PROGRAMIRANJE KORISNIČKIH INTERFEJSA 2025/2026

ELEKTOTEHNIČKI FAKULTET
UNIVERZITETA U BEOGRADU

Šta je Android?

- Operativni sistem za mobilne telefone i tablete
- Prva verzija – septembar 2008
- Google
- Stotine miliona telefona koriste Android u preko 190 zemalja
- Veliki broj aplikacija za Android koje se mogu preuzeti sa Google Play-a

Verzije Androida

API nivo

Svaka verzija Androida podržava određeni API (dostupne funkcionalnosti)

Npr:

- Android 10 – api level 29
- Android 11 – api level 30
- Android 12 – api level 31, 32
- Android 13 – api level 33
- Android 14 – api level 34
- Android 15 – api level 35
- Android 16 – api level 36

Instalacija alata

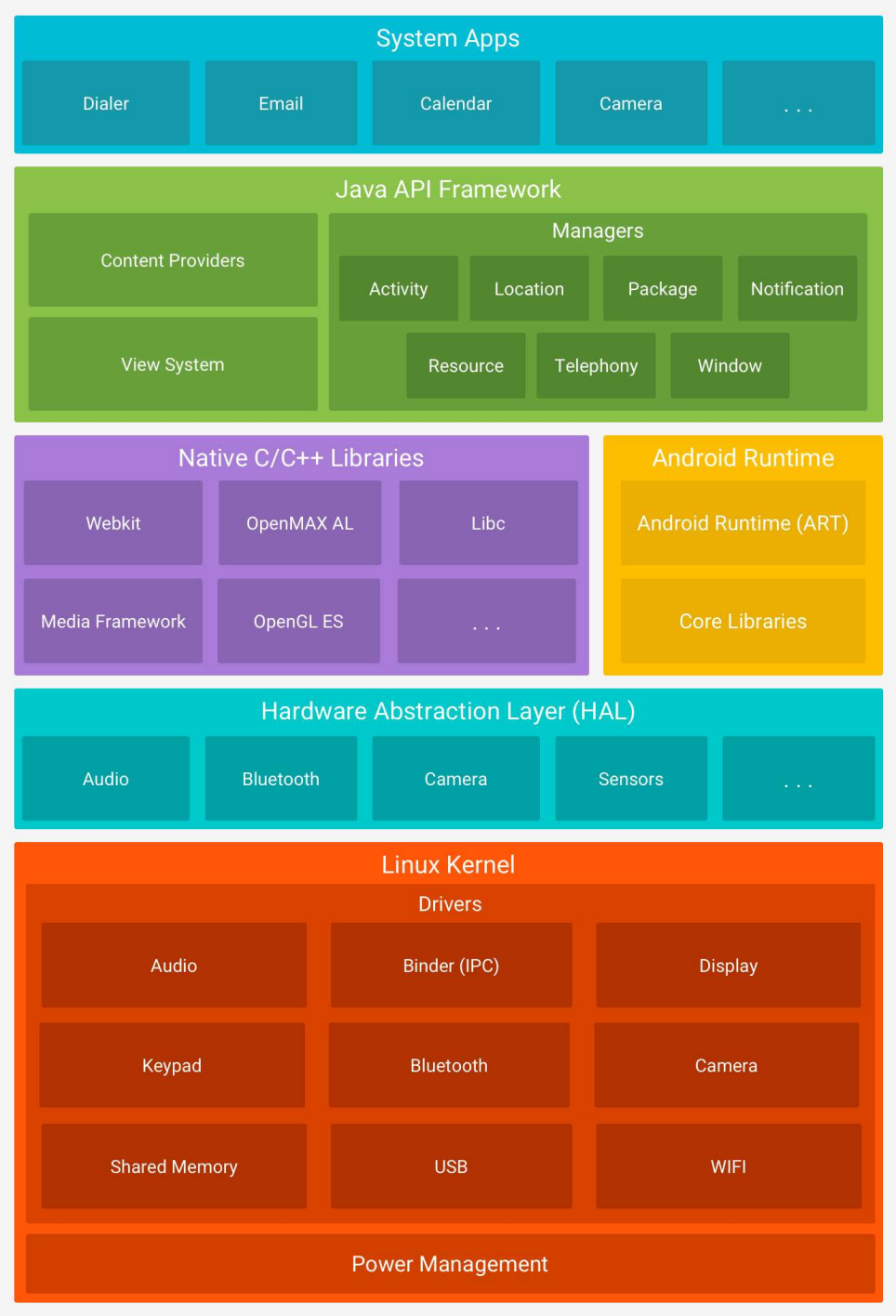
Android Studio

https://developer.android.com/studio/

Dokumentacija

https://developer.android.com/guide/

https://developer.android.com/reference/android/package-summary



Arhitektura Androida

Arhitektura Androida

Linux kernel – osnova Android platforme

Apstrakcija hardvera – ovaj sloj sadrži biblioteke koje standardizuju pristup hardverskim komponentama, i svaki hardverski modul ima svoj interfejs za povezivanje hardvera sa višim slojevima sistema

Svaka Android aplikacija ima svoj proces koji koristi svoju instancu **Android Runtime-a**. ART (ranije Dalvik VM) koristi napredne tehnike kompajliranja tako da poboljšava preformanse aplikacije i upravljanja memorijom

Osnovni delovi Android sistema, kao što su ART i HAL, napisani su u **C/C++**. Android pruža Java API-je koji omogućavaju pristup funkcionalnostima ovih nativnih biblioteka.

Sve funkcionalnosti Android operativnog sistema dostupne su kroz **Java API**-je

- Activity Manager – kotroliše životni ciklus aplikacije.
- Content Providers – dozvoljava deljenje sadržaja sa drugim aplikacijama.
- Resource Manager – omogućava pristup resursima.
- Notifications Manager – obezbeđuje rad sa notifikacijama.
- View System – služi za kreiranje korisničkog interfejsa.

Poslednji sloj čine **sistemske aplikacije**, poput aplikacija za SMS, kalendar, kontakte i slično.

Kotlin

Multiplatformski programski jezik, kompatibilan sa Javom

https://play.kotlinlang.org/

```
fun main() {
    var msgFromFunction = happyBirthday("Petar")
    msgFromFunction += " Yey!"
    print(msgFromFunction)
}

fun happyBirthday(name: String): String{
    val msg: String = "Happy birthday, $name!"
    /*
     Comments
    */
    return msg
}
```

```
fun main() {
    val trafficLightColor = "Amber"

    when (trafficLightColor) {
        "Red" -> println("Stop")
        "Yellow", "Amber" -> println("Slow")
        "Green" -> println("Go")
        else -> println("Invalid traffic-light color")
    }
}
```

```
fun main() {
    var favoriteActor: String? = "Sandra Oh"
    favoriteActor = null
}
```

```
class SmartDevice(val name: String, val category: String) {
    var deviceStatus = "online"

    constructor(name: String, category: String, statusCode: Int) : this(name, category)
        deviceStatus = when (statusCode) {
            0 -> "offline"
            1 -> "online"
            else -> "unknown"
        }
    }
    ...
}
```

```
var speakerVolume = 2
    get() = field
    set(value) {
        field = value
    }
```

```
open class SmartDevice(val name: String, val category: String) {
    ...
}
```

```
class SmartTvDevice(deviceName: String, deviceCategory: String) :
    SmartDevice(name = deviceName, category = deviceCategory) {
}
```

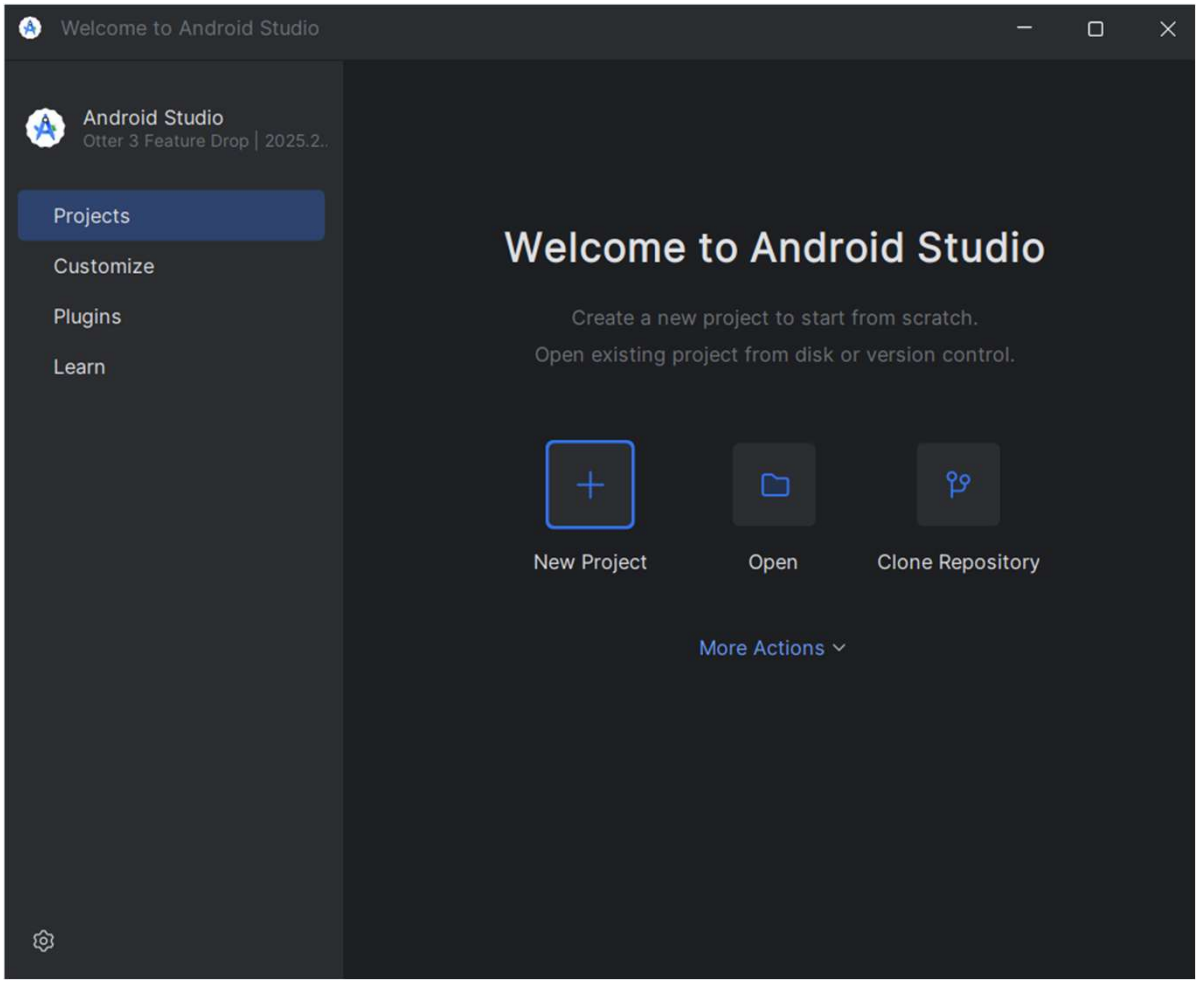
Lambda izrazi

```
fun trickOrTreat(isTrick: Boolean, extraTreat: (Int) -> String): () -> Unit {
    if (isTrick) {
        return trick
    } else {
        println(extraTreat(5))
        return treat
    }
}
```

```
val trick = {
    println("No treats!")
}

val treat = {
    println("Have a treat!")
}
```

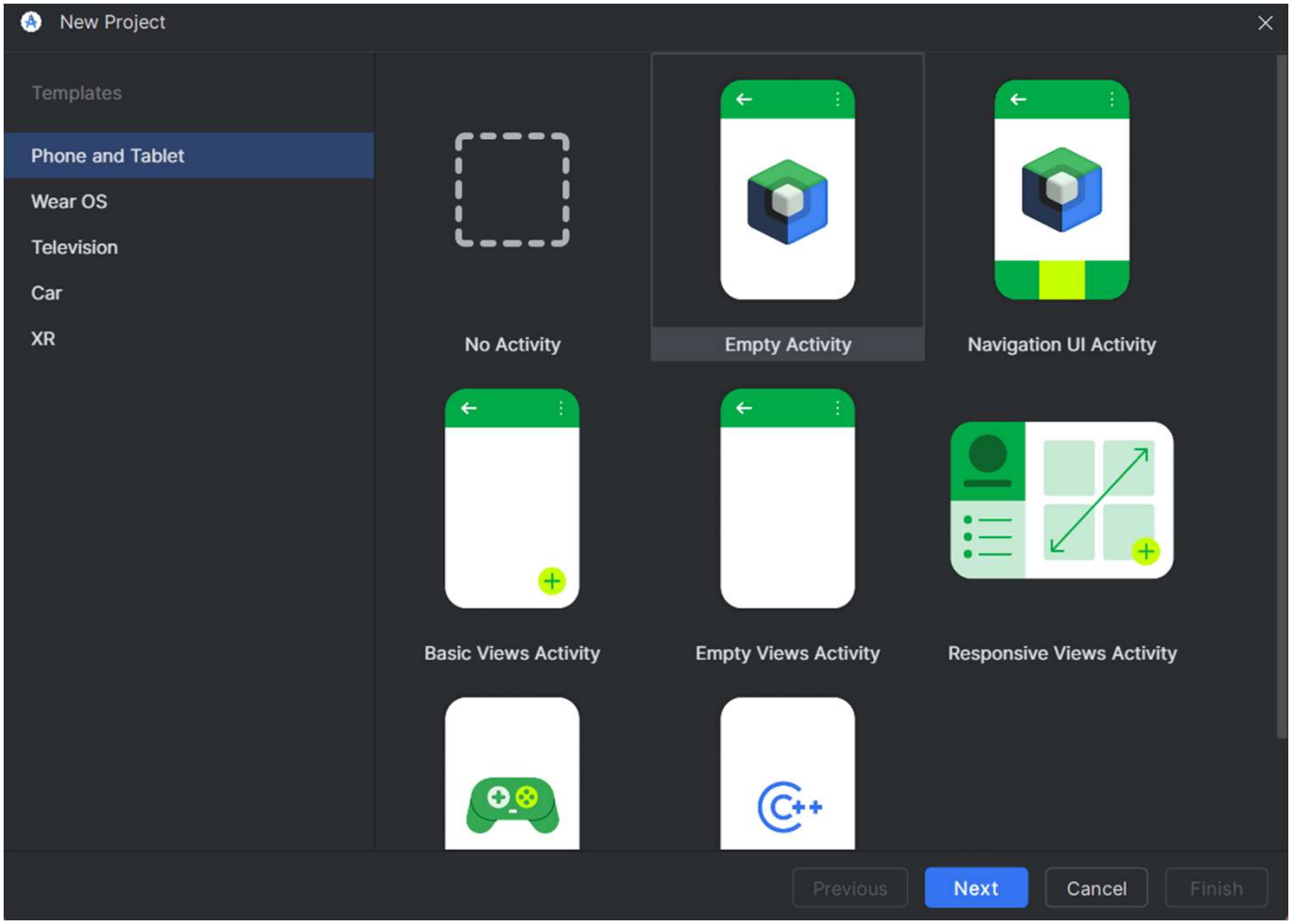
```
val coins: (Int) -> String = { quantity ->
    "$quantity quarters"
}
```

```
val coins: (Int) -> String = {
    "$it quarters"
}
```

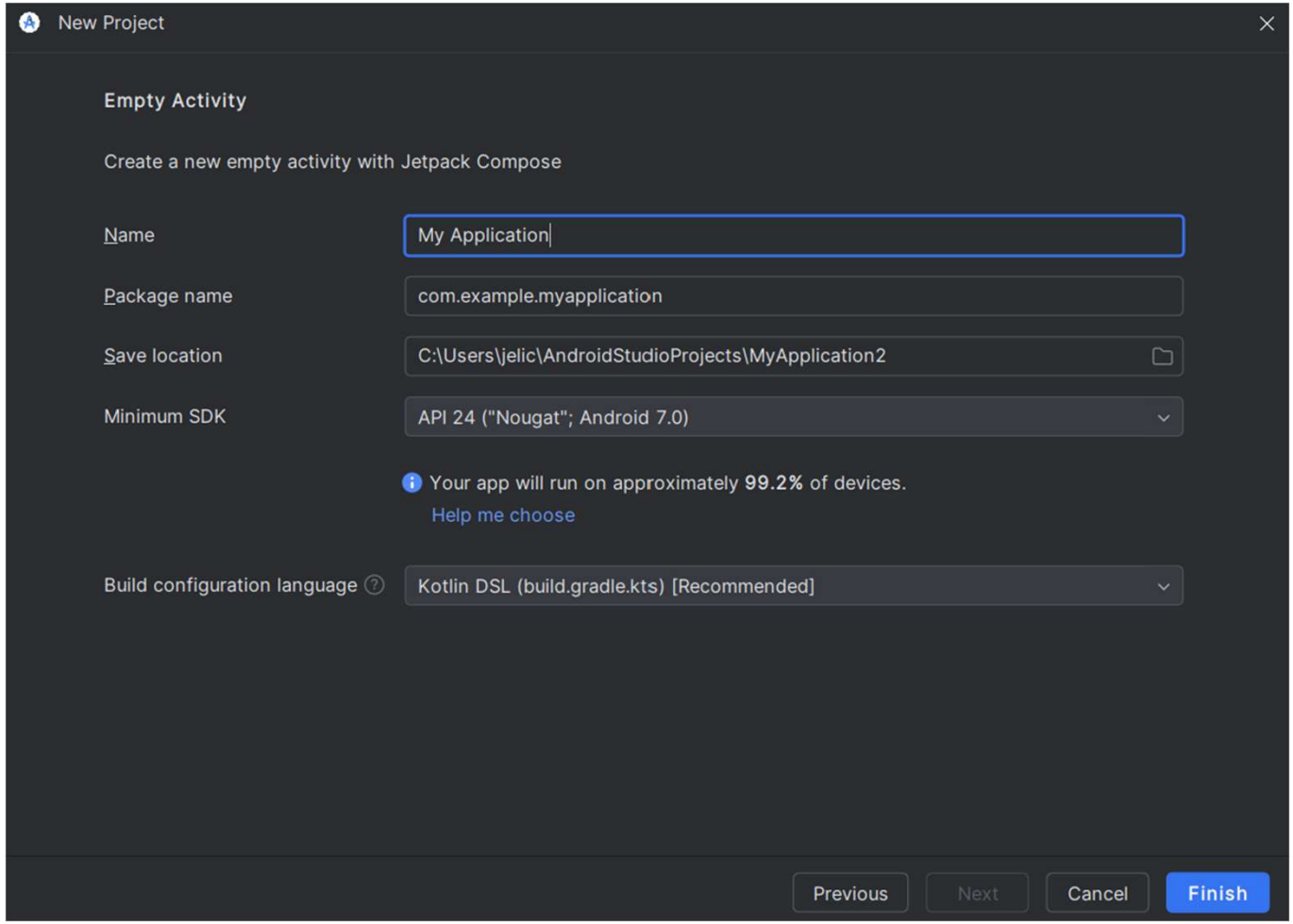

Welcome to Android Studio
Android Studio
Otter 3 Feature Drop | 2025.2..
Projects
Customize
Plugins
Learn
Welcome to Android Studio
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Clone Repository
More Actions

Hello World

Hello World


New Project
Templates
Phone and Tablet
Wear OS
Television
Car
XR
No Activity
Empty Activity
Navigation UI Activity
Basic Views Activity
Empty Views Activity
Responsive Views Activity
Previous
Next
Cancel
Finish

Hello world


New Project
Empty Activity
Create a new empty activity with Jetpack Compose
Name
My Application
Package name
com.example.myapplication
Save location
C:\Users\jelic\AndroidStudioProjects\MyApplication2
Minimum SDK
API 24 ("Nougat"; Android 7.0)
Your app will run on approximately 99.2% of devices.
Help me choose
Build configuration language
Kotlin DSL (build.gradle.kts) [Recommended]
Previous
Next
Cancel
Finish

Struktura Android aplikacije

Manifest folder

- AndroidManifest.xml - Fajl koji opisuje fundimentalne karakteristike naše aplikacije

Java folder (Kotlin + java)

- Sadrži .kt fajlove našeg projekta kao i glavnu aktivnost koja se pokreće pri startovanju aplikacije

Res folder

- Drawable - Resursi koje možemo iscrtati na ekranu (npr. slike)
- Mipmap – Launcher ikonica aplikacije
- Values - XML fajlovi koji sadrže kolekcije stringova, boja...

Gradle skripte

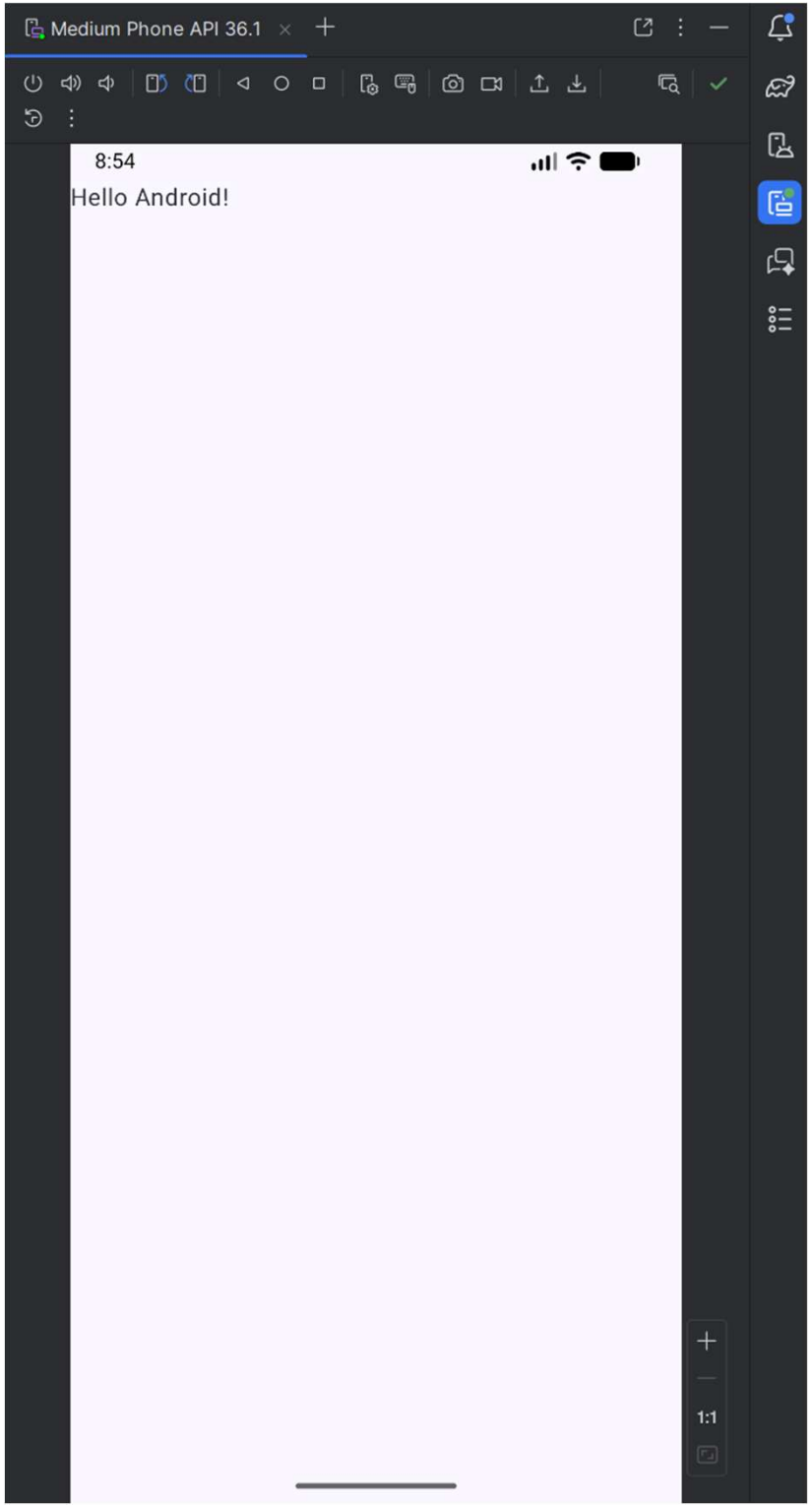
AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="My Application"
            android:theme="@style/Theme.MyApplication">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

```kotlin
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            MyApplicationTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        name = "Android",
                        modifier = Modifier.padding( paddingValues = innerPadding)
                    )
                }
            }
        }
    }
}
```

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

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    MyApplicationTheme {
        Greeting( name = "Android")
    }
}
```

MainActivity.kt



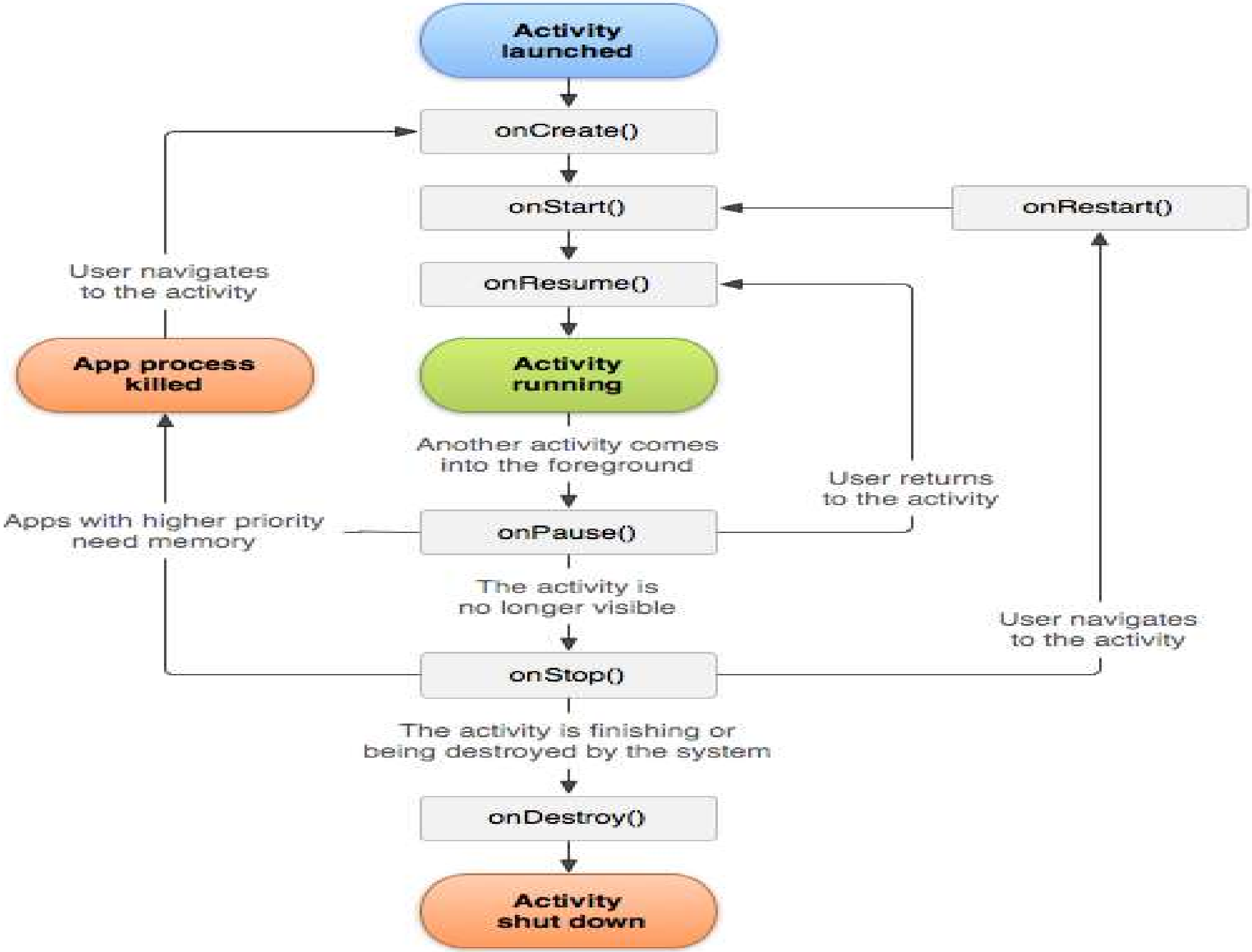
Pokretanje aplikacije

- Device Manager
- Android Emulator

Aktivnosti

Android sistem inicira program aktivnošću koja zove svoju onCreate() metodu

Svaka aktivnost ima svoj životni ciklus koji se sastoji od sekvence metoda koje startuju aktivnost i sekvence metoda koje gase aktivnost


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Sr.No	Callback & Description
1	**onCreate()** Zove se nakon što je aktivnost prvi put kreirana
2	**onStart()** Zove se nakon što je aktivnost postala vidljiva korisniku
3	**onResume()** Zove se nakon što je aktivnost postala spremna za interakciju sa korisnikom
4	**onPause()** Zove se nakon što je aktivnost pauzirana, a i dalje je vidljiva, dok je fokusu nešto drugo (na primer: dijalog)
5	**onStop()** Zove se kada aktivnost više nije vidiljiva
6	**onDestroy()** Zove se nakon što je aktivnost uništena od strane sistema
7	**onRestart()** Zove se nakon što se aktivnost restartuje nakon stopiranja

Primer - životni ciklus aplikacije

MainActivity.kt - override-ovati metode od značaja

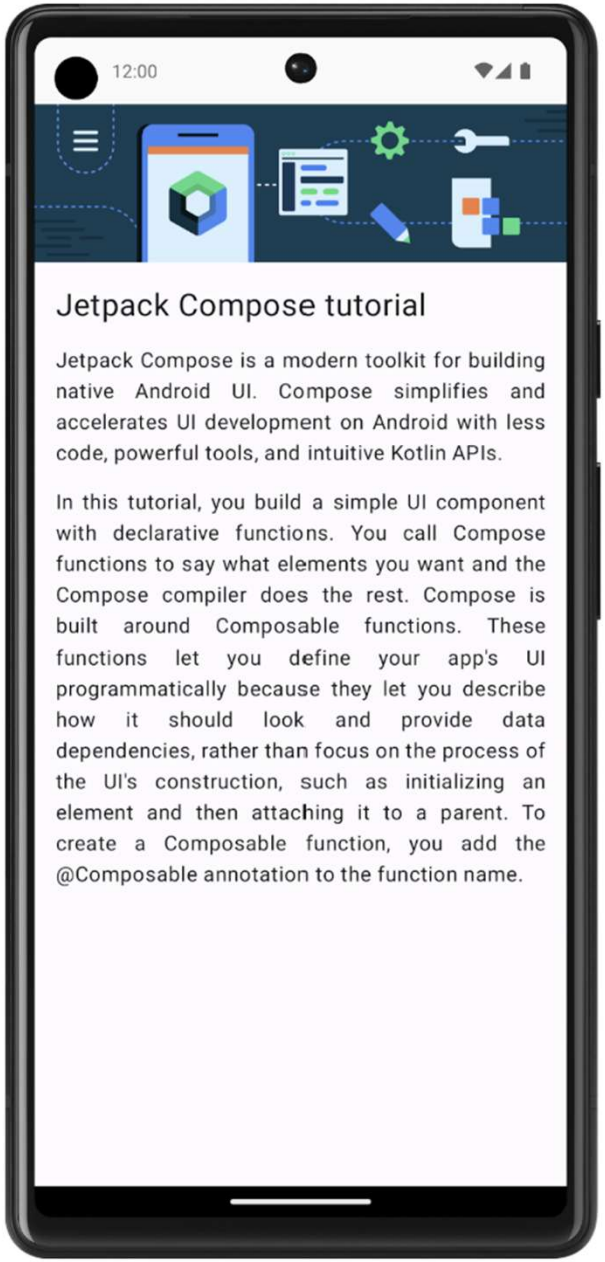
Pratiti sadržaj kozole u sledećim situacijama

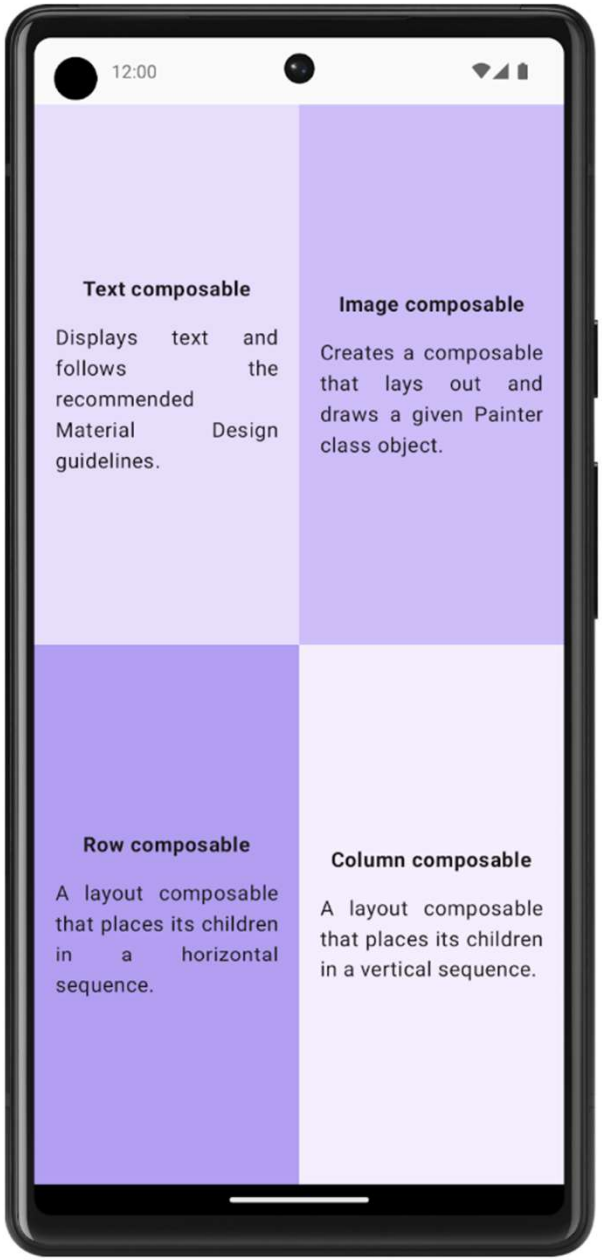
- Pri pokretanju aplikacije
- Nakon stavljanja aplikacije u background
- Nakon povratka aplikacije iz background-a
- Nakon klika na back

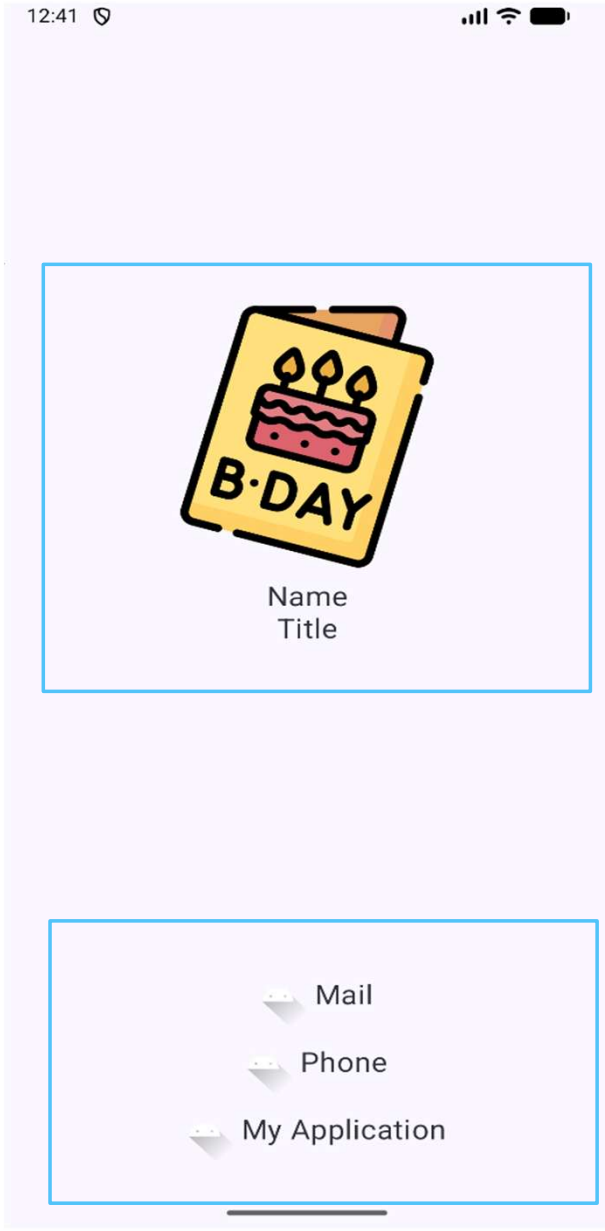
Dodavanje slike

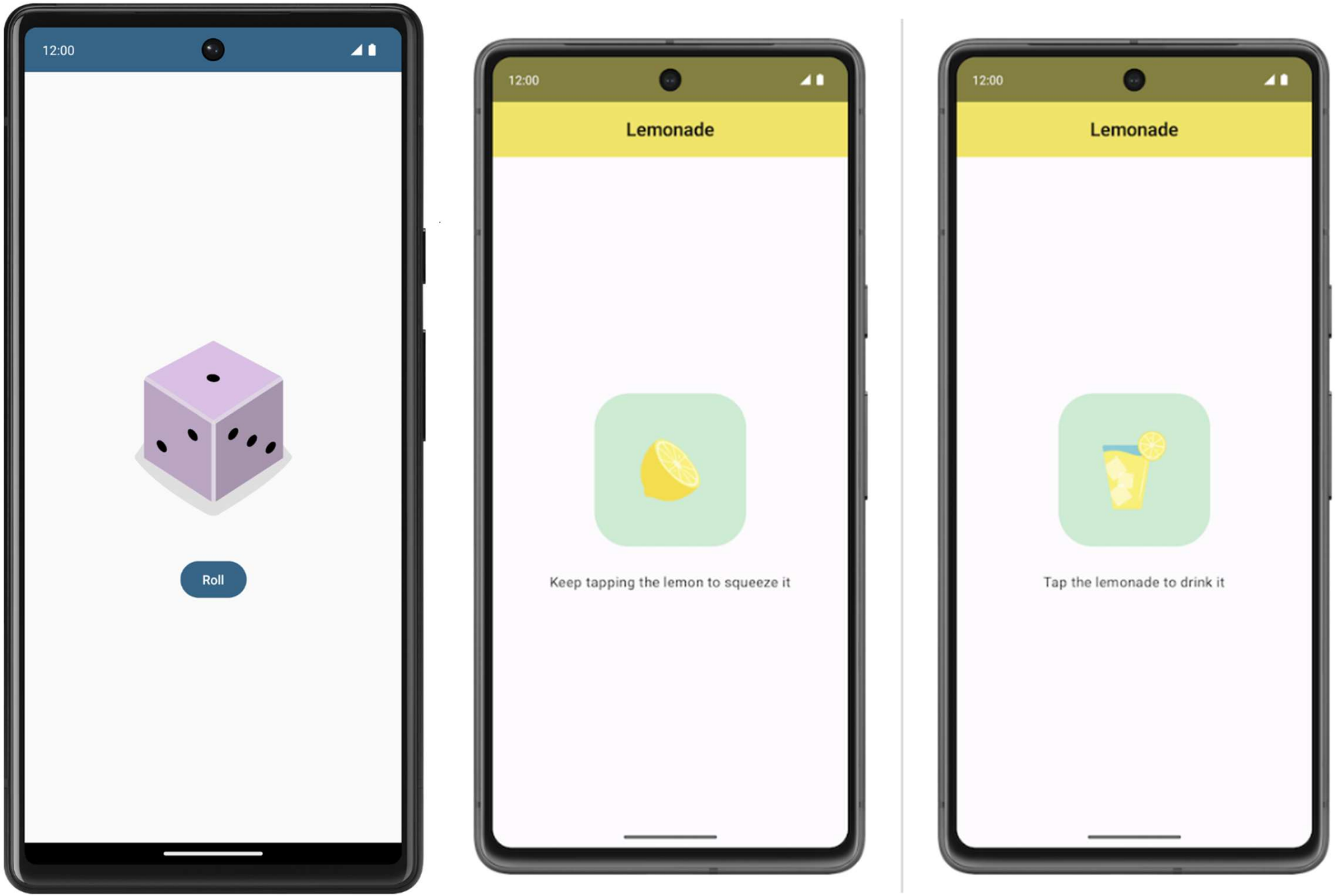
```
@Composable
fun GreetingText(name: String, from: String, modifier: Modifier = Modifier) {
    Column(verticalArrangement = Arrangement.Center,
        modifier = modifier.fillMaxSize()) {
        Text(
            text = "Happy birthday $name!",
            fontSize = 20.sp,
            textAlign = TextAlign.Center,
            modifier = Modifier.fillMaxWidth()
        )
        Text(
            text = "From $from",
            fontSize = 16.sp,
            modifier = Modifier.align(alignment = Alignment.CenterHorizontally)

        )
    }
}
```

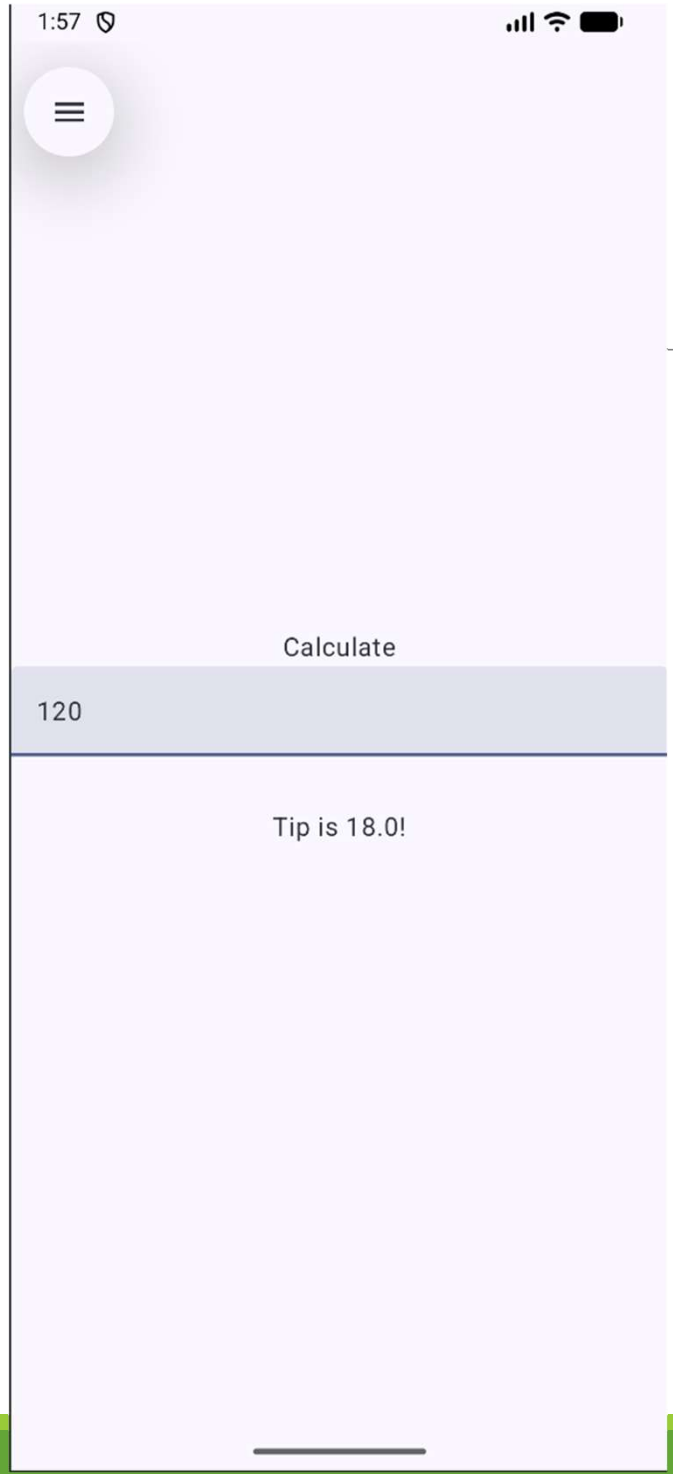
```
@Composable
fun Greeting(name: String, from: String, modifier: Modifier = Modifier){
    Column(verticalArrangement = Arrangement.Center,
        modifier = modifier.fillMaxSize()) {
        val image = painterResource( id = R.drawable.birthday_card)
        Image(painter = image, contentDescription = null,
            modifier = Modifier.align(alignment = Alignment.CenterHorizontally))
        GreetingText(name, from, modifier)
    }
}
```

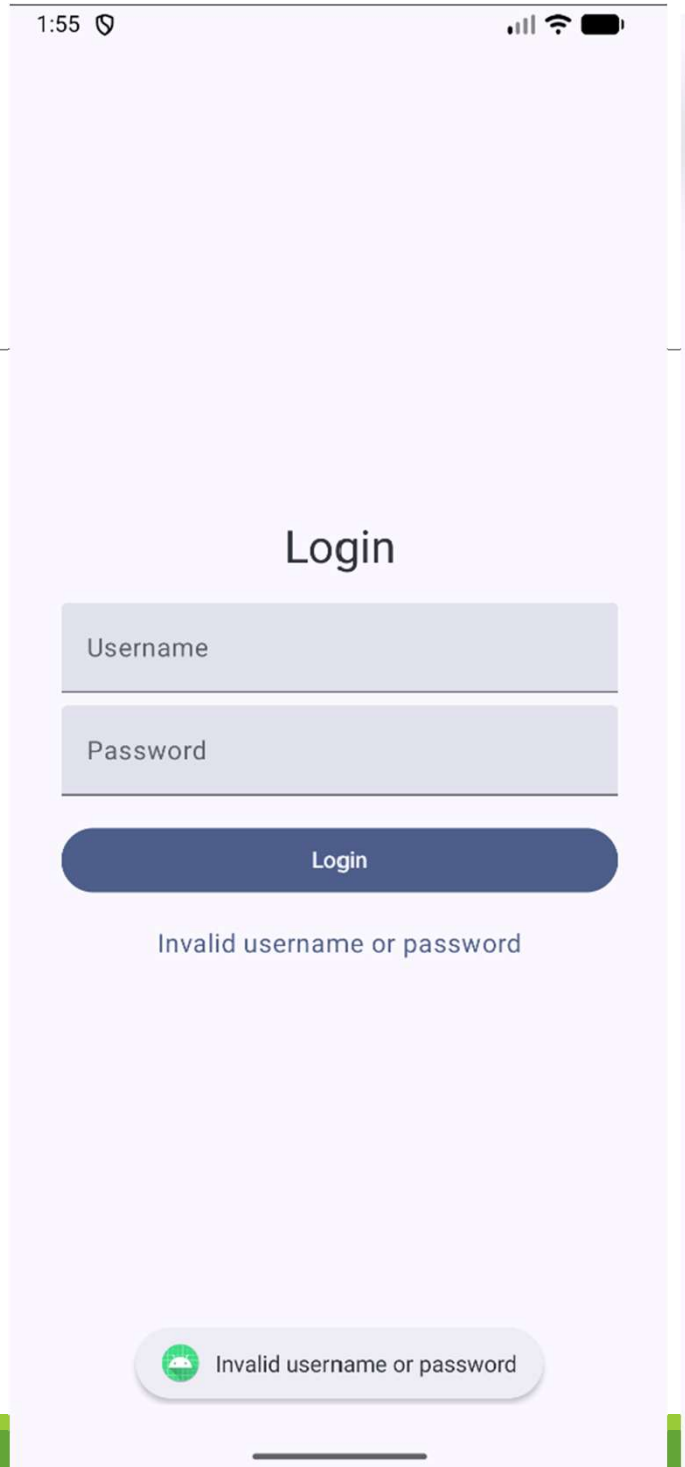

12:00
Jetpack Compose tutorial
Jetpack Compose is a modern toolkit for building native Android UI. Compose simplifies and accelerates UI development on Android with less code, powerful tools, and intuitive Kotlin APIs.
In this tutorial, you build a simple UI component with declarative functions. You call Compose functions to say what elements you want and the Compose compiler does the rest. Compose is built around Composable functions. These functions let you define your app's UI programmatically because they let you describe how it should look and provide data dependencies, rather than focus on the process of the UI's construction, such as initializing an element and then attaching it to a parent. To create a Composable function, you add the @Composable annotation to the function name.

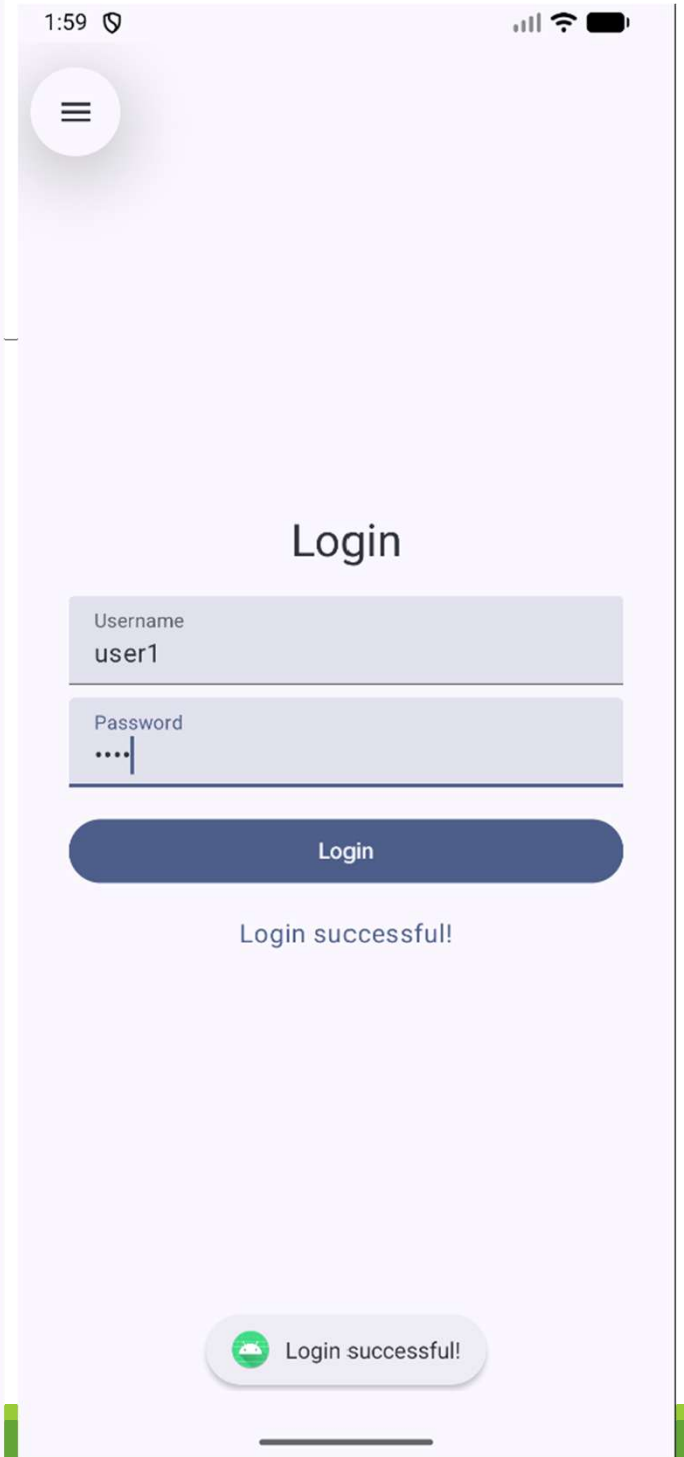

12:00
Text composable
Displays text and follows the recommended Material Design guidelines.
Image composable
Creates a composable that lays out and draws a given Painter class object.
Row composable
A layout composable that places its children in a horizontal sequence.
Column composable
A layout composable that places its children in a vertical sequence.

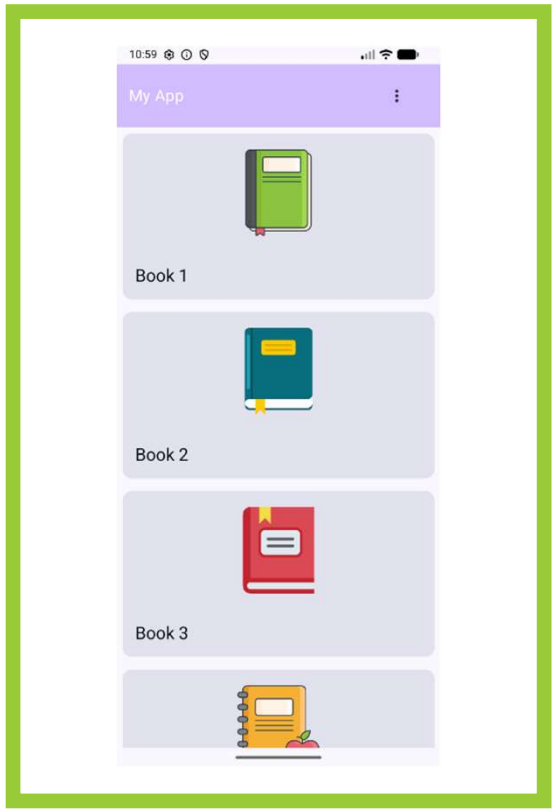

12:41
B·DAY
Name
Title
Mail
Phone
My Application

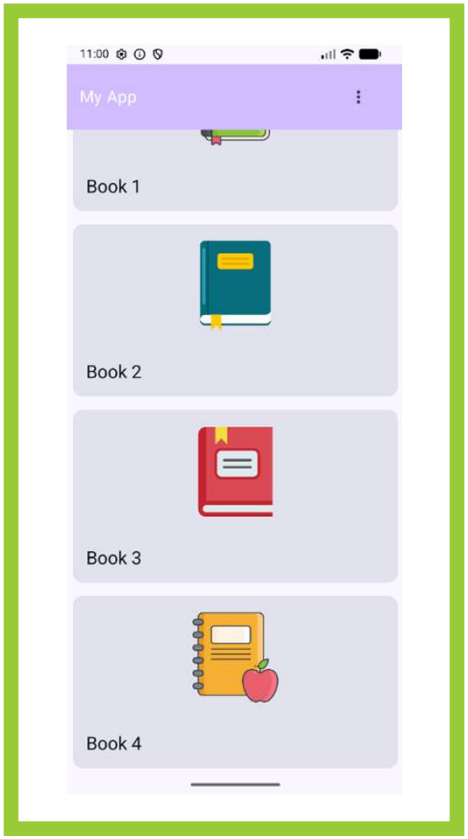

12:00
Roll
12:00
Lemonade
Keep tapping the lemon to squeeze it
12:00
Lemonade
Tap the lemonade to drink it


1:57
Calculate
120
Tip is 18.0!


1:55
Login
Username
Password
Login
Invalid username or password
Invalid username or password

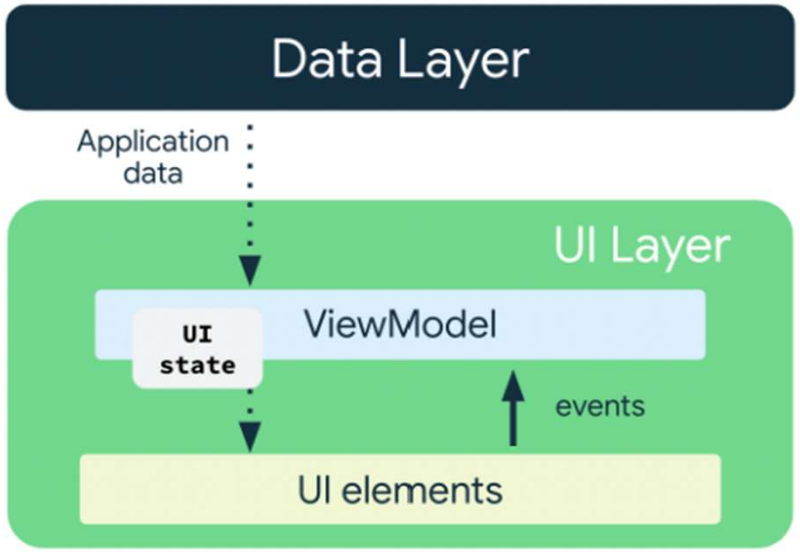

1:59
Login
Username
user1
Password
Login
Login successful!
Login successful!





Scroll lista

```
LazyColumn {
    items(100) { index ->
        Text(text = "Item $index")
    }
}
```



ViewModel

- Čuva i upravlja stanjima
- Odvaja UI od poslovne logike
- Preživljava promene konfiguracije (rotacija)
- Sprečava gubitak podataka pri recomposition-u
- Omogućava lakše testiranje i održavanje koda

Navigacija

```
NavHost(navController = navController, startDestination = "books")
{
    composable("books") {
        BooksScreen(
            viewModel = booksViewModel,
            onBookClick = { book ->
                val bookName = book.stringResourceId
                navController.navigate("bookDetails/$bookName")
            },
            modifier
        )
    }

    composable("bookDetails/{bookName}") { backStackEntry ->
        val bookName =
backStackEntry.arguments?.getString("bookName")?.toInt() ?: 0
        BookDetailsScreen(bookName = bookName, viewModel =
booksViewModel, modifier)
    }
}
```

11:00

My App

Option 1

Option 2

Book 1

Book 2

Book 3

11:00

My App

Hello movies

Open Camera

11:01

My App

Hello movies

Open Camera

Pitanja?