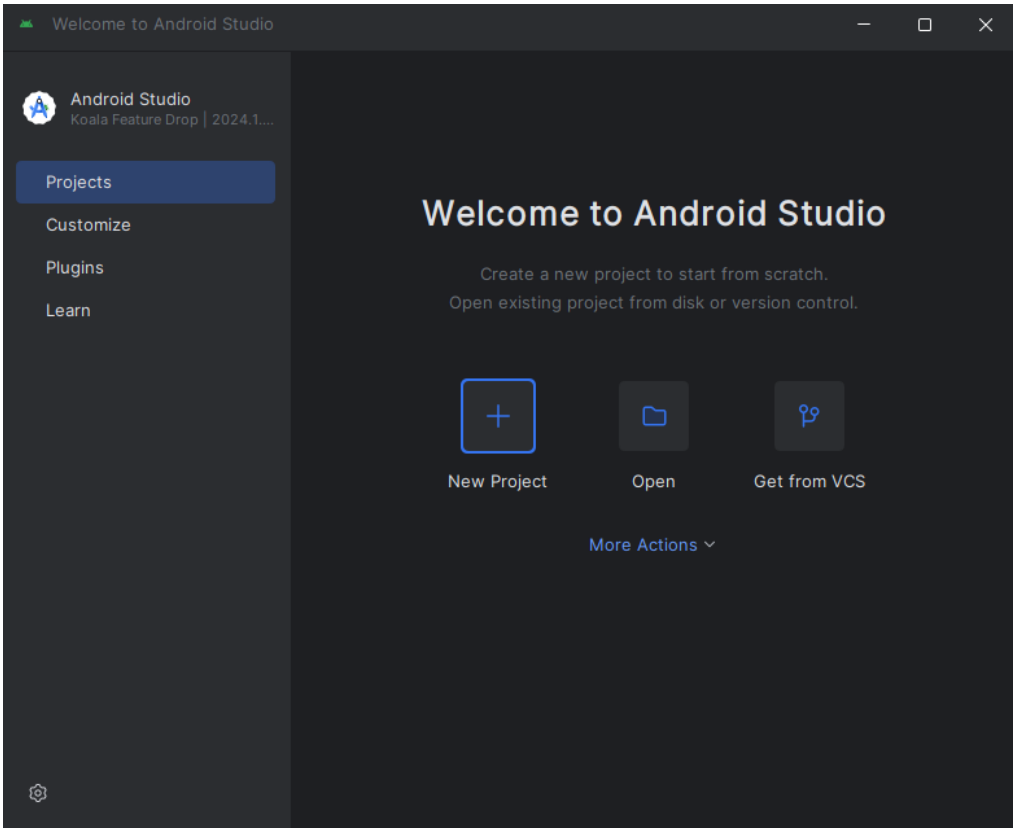
RAZVOJ MOBILNIH APLIKACIJA:

LV 2: Android Studio i uvod u Jetpack Compose

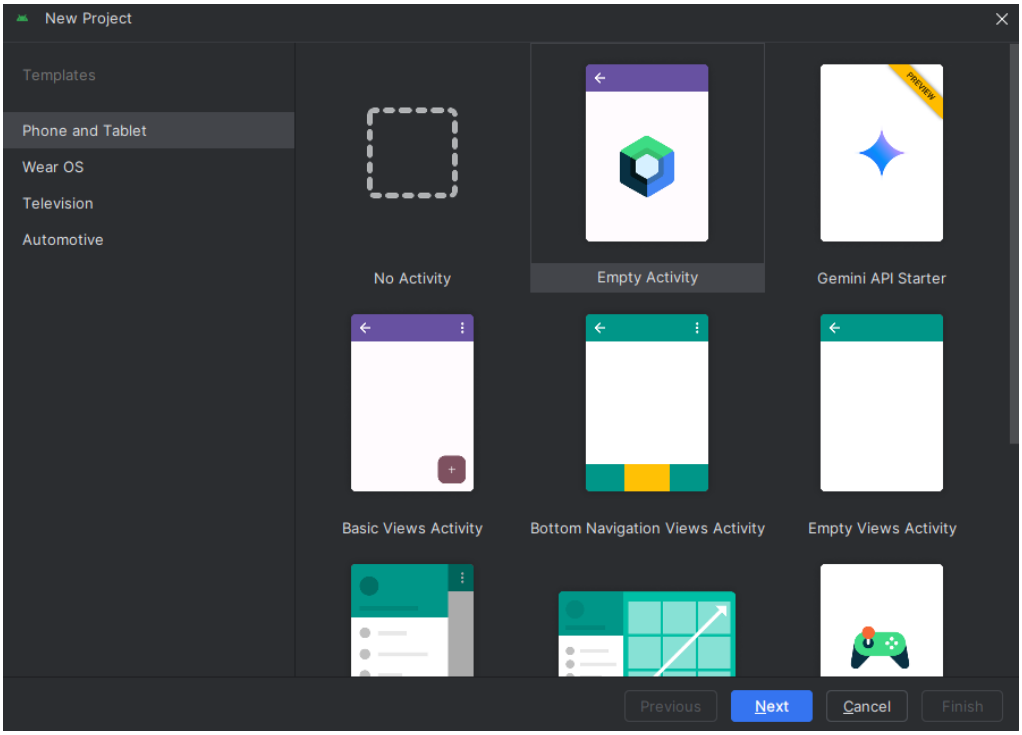
1. Stvaranje prve Andorid aplikacije

Tijekom laboratorijskih vježbi koristit ćemo Android Studio kao glavno razvojno okruženje za izradu Android aplikacija. Sve vježbe su pripremljene u verziji **Android Studio Merkat**, stoga je preporučljivo instalirati ovu verziju kako biste izbjegli moguće razlike u sučelju ili funkcionalnostima. Nakon pokretanja Android Studija, otvoriti će se početni zaslon prikazan na slici 1. Ovaj zaslon služi kao polazna točka za stvaranje novih projekata ili otvaranje postojećih. Kako bismo započeli rad na našoj prvoj aplikaciji, kliknite na gumb "New Project".



Slika 1. *Početni zaslon Android studio merkat*

Klikom na "New Project" otvara se prozor za odabir vrste projekta, kao što je prikazano na slici 2. Ovdje ćemo odabrati **"Empty Activity"**. Ova opcija predstavlja najjednostavniji predložak za Android aplikaciju – praznu aktivnost koja služi kao osnovni zaslon bez unaprijed definiranog sadržaja. "Empty Activity" idealna je početna točka za učenje jer nam omogućava da sami kreiramo korisničko sučelje od nule, koristeći alate poput Jetpack Composea. Nakon što odaberete "Empty Activity", kliknite na gumb **"Next"**.

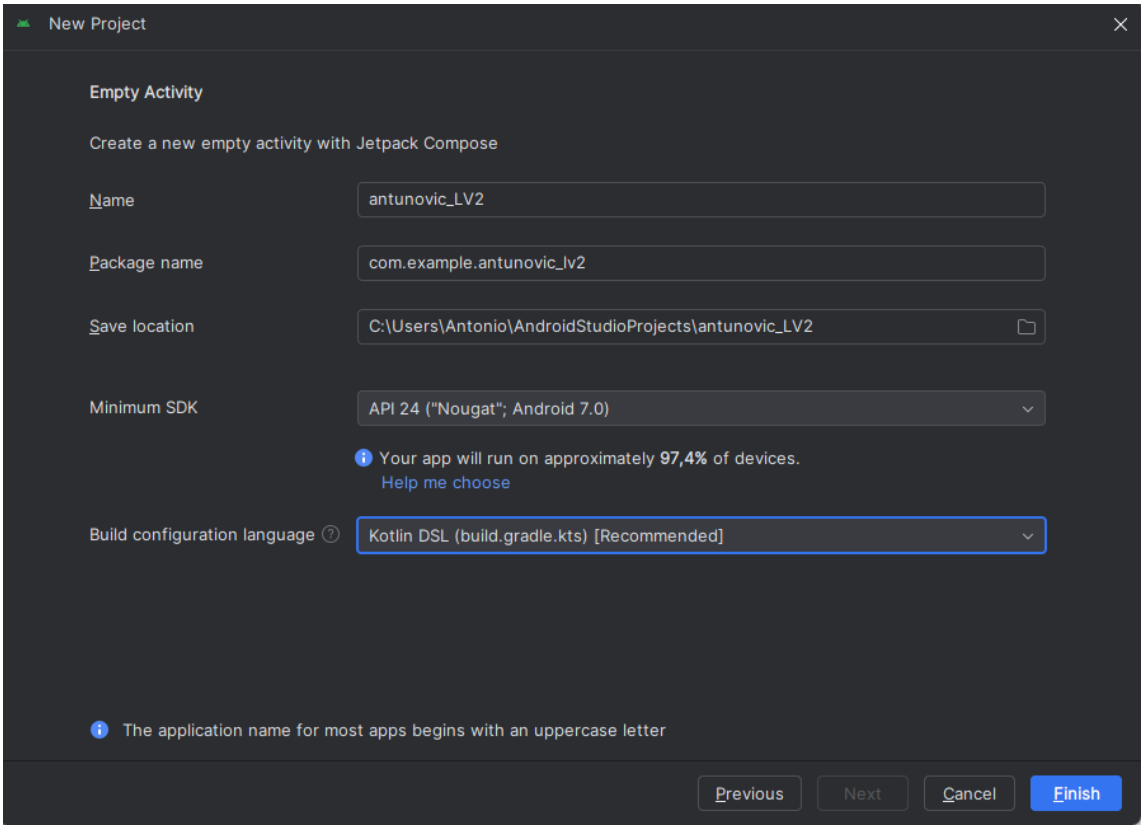


Slika 2. *Odabir vrste projekta*

Nakon klika na "Next", otvorit će se prozor za konfiguraciju projekta, prikazan na slici 3. Ovdje definiramo osnovne postavke aplikacije:

- **Name**: Unesite naziv projekta, primjerice "prezime_LV2". Ovaj naziv ne mora biti jedinstven i služi samo za vašu osobnu identifikaciju projekta u Android Studiju.
- **Package Name**: Ovo je jedinstveni identifikator aplikacije. Preporučuje se korištenje obrnutog oblika domene (npr. com.example.prezime_LV2).
- **Save Location**: Odaberite direktorij na računalu gdje želite spremiti projekt.
- **Minimum SDK**: Postavite na **API 24 (Android 7.0, Nougat)**. Ovo osigurava da aplikacija radi na većini modernih uređaja, uz dobar balans između kompatibilnosti i dostupnih funkcionalnosti.

Kada završite s postavljanjem, kliknite na **"Finish"**. Android Studio će tada stvoriti projekt i sinkronizirati potrebne datoteke (Gradle build sustav), što može potrajati nekoliko minuta.



Slika 3. *Konfiguracija projekta*

2. Jetpack compose

Jetpack Compose moderni je UI toolkit za razvoj aplikacija na Android platformi. Radi se o deklarativnom pristupu izgradnji korisničkog sučelja (UI), gdje se koristi Kotlin kod za opisivanje kako bi UI trebao izgledati i ponašati se u različitim situacijama. Umjesto pisanja zasebnih datoteka za strukturu i logiku, Compose omogućava definiranje cijelog korisničkog sučelja unutar jednog dijela koda, čime se postiže intuitivan i reaktivni programski model.

Osnovni građevni blokovi u Jetpack Composeu su **Composable funkcije**. Svaka Composable funkcija opisuje dio korisničkog sučelja (npr. tekst, gumb, sliku) i može se sastojati od drugih

Composable funkcija, omogućujući stvaranje složenih hijerarhija korisničkog sučelja na jednostavan način.

2.1. Composable funkcije

Composable funkcije su osnovni građevni blokovi korisničkog sučelja u Jetpack Composeu. Composable Funkcije:

- Opisuju dio korisničkog sučelja koji se prikazije na zaslonu (npr. Tekst, gumb, sliku)
- Ne vraćaju nikakvu vrijednost
- Primaju parametre i generiraju prikaz na temelju tog unosa

Programski kod 1. prikazuje deklaraciju composable funkcije. Svaka Composable funkcija mora biti označena anotacijom @Composable, koja označava da funkcija može stvarati UI elemente. Također, često se koristi anotacija @Preview za prikazivanje UI-a u Android Studiju bez pokretanja aplikacije.

```
@Preview(showBackground = true)
@Composable
fun Greeting(name: String) {
    Text(text = "Pozdrav, $name!")
}
```

Programski kod 1. *Primjer composable funkcije*

Composable funkcija mora slijediti konvenciju imenovanja **PascalCase**. Ova konvencija zahtijeva da prvo slovo svake riječi u složenom imenu bude veliko (npr. Greeting, UserProfile).

Razlike u stilovima imenovanja

- **PascalCase**: Sva prva slova riječi su velika (npr. DoneButton).
- **CamelCase**: Prva riječ počinje malim slovom, a ostale riječi velikim (npr. doneButton – ne koristi se za Composable funkcije).

Pravila za imenovanje

- **Funkcija mora biti imenica**: DoneButton()
- **Ne smije biti glagol ili glagolska fraza**: DrawTextField()
- **Ne smije biti prijedlog s imenicom**: TextFieldWithLink()
- **Ne smije biti pridjev**: tamno()
- **Ne smije biti prilog**: izvana()
- **Dozvoljeno je dodavanje opisnih pridjeva kao prefiks**: RoundIcon()

2.2. Okno dizajna u Android studiju

Android Studio omogućuje pregled Composable funkcija izravno unutar IDE-a, bez potrebe za pokretanjem aplikacije na emulatoru ili fizičkom Android uređaju. Ova značajka štedi vrijeme tijekom razvoja jer omogućuje brzo testiranje i prilagođavanje korisničkog sučelja (UI). Pregled je dostupan u oknu dizajna (Design pane), koje se nalazi u uređivaču koda, a prikazano je na slici 3.

Slika 3. *Okno dizajna*

2.3. Hijerarhija korisničkog sučelja u Jetpack Composeu

U Jetpack Composeu korisničko sučelje (UI) organizirano je hijerarhijski putem layout komponenata koje sadrže druge UI elemente. Pojmovi **roditelj** (parent) i **dijete** (child) opisuju ovu strukturu:

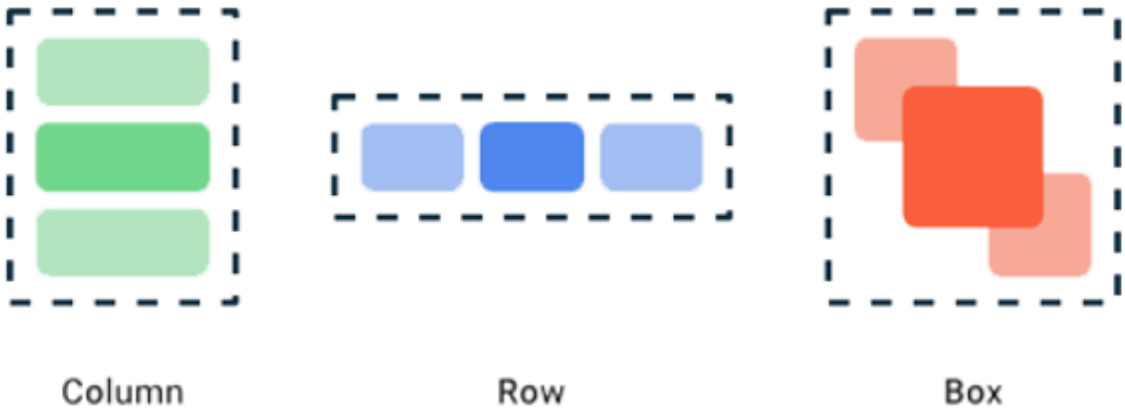
- Roditeljski element (npr. Column, Row ili Box) sadrži jedan ili više podređenih elemenata (npr. Text ili Button).

- Podređeni elementi mogu sami biti roditelji drugih elemenata, stvarajući složenu hijerarhiju.

Ključne layout komponente za raspored elemenata uključuju:

- **Column**: Raspoređuje elemente vertikalno
- **Row**: Raspoređuje elemente horizontalno
- **Box**: Omogućuje preklapanje elemenata

Na slici 4 prikazani su primjeri rasporeda elemenata koristeći Column, Row i Box, dok programski kod 2. prikazuje primjer korištenja Column komponente.



Slika 4. *Elementi smješteni u reda, stupac i box*

```
Column {
    Text(
        text = "Hello $name!", modifier = modifier
    )
    val bmi = 15
    Text(
        text = "Tvoj BMI je: $bmi", modifier = modifier
    )
}
```

Programski kod 2. *Primjer Column komponente*

2.4. Promjena veličine i poravnanje fonta

Dodali ste tekst u svoje korisničko sučelje, ali još ne izgleda kao konačna aplikacija. Potrebno je promijeniti veličinu, boju teksta i druge atribute koji utječu na izgled elementa teksta. Također možete eksperimentirati s različitim veličinama i bojama fonta.

U Android aplikacijama koriste se dvije glavne mjerne jedinice za veličinu elemenata:

- **DP (Density-Independent Pixels)**: Jedinica neovisna o gustoći zaslona, korištena za dimenzije elemenata poput razmaka ili širine.

- **SP (Scalable Pixels)**: Skalabilni pikseli specifični za veličinu fonta. Po zadanim postavkama, SP je isti kao DP, ali se prilagođava korisnikovim postavkama veličine teksta na telefonu (npr. povećanje teksta za osobe s lošim vidom)

Za prilagođavanje veličine teksta u Jetpack Composeu preporučuje se korištenje SP jer omogućava fleksibilnost i pristupačnost. Da biste koristili SP, potrebno je uvesti paket androidx.compose.ui.unit.sp

Veličina fonta definirana je parametrom fontSize unutar Text komponente. Poravnanje teksta može se postaviti pomoću modifikatora textAlign unutar Text komponente. Dostupne opcije uključuju:

- **TextAlign.Left** (lijevo poravnanje)
- **TextAlign.Center** (centrirano poravnanje)
- **TextAlign.Right** (desno poravnanje)
- **TextAlign.Justify** (obostrano poravnanje)

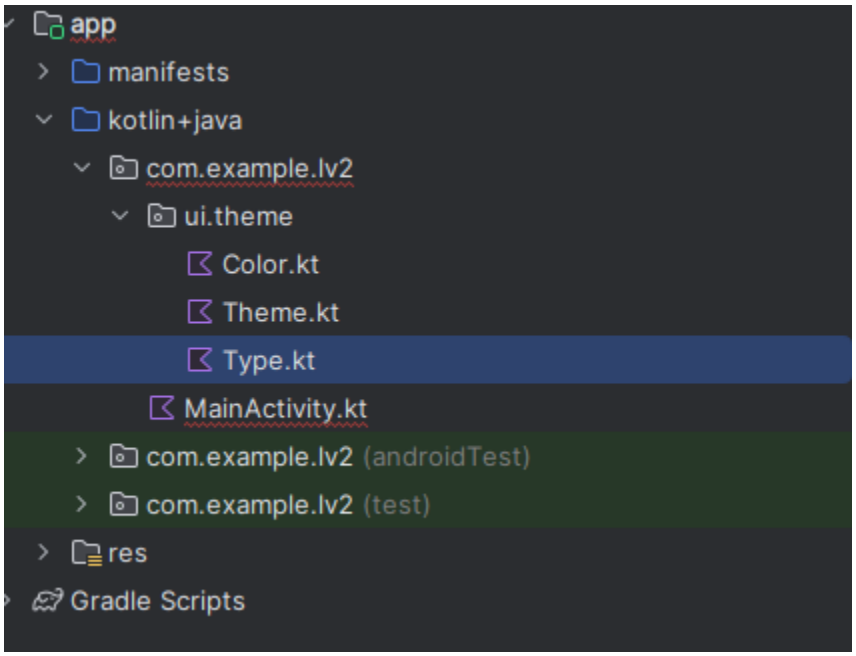
Programski kod 3 prikazuje promjenu veličine i poravnanje teksta

```
Text(
    text = "Hello",
    fontSize = 20.sp,
    textAlign = TextAlign.Center
)
```

Programski kod 3. *Promjena veličine teksta i poravnanje*

2.5. Korištenje novog fonta

Do sada smo radili uglavnom s datotekom MainActivity.kt, ali Android projekat sadrži mnoge druge datoteke koje podržavaju funkcionalnost i dizajn aplikacije. Organizaciju datoteka možete vidjeti na slici 5. Posebno je važan folder ui.theme, koji sadrži datoteke poput Color.kt, theme.kt, Type.kt. fontove mijenjamo u datoteci Type.kt.



Slika 5. *Organizacija datoteka*

Ako želite koristiti vlastiti font (npr. preuzet s interneta), možete ga dodati u projekt na sljedeći način:

1. Preuzmite datoteku fonta u formatu .ttf ili .otf (npr. sa stranica poput Google Fonts).

2. U folderu res (resources) vašeg projekta stvorite novi podfolder pod imenom **font** (npr. app/src/main/res/font).
3. Kopirajte preuzetu datoteku fonta (npr. custom_font.ttf) u taj folder.
4. U datoteci Type.kt, definirajte vlastiti font koristeći FontFamily i referencirajte datoteku fonta.

Programski kod 4 prikazuje kako dodati vlastiti font u Type.kt

```
val CustomFont = FontFamily(
    Font(R.font.custom_font)
)
// Set of Material typography styles to start with
val Typography = Typography(
    bodyLarge = TextStyle(
        fontFamily = CustomFont,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp,
        lineHeight = 24.sp,
        letterSpacing = 0.5.sp
    )
)
```

Programski kod 4. *Dodavanje custom fonta*

2.6. Modifikatori

Modifikatori u Jetpack Compose omogućuju preciznu kontrolu nad izgledom i ponašanjem Composable elemenata. Pomoću njih možemo definirati dimenzije, poravnanje, margine, pozadinske boje, obrube i mnoge druge stilizacijske i funkcionalne karakteristike elemenata. Programski kod 5 prikazuje primjer korištenja modifikatora unutar *Text* composable funkcije. U ovom primjeru tekst će ispunit cijeli dostupni prostor zahvaljujući, imati razmak od 16 dp od rubova, a pozadina je postavljena na svijetlosivu boju. Modifikatori se primjenjuju lančano, što omogućuje jednostavnu i fleksibilnu prilagodbu.

```
@Composable
fun ModifierExample() {
    Text(
        text = "Primjer modifikatora",
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
            .background(Color.LightGray)
    )
}
```

Programski kod 5. *Primjer korištenja modifikatora*

3. Rad na vježbi

U ovoj vježbi potrebno je izraditi prvi zaslon Android aplikacije, prikazan na slici 6. Zaslon će uključivati tri *Text* elementa organizirana unutar *Column* komponente za vertikalni raspored i pozadinsku sliku, te je cijeli zaslon potrebno prikazati.

Slika 6. *Prvi zaslon aplikacije*

Programski kod 6. prikazuje *UserPreviw* composable funkciju koja služi za prikaz sadržaja početnog zaslona.

```
@Composable
fun UserPreview(name: String,visina: Float,tezina: Float,
       modifier : Modifier = Modifier)

{
    val bmi = tezina / (visina * visina)
    val formattedBmi = String.format("%.2f", bmi)

    Column(
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally,
        modifier = modifier
    ) {
        Text(
            text = "Pozdrav $name!",
            fontSize = 20.sp,
            lineHeight = 56.sp,
            modifier = Modifier.padding(top = 8.dp, start = 10.dp)
        )
        Text(
            text = "Tvoj BMI je:",
            fontSize = 55.sp,
            lineHeight = 61.sp
        )
        Text(
            text = formattedBmi,
            fontSize = 70.sp,
            lineHeight = 72.sp,
            fontWeight = FontWeight.Bold
        )
    }
}
```

Programski kod 6. *UserPreview funkcija*

Sadržaj je organiziran vertikalno pomoću *Column* komponente. Unutar *Column* definirano je da će elementi biti centrirani horizontalno postavljanjem *horizontalAlignment* na *Alignment.CenterHorizontally*, dok parametrom *verticalArrangement* je postavljen na *Arrangement.Center* kako bi cijeli sadržaj bio vertikalno centriran na zaslonu. Modifikatorom se

omogućuje da *Column* element zauzme sav dostupan prostor. Unutar *Columna* smještena su tri *Text* elementa kojima se ispisuju korisnički podatci.

Programski kod 7 prikazuje *BackgroundImage* funkciju kojom se dodaje pozadinska slika na zaslon.

```
@Composable
fun BackgroundImage() {
    Box{
        Image(
            painter = painterResource(id = R.drawable.androidparty),
            contentDescription = null,
            contentScale = ContentScale.Crop,
            alpha = 0.1f
        )
        UserPreview(name = "Miljenko",
                    visina = 1.91f,
                    tezina = 80f,
                    modifier = Modifier.fillMaxSize())
    }
}
```

Programski kod 7. *BackgroundImage funkcija*

Funkcija *BackgroundImage* definira pozadinsku sliku za zaslon koristeći *Box* komponentu koja omogućuje postavljanje elemenata jedan na drugi, pri čemu se slika učitava iz resursa *R.drawable.androidparty* s prozirnošću od 0.1 kako bi bila diskretna, *contentScale* parametar je postavljen na *ContentScale.Crop* što osigurava da slika ispuni prostor bez izobličenja. Iznad slike poziva se funkcija *UserPreview* s fiksnim vrijednostima za ime, visinu i težinu, čime se prikazuje sadržaj zaslona preko pozadine.

Funkcija *HomeScreen* prikazana programskim kodom 8, označena anotacijom *@Preview(showBackground = true* ključna je za prikazivanje cijelog zaslona aplikacije unutar okna dizajna u Android Studiju, omogućavajući vizualno testiranje izgleda bez pokretanja aplikacije na emulatoru ili uređaju. Unutar funkcije, *Lv2Theme* primjenjuje definiranu temu aplikacije iz foldera *ui.theme*. Unutar Jetpack Compose teme se koriste za dosljedno stiliziranje aplikacije, omogućujući jednostavnu primjenu boja, tipografije i oblika za sve Composable

Komponente. *BackgroundImage* se poziva kako bi se prikazala pozadinska slika zajedno s tekstualnim sadržajem, stvarajući potpun prikaz zaslona prikazanog na slici 6.

```
@Preview(showBackground = true)
@Composable
fun HomeScreen() {
    Lv2Theme {
        BackgroundImage()
    }
}
```

Programski kod 8. *HomeScreen funkcija*

4. Zadatak za samostalni rad

Vaš zadatak je modificirati postojeću aplikaciju kako bi se izgled zaslona prilagodio prema novim zahtjevima koji su prikazani na slici 7. Cilj je prilagoditi korisničko sučelje tako da se u gornjem lijevom kutu prikazuje pozdrav "Pozdrav {Ime}!", dok će se u sredini zaslona nalaziti Column s dva Row elementa za prikaz težine i visine korisnika. Ispod toga prikazat će se "Tvoj BMI" i izračunati BMI, pri čemu će BMI biti obojen zeleno ako je manji od 20, a crveno ako je veći od 20. Uz to potrebno je preuzeti font po želji s https://fonts.google.com/ i primijeniti ga u aplikaciji.

Slika 7. *Izgled zaslona aplikacije*