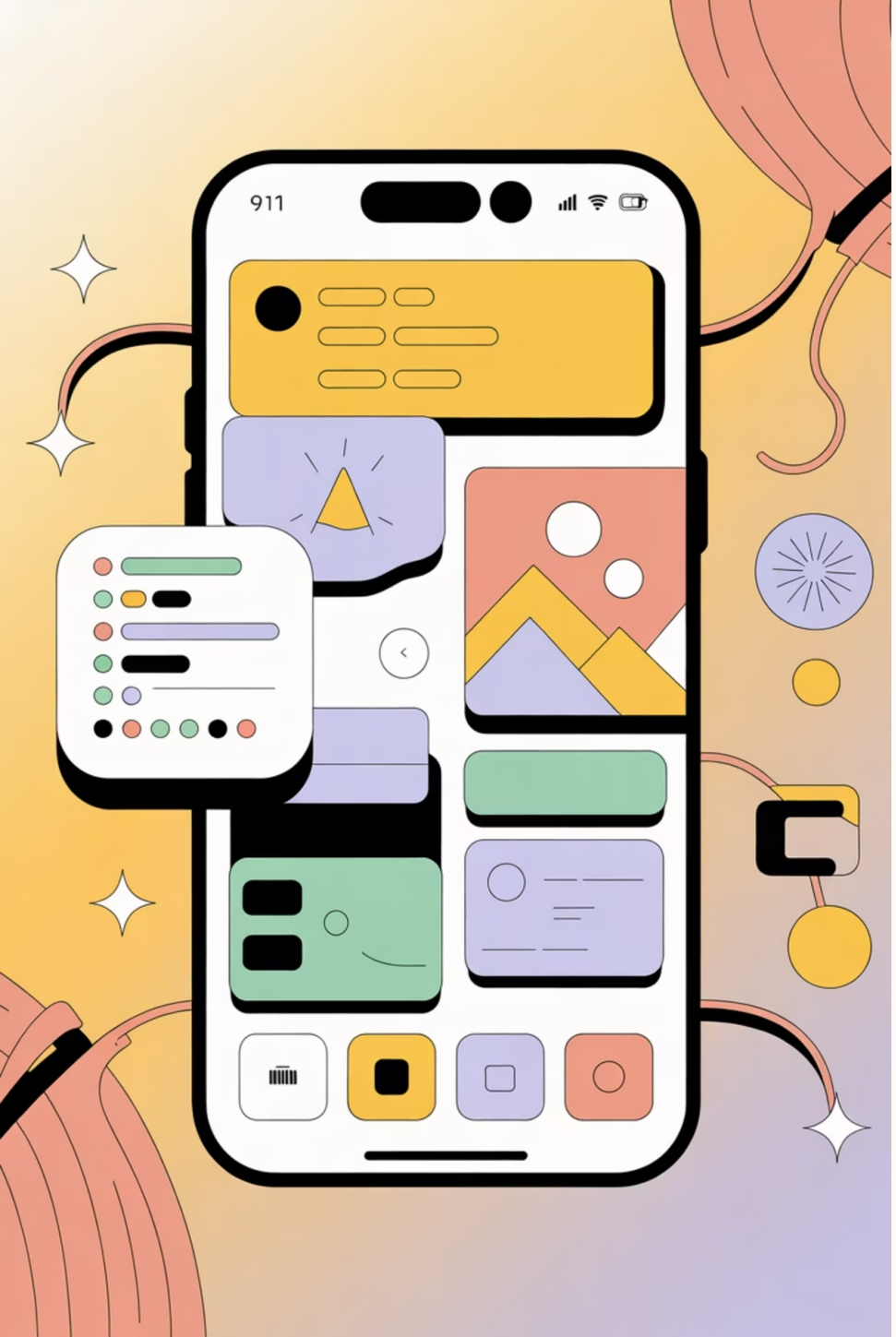
Predavanje br. 5

Razvoj mobilnih aplikacija i servisa

Uvod u Jetpack Compose

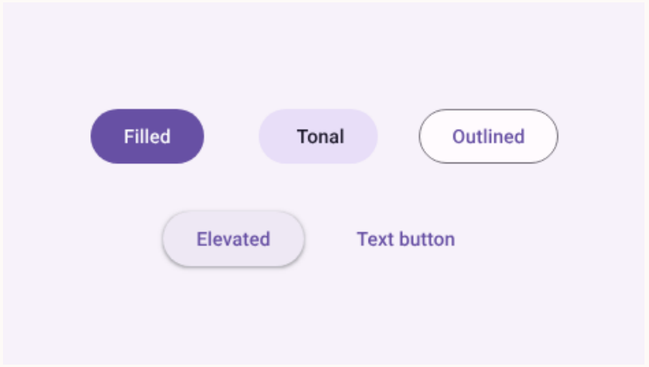


Šta je korisnički interfejs (UI)?

Korisnički interfejs (UI) aplikacije je sve što vidite na ekranu: tekst, slike, dugmad i mnoge druge vrste elemenata, te način na koji su raspoređeni. On određuje kako aplikacija prikazuje informacije korisniku i kako korisnik sa njom interaguje.

Svaki od ovih elemenata naziva se UI komponentom. Gotovo sve što vidite na ekranu vaše aplikacije je UI element. Mogu biti interaktivni, poput dugmadi za klikanje ili polja za unos teksta, ili dekorativni, poput slika.

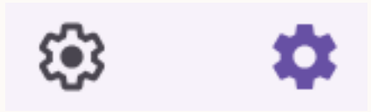
UI elementi



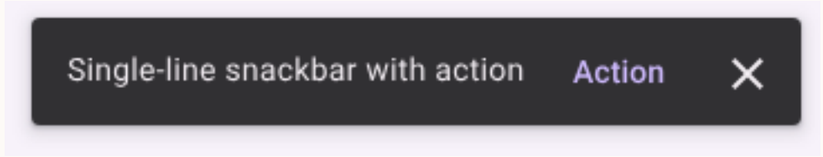
Button

Floating Action Button

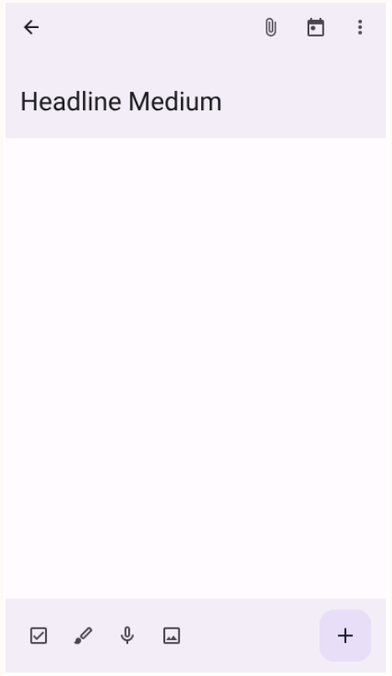
Progress indicator



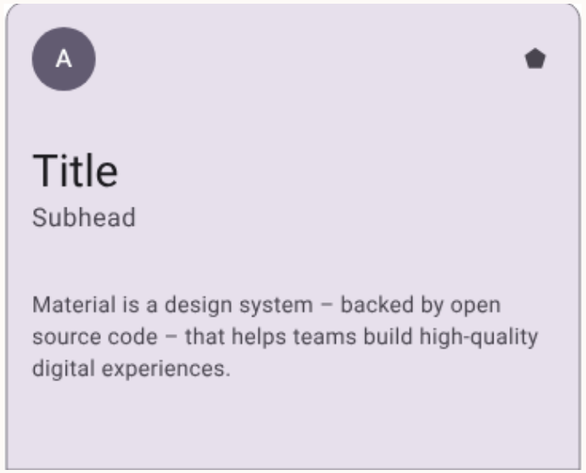
Icon Button



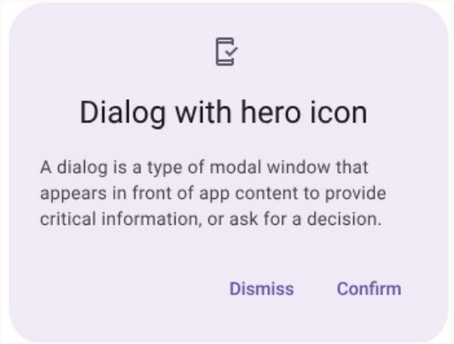
Snackbar



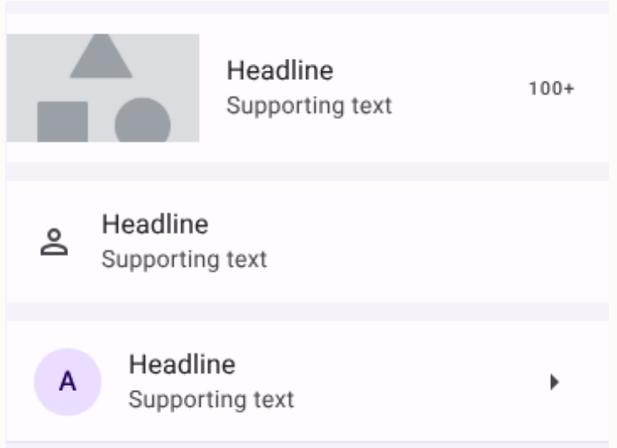
Scaffold



Card



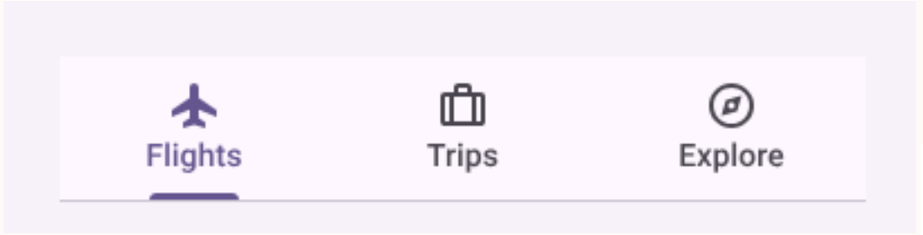
Dialog



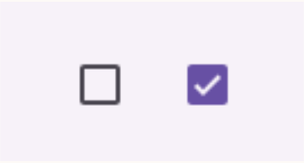
List

UI elementi

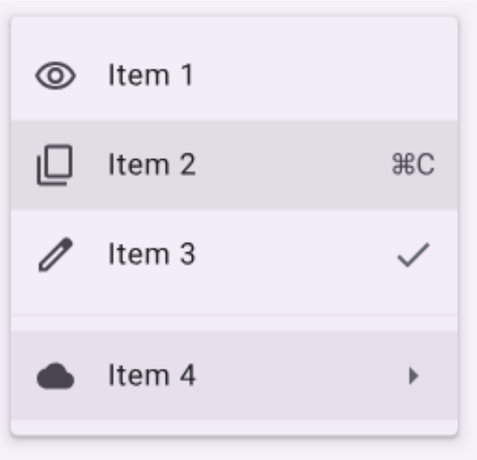
App bar



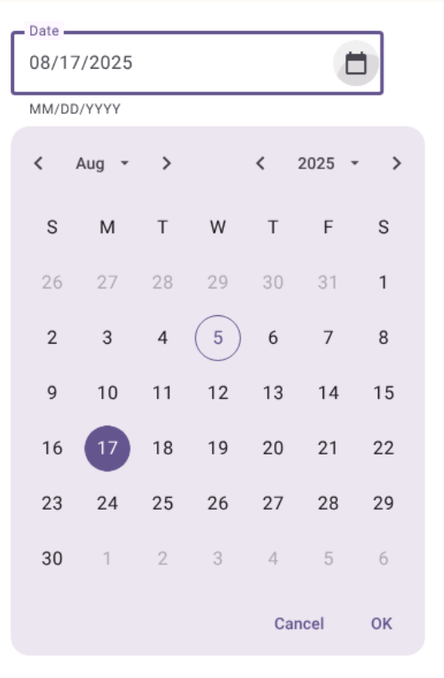
Tabs



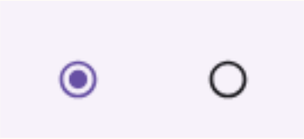
Checkbox



Menu

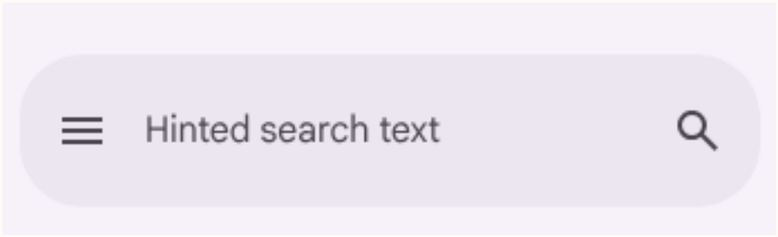


Date picker

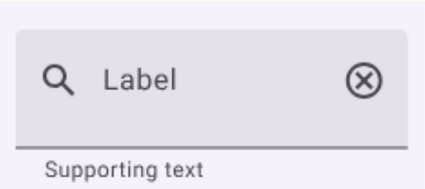


Radio Button

Navigation bar

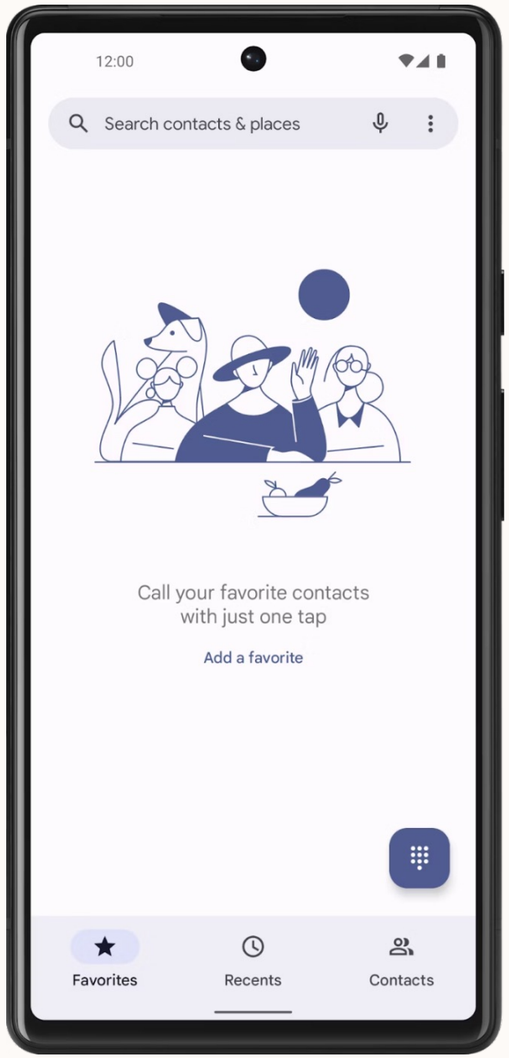


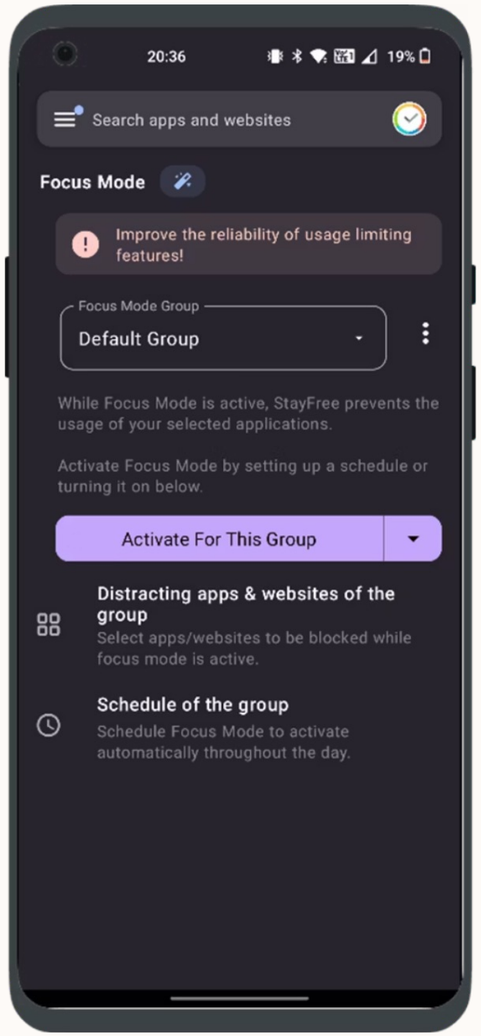
Search

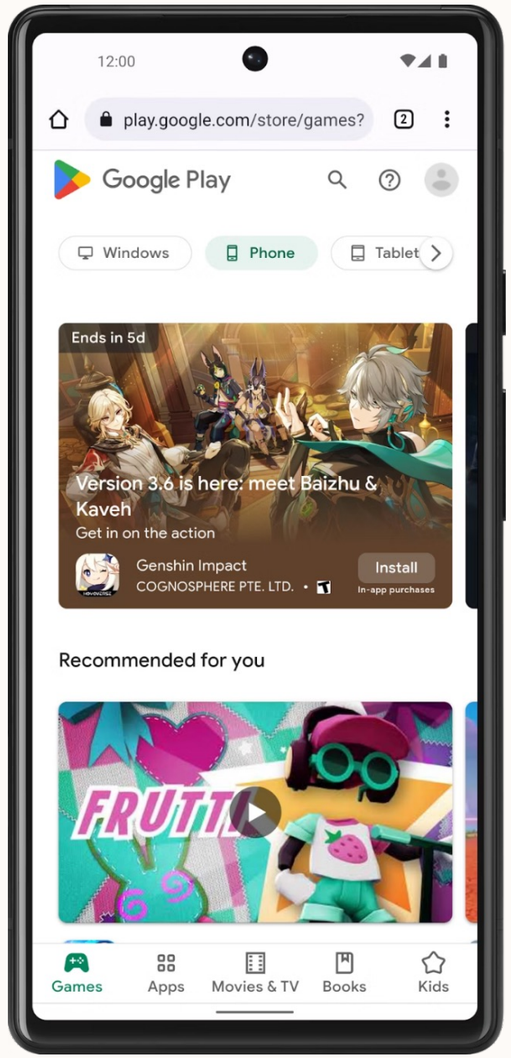


Text Input

Pronađi UI elemente







Šta je Jetpack Compose?

Framework izgrađen na Kotlinu

Compose je framework koji nadograđuje Kotlin i Android Studio, omogućavajući vam da gradite korisnički interfejs pomoću funkcija umjesto XML datoteka.

Važna distinkcija:

- **Kotlin** je programski jezik koji koristite za logiku
- **Compose** je framework koji vam omogućava da gradite **UI sa Kotlinom**

Ovo znači da možete koristiti sve moćne osobine Kotlina - lambda izraze, ekstenzije,
- direktno u vašem UI kodu!

@Composable

Ključna komponenta Compose frameworka je **@Composable** anotacija. Ova anotacija transformiše običnu Kotlin funkciju u UI komponentu.

Obična Kotlin funkcija

```
fun calculateScore(): Int {
    // logika proračuna
    return score
}
```

Vraća Integer vrijednost - čista logika bez UI-a

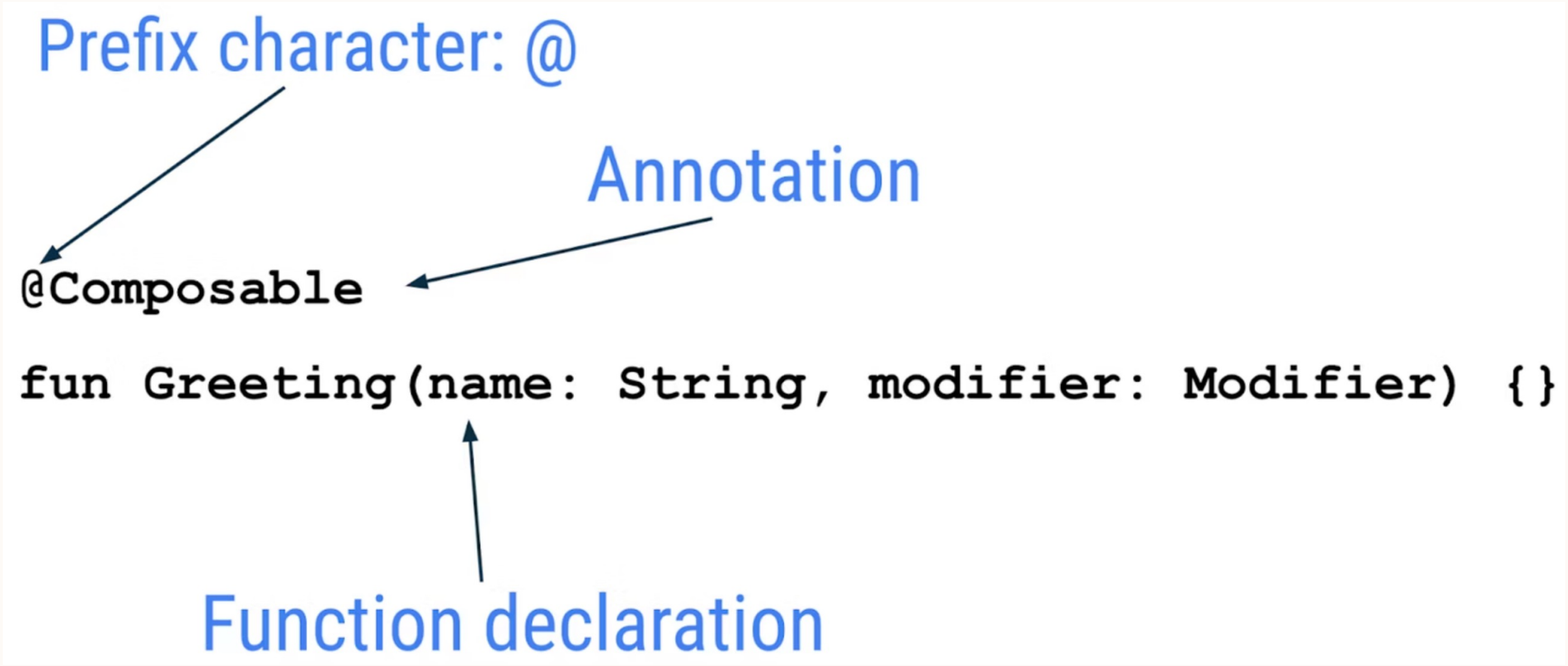
Composable funkcija

```
@Composable
fun EateryButton() {
        // UI komponenta
        Button(onClick = {}) {
                Text("Klikni me")
        }
}
```

Prikazuje UI komponentu - vizuelni element na ekranu

Composable funkcije ne vraćaju vrijednosti - umjesto toga, one *kreiraju* UI koji se prikazuje korisniku.

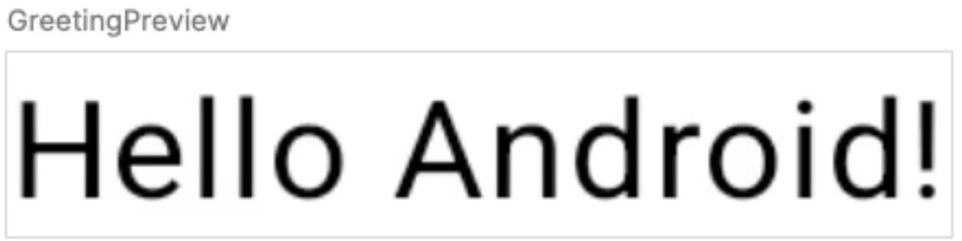
Anotacija



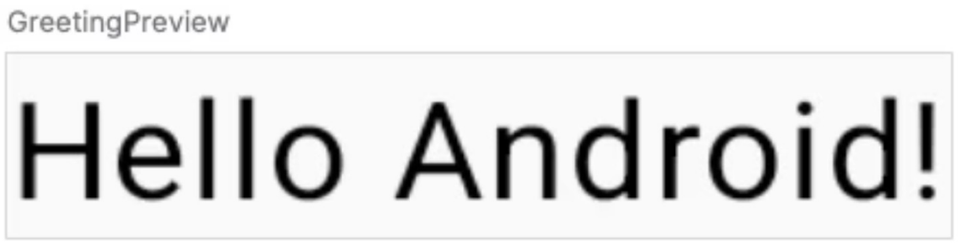

```
@Json
val imgSrcUrl: String

@Volatile
private var INSTANCE: AppDatabase? = null
```

Anotacija

```
44  @Preview
45  @Composable
46  fun GreetingPreview() {
47      HappyBirthdayTheme {
48          Greeting( name: "Android")
49      }
50  }
```




```
44  @Preview(showBackground = true)
45  @Composable
46  fun GreetingPreview() {
47      HappyBirthdayTheme {
48          Greeting( name: "Android")
49      }
50  }
```



Composable funkcije: Pravila imenovanja

Pascal Case

MORA biti imenica

Primjer: `DoneButton()`

👉 **Zašto je ovo važno?** Imenovanje funkcija kao imenica (što predstavljaju) olakšava razumijevanje njihove uloge u korisničkom sučelju i čini ih modularnijima.

MOGU biti prefiksovane opisnim pridjevima

Primjer: `RoundIcon()`

👉 **Zašto je ovo važno?** Pridjevi pružaju dodatni kontekst o izgledu ili karakteristikama komponente, a da pritom zadrže imenicu kao srž njene funkcionalnosti.

NIJE glagol ili glagolska fraza

Primjer: `DrawTextField()`

👉 **Zašto je ovo važno?** Kompozabilne funkcije opisuju UI, a ne "rade" nešto direktno. Glagoli impliciraju akciju, što može biti zbunjujuće u deklarativnom UI-u.

NIJE imenica s prijedlogom

Primjer: `TextFieldWithLink()`

👉 **Zašto je ovo važno?** Ovakvi nazivi mogu biti predugački i manje intuitivni. Bolje je kreirati zasebnu komponentu ili koristiti odgovarajuće parametre.

NIJE pridjev

Primjer: `Bright()`

👉 **Zašto je ovo važno?** Pridjevi opisuju, ali ne identificiraju jasno UI element. Naziv bi trebao odražavati ono što komponenta *jest*, a ne samo kakva je.

NIJE prilog

Primjer: `Outside()`

👉 **Zašto je ovo važno?** Prilozi opisuju kako se nešto radi ili gdje se nešto nalazi, što nije prikladno za imenovanje UI komponenti.

Primjer

```
// A
@Composable
fun FancyButton(text: String) {}

// B
@Composable
fun BackButtonHandler() {}

// C
@Composable
fun fancyButton(text: String) {}//

// D
@Composable
fun RenderFancyButton(text: String) {}

// E
@Composable
fun drawProfileImage(image: ImageAsset) {}
```

Izgradnja UI-a sa Composables

Kombinovanje Composable funkcija

Moć Compose-a dolazi iz mogućnosti **pozivanja Composable funkcija unutar drugih Composable funkcija**. Ovaj pristup vam omogućava da gradite složene interfejse iz manjih, ponovnih komponenti.

Compose već dolazi sa mnogo ugrađenih UI komponenti koje možete koristiti kao građevne blokove:

- Text, Image, Button za osnovne elemente
- Column, Row, Box za raspored
- Card, Surface za kontejnere
- I mnoge druge specijalizirane komponente

Primjer: Izgradnja EateryCard komponente

Vanjski Composable

```
@Composable
fun EateryCard(name: String, ...){
        // Glavni kontejner
        TitleRow(name)
        // Ostali elementi...
}
```

EateryCard je glavni Composable koji organizuje sve elemente kartice restorana.

Unutrašnji Composables

```
@Composable
fun TitleRow(name: String, ...){
        Row {
                Text(name)
                FavoriteButton()
        }
}
```

TitleRow je manji Composable koji prikazuje naziv restorana i dugme za omiljene.

Ovaj pristup omogućava čitljivost koda i lakše održavanje. Svaki Composable ima jasnu odgovornost i može se testirati nezavisno.

Ponovna upotreba Composable funkcija

1

Definirate jednom

Kreirajte EateryCard komponentu sa svim potrebnim elementima i stilizacijom.

2

Koristite bilo gdje

Nakon što je komponenta gotova, možete je pozivati na bilo kojem mjestu u vašoj aplikaciji.

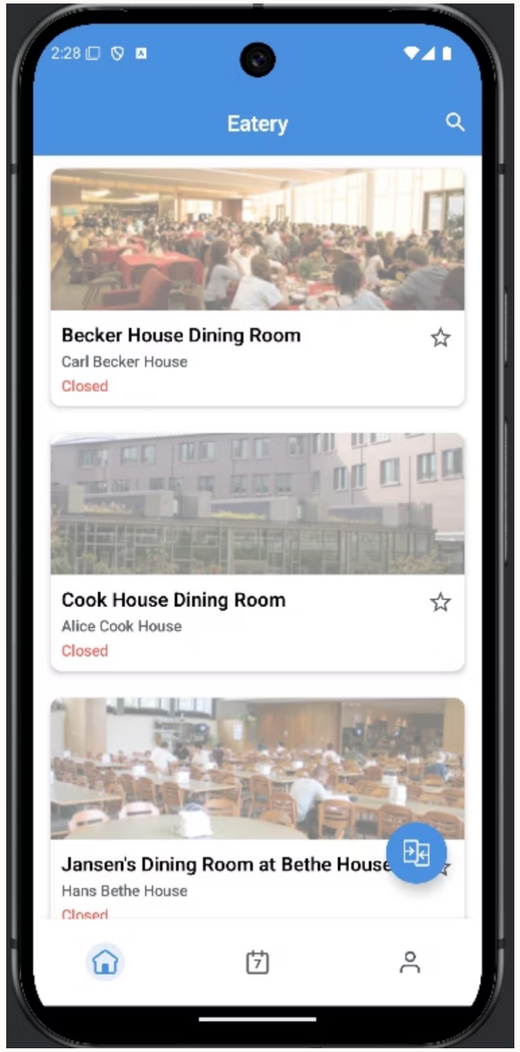
3

Modifikujte parametrima

Prosljeđivanjem različitih parametara, ista komponenta može prikazati različit sadržaj.

Ovo je jedna od najvećih prednosti Compose pristupa - **ponovna upotreba koda** postaje prirodna i jednostavna. Jednom kada izgradite EateryCard, možete je koristiti za prikazivanje bilo kojeg drugog restorana!

Primjer: HomeScreen sa više kartica



Glavni ekran

```
@Composable
fun HomeScreen(...){
        Column {
                EateryCard("Becker")
                EateryCard("Cook")
                EateryCard("Bethe")
        }
}
```

Composable komponenta

```
@Composable
fun EateryCard(name: String, ...){
        // Implementacija kartice    ...
}
```

Pozivamo **istu EateryCard funkciju** tri puta sa različitim imenima. Svaki poziv kreira novu instancu kartice sa specifičnim podacima.

Layout i View

Dva glavna tipa komponenti

Većina ugrađenih Compose komponenti može se klasifikovati kao:

Layout

UI komponenta koja organizuje ili drži druge UI komponente. Layout-i određuju kako se elementi raspoređuju na ekranu.

View

UI komponenta koja prikazuje sadržaj ili omogućava interakciju. Pogledi su vizuelni elementi sa kojima korisnici direktno interaguju.

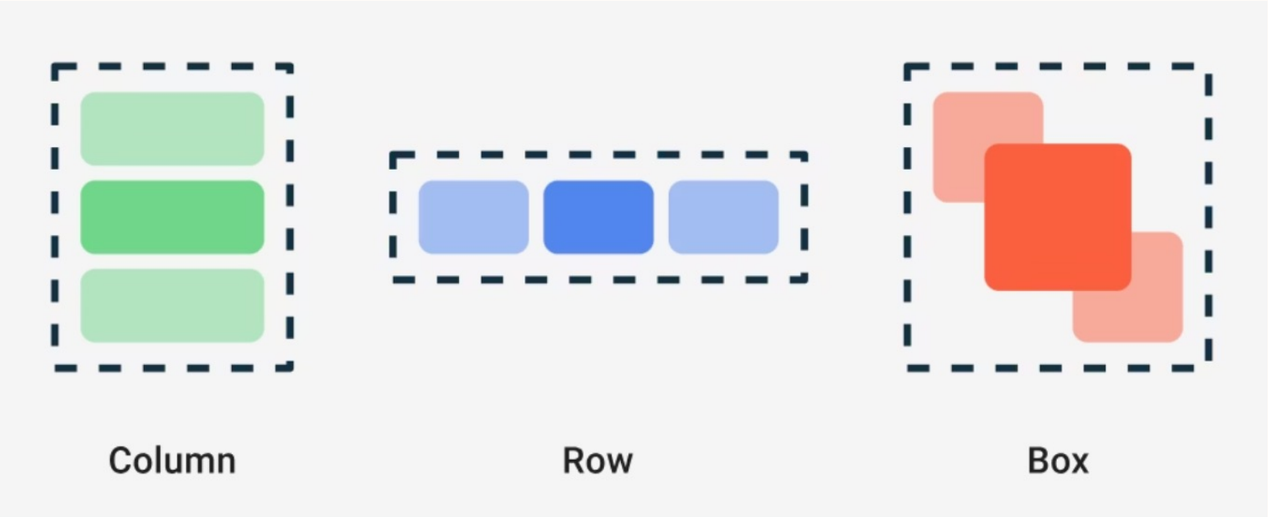
Layout-i mogu držati druge layout-e i/ili view-e, stvarajući fleksibilnu hijerarhijsku strukturu vašeg interfejsa!

Vrste layout-a

Osnovni layout

Tu spadaju:

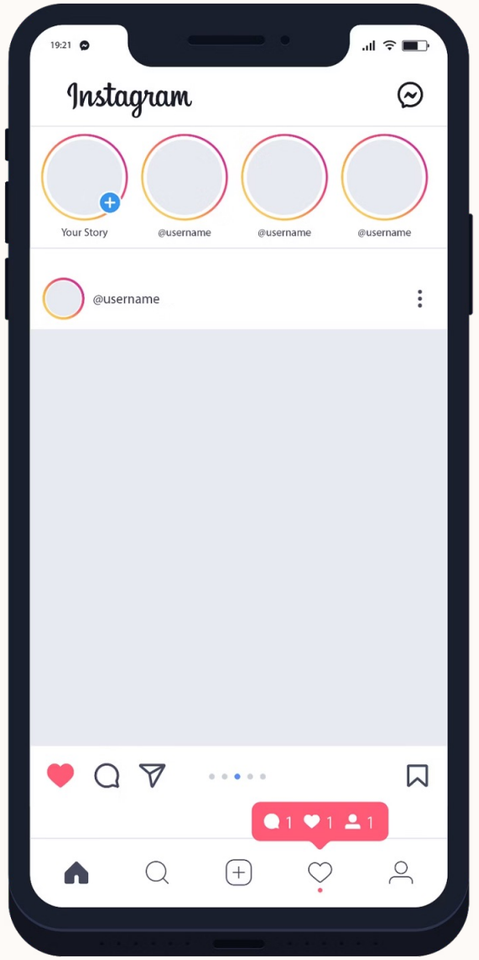
- Row
- Column
- Box



Napredni kontejneri

Za naprednije funkcionalnosti postoje:

- **LazyRow** i **LazyColumn** - za efikasno prikazivanje velikih listi



Column layout

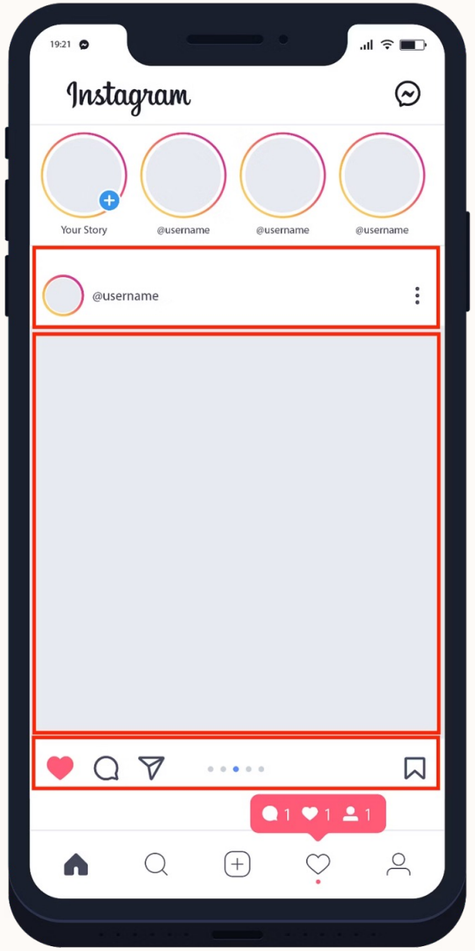
Vertikalni raspored

Column organizuje svoje child elemente vertikalno, jedan ispod drugog, kao što su komponente u Instagram postu.

```
@Composable
fun InstagramPost(...){
        Column(...) {
                Header(...)
                Image(...)
                BottomRow(...)
        }
}
```

Svaki element unutar Column-a automatski se smješta ispod prethodnog, kreirajući prirodan vertikalni tok sadržaja.

Primjer Instagram posta gdje Column organizuje Header, Image i BottomRow vertikalno.



Row layout

Horizontalni raspored

Row organizuje svoje child elemente horizontalno, jedan pored drugog. Ovo je idealno za elemente koje želite prikazati u istom redu.


```
@Composable
fun PostHeader(...){
      Row(...) {
            Row(...) {
                  ProfilePicture(...)
                  Text("@username")
            }
            OptionsButton(...)
      }
}
```

U ovom primjeru, unutrašnji Row drži profilnu sliku i korisničko ime zajedno, dok vanjski Row raspoređuje taj grupisani element i dugme za opcije horizontalno. **Row-ovi se mogu gnijezditi** za kreiranje složenijih rasporeda!

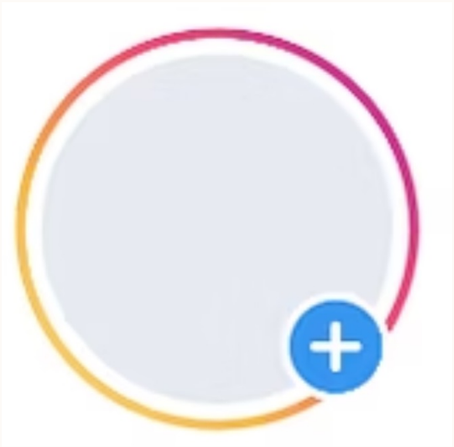
Box layout

Preklapanje elemenata

Box omogućava stavljanje elemenata jedan preko drugog. Ovo je korisno kada želite preklapanje, poput dodavanja ikone preko slike.

```
@Composable
fun InstaStoryBubble(...){
	Box(...) {
		Image(...)
		Icon(...)
	}
}
```

Elementi unutar Box-a se renderuju redoslijedom kojim su navedeni - prvi element je na dnu, posljednji na vrhu. U ovom primjeru, ikona se prikazuje preko slike, kreirajući efekat story bubble-a sa statusom.



Story bubble gdje je ikona postavljena preko slike koristeći Box kontejner.

Views - osnovni elementi

Najčešće korišteni pogledi

- **Text** - prikazuje tekst
- **Image** - prikazuje slike
- **Button** - dugme za interakciju

Složeniji pogledi

- **TextField** - za unos teksta
- **Slider** - za odabir vrijednosti

Primjer Text pogleda:

```
@Composable
fun EateryCard(name: String, ...){
      ...
      Text("Straight from the Market")
...
}
```

Text komponenta prikazuje string na ekranu. Možete koristiti bilo koji string literal ili varijablu.

Image

Prikazivanje slika

Image komponenta prikazuje grafičke resurse u vašoj aplikaciji.

```
@Composablefun EateryCard(name: String, ...){
        ...
        Image(
        painterResource(R.drawable.market),
        contentDescription = "marketplace"
        )
        ...
}
```

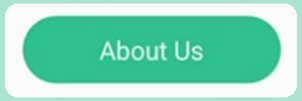
Slike se učitavaju iz `drawable` foldera vašeg projekta koristeći `painterResource`. **contentDescription** je obavezan parametar za pristupačnost - opisuje sliku korisnicima sa oštećenjima vida.

Button

Primjer Button:

```
Button(onClick = {...})
        {
                Text("About Us")
}
```

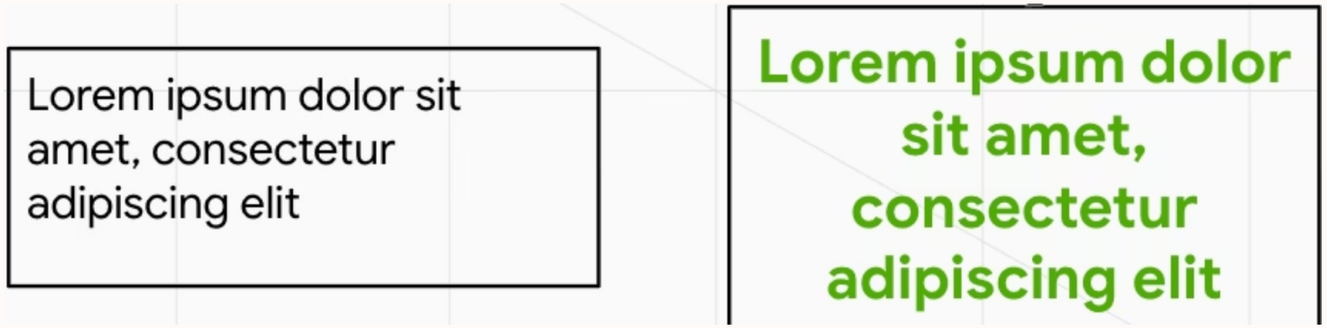
Button prima `onClick` lambda funkciju koja definiše šta se dešava kada korisnik klikne dugme. Sadržaj dugmeta (obično Text) se definiše unutar lambda bloka.



Argumenti - fino uređivanje UI elemenata

Kako kontrolisati izgled komponenti?

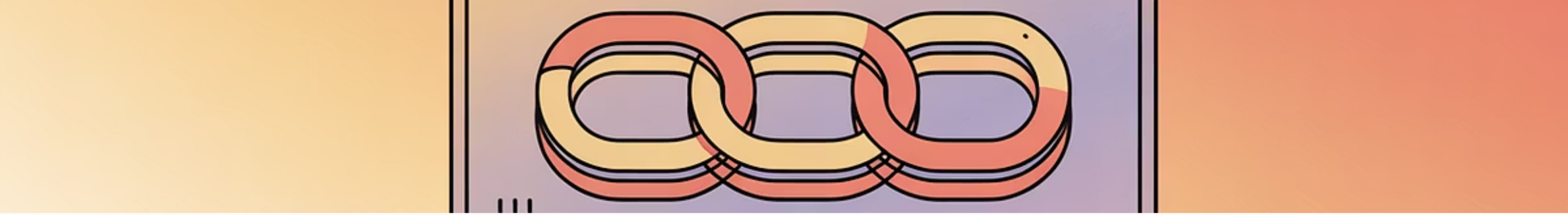
Za detaljne promjene izgleda vaših UI komponenti, koristite **argumente** Composable funkcije. Svaka Compose komponenta ima set parametara koje možete podesiti.



Primjer: Stilizovanje teksta

```
Text(
        text = "Lorem ipsum...",
        fontSize = 24.sp,
        fontWeight = FontWeight.Bold,
        color = Color.Green,
        textAlign = TextAlign.Center,
...)
```

Argumenti omogućavaju kontrolu nad veličinom fonta, težinom, bojom, poravnanjem i mnogim drugim osobinama specifičnim za svaki tip komponente!



Modifikatori - generalne promjene

Šta su modifikatori?

Za generalnije promjene u načinu na koji se komponenta renderuje - kao što su širina, visina, pozadina, padding - koristite **Modifikatore**.

Ključne karakteristike:

- Modifikatori se mogu **ulančavati** (chain-ovati)
- Redoslijed je važan! Modifikatori se primjenjuju slijeva na desno
- Modifikatori su univerzalni - rade sa svim Composable komponentama

Primjer sa Modifikatorima:

```
Text(
        text = "Lorem ipsum...",
        modifier = Modifier
                .fillMaxWidth()
                 .background(Color.Green))
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit

vs.

Lorem ipsum dolor sit amet, consectetur adipiscing elit

U primjeru iznad, `fillMaxWidth()` proširuje tekst na punu širinu, a zatim `background()` dodaje zelenu pozadinu. Promjena redoslijeda može dati različite rezultate!

Modifikatori

Postoji MNOGO modifikatora, i izuzetno su korisni za precizno oblikovanje i prilagođavanje vašeg korisničkog interfejsa!

.fillMaxWidth() & .fillMaxHeight()

Prilagođavaju komponentu da zauzme svu dostupnu širinu ili visinu roditeljskog elementa.

```
Text(    text = "Širok tekst",
modifier = Modifier.fillMaxWidth())
```

.weight(...)

Koristi se unutar `Row` ili `Column` da se komponenti dodijeli proporcionalan udio raspoloživog prostora.

```
Row {    Box(Modifier.weight(1f))
    Box(Modifier.weight(2f))}
```

.background(...)

Postavlja boju ili grafiku pozadine komponente.

```
Text(    text = "Zelena pozadina",
            modifier =
Modifier.background(Color.Green))
```

.padding(...)

Dodaje unutrašnji razmak oko sadržaja komponente, gurajući ga dalje od njenih ivica.

```
Text(    text = "Padding 16dp",
modifier = Modifier.padding(16.dp))
```

.width(...) & .height(...)

Postavljaju specifičnu, fiksnu širinu ili visinu komponente u dp jedinicama.

```
Text(    text = "Fiksna veličina",
modifier = Modifier.width(100.dp))
```

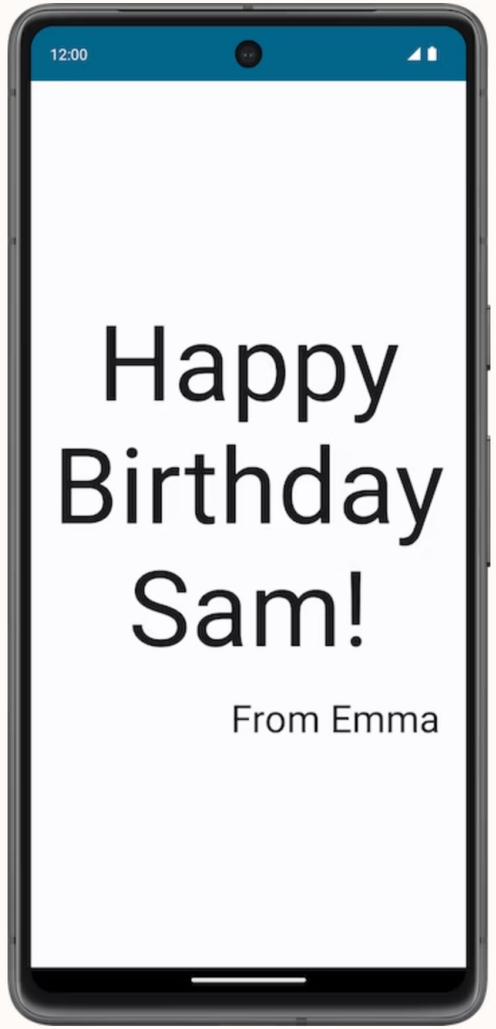
.clickable(...)

Omogućava komponenti da reaguje na dodir ili klik, izvršavajući određenu akciju.

```
Text(    text = "Klikni me!",    modifier
 = Modifier.clickable { /* akcija */ })
```

Primjer

Vaša prva Jetpack Compose aplikacija



Zadatak

Poravnaj sav sadržaj horizontalno i vertikalno na sredinu ekrana.
Za prvi Text element postavi:

- Bold stil fonta,
- gornji razmak (paddingTop) od 24dp,
- donji razmak (paddingBottom) od 8dp.

Za drugi Text element postavi veličinu fonta na 16sp.

Resursi
Preuzmi ovu ili proizvoljnu sliku, dodaj je u svoj projekat,
i koristi sljedeće tekstualne stringove:

- All tasks completed
- Nice work!