



MOBILNE APLIKACIJE

Vežbe 4

GUI I

2022/2023

Sadržaj

1. Resursi	3
1.1 Povezivanje resursa i Java klasa	4
2. Tipovi pogleda	6
3. Rasporedi	11
4. Domaći	13

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://material.io/>

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

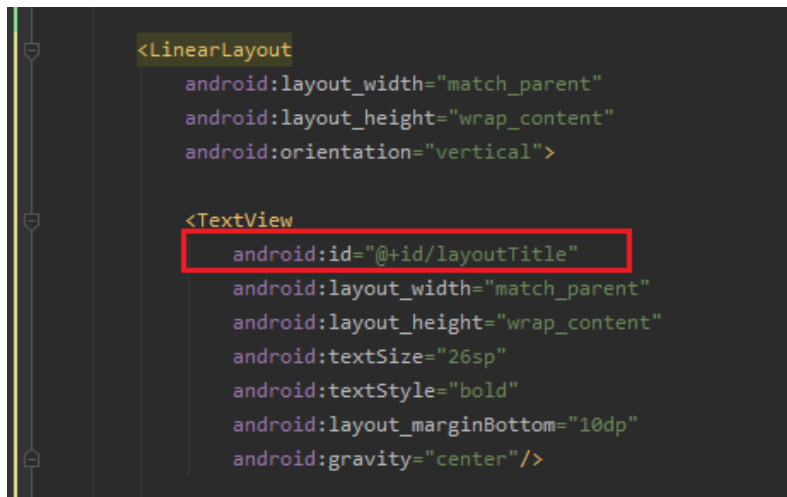
U *values* direktorijumu se navode boje, teme, stilovi, jezici, nizovi itd. Postoji nekoliko xml datoteka u ovom direktorijumu:

- *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.

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).



```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <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 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 (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

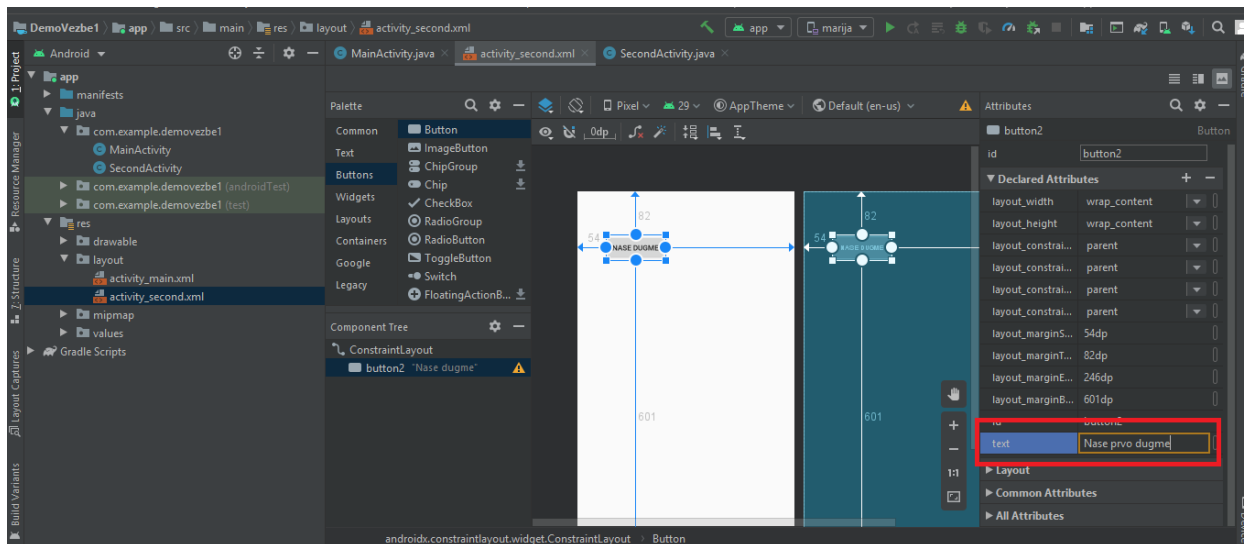
Tipovi pogleda koji postoje su:

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

Button

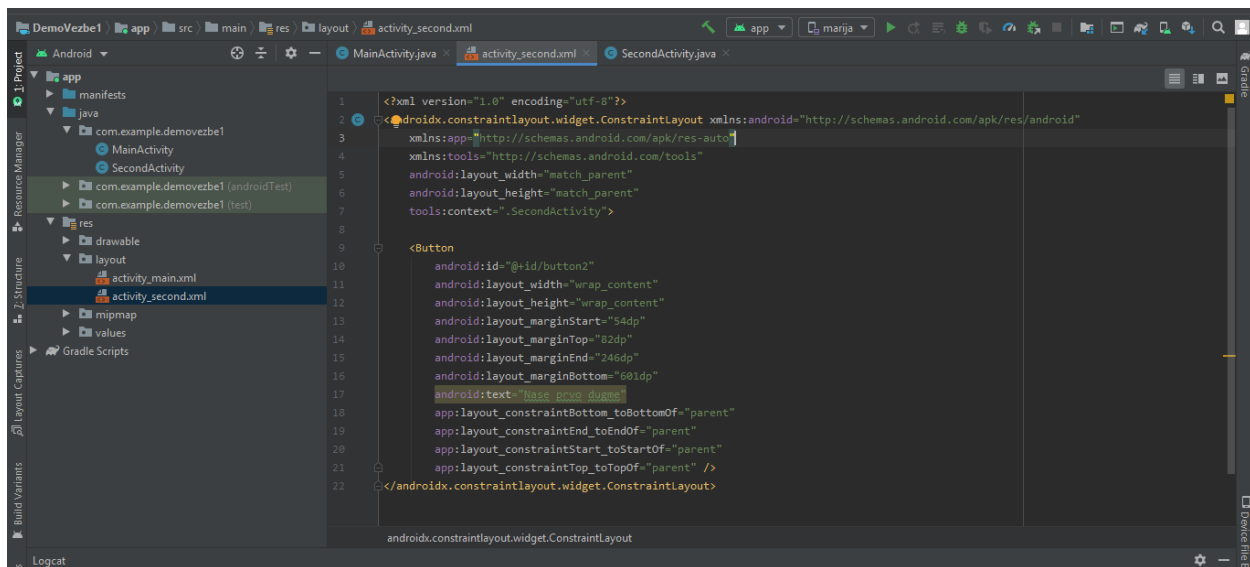
Dugme (*button*) prikazuje tekst ili sliku koja simbolizuje određenu akciju. Kada korisnik pritisne dugme generiše se *click* događaj, koji obrađuje *onClick* metoda.

Kada otvorimo jedan od layout-a u *Design* režimu, možemo iz palete da prevlačimo i isctavamo 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.



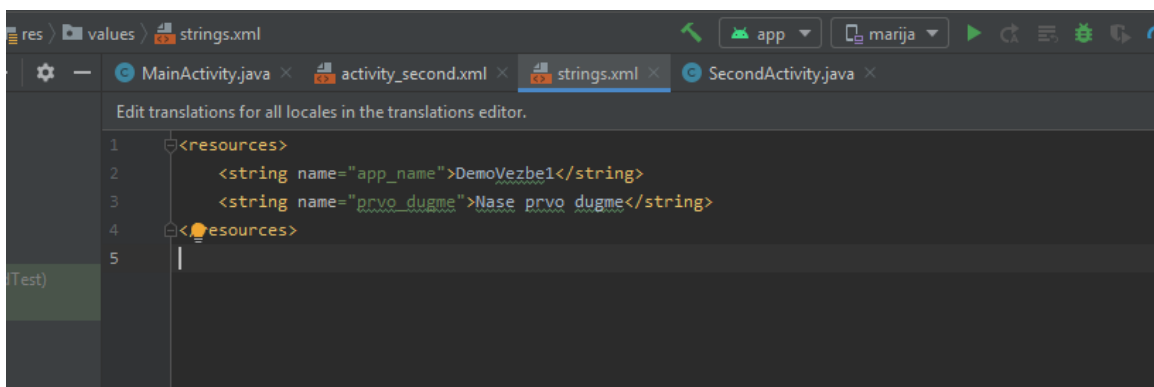
Slika 3. Kreiranje novog dugmeta u *Design* režimu

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*.



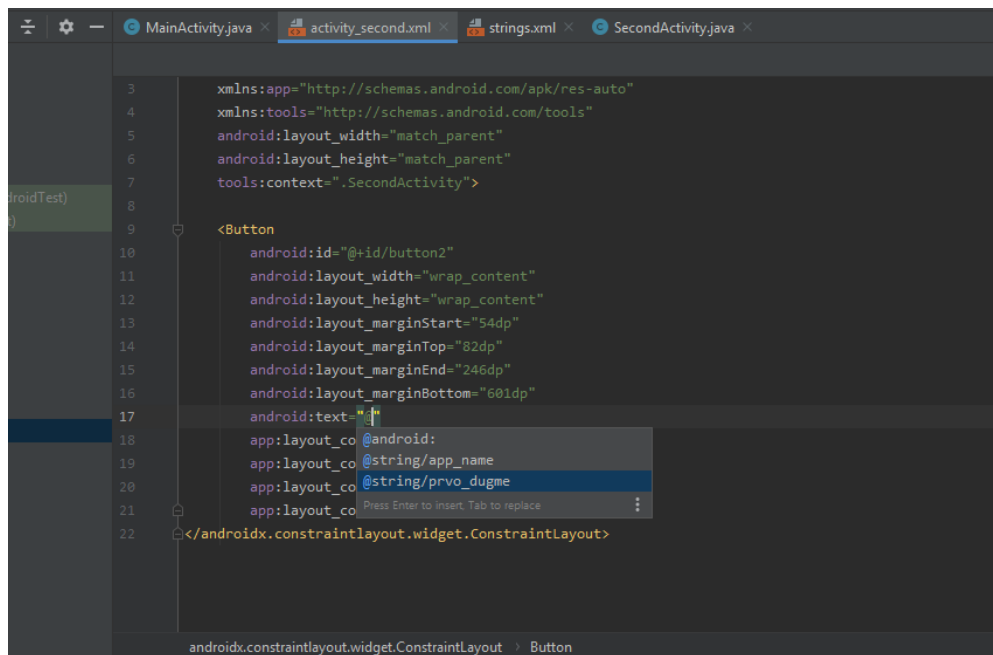
Slika 4. *activity_second.xml* sa novim elementom

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.

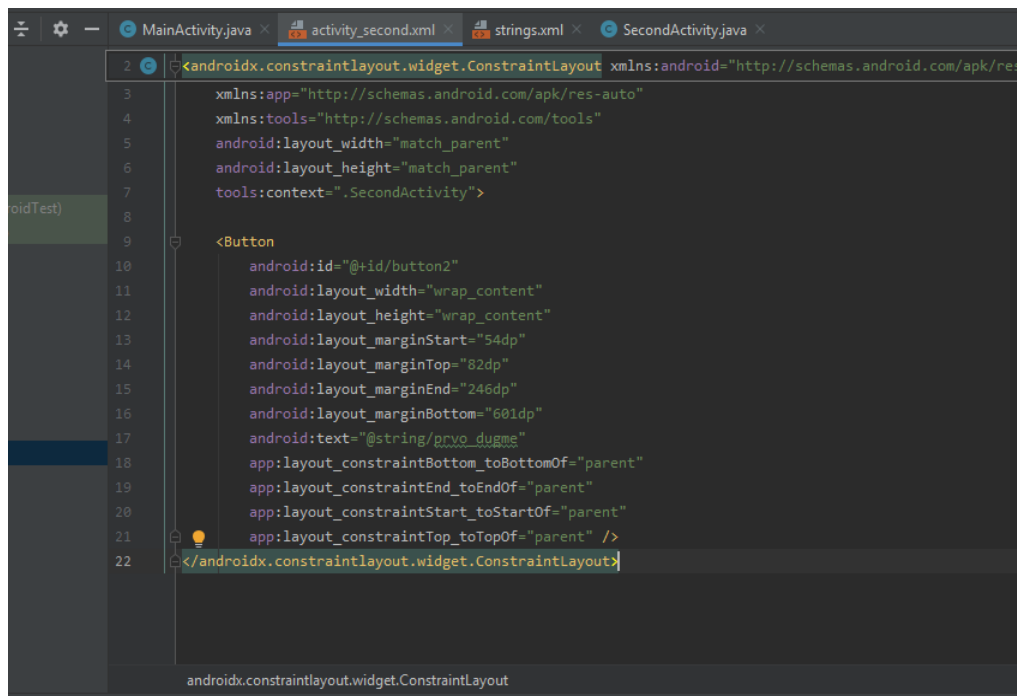


Slika 5. *strings.xml* proširen sa novim stringom

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

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*

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*

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*

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*

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*

EditText

EditText omogućava unos teksta (slika 13).

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

Slika 13. *EditText*

3. Rasporedi

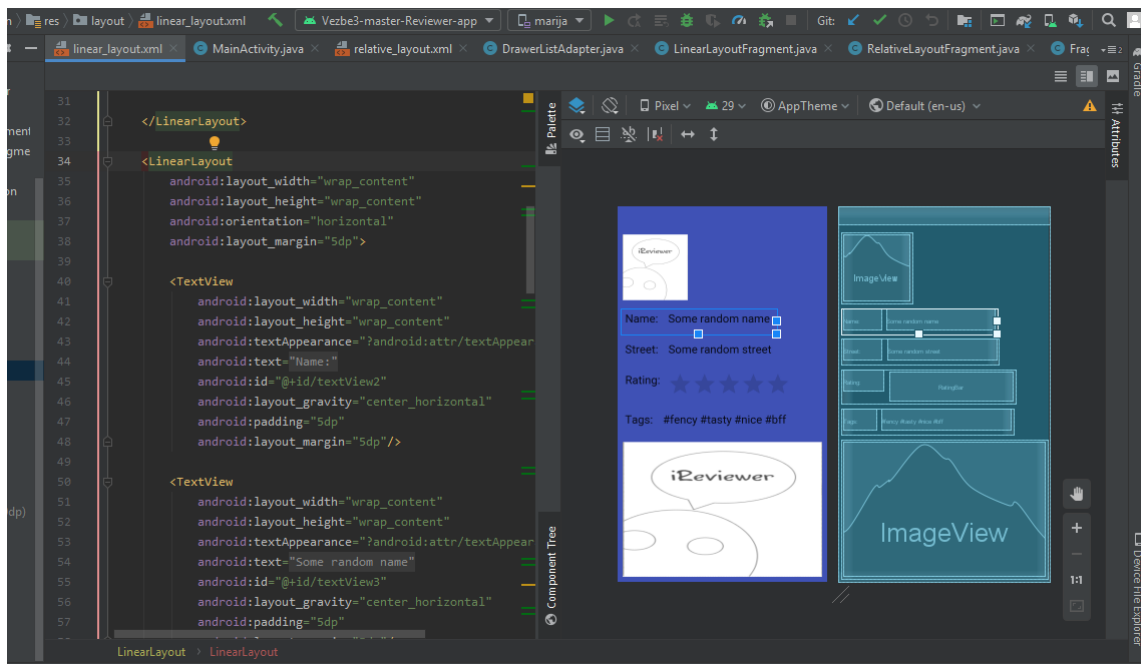
Raspored (*layout*) određuje na koji način se drugi rasporedi prikazuju na ekranu. Raspored možemo da definišemo u okviru XML datoteke ili u Java kodu.

Postoji nekoliko vrsta rasporeda:

- Linearni (*LinearLayout*)
- Relativni (*RelativeLayout*)
- Ograničavajući (*ConstraintLayout*)
- *FrameLayout*
- Koordinirajući (*CoordinatorLayout*)

Linearni raspored

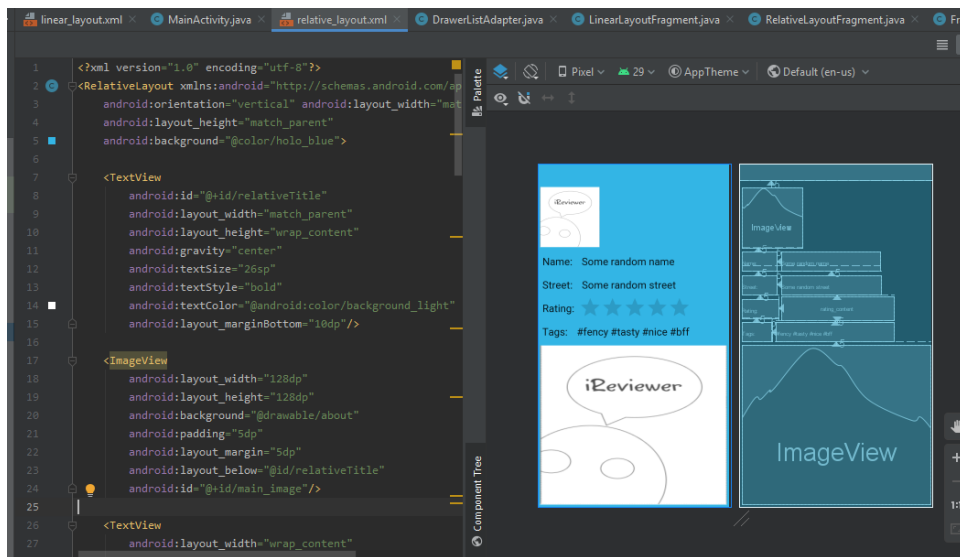
Pogledi koji se nalaze u linearnom rasporedu se raspoređuju u jednom pravcu (vertikalno ili horizontalno). Na slici 14 se nalazi vertikalni linearni raspored (jedan pogled u vrsti) u okviru kog se nalaze horizontalni linearni rasporedi (jedan pogled u koloni).



Slika 14. Linearni raspored

Relativni raspored

Relativni raspored raspoređuje poglede relativno u odnosu na sebe i jedan pogled u odnosu na drugi.



Slika 15. Relativni raspored

Ograničavajući raspored

Ograničavajući raspored omogućava određivanje pozicije i veličine pogleda. Sličan relativnom rasporedu ali je fleksibilniji od njega.

Frame Layout

Ovaj raspored prikazuje više pogleda, koji će biti raspoređeni jedan na drugom. Pogodan za kreiranje dijaloga i notifikacija u aplikaciji.

Koordinirajući raspored

Koordinirajući raspored upravlja interakcijom između pogleda koje sadrži i ti pogledi također mogu da komuniciraju jedni sa drugima. Namenjen je za dva slučaja: kao dekor aplikacije najvišeg nivoa ili hromirani izgled i kao kontejner za određenu interakciju sa jednim ili više podređenih prikaza.

4. Domaći

Domaći se nalazi na Canvas-u (canvas.ftn.uns.ac.rs) na putanji Vežbe/04 Zadatak.pdf.

Primer možete preuzeti na sledećem linku: <https://gitlab.com/antesevicceca/mobilne-aplikacije-sit>

Za dodatna pitanja možete se obratiti asistentima:

- ✉ Svetlana Antešević (svetlanaantesevic@uns.ac.rs)
- ✉ Jelena Matković (matkovic.jelena@uns.ac.rs)