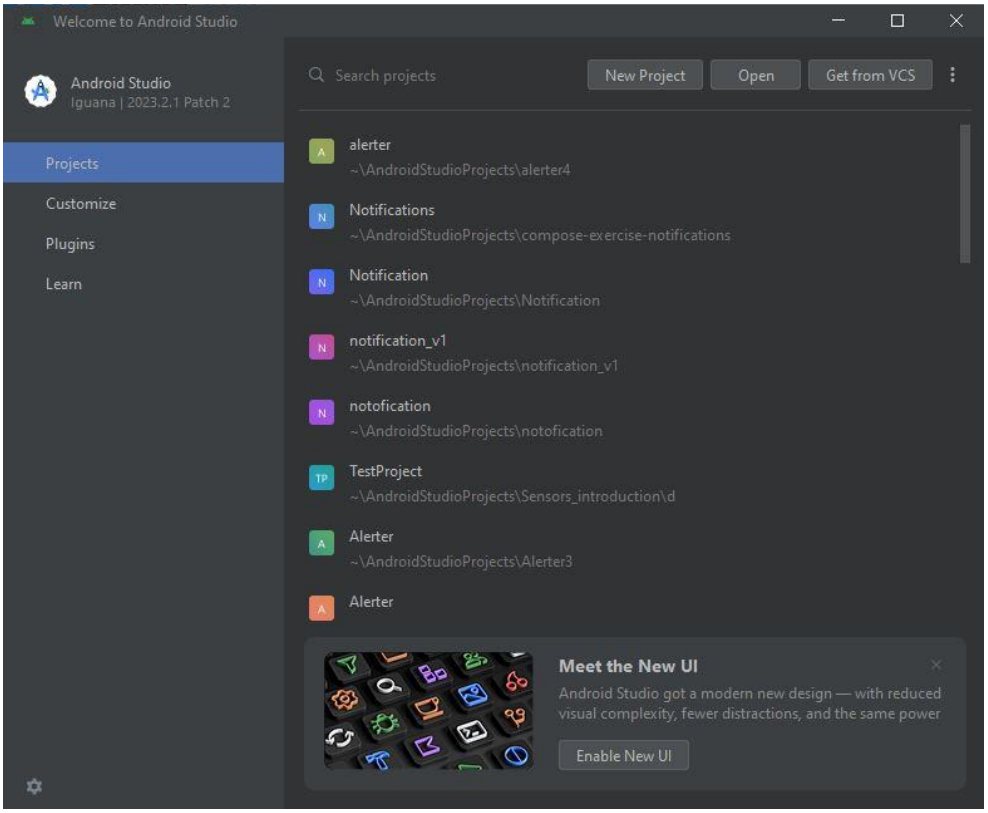
Android Studio i uvod u Jetpack Compose

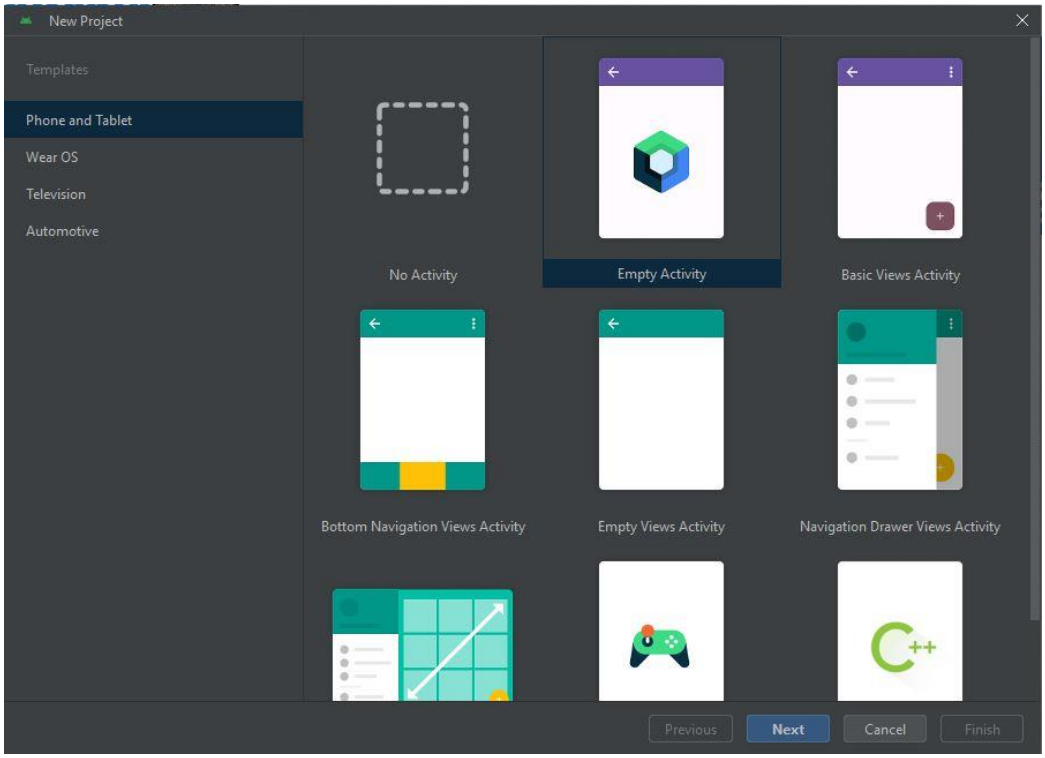
Stvaranje prve Android aplikacije

Do kraja laboratorijskih vježbi, koristiti ćemo Android studio. Vježbe su napravljene u verziji Iguana. Nakon pokretanja Android studija, dolazimo do početnog zaslona kao na slici 1.



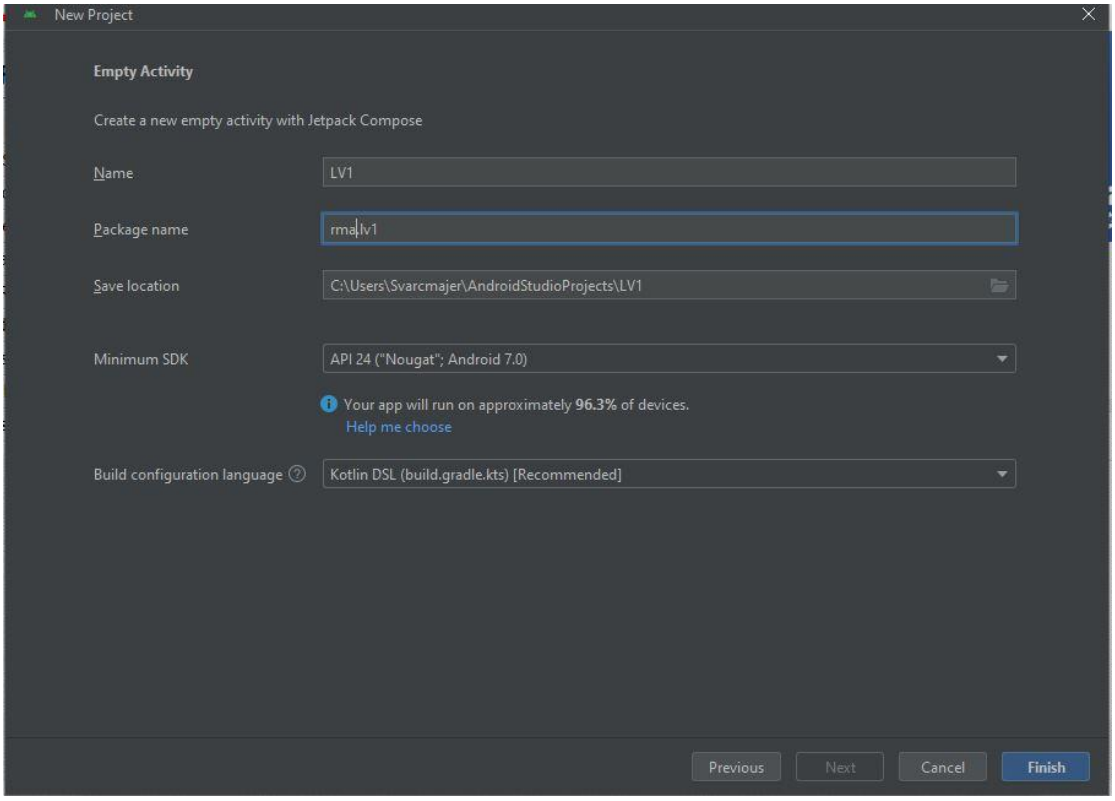
Slika 1. Početni zaslon Android Studio Iguana

Nakon klika na gumb New Project, otvara se prozor kao na slici 2. Na ovoj slici biramo Empty activity. Empty activity (prazna aktivnost) je najjednostavniji tip aktivnosti u Androidu. To je jednostavno prazan ekran bez ikakvog sadržaja. Koristi se kao početna točka za izradu novih aktivnosti u aplikaciji.



Slika 2. Odabir vrste projekta

Klikom na gumb Next, otvara se prozor projekta kao na slici 3. Ovdje određujemo ime projekta koje ne mora biti unikatno. Unikatan treba biti package name. Preporuka je koristiti obrnuti oblik domene. Za potrebe laboratorijskih vježbi, Minimum SDK će nam biti API 24, Android 7.0.



Slika 3. Određivanje postavki projekta.

Jetpack compose

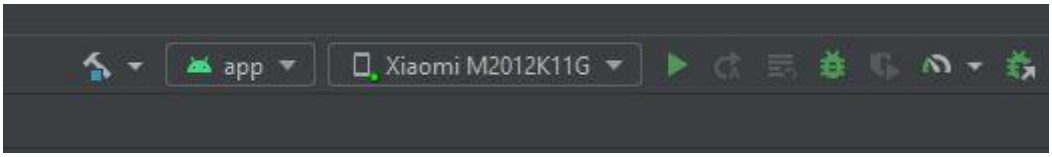
Jetpack Compose je moderni UI toolkit za razvoj aplikacija za Android platformu. To je biblioteka za izgradnju korisničkog sučelja (UI) koja koristi deklarativni pristup stvaranju korisničkog sučelja, gdje se koristi jednostavan, intuitivan i reaktivni programski model za opisivanje kako bi UI trebao izgledati i kako bi se ponašao u različitim situacijama. Composable funkcije u Jetpack Composeu su sastavni dijelovi korisničkog sučelja, a oni se mogu sastojati od drugih composable funkcija, stvarajući složene hijerarhije. Prednosti korištenja Jetpack Composea uključuju bolju produktivnost razvojnih programera, manje grešaka tijekom razvoja i brže izmjene UI-ja. Jetpack Compose se također integrira s drugim Jetpack bibliotekama kao što su Room za upravljanje bazama podataka i LiveData za povezivanje podataka sa korisničkim sučeljem. Jetpack Compose i XML su oba alati za izgradnju korisničkog sučelja u aplikacijama za Android platformu, ali postoje značajne razlike između njih. XML (Extensible Markup Language) je jezik za označavanje koji se koristi za definiranje strukture i izgleda korisničkog sučelja u Androidu. U XML-u, korisničko sučelje se definira pomoću tagova, atributa i vrijednosti, a stilovi se definiraju u posebnim XML datotekama. XML je statičan, što znači da su promjene na korisničkom sučelju složene i zahtijevaju mnogo koda za upravljanje. Jetpack Compose, s druge strane, koristi deklarativni pristup za stvaranje korisničkog sučelja, što znači da razvijatelji opisuju željeni izgled i ponašanje korisničkog sučelja koristeći kotlin kod. Compose se temelji na funkcijama koje proizvode komponente korisničkog sučelja, poznate kao Composable funkcije. Compose također omogućuje dinamičko ažuriranje korisničkog sučelja, što znači da se promjene na korisničkom sučelju mogu napraviti jednostavno i brzo. Glavna prednost Jetpack Composea u odnosu na XML je njegova jednostavnost i fleksibilnost. Compose je intuitivan i ima manje koda, što olakšava razvoj korisničkog sučelja i smanjuje mogućnost grešaka. Također omogućuje brzo ažuriranje korisničkog sučelja i pruža bolju interakciju s drugim Jetpack bibliotekama.

Jetpack compose kroz prvu aplikaciju

U ovom zadatku ćemo napisati jednostavnu aplikaciju koja pokazuje korisnikovo ime i BMI.

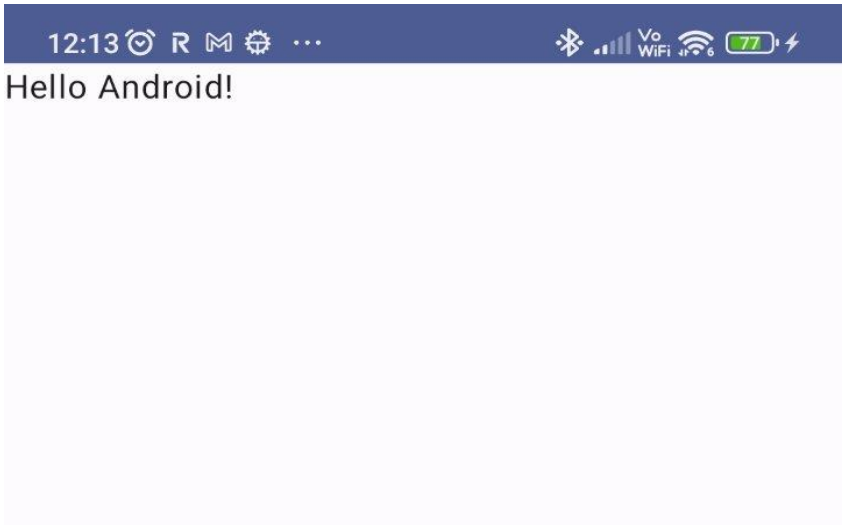
1. Pokrenuti novi Android projekt
2. Nazvati ga Fitness app
3. Nakon što Android studio napravi projekt, možemo testirati početnu aplikaciju.

Android studio nudi pregled aplikacije na emulatoru ili na Android uređaju (mobitel ili teblet). U gornjem desnom uglu Android studija možete odabrati uređaj te pokrenuti aplikaciju kako je prikazano na slici 4.



Slika 4. Odabir uređaja za prikaz aplikacije.

Aplikacija bi trebala izgledati kao na slici 5.



Slika 5. Početni izgled aplikacije

Kada ste izradili ovu aplikaciju Fitness app s predloškom Empty Activity, Android Studio je postavio resurse za osnovnu Android aplikaciju, uključujući Hello Android! poruku na ekranu. U ovoj laboratorijskoj vježbi naučit ćete kako ta poruka stigne tamo, kako promijeniti njezin tekst u svoju aplikaciju i kako dodati i oblikovati dodatne poruke i elemente.

Jetpack compose nam koristi za definiranje korisničkog sučelja (UI). Elementi sučelja su: tekst, slike, gumbi i sl. Pomoću Jetpack composea definiramo i način na koji su postavljeni na ekran te način na koji korisnici mogu raditi interakciju s njima.

Composable funkcije

Ove funkcije su osnovni građevni blok korisničkog sučelja u Composeu. Composable funkcija:

- Opisuje dio vašeg korisničkog sučelja.
- Ne vraća ništa.
- Uzima neki unos i generira ono što je prikazano na zaslonu.

Anotacije

Anotacije su sredstva za dodavanje dodatnih informacija kodu. Ove informacije pomažu alatima kao što je compiler Jetpack Composea i drugim programerima da razumiju kod aplikacije.

Anotacija se primjenjuje tako da se njenom nazivu (bilješka) doda znak @ na početku deklaracije koju označavate. Različiti elementi koda, uključujući svojstva, funkcije i klase, mogu biti anotirani.

Sljedeći dijagram primjer je anotirane funkcije:

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
}
```

Slika 6. Primjer anotirane funkcije

Anotacija s parametrima

Anotacije mogu imati parametre. Parametri daju dodatne informacije alatima koji ih obrađuju. Slijedi nekoliko primjera anotacije @Preview s parametrima i bez njih.

Pokušajte anotaciji :

@Preview(showBackground= true)

Maknuti parametar showbackground. Napomena: Anotacija može imati i više parametara. Tada ih pišemo:

```
@Preview(
  showBackground= true,
  showSystemUi = true
)
```

Imenovanje Composable funkcija

Composable funkcija koja ne vraća ništa i nosi oznaku @Composable MORA se imenovati korištenjem velikih i malih slova (tzv.Pascal case). Ovaj način se odnosi na konvenciju imenovanja u kojoj je prvo slovo svake riječi u složenici veliko. Razlika između velikih i velikih slova Pascal i velikih i velikih slova je u tome što su sve riječi u Pascal velikim slovima. U tzv. Camel caseu, prva riječ počinje malim slovom.

Funkcija Compose:

- MORA biti imenica: DoneButton()
- NIJE glagol ili glagolska fraza: DrawTextField()
- NIJE prijedlog s imenicom: TextFieldWithLink()
- NIJE pridjev: svijetlo()
- NIJE prilog: izvana()
- Imenice MOGU imati prefiks opisnih pridjeva: RoundIcon()

Okno dizajna u Android Studiju

Android Studio vam omogućuje pregled funkcija koje možete sastaviti unutar IDE-a, umjesto instaliranja aplikacije na Android uređaj ili emulator. Možete pregledati kako vaša aplikacija izgleda u oknu Dizajn u Android Studiju. Okno dizajna je prikazano na slici 7.

Slika 7. Okno dizajna

Composable funkcija mora dati zadane vrijednosti za bilo koji parametar da bi je pregledala. Iz tog razloga, preporučuje se da ne koristite preview funkcije Greeting() izravno. Umjesto toga, trebate dodati drugu funkciju, u ovom slučaju funkciju UserPreview(), koja poziva funkciju Greeting() s odgovarajućim parametrom.

```
@Preview(showBackground = true)
@Composable
fun UserPreview() {
    LV1Theme {
        UserPreview("Miljenko")
    }
}
```

Zamijenite argument “Android” sa svojim imenom u Greeting() funkciji.

Dodavanje tekstualnog elementa

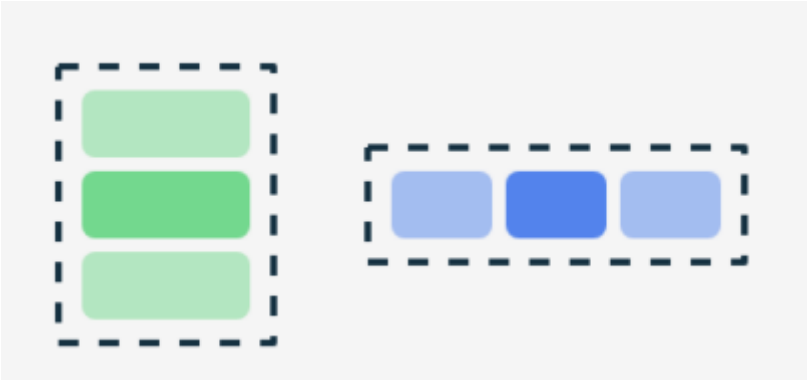
Uz tekstualni element Hello $name, dodati ćemo još jedan tekst

```
Text(
    text = "Hello $name!",
    modifier = modifier
)
Text(
    text = "Tvoj BMI je: $bmi",
    modifier = modifier
)
```

Sada moramo funkciju UserPreview nadopuniti argumentom bmi. Za sada neka tip podatka bude int. Primjetiti ćete da je sada u aplikaciji tekst prikazan jedan preko drugoga.

Raspoređivanje teksta u redove i stupce

Hijerarhija korisničkog sučelja temelji se na “containerima”, što znači da jedna komponenta može sadržavati jednu ili više komponenti, a ponekad se koriste pojmovi roditelj i dijete. Kontekst ovdje je da nadređeni UI elementi sadrže podređene UI elemente, koji zauzvrat mogu sadržavati podređene UI elemente. Najjednostavni ovakvi primjeri su stupac, redak i okvir koji mogu djelovati kao nadređeni elementi korisničkog sučelja. Na slici 8 možemo vidjeti razmještaj elemenata u stupac sa lijeve strane i redak sa desne strane. U ovom slučaju je parent element stupac/redak dok je child element tekst.



Slika 8. Elementi smješteni u redak i stupac

Kako bismo izbjegli da se tekst iz naše aplikacije preklapa, smjestiti ćemo ga u Column. Da bismo koristili column, moramo uvesti: androidx.compose.foundation.layout.Column u naš projekt. Ovaj paket je dio Jetpack Compose UI toolkita za izradu korisničkih sučelja. Potrebno je samo ovo kopirati pod import sekciju našeg projekta. Kada smo ovo napravili, enkapsuliramo oba tekstualna elementa u Column. Kod sada izgleda ovako:

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

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.

Skalabilni pikseli (SP) su mjerna jedinica za veličinu fonta. Elementi korisničkog sučelja u Android aplikacijama koriste dvije različite mjerne jedinice: piksele neovisne o gustoći (DP), koje kasnije koristite za izgled, i skalabilne piksele (SP). Prema zadanim postavkama, SP jedinica je iste veličine kao DP jedinica, ali se mijenja na temelju željene veličine teksta korisnika u postavkama telefona.

Koristiti ćemo skalabilne piksele a da bismo ih koristili, potrebno ih je uvesti u projekt. To ćemo napraviti kopiranjem androidx.compose.ui.unit.sp u import. Nakon ovoga, možemo koristi argument fontSize u funkciji tekst.

Postavimo veličinu teksta na 100.sp. Primjetiti ćete da se tekst sada preklapa. Ovaj puta, preklapa nam se samo tekst sa BMI-jem. Ovaj tekst ne ide preko teksta Hello jer zauzima samo svoj container. Kako se slova ne bi preklapala, možemo ili smanjiti veličinu ili postaviti tekst u više redova odnosno, definirati argument lineHeight. Postavite ovu vrijednost na 116.

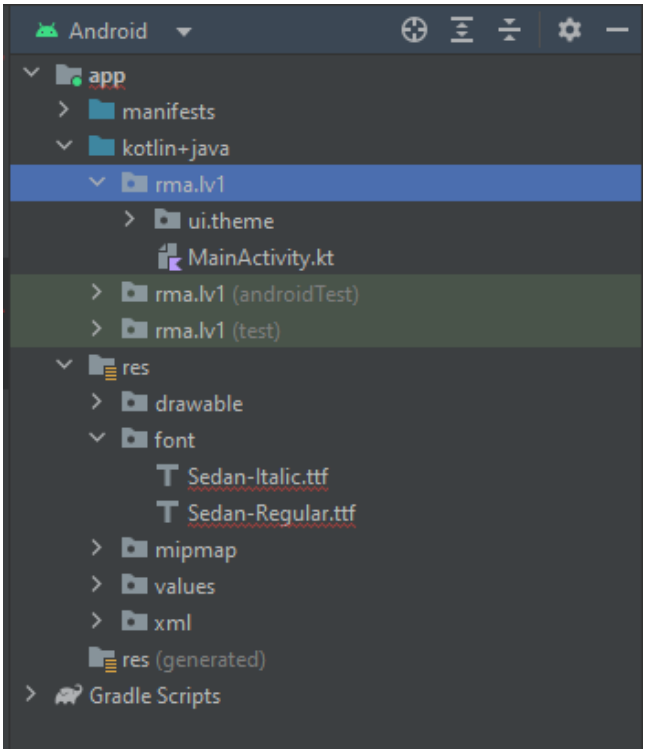
Ukoliko želimo odrediti poravnanje teksta, to radimo uz pomoć textAlign. Postavimo BMI na sredinu dodajući :

```
textAlign = TextAlign.Center
```

Greške koje su se sada pojavile u kodu, ispravljamo putem alt+enter kombinacije. Ovo je korisna informacija za druge stvari. U ovom slučaju, u projekt će se automatski importati potrebni paketi.

Korištenje novog fonta

Do sada smo radili samo na datoteci MainActivity.kt. Međutim, naš Android projekt ima mnogo drugih datoteka. Organizaciju možemo vidjeti na slici 10. Unutar foldera ui.theme postoje datoteke Color.kt, Theme.kt i Type.kt. Fontove mijenjamo u datoteci Type.kt.



Slika 10. Organizacija datoteka

Ukoliko želimo koristiti novi font, možemo ga skinuti i ubaciti u folder res. Najprije je potrebno napraviti novi folder font u njemu. Za sada ćemo u datoteci Type.kt promijeniti iz default u cursive. Pogledajte sada dizajn u Design okviru.

Više o radu sa fontovima možete pogledati na poveznici .

Zadatak:

Promijeniti pozadinu ekrana u zelenu. Tekst “Pozdrav” i tekst $name prikazati u istom redu. Ispod toga ispisati Tvoj BMI je: te ispod toga broj koji ste dobili unosom težine i visine. Koristiti font Roboto. Ukoliko BMI nije u granicama normale, obojati ga crvenom bojom. Ukoliko je normalan, obojati ga tamnom zelenom bojom. Pozdrav i $name postaviti u isti row, za elemente jedan ispod drugog koristiti column.

Konačan rezultat treba izgledati ovako:

Dodatni zadatak:

Proći codelab: https://developer.android.com/codelabs/basic-android-kotlin-compose-viewmodel-and-state#0