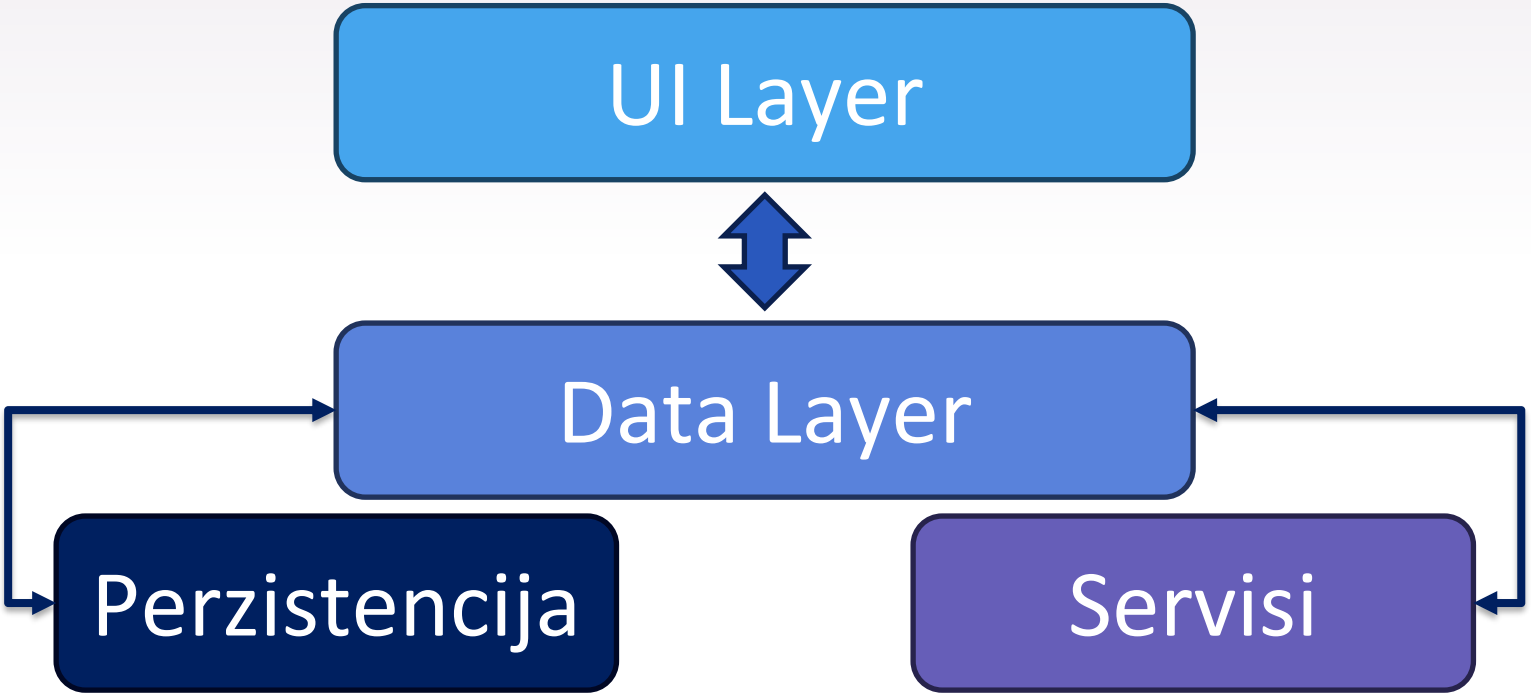
# Jetpack Compose

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

**Razvoj mobilnih aplikacija i servisa**
Računarstvo i informatika

2024.

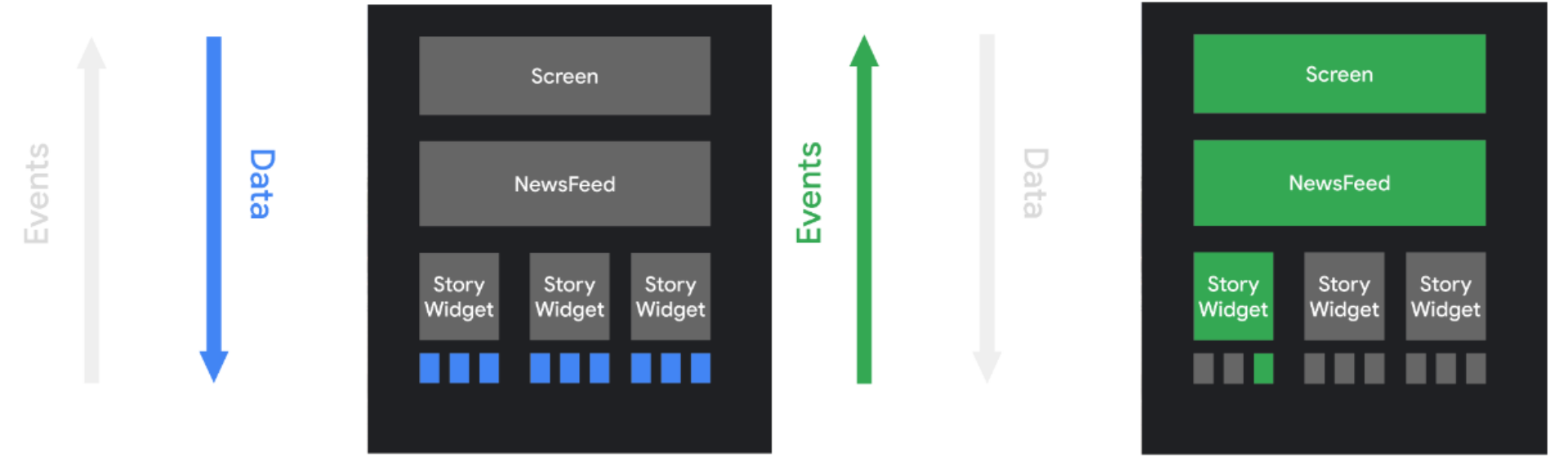
# Struktura Android aplikacije



# Jetpack Compose

- Alat (biblioteka) za izradu native Android korisničkog interfejsa
- Deklarativno programiranje
- Lako povezivanje sa slojem podataka
- Ponovno korišćenje komponenti (reusability)

# Deklarativna paradigma



# Composable

- Composable je funkcija koja opisuje neku UI komponentu
- Composable funkcije mogu sadržati druge composable funkcije
- Ulazni parametri su podaci, a ne vraća ništa (no return value)
- Brza, idempotentna i bez bočnih efekata
  - Pri istim ulaznim vrednostima generiše potpuno istu stvar
  - Ne pravi promene u globalnim promenljivama (bočni efekti)

# Composable

- Da bi funkcija bila composable, potrebno je anotirati je sa ***@Composable:***

```
@Composable
fun TextDisplay(modifier: Modifier = Modifier.fillMaxSize())
{
    Surface(modifier.padding(24.dp)) {
        Text("Hello World")
    }
}
```

# Composable

- Pošto su composables Kotlin funkcije, moguće je koristiti bilo koje naredbe ili izračunavanja u okviru njih:

```
@Composable
fun Greeting(names: List<String>) {
    for (name in names) {
        Text("Hello $name")
    }
}
```

# Rekompozicija

- Rekompozicija je proces ažuriranja UI-ja aplikacije onda kada dođe do promene u okviru composable hijerarhije
- Rekompozicija se vrši samo u onim delovima koji su pod uticajem promene ulaznih podataka
- Voditi računa da composable funkcije budu brze, jer se rekompozicija može izvršiti u svakom frejmu (npr. kod animacija)

# Rekompozicija - primer

```
@Composable
fun ClickCounter(clicks: Int, onClick: () -> Unit) {
    Button(onClick = onClick) {
        Text("I've been clicked $clicks times")
    }
}
```

# Rekompozicija i bočni efekti

- Nikako se ne oslanjati na bočne efekte za rekompoziciju
- Rekompozicija se sigurno izvršava onda kada se ulazni parametri composable funkcija promene
- Ako u funkciji postoji neki „on change listener“, nema garancije da će se u okviru UI elementa naći najsvežija vrednost
- Čitanje podataka (npr. iz baze) ne treba raditi u okviru composable funkcija, već im proslediti kao parametar

# Pravila

- Composable funkcije se mogu pozvati u bilo kom redosledu
- Mogu se izvršavati i u paraleli
- Rekompozicija preskače što više funkcija i lambda izraza radi optimalnosti izvršenja
- Rekompozicija je „optimistic“ – ukoliko se neki ulazni parametar promeni pre nego što je rekompozicija završena, ona se prekida i otpočinje nova

# Redosled - primer

```
@Composable
fun ButtonRow() {
    MyFancyNavigation {
        StartScreen()
        MiddleScreen()
        EndScreen()
    }
}
```

# Bočni efekti - primer

```
@Composable
fun ListWithBug(myList: List<String>) {
    var items = 0
    Row(horizontalArrangement = Arrangement.SpaceBetween) {
        Column {
            for (item in myList) {
                Text("Item: $item")
                items++ // Side effect
            }
        }
        Text("Count: $items")
    }
}
```

# Jetpack UI Composables

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

**Jetpack Compose**
Razvoj mobilnih aplikacija i servisa

# Pisanje composable funkcije

- Moguće je proslediti joj proizvoljne parametre
- Korisno je proslediti joj Modifier parametar, jer se on koristi za formatiranje sadržaja koji composable formira
- Modifier se vezuje za Material Theme, kao i većina built-in composable elemenata

# Primer

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Surface(color = MaterialTheme.colorScheme.primary) {
        Text(
            text = "Hello $name!",
            modifier = modifier
        )
    }
}
```

# Neki UI elementi

- Surface
- Text
- Button
- Column
- Row
- ...

# Surface

- Površina – moguće je stilizovati, sadrži druge elemente
- Surface (Material Design)

```
@Composable
fun Surface(
    modifier: Modifier = Modifier,
    shape: Shape = RectangleShape,
    color: Color = MaterialTheme.colors.surface,
    contentColor: Color = contentColorFor(color),
    border: BorderStroke? = null,
    elevation: Dp = 0.dp,
    content: @Composable () -> Unit
): Unit
```

# State u composable funkciji

- Ukoliko se u composable funkciji definiše promenljiva i dodeli joj inicijalna vrednost, ona će pri svakoj rekompoziciji dobiti tu vrednost bez obzira na dalje promene u funkciji
- Kako bi se stara vrednost sačuvala, potrebno je definisati je kao state

# State u composable funkciji

```
@Composable
fun MyApp(modifier: Modifier = Modifier) {
    val showOnboarding = remember { mutableStateOf(true) }
    Surface(modifier) {
        if (showOnboarding) {
            OnboardingScreen(onContinueClicked = {
                        showOnboarding.value = false })
        } else {
            Greetings()
        }
    }
}
```

# State u composable funkciji

```
val shouldShowOnboarding = remember {
        mutableStateOf(true)
    }
shouldShowOnboarding.value = false;

var showOnboarding by remember { mutableStateOf(true) }
showOnboarding = false;

var showWithDestroy by rememberSaveable {
        mutableStateOf(true)
    }
```

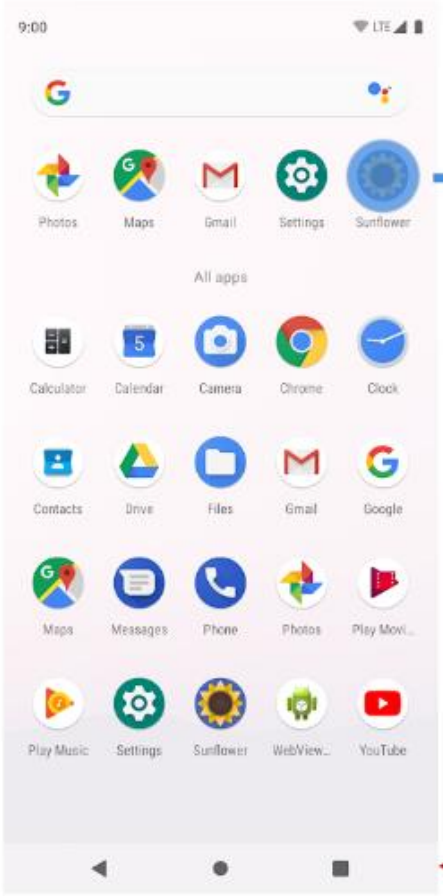
# Jetpack Compose Navigation

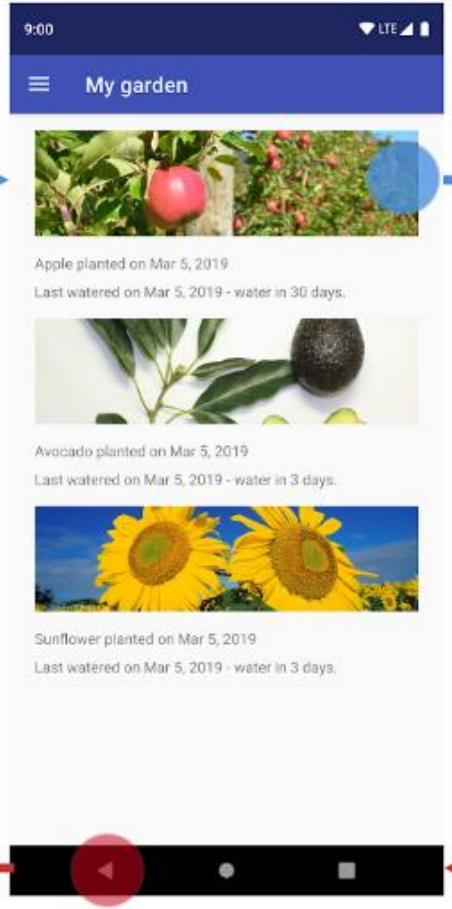
prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

**Jetpack Compose**
Razvoj mobilnih aplikacija i servisa

# Navigacija u aplikaciji

- Navigacija je promena sa jednog ekrana na drugi
- Principi dobre navigacije u aplikaciji:
  - Fiksna početna strana aplikacije
  - Stanje navigacije predstavljeno je stekom destinacija
  - Up i Back dugme rade istu stvar (vode na prethodnu destinaciju)
  - Up dugme ne može da izađe iz aplikacije, dok Back može





**Launcher** **List Screen** **Detail Screen**

# Navigation Component

- Tri glavna dela Navigation komponente Androida su:
  - **NavHost** – UI element koji sadrži trenutnu destinaciju navigacije
  - **NavGraph** – graf mogućih prelaza između destinacija u okviru aplikacije
  - **NavController** – matična komponenta navigacije koja upravlja svakom akcijom navigacije, sadrži i vodi računa o back stack-u

# Navigation Controller

- Najbolje je kreirati ga u najvišoj komponenti aplikacije, budući da je potreban svuda u aplikaciji
- Kreiranje: `val navController = rememberNavController()`
- Za svaki NavController vezuje se jedan NavGraph i jedan back stack

# Primer

- Aplikacija koja ima dva ekrana – Friends i Profile
- Kreiraju se composables za ova dva ekrana i glavnu aplikaciju
- Aplikacija sadrži NavHost composable koji prikazuje trenutnu destinaciju
- Friends i Profile composables primaju funkciju za navigaciju kao ulazni parametar

# Primer

```
@Composable
fun Profile(onNavigateToFriendsList: () -> Unit) {
  Text("Profile")
  Button(onClick = { onNavigateToFriendsList() }) {
    Text("Go to Friends List")
  }
}
```

# Primer

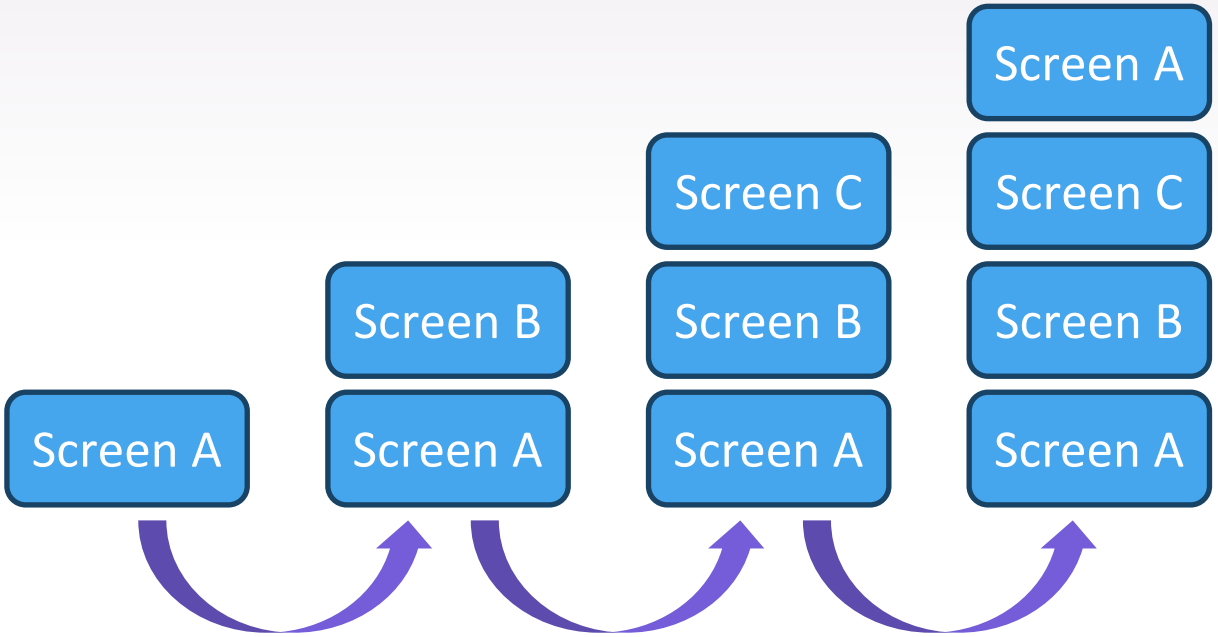
```
@Composable
fun FriendsList(onNavigateToProfile: () -> Unit) {
  Text("Friends List")
  Button(onClick = { onNavigateToProfile() }) {
    Text("Go to Profile")
  }
}
```

```
@Composable
fun MyApp() {
    val navController = rememberNavController()
    NavHost(navController, startDestination = "profile") {
        composable("profile") {
            Profile(
                onNavigateToFriendsList = { navController.navigate("friendslist")})
        }
        composable("friendslist") {
            FriendsList(
                onNavigateToProfile = { navController.navigate("profile") })
        }
    }
}
```

# Navigation Stack

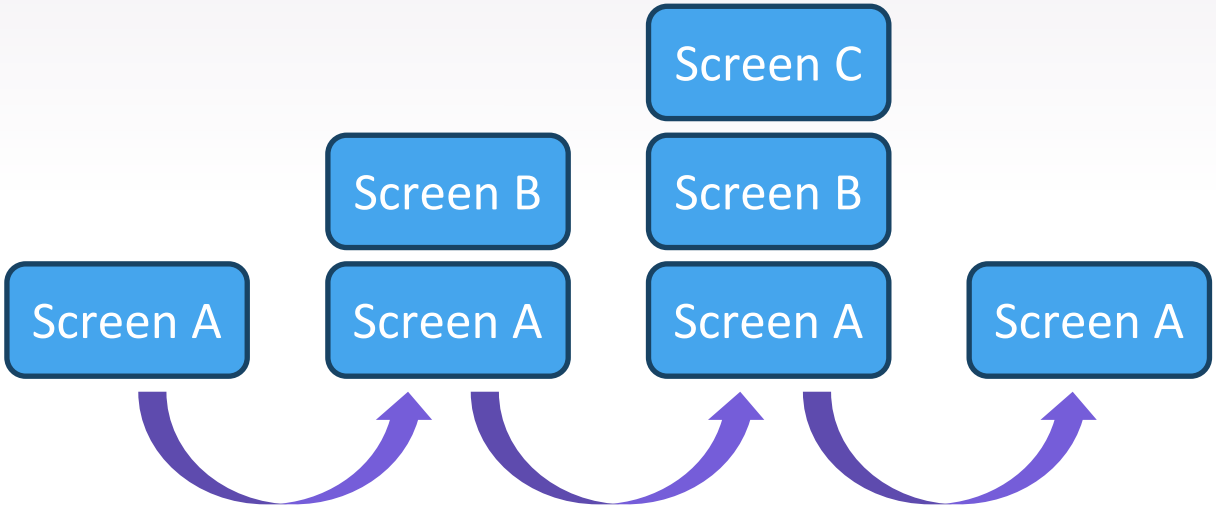
- Pri pokretanju aplikacije kreira se navigation stack (ili back stack)
- Na dnu steka se uvek nalazi startni ekran aplikacije
- Svaka akcija navigacije dodaje ekran na vrh steka
- Svaki pritisak Back ili Up dugmeta treba da skine ekran sa vrha steka (pop back stack)

# Pop Back Stack




- Potrebno je voditi računa o back stack-u
- Ako je potrebno vratiti se na neki od prethodnih ekrana, nikako ne treba pozvati navigate() jer će to staviti ekran na stek

# Pop Back Stack - rešenje




- Umesto navigate(), pozvati popBackStack() kojoj treba proslediti destinaciju do koje prazni stek

```
navController.popBackStack(destinationId, inclusive)
```

# Literatura

- Principi Jetpack Compose-a
- Jetpack Composables dokumentacija (UI komponente)
- Navigation Codelab
- Principi navigacije u Androidu

# Hvala na pažnji!