OSNOVE RAZVOJA WEB I MOBILNIH APLIKACIJA

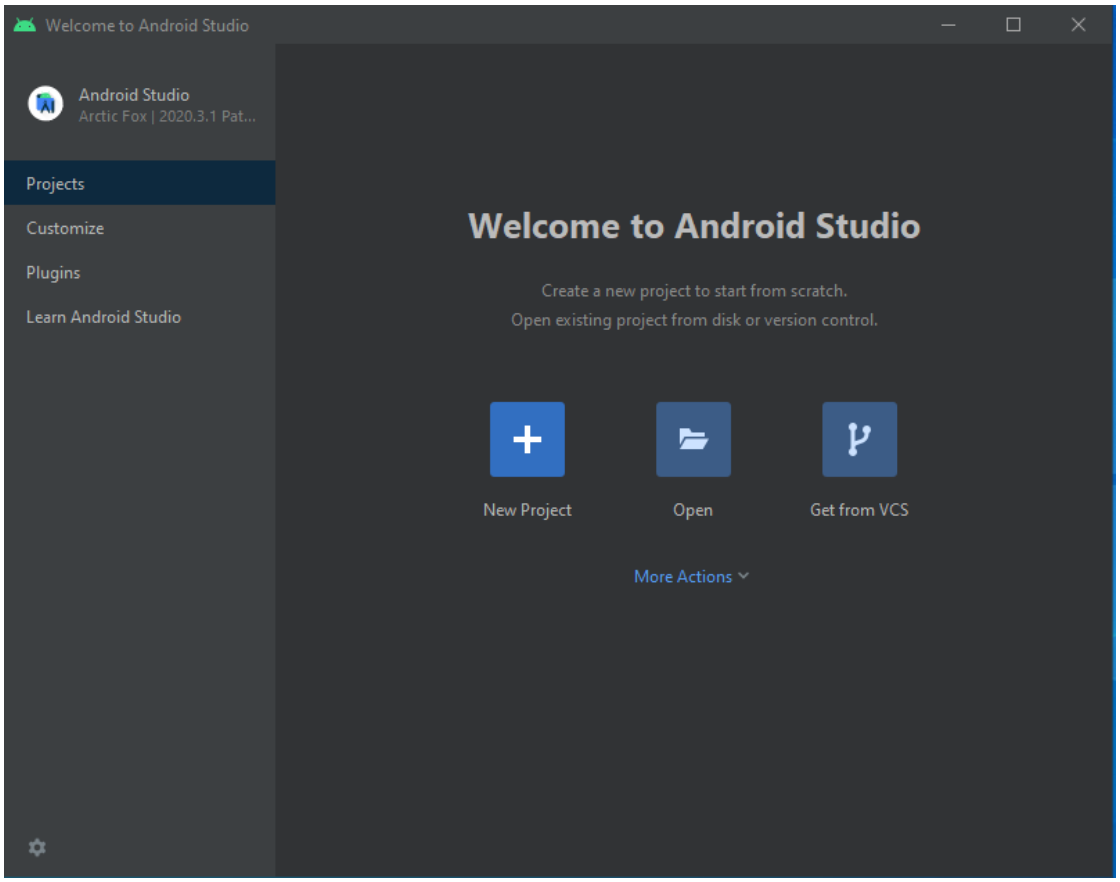
LV 5: Uvod u Android

Osijek, 2022.

1. PRIPREMA

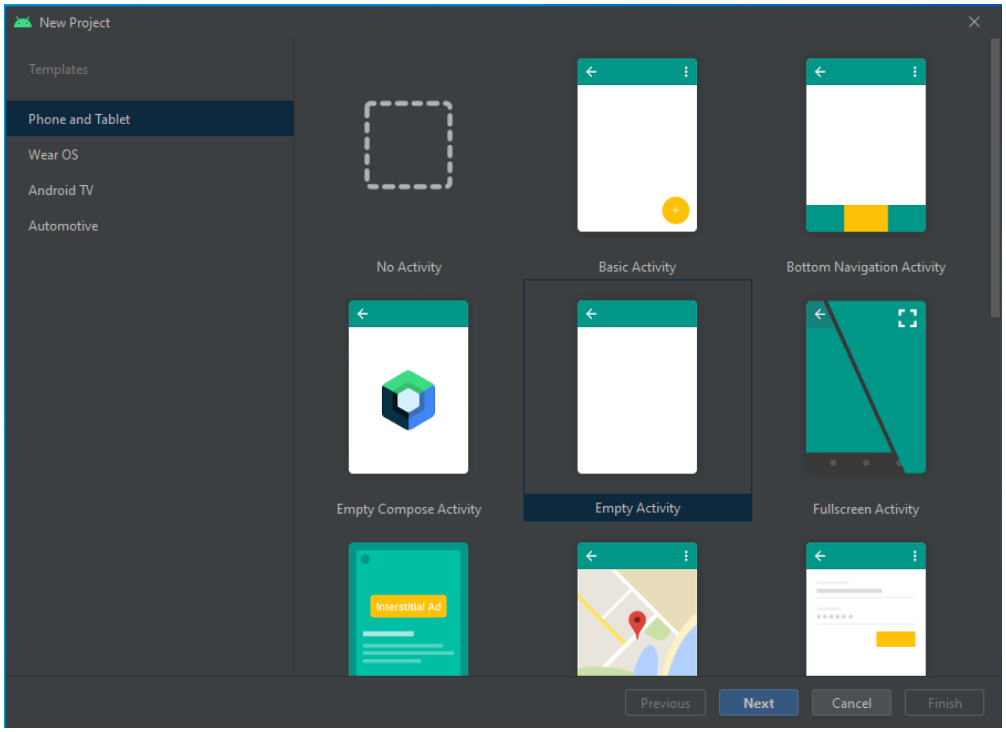
U ovoj je vježbi prikazano kreiranje prve aplikacije za Android sustav. Razmotreni su njeni osnovni dijelovi, od osnovnih gradivnih elemenata do korisničkog sučelja (engl. user interface, UI). Kroz primjere je pokazana izgradnja UI-ja u XML opisnom jeziku (engl. eXtensible Markup Language) korištenjem različitih kontrola, povezivanje događaja poput klika na gumb s kodom koji ga obrađuje te pokretanje aplikacije.

1.1. Kreiranje Android projekta



Slika 1.1. Prikaz glavnog izbornika unutar Android Studio programa.

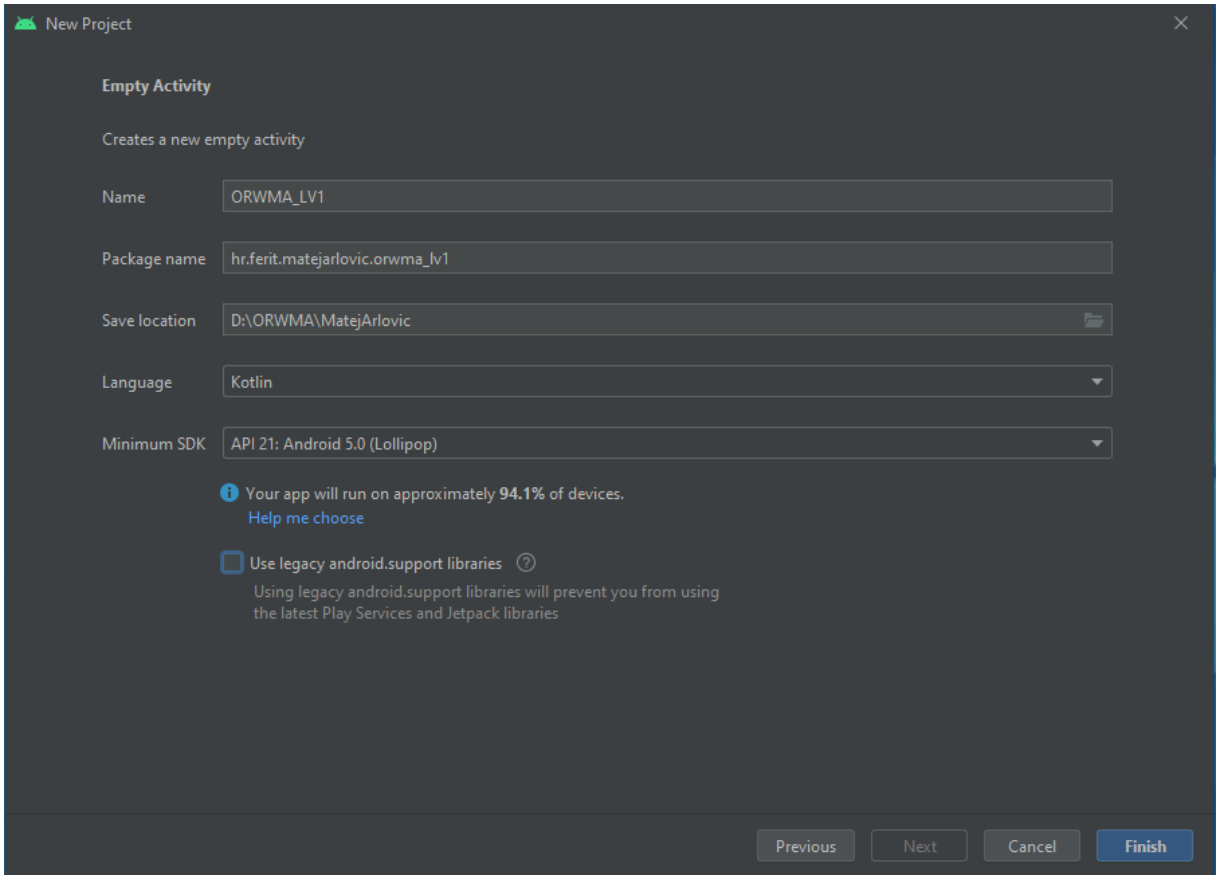
Kreiranje novog Android projekta moguće je obaviti iz glavnog izbornika kod pokretanja Android studija, kako je prikazano slikom 1.1. Klikom na *New Project* gumb otvara se novi izbornik nalik onome na slici 1.2., a koji omogućuje odabir početnog *Activitya*.



Slika 1.2. Prikaz izbornika za odabir početnog Activitya.

Ponuđeni su različiti predlošci, čak i mogućnost izostanka Activitya, a za potrebe ovih vježbi svi će novi projekti započinjati dodavanjem praznog Activitya (*Empty Activity*). Klikom na gumb *Next* korisniku se prikazuje zaslon nalik onome prikazanom na slici 1.3., koji omogućuje unos detalja vezanih uz sam projekt.

Prvenstveno se ovdje radi o nazivu aplikacije, nazivu paketa, lokaciji spremanja aplikacije, programskom jeziku u kojem se programira aplikacija i minimalnoj verziji Androida koju aplikacija podržava. Naziv aplikacije ne mora biti jedinstven u odnosu na druge aplikacije, ali naziv paketa aplikacije **MORA**. Android koristi standardnu konvenciju naziva paketa (*Package name*) još kada se koristio programski jezik Java za pravljenje Android aplikacija.



Slika 1.3. Prikaz izbornika za unos detalja o projektu.

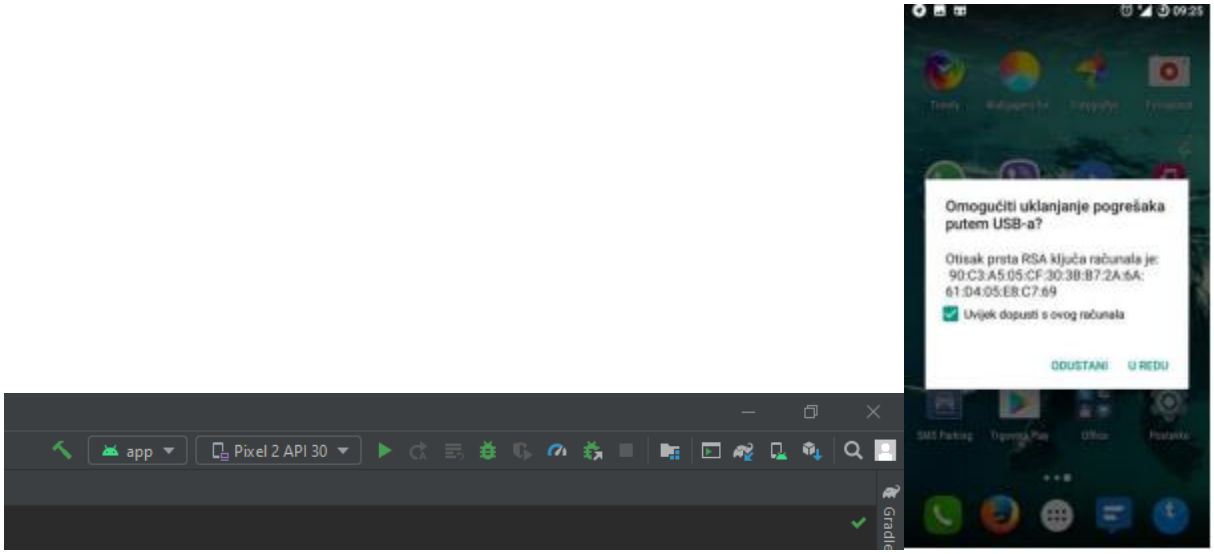
Paket mora biti jedinstven kako bi bilo moguće razlikovati projekte, odnosno aplikacije. Ime paketa određuje se stoga upravo kao obrnuta domena, s ciljem olakšavanja održavanja jedinstvenosti. Ukoliko ne posjedujete domenu, za učenje možete unijeti bilo koji naziv paketa. Za potrebe ovih vježbi koristite sljedeću konvenciju: **hr.ferit.imeprezime.naziv_projekta**. Za razvojni jezik odaberite **Kotlin**, za minimalnu podržanu verziju Androida odaberite **Android 5.0 (Lollipop)**.

Sve projekte potrebno je u laboratoriju smjestiti u **D:\ORWMA\ImePrezime**

Nakon klika gumba *Finish* IDE generira sve potrebne klase i zapise. Nakon kraćeg čekanja se otvaraju dvije datoteke i prikazuje čitav projekt. Android odvaja funkcionalnosti od prikaza stoga se sva funkcionalnost aplikacije nalazi unutar *MainActivity.kt* datoteke, a izgled (*layout*) se definira pomoću XML opisnog jezika unutar *layout_main.xml* datoteke.

1.1.1. Pokretanje Android projekta

Da bi se Android projekt pokrenuo, potrebno je imati spojen i uključen uređaj. Ovdje može biti riječ ili o stvarnom ili o virtualnom uređaju. Svi spojeni uređaji dostupni preko ADB-a vidljivi su unutar Android monitora na vrhu sučelja razvojnog okruženja, kako je prikazano slikom 1.4.



Slika 1.4. Prikaz odabira uređaja.

Virtualni uređaji radit će bez ikakvih problema, dok je za korištenje fizičkih uređaja potrebno dati dozvolu na samom uređaju. Klikom na gumb u alatnoj traci ili pritiskom na kombinaciju tipku Shift i F10 pokreće se aplikacija na odabranom uređaju.

1.2. Resursi

Kod razvoja aplikacija za Android platformu, kod se u pravilu razdvaja od izgleda, slika i drugih podataka koji se zajednički nazivaju resursima. Ovo razdvajanje u značajnoj mjeri olakšava izmjene, ali i prilagodbe različitim uređajima, tržištima i slično, jer je vrlo lako imati više različitih inačica istog resursa. Ove prilagodbe sustav će odraditi bez zahtjeva za dodatnim pisanjem koda ili intervencijama programera. Resursi za aplikaciju smješteni su u res mapi unutar projekta, a pakiraju se u .apk datoteku prilikom njena generiranja.

1.2.1. Izgled

Izgled Android aplikacija definira se resursima koji se nazivaju *layout*. Riječ je o datotekama napisanim uporabom XML opisnog jezika unutar kojih se definiraju elementi korisničkog sučelja. Svaki *layout* sadrži jedan korijenski element, koji mora biti ili *View* ili *ViewGroup* objekt. U korijenski se element mogu zatim ugnijezditi drugi elementi, odnosno moguće je izgraditi hijerarhiju koja sačinjava izgled zaslona aplikacije.

1.2.1.1. View

View je osnovna klasa koja predstavlja UI kontrolu. Osnovne i najčešće korišteni derivati *View* klase će se objasniti u nastavku. Najvažnije svojstvo svih *Viewa* jest id koji mora biti unikatan unutar cijelog projekta, a dodaje se tako da se u XML-u svojstvo id postavi na „***@+id/some_custom_id***“. Pomoću njega je moguće pristupiti elementu sučelja ili referencirati druge elemente.

Više informacija o View klasi i njenim derivatima moguće je pronaći na https://developer.android.com/reference/android/view/View.html

ConstraintLayout

ConstraintLayout je *layout* koji omogućava pravljenje velikih i kompleksnih responzivnih izgleda bez ugniježdavanja svih *Viewova* unutar istoga. Svi elementi unutar *ConstraintLayouta* se zatim pozicioniraju pomoću četiri točke vezanja i mijenjaju svoje pozicije u ovisnosti o dimenzijama zaslona. Primjer korištenja *ConstraintLayouta* prikazan je na slici 1.6.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Slika 1.6. Prikaz korištenja *ConstraintLayouta*.

TextView

TextView je klasa koja predstavlja UI kontrolu za prikaz tekstualnog sadržaja koji se ne može izravno uređivati od strane korisnika. Primjer korištenja *TextViewa* unutar izgleda prikazan je na slici 1.7. Da bi se dodao *TextView* unutar *layouta* potrebno je unutar XML datoteke dodati oznaku <TextView> (koja se također u istoj liniji može zatvoriti). Svojstvima *layout_width* i *layout_height* definiraju se dimenzije koje govore da visina i širina *TextView* elementa treba biti jednaka veličini sadržaja koji se nalazi unutar njega. Sa *layout_constraint(...)* definira se položaj *Viewa* u odnosu na određeni element (u ovome slučaju je to *parent* odnosno *ConstraintLayout*). Ta ista svojstva se odnose na druge *View* elemente (ButtonView, ImageView, itd.). Svojstvo *text* određuje sadržaj koji se nalazi unutar *Viewa*, no najbolje je sadržaj izdvojiti kao zaseban resurs koji je kasnije objašnjen.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Slika 1.7. Prikaz korištenja *TextViewa* unutar *layouta*.

EditText

EditText je klasa koja predstavlja UI kontrolu za unos niza znakova. Primjer korištenja *EditViewa* unutar izgleda prikazan je na slici 1.8.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editTextTextPersonName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Slika 1.8. Prikaz korištenja EditView unutar layouta.

Da bi se dodao *EditText* unutar *layouta* potrebno je unutar XML datoteke dodati oznaku <EditText>. Sadrži većinu svojstava kao i svi ostali *View* elementi. Svojstvo *text* određuje sadržaj koji se nalazi unutar *Viewa prije no što korisnik klikne na njega*. Svojstvom *inputType* se određuju pravila unosa znakova odnosno koja vrsta tipkovnice će se prikazati korisniku kada se klikne na *EditText*.

Button

Klasa Button predstavlja standardni gumb pritiskom na koji je moguće obaviti neku radnju. Da bi se *Button* dodao unutar *layouta* potrebno je unutar XML datoteke dodati oznaku <Button>. Sadrži većinu svojstava kao i svi ostali *View* elementi. Svojstvo *text* određuje sadržaj koji se nalazi unutar *Buttona*. Na slici 1.9. prikazano je korištenje *Buttona* unutar *layouta*.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Slika 1.9. Prikaz korištenja Buttona unutar layouta.

ImageView

Klasa ImageView predstavlja UI kontrolu za prikaz slika na zaslonu. Da bi se *ImageView* dodao unutar *layouta* potrebno je unutar XML datoteke dodati oznaku <ImageView>. Sadrži većinu svojstava kao i svi ostali *View* elementi. Svojstvo *srcCompat* određuje sadržaj koji se nalazi unutar *ImageViewa*. Na slici 1.10. prikazano je korištenje *ImageViewa* unutar layouta.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_launcher_background" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Slika 1.10. Prikaz korištenja ImageViewa unutar layouta.

1.2.2. Stringovi

Prilikom razvoja mobilnih aplikacija dobra je praksa razdvojiti stringove od koda i prikaza. *Stringovi* se izdvajaju kao resursi u *strings.xml* datoteku, smještenu u res/values mapu unutar hijerarhije projekta. Svaki je *string* smješten unutar oznaka <string> i </string>, a unutar otvarajuće oznake dodjeljeno mu je svojstvo *name*. Upravo ovo svojstvo predstavlja jedinstveni identifikator resursa. Prikazano je definiranje i korištenje stringova na slici 1.11.

```
<resources>
    <string name="app_name">ORWMA_LV1</string>
</resources>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/app_name"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Slika 1.11. Prikaz definiranja i korištenja stringova u layouta.

1.2.3. Boje

Heksadekadska reprezentacija RGB boje se izdvajaja kao resurs unutar colors.xml datoteke, koja je smještena unutar res/values mape. Svaka je boja smještena unutar oznaka <color> i </color>, a unutar otvarajuće oznake dodjeljeno joj je svojstvo *name* koji predstavlja jedinstveni identifikator resursa. Definiranje i korištenje boja prikazano je na slici 1.12.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

```xml
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/app_name"
    android:textColor="@color/purple_700"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Slika 1.12. Prikaz definiranja i korištenja boja unutar layouta.

1.2.4. Slike

Drawables je zajednički naziv za slike koje se koriste u aplikaciji. Podržani rasterski formati su: JPEG, GIF, PNG, WebP i NinePatch. Iako su podržani, dobra je praksa ne koristiti JPEG i GIF formate. Rasterske slike se uvijek kreiraju u nekoliko različitih veličina kako bi se podržale različite gustoće ekrana. Svaku sliku dobro je kreirati barem u 5 veličina i postaviti u odgovarajuće mape unutar hijerarhije projekta (mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi). U slučaju da za određenu rezoluciju nisu pruženi resursi, sustav će odabrati najbliži resurs i skalirati ga.

Više informacija o Drawable resursima moguće je pronaći na https://developer.android.com/guide/topics/resources/drawable-resource.html

2. RAD NA VJEŽBI

Iz teorije i koraka prikazanih u pripremi ove laboratorijske vježbe potrebno je stvoriti prvu Android aplikaciju pomoću čarobnjaka za stvaranje projekta u Android Studiju.

2.1. Obrada klika Button elementa

U ovome primjeru bi trebalo realizirati brojač gore/dole klikom na dva gumba, rezultat bi trebalo ispisati na zaslon korisnika. Nakon stvaranja novog projekta potrebno je unutar *activity_main.xml* stvoriti dva *Buttona* i jedan *TextView*. Na slici 2.1. prikazana je definicija zadanih elemenata.

strings.xml

```
<resources>
    <string name="app_name">ORWMA_LV2</string>
    <string name="increase_text">Povećaj</string>
    <string name="decrease_text">Smanji</string>
</resources>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="148dp"
        android:text="TextView"
        android:textColor="@color/teal_700"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="64dp"
        android:text="@string/increase_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />
```

```
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/decrease_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button2" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Slika 2.1. Prikaz definiranja zadanih elemenata.

Nakon postavke *layouta* potrebno je u *MainActivity* dohvatiti sve *Button* elemente iz *layouta* i namjestiti im da osluškuju klik elementa. Kako bi se pristupilo elementu iz layouta mora se koristiti *findViewById* metoda. *setOnClickListener* metoda se poziva svaki put kada se odabrani element klikne. To može biti gumb ili bilo koji drugi element koji nasljeđuje *View* klasu. Klikom na prvi gumb se povećava brojač za jedan i automatski se nova vrijednost brojača prikazuje na *TextViewu*. Klikom na drugi gumb se brojač smanjuje za jedan, te se automatski nova vrijednost brojača prikazuje na *TextViewu*. Programsko rješenje za *MainActivity* prikazano je na slici 2.2.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        var counter = 0
        val textView = findViewById<TextView>(R.id.textView2)
        findViewById<Button>(R.id.button2).setOnClickListener {
            counter += 1
            textView.text = counter.toString()
        }
        findViewById<Button>(R.id.button3).setOnClickListener {
            counter -= 1
            textView.text = counter.toString()
        }
    }
}
```

Slika 2.2. Prikaz programskog rješenja za MainActivity.kt.

2.2. Obrada klika na ImageView element

U ovome primjeru bi trebalo realizirati nestajanje slike kada korisnik na nju klikne.

Potrebno je unutar *activity_main.xml* stvoriti jedan *ImageView* element. Na slici 2.3. prikazana je definicija zadanog elementa. Naime slikama je moguće upravljati na različite načine kroz XML ili direktno u kodu. Svakom UI elementu koji je derivat klase *View* se može postaviti njegova vidljivost pomoću svojstva visibility. Tako *ImageView* može biti: vidljiv (VISIBLE), skriven (INVISIBLE) ili uklonjen sa zaslona (GONE). Osim toga *ImageViewu* se može urediti i alpha svojstvo koje upravlja prozirnošću *View* elementa.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_launcher_background" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Slika 2.3. Prikaz definiranja ImageView elementa unutar layouta.

Unutar MainActivity.kt potrebno je dohvatiti *ImageView* iz *layouta* pozivom *findViewById()* metode. Nakon toga je potrebno na taj *ImageView* staviti osluškivanje klik događaja korištenjem *setOnClickListener* metode. Programsko rješenje za *MainActivity* prikazano je na slici 2.4.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        var state = true
        val imageView = findViewById<ImageView>(R.id.imageView2)
        imageView.setOnClickListener {
            state = !state
            if (!state) {
                imageView.visibility = View.INVISIBLE
            }
            else {
                imageView.visibility = View.VISIBLE
            }
        }
    }
}
```

Slika 2.4. Prikaz programskog rješenja za MainActivity.kt.

2.3. Toast poruke

Obavijesti se na Android platformi korisniku pružiti na različite načine, a jedan od najčešćih je korištenje kratkotrajnih tekstualnih poruka koje se nazivaju Toast. S ovim porukama ne postoji mogućnost interakcije, a kreiraju se korištenjem statičkih metoda Toast klase.

strings.xml

```
<resources>
    <string name="app_name">ORWMA_LV1</string>
    <string name="push_text">Pritisni</string>
</resources>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/push_text"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val button = findViewById<Button>(R.id.button4)
        button.setOnClickListener {
            Toast.makeText(this, getString(R.string.app_name),
Toast.LENGTH_LONG).show()
        }
    }
}
```

2.4. Log poruke

Log klasa i njene statičke metode omogućuju ispis poruka različite razine važnosti unutar terminala odnosno logcata. Podržane su sljedeće razine: *verbose* (Log.v()), *debug* (Log.d()), info (Log.i()), *warning* (Log.w()), *error* (Log.e()) i *what a terrible failure* (Log.wtf()). Dobra je praksa definirati vlastite oznake (tagove) kako bi se mogli stvoriti filteri poruka.

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        Log.d("MAIN ACTIVITY", "Ovo je Log poruka u Main Activityu!")
    }
}
```

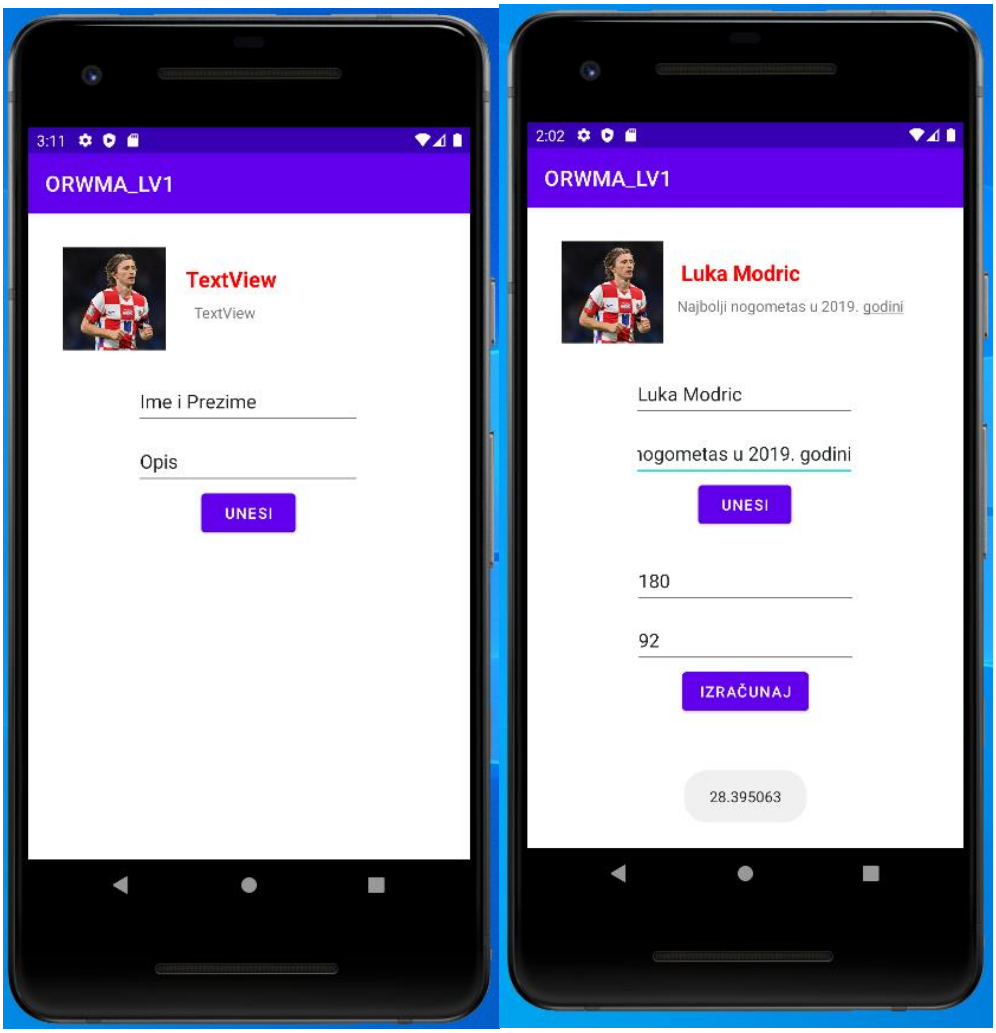
Logcat

```
D/libEGL: loaded /vendor/lib/egl/libGLESv1_CM_emulation.so
D/libEGL: loaded /vendor/lib/egl/libGLESv2_emulation.so
W/lovic.orwma_lv: Accessing hidden method Landroid/view/View;->computeFitSystemWindows(Landroid/g
    Accessing hidden method Landroid/view/ViewGroup;->makeOptionalFitsSystemWindows()V (greylist,
D/MAIN ACTIVITY: Ovo je Log poruka u Main Activityu!
D/HostConnection: HostConnection::get() New Host Connection established 0xf47b0bb0, tid 22418
D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 A
W/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
```

3. ZADACI ZA SAMOSTALAN RAD

1. Potrebno je napraviti Android aplikaciju koja će sadržavati sljedeće elemente: 1x *ImageView*, 2x *TextViewa*, 2x *EditTexta* i 1x *Button*. Promijenite boju sadržaja prvog *TextViewa* u crvenu tako da ju definirate u colors.xml i onda pozovete u *layoutu*. Osim toga promjenite sliku *ImageViewa* u neku sliku s interneta. Nakon što završite postavljanje izgleda (*layouta*) potrebno je sve povezati u programskom kodu. Kada korisnik klikne *Button* tada aplikacija treba ažurirati dva *TextViewa* sadržajem koji se nalazi unutar *EditTexta*.

2. Potrebno je unaprijediti Android aplikaciju iz zadatka 1. tako da dodate još dva *EditTexta* u koje se upisuju visina (u centimetrima) i težina (u kilogramima). Nakon pritiska na gumb „Izračunaj“ potrebno je u *Toastu* ispisati vaš BMI indeks. BMI se mora izračunati unutar zasebne funkcije , a formula za izračun BMI indeksa: kg/m^2



Slika 3.1. Prikaz kako bi trebalo izgledati rješenje zadatka 1 (lijeva) i zadataka 2 (desno).