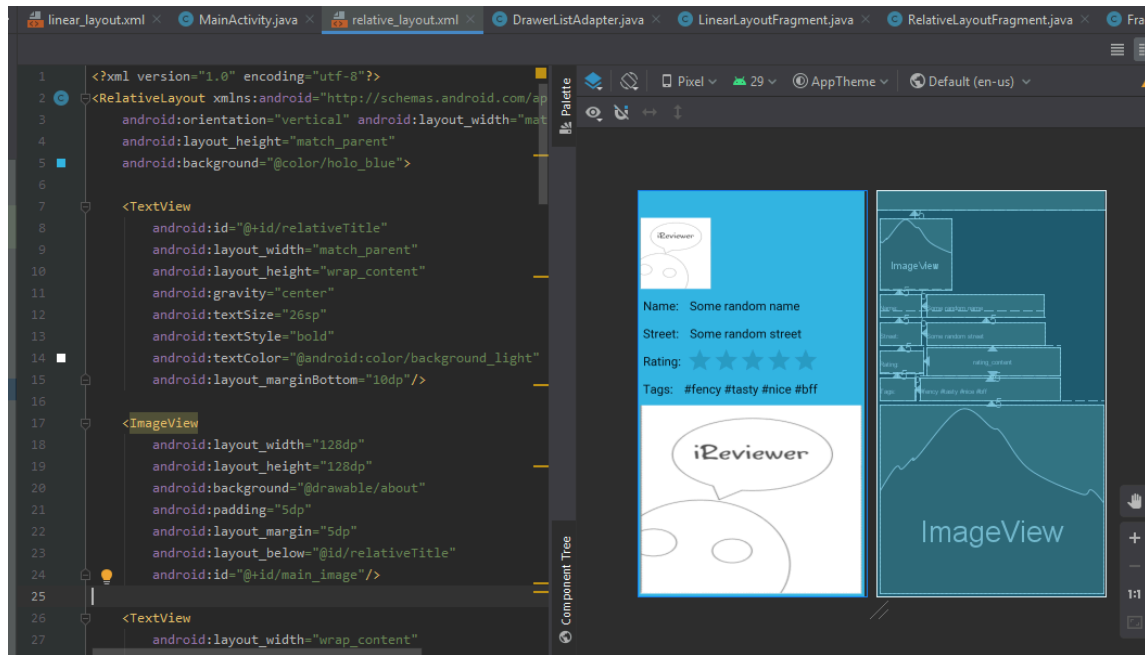
Relativni raspored raspoređuje pogledke relativno jedan u odnosu na drugi ili u odnosu na roditeljski pogled.

Možeš definisati pravila kao što su:

- poravnanje jednog pogleda sa vrhom, dnom, levom ili desnom ivicom drugog pogleda,
- pozicioniranje ispod ili pored određenog pogleda,
- ili postavljanje pogleda u odnosu na ivice roditeljskog rasporeda.

```
android:layout_below="@id/naslov"
android:layout_alignParentRight="true"
```

Ovaj tip rasporeda pruža veliku fleksibilnost u raspoređivanju elemenata, ali može postati nepregledan kod složenih UI struktura. Zato se danas često zamenjuje ConstraintLayout-om, koji pruža slične mogućnosti uz bolju optimizaciju.



Slika 15. Relativni raspored

### 3.3 Ograničavajući raspored (Constraint Layout)

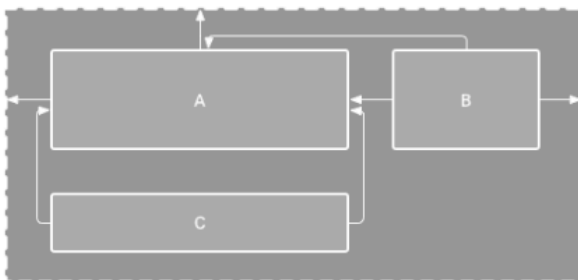
ConstraintLayout koristi ograničenja (constraints) kako bi definisao poziciju i veličinu pogleda u odnosu na druge poglede ili na sam roditeljski raspored.

To je fleksibilan i moćan raspored koji omogućava kreiranje kompleksnih i responzivnih korisničkih interfejsa bez potrebe za dubokom hijerarhijom pogleda (npr. bez ugnježđenih LinearLayout-ova).

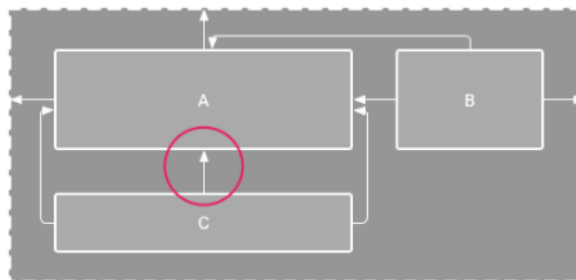
Umesto ugnježđivanja više rasporeda, svaki element u ConstraintLayout-u može biti povezan:

- sa drugim elementima (npr. levo/desno/iznad/ispod),
- ili sa granicama roditeljskog rasporeda.

Zbog svoje efikasnosti i prilagodljivosti, ConstraintLayout se preporučuje za pravljenje složenih i modernih UI dizajna u Android aplikacijama.



**Figure 1.** The editor shows view C below A, but it has no vertical constraint.



**Figure 2.** View C is now vertically constrained below view A.

Slika 16. Ograničavajući raspored

### 3.4 Raspored okvira (FrameLayout)

FrameLayout prikazuje više pogleda, ali ih raspoređuje jedan preko drugog, kao slojeve (eng. *stack*). U praksi, najčešće se koristi za prikaz jednog pogleda u određenom trenutku, dok su ostali skriveni.

Ovaj raspored je posebno koristan za dinamičko menjanje sadržaja, kao što je učitavanje fragmenata unutar kontejnera:

```
<FrameLayout
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

### 3.5 Koordinirajući raspored (CoordinatorLayout)

CoordinatorLayout je napredni raspored koji upravlja međusobnom interakcijom podpogleda koje sadrži. Dizajniran je za koordinaciju animacija i ponašanja elemenata, kao što su automatsko sakrivanje Toolbar-a prilikom skrolovanja ili sinhronizacija pokreta između AppBarLayout-a i FloatingActionButton-a.

Često se koristi u kombinaciji sa komponentama iz Android Design Support biblioteke (npr. Snackbar, AppBarLayout, CollapsingToolbarLayout, BottomSheetBehavior), i idealan je za kreiranje kompleksnih i interaktivnih UI elemenata.

```
<androidx.coordinatorlayout.widget.CoordinatorLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <!-- Toolbar itd. -->
```

```

</com.google.android.material.appbar.AppBarLayout>
<androidx.recyclerview.widget.RecyclerView
    android:layout_width="match_parent"
    android:layout_height="match_parent"

app:layout_behavior="@string/appbar_scrolling_view_behavior"
"/>

<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_margin="16dp"
    android:src="@drawable/ic_add" />
</androidx.coordinatorlayout.widget.CoordinatorLayout>

```

### 3.6 Tabelarni raspored (TableLayout)

TableLayout se koristi za kreiranje rasporeda u obliku tabele, sa redovima i kolonama, slično HTML tabelama. Svaki red se definiše pomoću TableRow elementa, a unutar svakog reda mogu se nalaziti različiti UI elementi (npr. TextView, EditText, Button).

TableLayout automatski organizuje ćelije u kolone prema broju elemenata u najdužem redu. Nema eksplicitno definisanih granica kolona kao u HTML-u – raspored zavisi od rasporeda sadržaja unutar redova. Osobine:

- Podržava više redova (TableRow)
- Kolone se automatski prilagođavaju sadržaju
- Možeš koristiti atribut android:layout\_span da spojiš ćelije
- Nema vidljivih linija – treba ih posebno dodati ako su potrebne

```

<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="Ime"
            android:padding="8dp"/>
        <EditText
            android:hint="Unesite ime"/>
    </TableRow>
    <TableRow>
        <TextView
            android:text="Email"
            android:padding="8dp"/>
        <EditText

```

```
        android:hint="Unesite email"/>
    </TableRow>
</TableLayout>
```



## 4. Domaći i primer

Domaći i primer se nalazi na *Canvas-u* ([canvas.ftn.uns.ac.rs](https://canvas.ftn.uns.ac.rs)) i na *Teams platformi* ([Files](#)) na putanji *Vežbe/03 Zadatak.pdf*.

Za dodatna pitanja možete se obratiti asistentu:

- Milan Podunavac ([milan.podunavac@uns.ac.rs](mailto:milan.podunavac@uns.ac.rs))
- Jelena Matković ([matkovic.jelena@uns.ac.rs](mailto:matkovic.jelena@uns.ac.rs))