



#R4E

Software Developer

Android

Primeiro Projeto

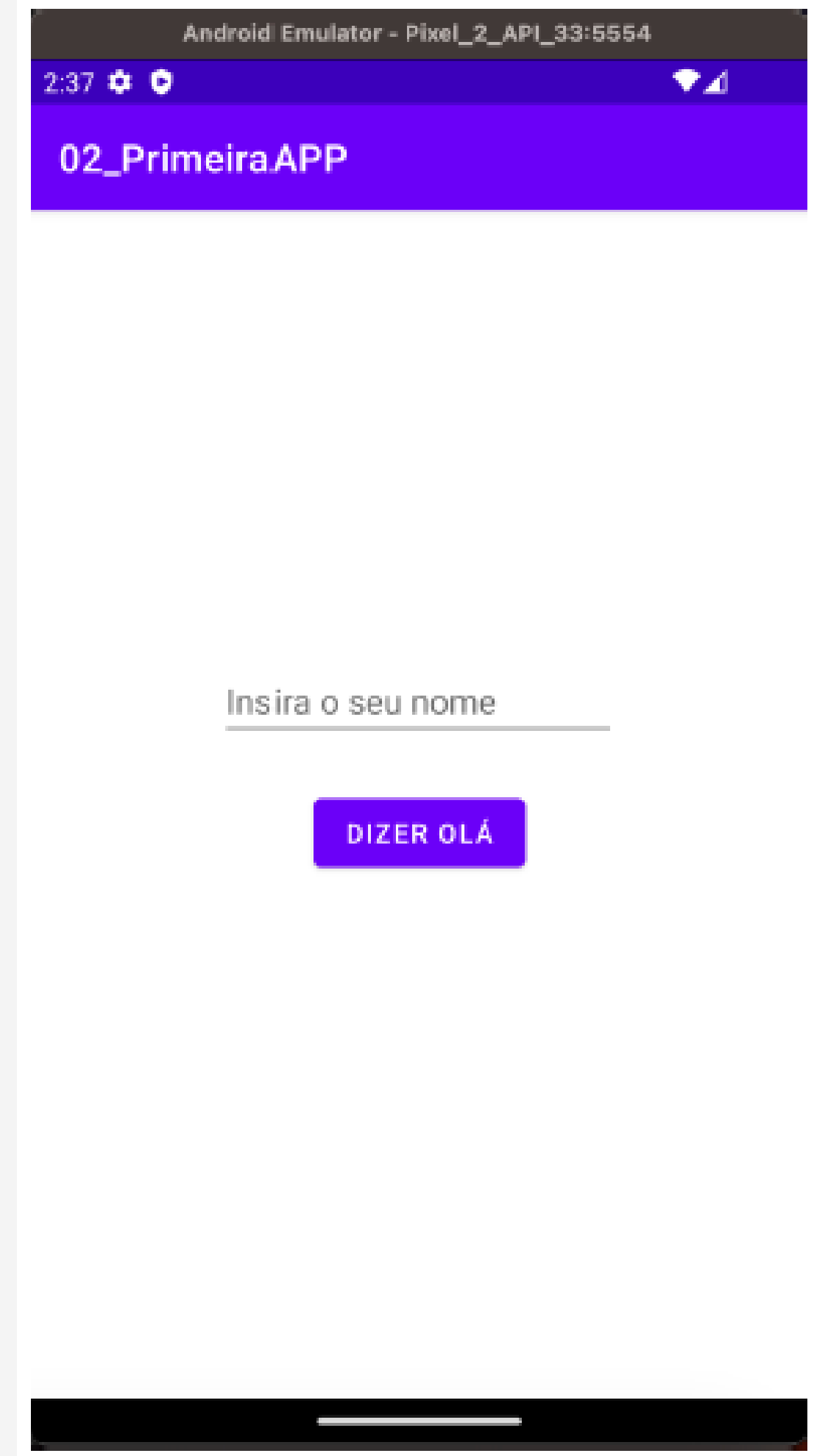


Conteúdo

- Elaboração da Primeira App

Primeira App

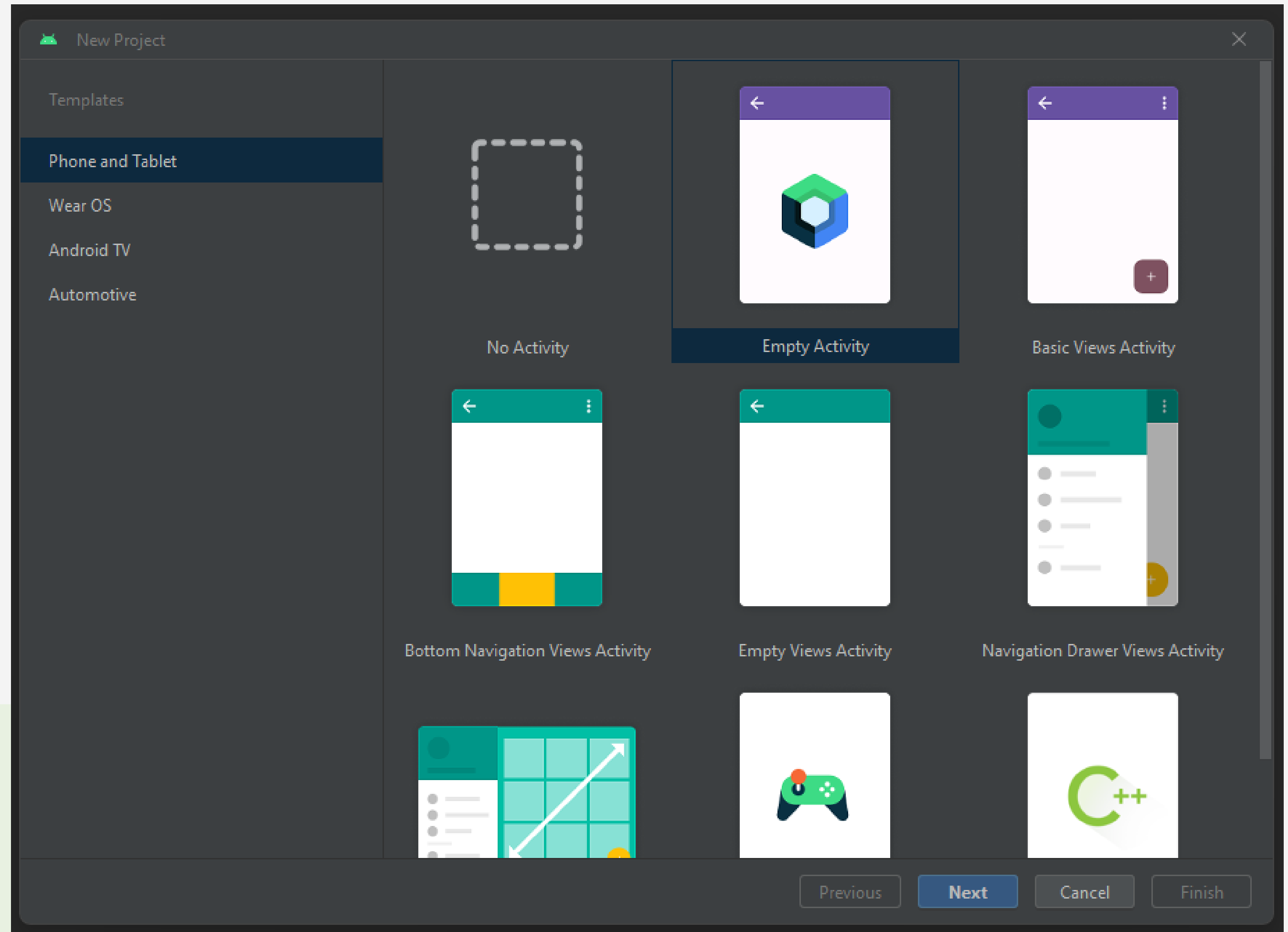
- Vamos criar uma aplicação que peça ao utilizador o seu nome e apresente uma mensagem de boas-vindas seguida do nome do utilizador.



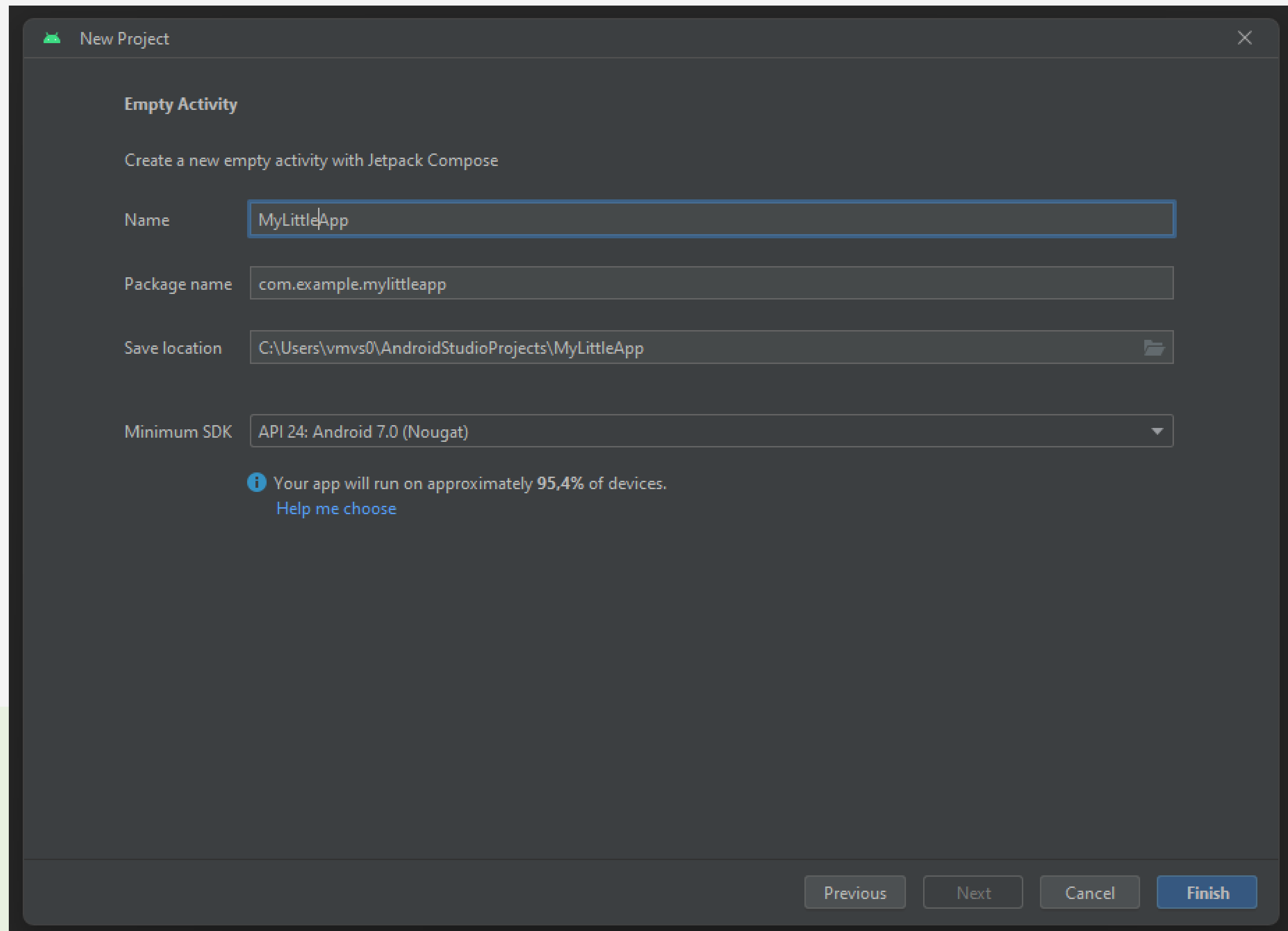
Primeira App

- Para criar a nova aplicação devemos:
 1. Abrir Android Studio;
 2. Selecionar “New Project”;
 3. Escolher o template “Empty Activity”
 4. Alterar o “Name” para um nome sugestivo que permita sabermos qual a aplicação que estamos a criar;
 5. Selecionar a localização da aplicação;
 6. Verificar que a linguagem selecionada é Kotlin.

Primeira App



Primeira App



New Project

Empty Activity

Create a new empty activity with Jetpack Compose

Name: MyLittleApp

Package name: com.example.mylittleapp

Save location: C:\Users\vmvs0\AndroidStudioProjects\MyLittleApp

Minimum SDK: API 24: Android 7.0 (Nougat)

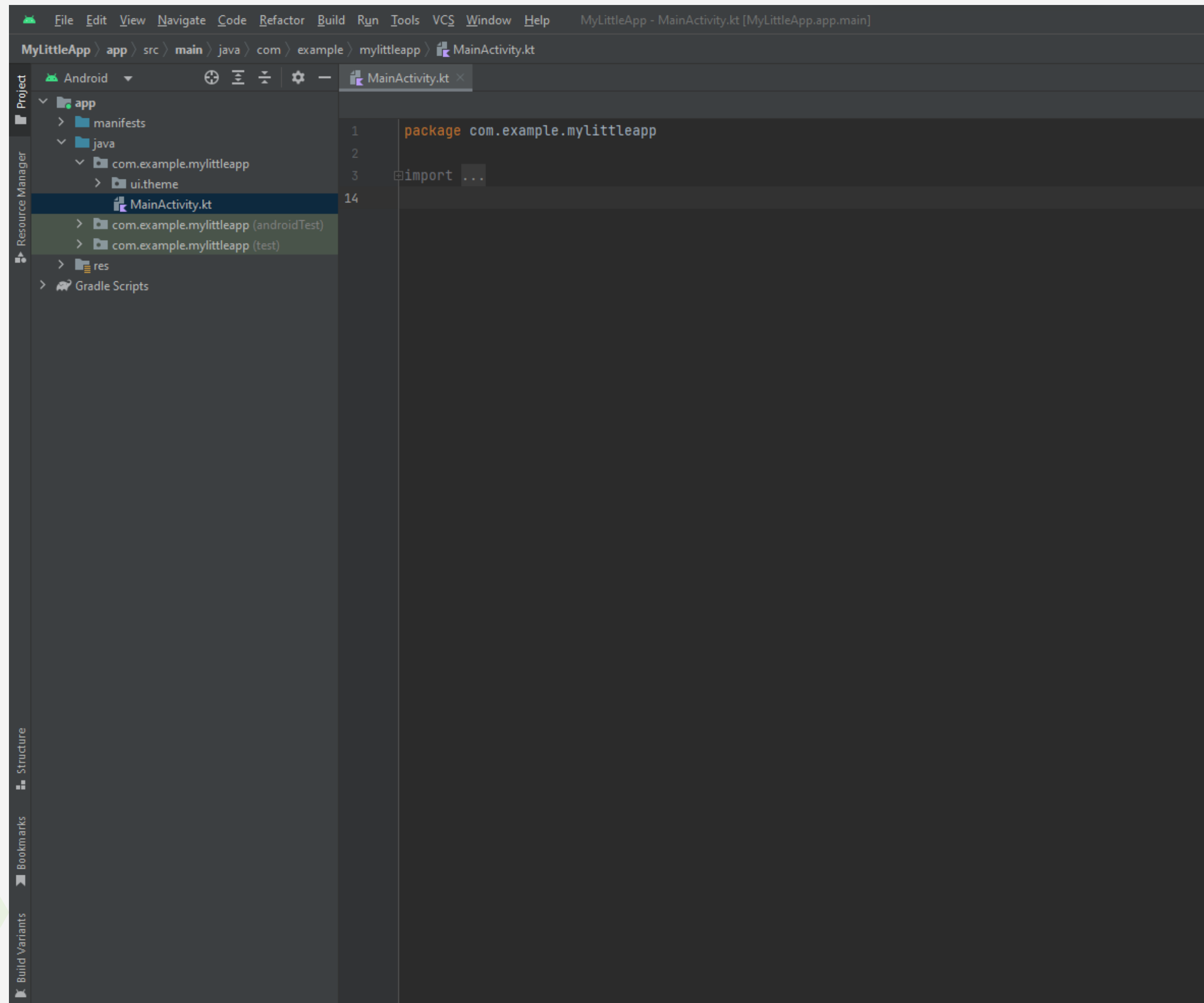
i Your app will run on approximately 95,4% of devices.
[Help me choose](#)

Previous Next Cancel Finish

Primeira App

```
1 package com.example.mylittleapp
2
3 import ...
4
14
15 class MainActivity : AppCompatActivity() {
16     override fun onCreate(savedInstanceState: Bundle?) {
17         super.onCreate(savedInstanceState)
18         setContent {
19             MyLittleAppTheme {
20                 // A surface container using the 'background' color from the theme
21                 Surface(
22                     modifier = Modifier.fillMaxSize(),
23                     color = MaterialTheme.colorScheme.background
24                 ) {
25                     Greeting(name: "Android")
26                 }
27             }
28         }
29     }
30 }
31
32 @Composable
33 fun Greeting(name: String, modifier: Modifier = Modifier) {
34     Text(
35         text = "Hello $name!",
36         modifier = modifier
37     )
38 }
39
40 @Preview(showBackground = true)
41 @Composable
42 fun GreetingPreview() {
43     MyLittleAppTheme {
44         Greeting(name: "Android")
45     }
46 }
```

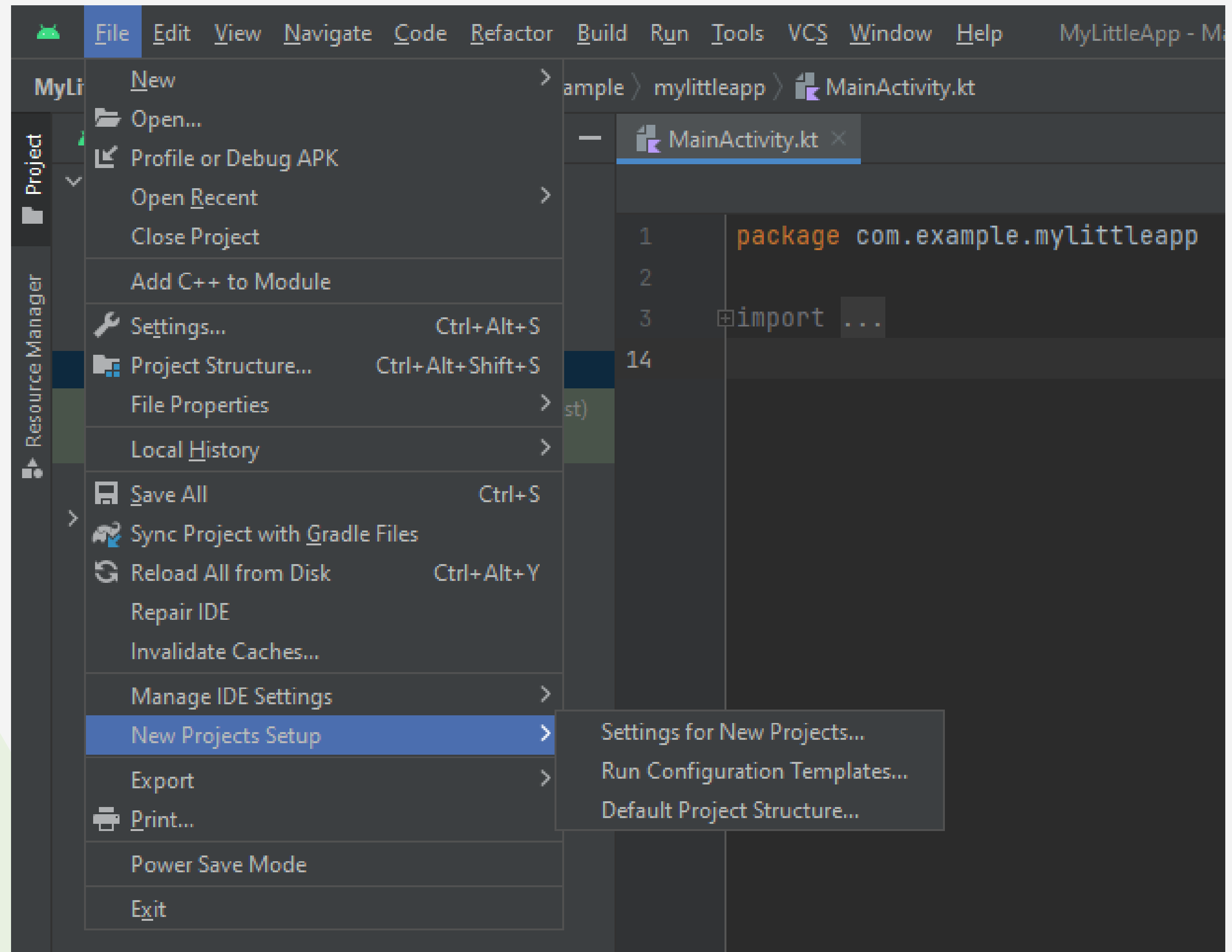
Primeira App



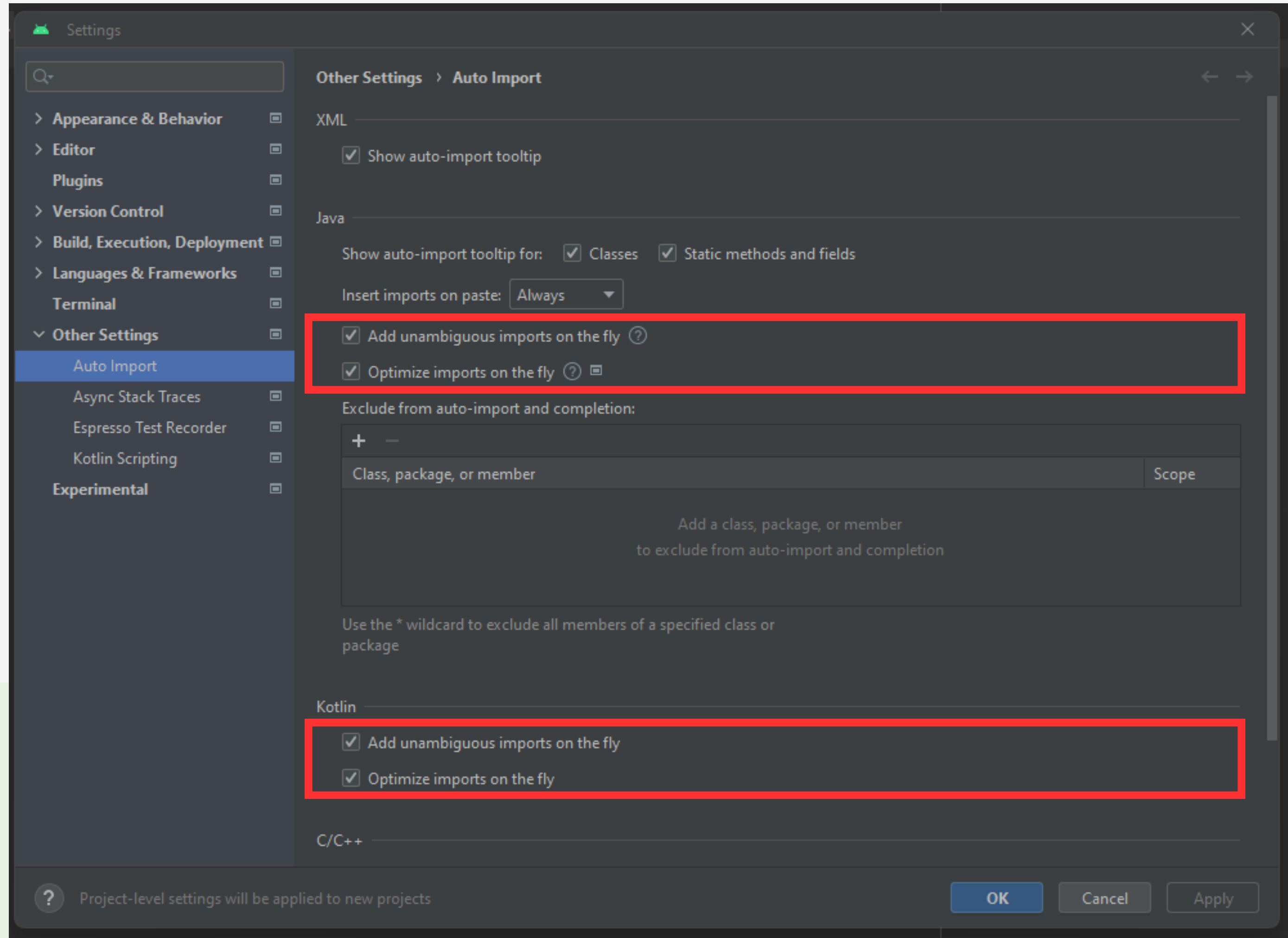
Primeira App - Configurações

- Para que todos os imports sejam feitos de forma automáticos podemos ativar a opção em:
 1. Menu File -> New Project Settings -> Preferences for New Projects
 2. Other Settings -> Auto Import
 3. Em Kotlin marcar as opções (opcional para Java)
 - a. Add unambiguous imports on the fly
 - b. Optimize on the fly

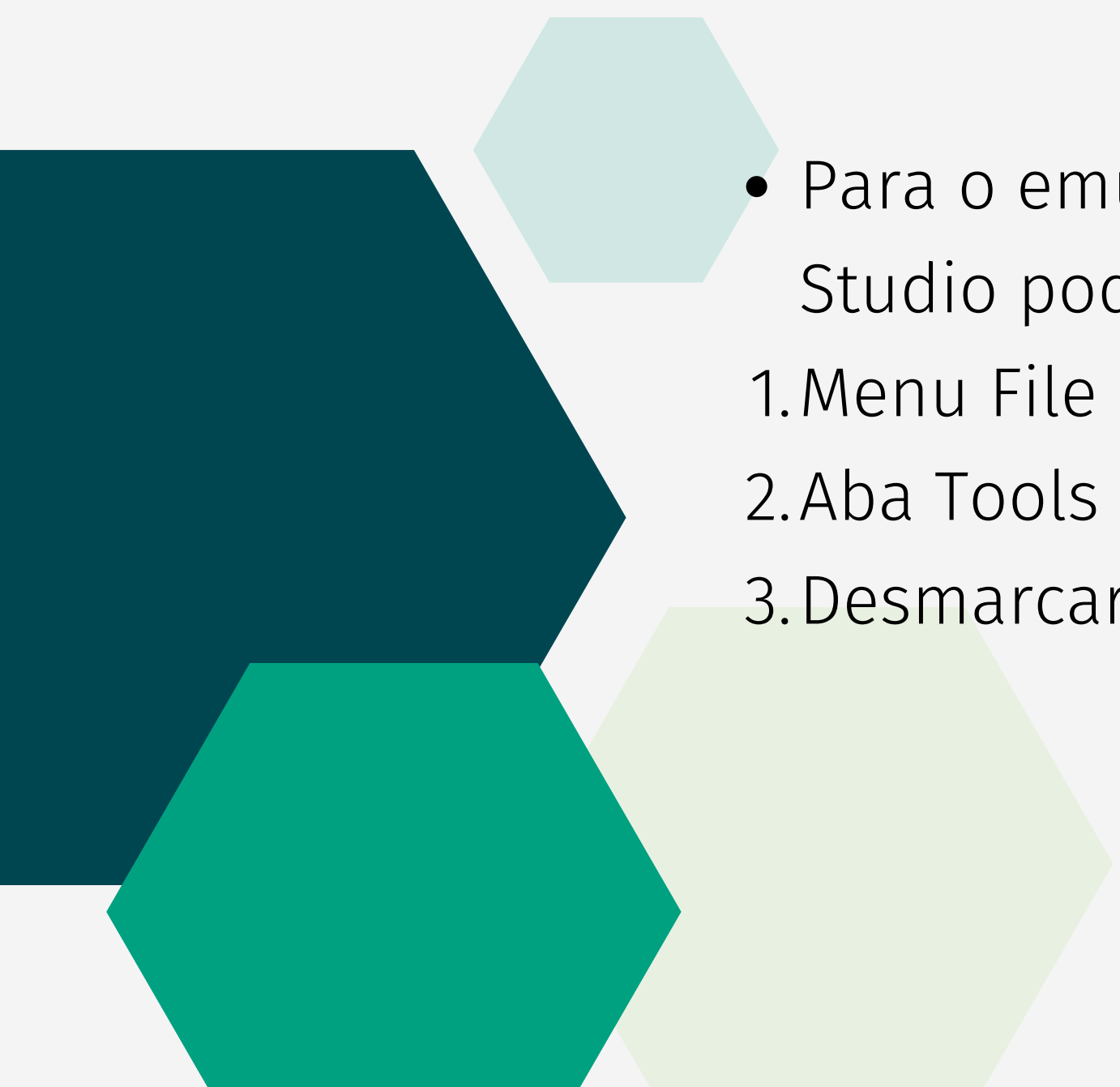
Primeira App



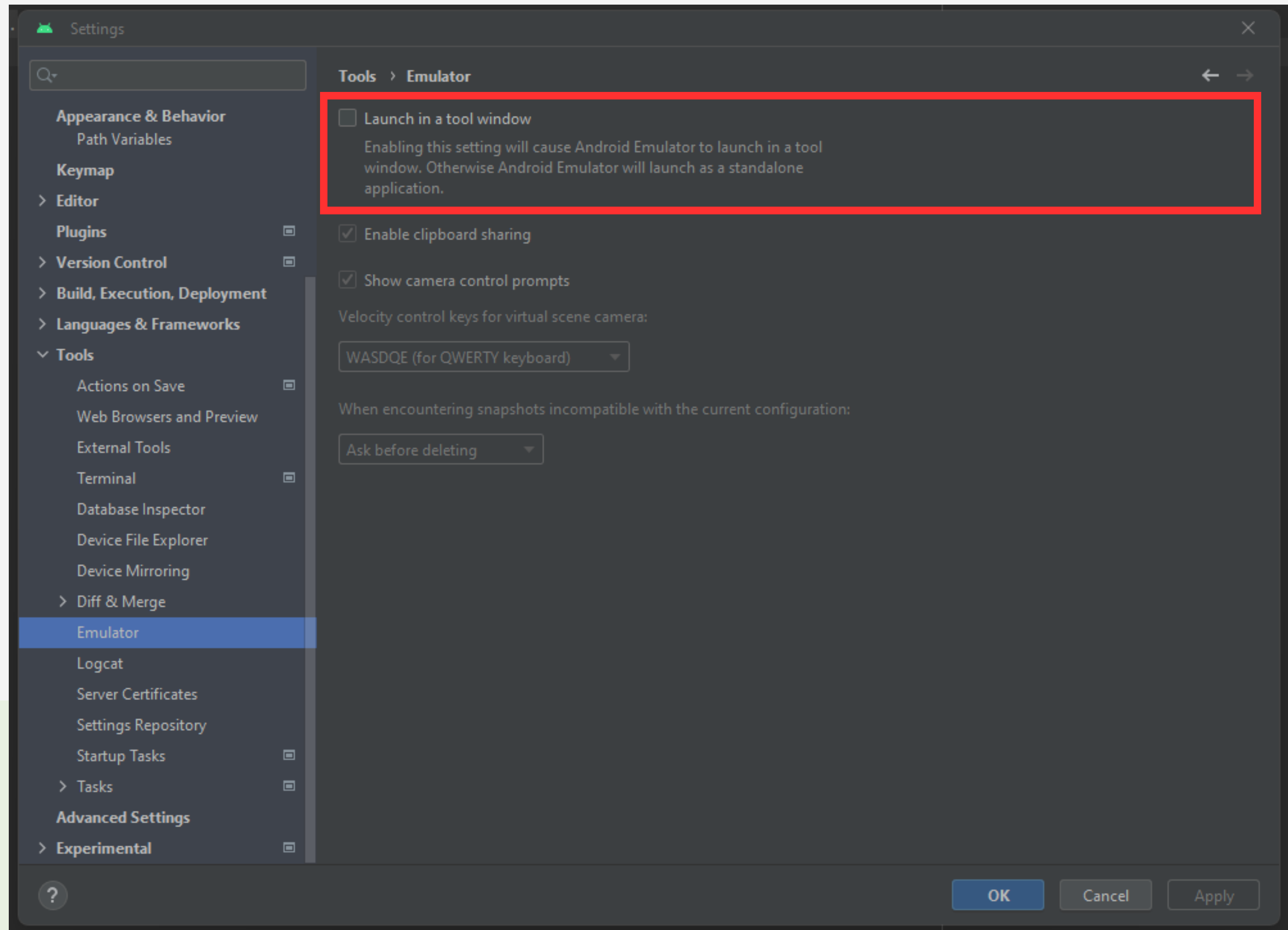
Primeira App



Primeira App - Configurações

- 
- Para o emulador ser apresentado numa janela a parte do Android Studio pode ser ativada a opção em:
 1. Menu File -> Preferences
 2. Aba Tools -> Emulator
 3. Desmarcar a opção “Launch in a tool window”

Primeira App



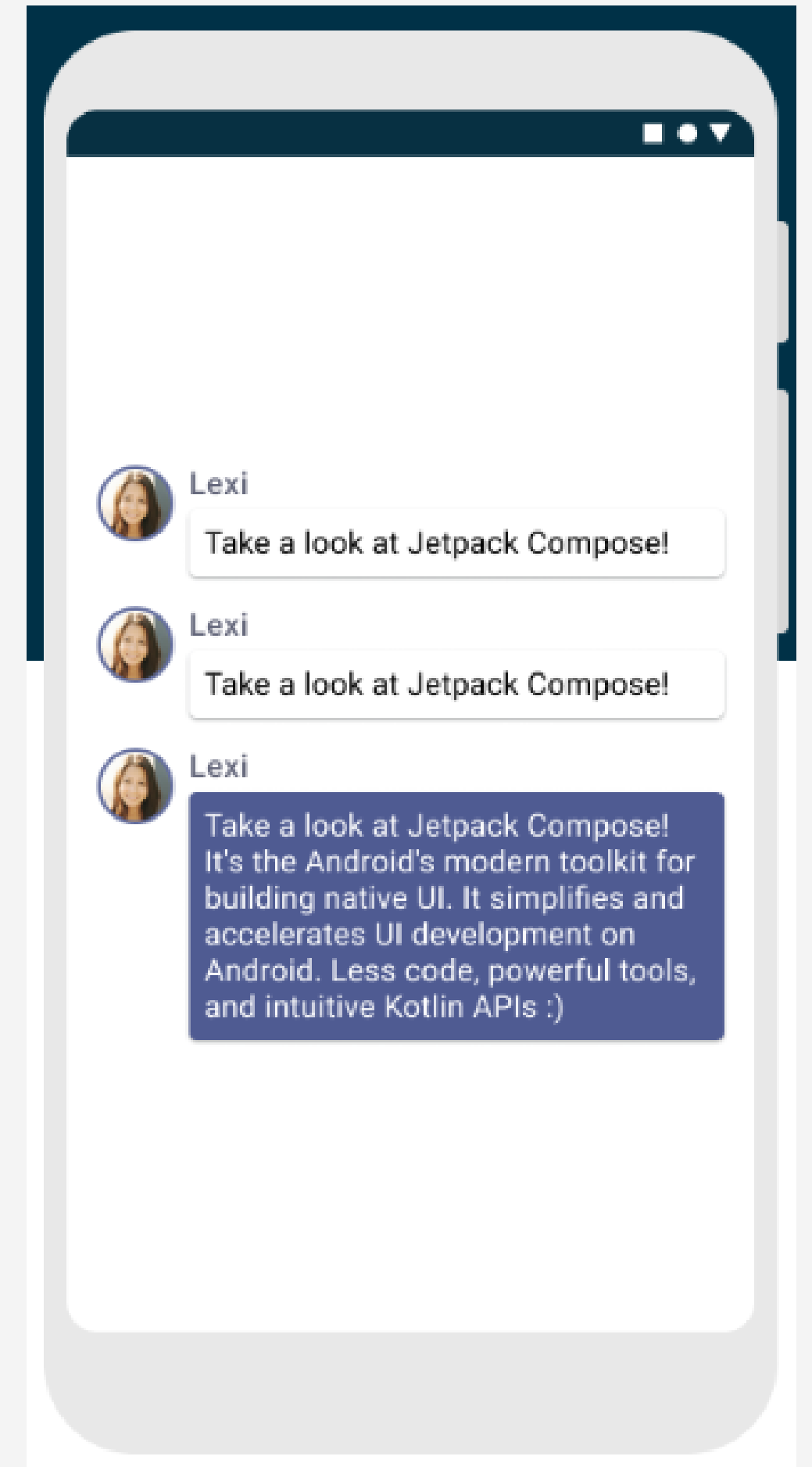
Primeira App

- Criada a aplicação é apresentado apenas um ficheiro aberto:
 - MainActivity.kt
- O MainActivity.kt é o ficheiro onde vamos programar todos os eventos e lógica da Activity.
- Mais à frente vamos verificar e perceber todos os elementos presentes no ficheiro.

Jetpack Compose

- O Jetpack Compose é um toolkit moderno para criação de IU nativa do Android. Ele simplifica e acelera o desenvolvimento de IUs no Android com menos código, ferramentas poderosas e APIs Kotlin intuitivas.
- Neste tutorial, vamos aprender a criar um componente de IU simples com funções declarativas. Você não vai editar nenhum layout XML nem usar o Layout Editor. Em vez disso, você chamará funções de composição para definir quais elementos quer usar e o compilador do Compose fará o restante.

Jetpack Compose



Jetpack Compose

O Jetpack Compose foi criado com base em funções que podem ser compostas. Essas funções permitem que você defina a IU do app de maneira programática, descrevendo as dependências de dados e de formas dela, em vez de se concentrar no processo de construção da IU (inicializando um elemento, anexando esse elemento a um pai etc.). Para criar uma função que pode ser composta, basta adicionar a anotação `@Composable` ao nome da função.

A screenshot of an Android application window. The window has a dark blue header bar at the top containing three white navigation icons: a square, a circle, and a triangle. Below the header, the text "Hello Android!" is displayed in a bold, black, sans-serif font on a white background.

Hello Android!

Adicionar um elemento de texto

Primeiro, vamos mostrar a mensagem "Hello world!" adicionando um elemento de texto ao método `onCreate`. Para fazer isso, defina um bloco de conteúdo e chame a função de composição `Text`. O bloco `setContent` define o layout da atividade em que as funções de composição são chamadas. Elas só podem ser chamadas usando outras funções desse tipo.

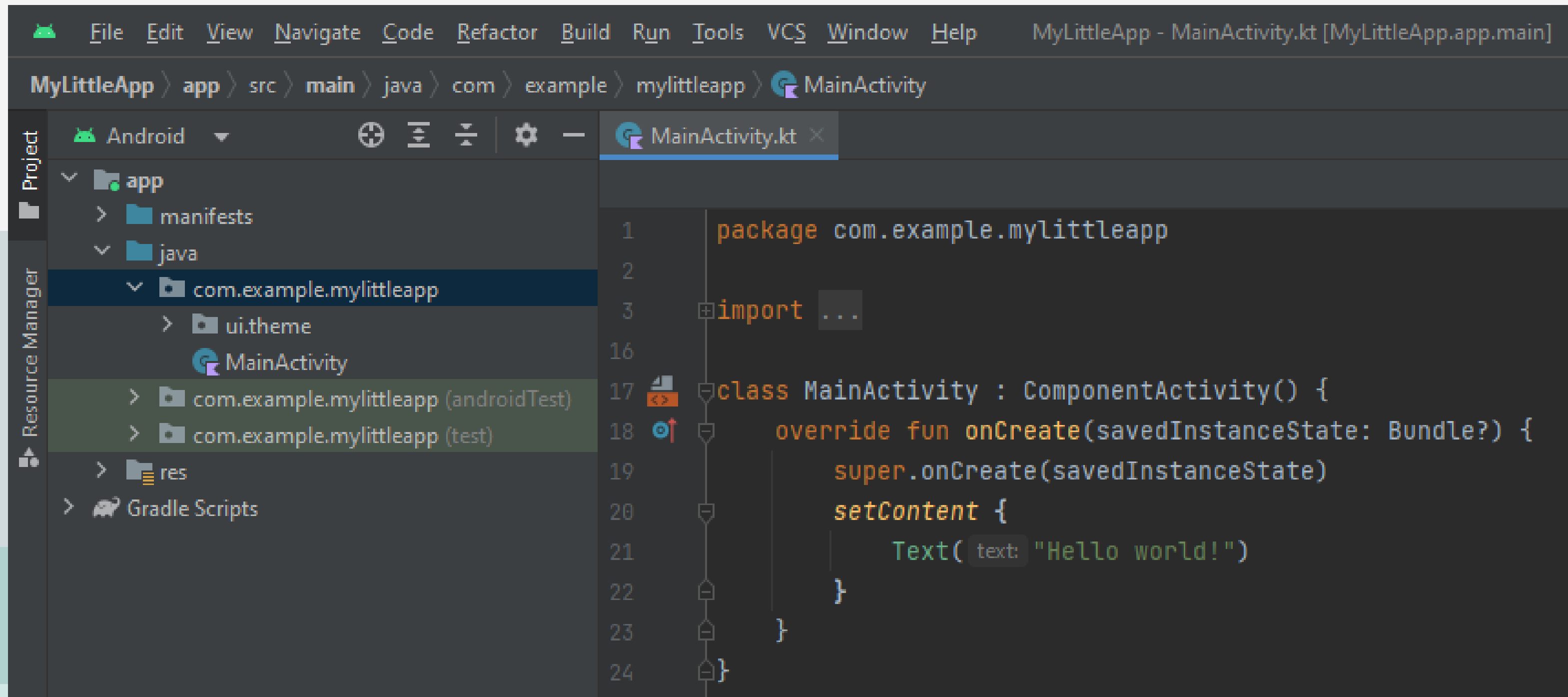
O Jetpack Compose usa um plug-in do compilador Kotlin para transformar essas funções de composição nos elementos de IU do app. Por exemplo, a função `Text` que é definida pela biblioteca de IU do Compose mostra um identificador de texto na tela.

Adicionar um elemento de texto

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.material.Text

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Text("Hello world!")
        }
    }
}
```

Adicionar um elemento de texto



```
MyLittleApp - MainActivity.kt [MyLittleApp.app.main]

MyLittleApp > app > src > main > java > com > example > mylittleapp > MainActivity

Project
└─ app
   └─ manifests
   └─ java
      └─ com.example.mylittleapp
         └─ ui.theme
            └─ MainActivity

Resource Manager
└─ com.example.mylittleapp (androidTest)
└─ com.example.mylittleapp (test)
└─ res
└─ Gradle Scripts

1 package com.example.mylittleapp
2
3 import ...
16
17 class MainActivity : ComponentActivity() {
18     override fun onCreate(savedInstanceState: Bundle?) {
19         super.onCreate(savedInstanceState)
20         setContent {
21             Text(text: "Hello world!")
22         }
23     }
24 }
```

Definir uma função que pode ser composta

Para tornar uma função composta, adicione a anotação `@Composable`. Para testar isso, defina uma função `MessageCard` que recebe um nome e o usa para configurar o elemento de texto.

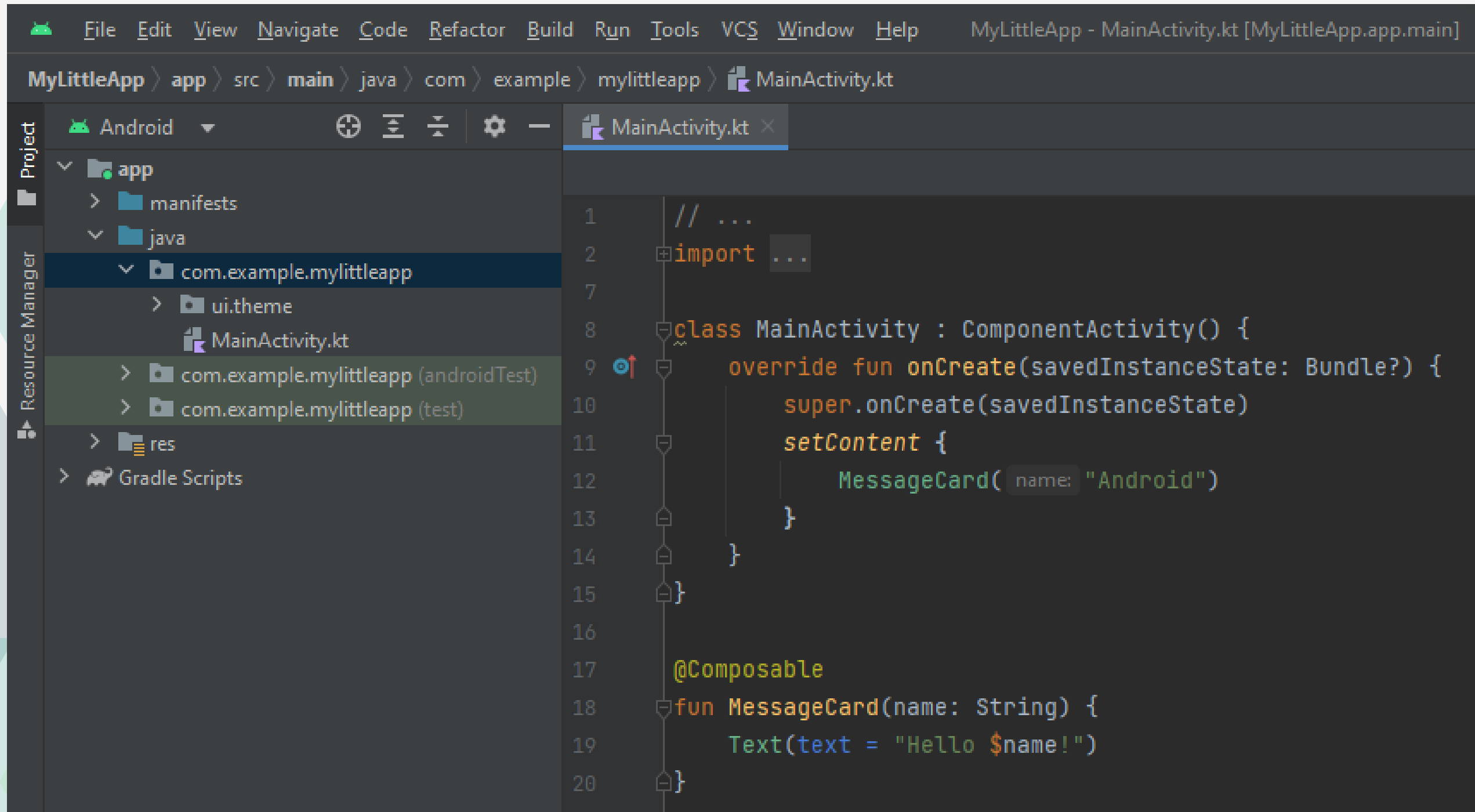
Definir uma função que pode ser composta

```
// ...
import androidx.compose.runtime.Composable

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MessageCard("Android")
        }
    }
}

@Composable
fun MessageCard(name: String) {
    Text(text = "Hello $name!")
}
```

Definir uma função que pode ser composta



```
1 // ...
2 import ...
7
8 class MainActivity : ComponentActivity() {
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContent {
12             MessageCard(name: "Android")
13         }
14     }
15 }
16
17 @Composable
18 fun MessageCard(name: String) {
19     Text(text = "Hello $name!")
20 }
```

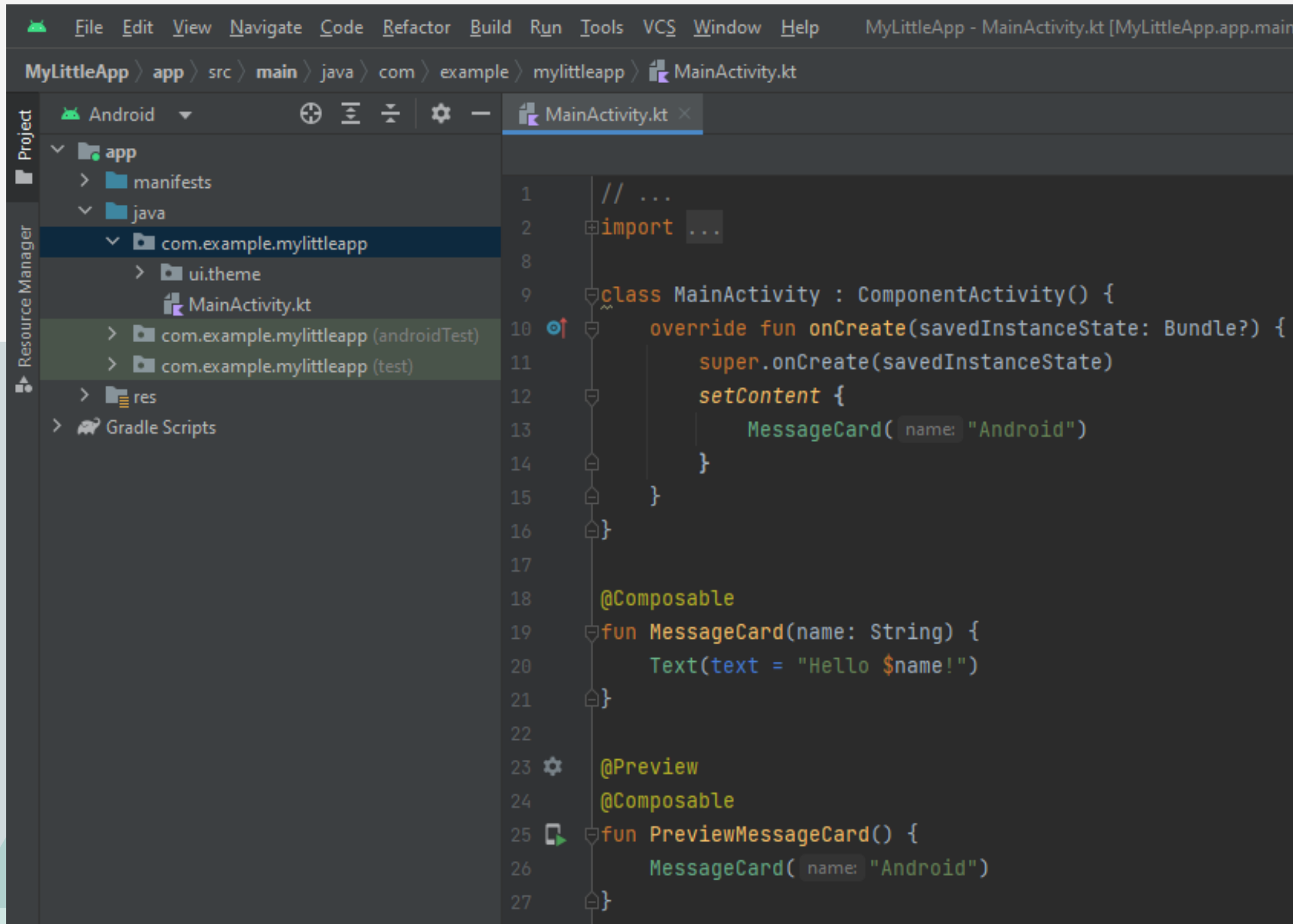
Visualizar a função no Android Studio

A anotação `@Preview` permite visualizar as funções de composição no Android Studio sem precisar criar e instalar o app em um emulador ou dispositivo Android. A anotação precisa ser usada em uma função de composição que não use parâmetros. Por esse motivo, não é possível visualizar a função `MessageCard` diretamente. Em vez disso, crie uma segunda função nomeada como `PreviewMessageCard`, que chama `MessageCard` com um parâmetro adequado. Adicione a anotação `@Preview` antes da `@Composable`.

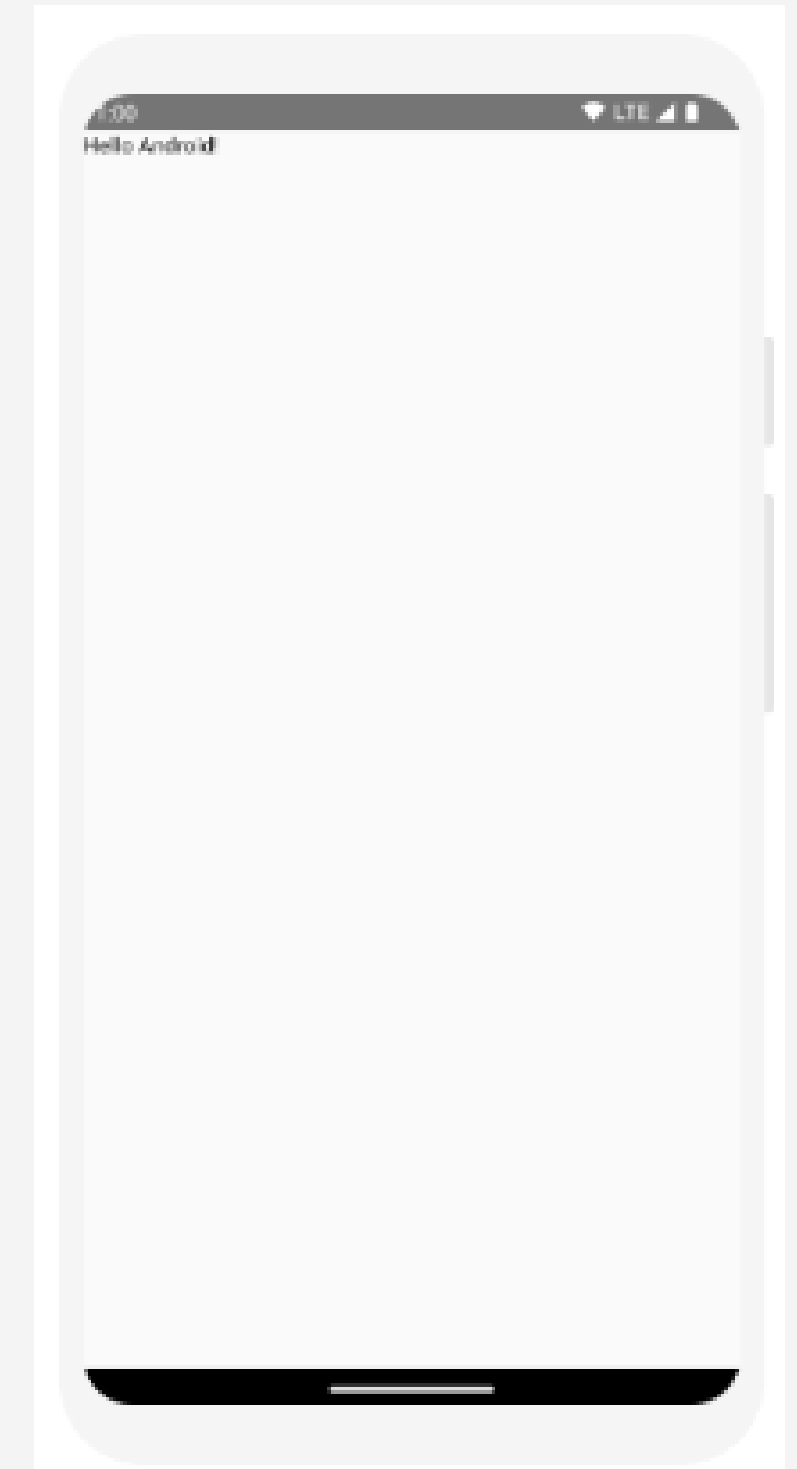
Visualizar a função no Android Studio

```
// ...  
import androidx.compose.ui.tooling.preview.Preview  
  
@Composable  
fun MessageCard(name: String) {  
    Text(text = "Hello $name!")  
}  
  
@Preview  
@Composable  
fun PreviewMessageCard() {  
    MessageCard("Android")  
}
```

Visualizar a função no Android Studio



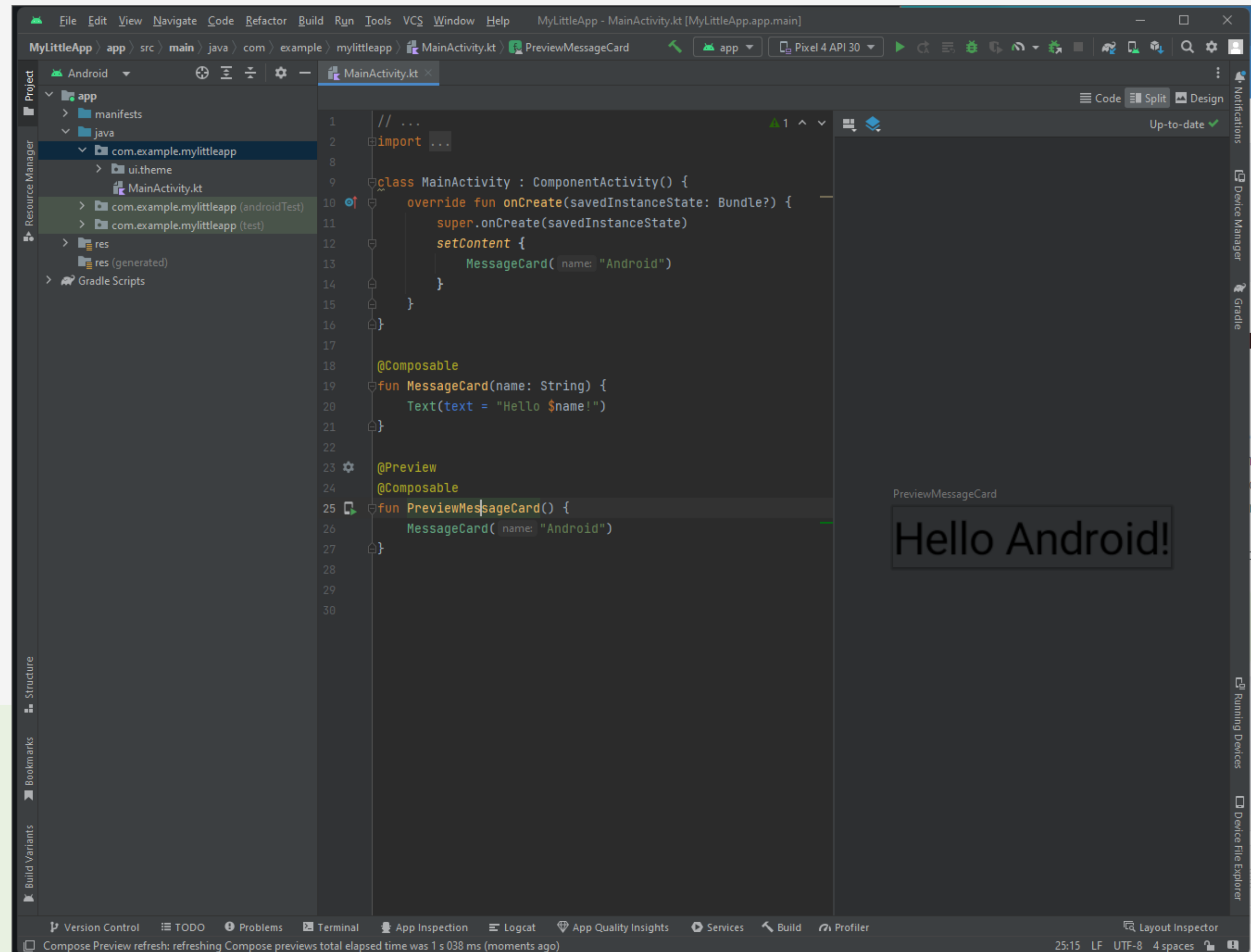
```
1 // ...
2 import ...
8
9 class MainActivity : AppCompatActivity() {
10     override fun onCreate(savedInstanceState: Bundle?) {
11         super.onCreate(savedInstanceState)
12         setContent {
13             MessageCard(name: "Android")
14         }
15     }
16 }
17
18 @Composable
19 fun MessageCard(name: String) {
20     Text(text = "Hello $name!")
21 }
22
23 @Preview
24 @Composable
25 fun PreviewMessageCard() {
26     MessageCard(name: "Android")
27 }
```



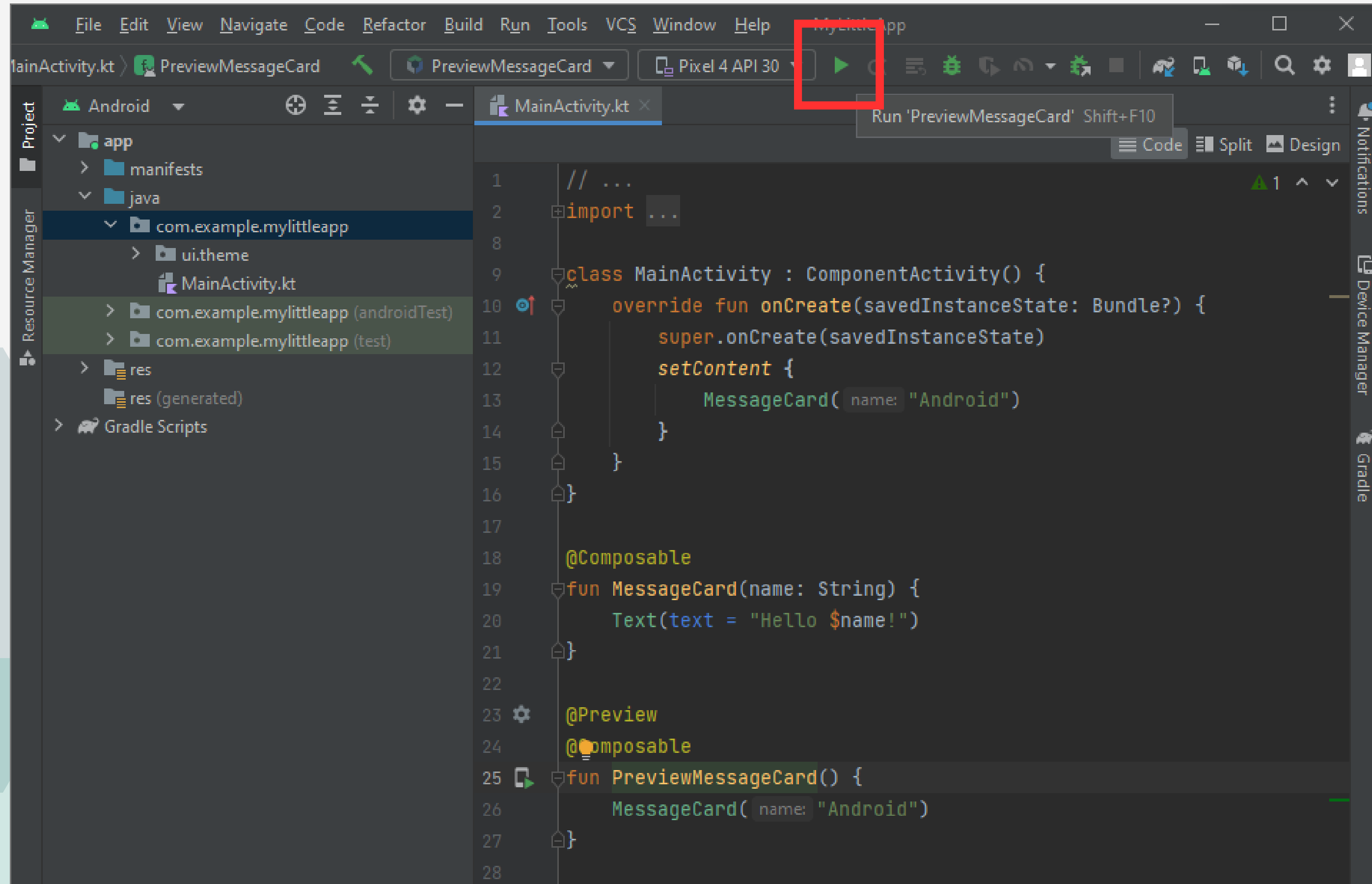
Visualizar a função no Android Studio

Recrie seu projeto. O app em si não muda, porque a nova função `PreviewMessageCard` não é chamada em nenhum lugar, mas o Android Studio adiciona uma janela de prévia que pode ser aberta clicando na visualização (de design/código) dividida. Essa janela mostra uma prévia dos elementos da IU criados por funções de composição marcadas com a anotação `@Preview`. Para atualizar as prévias a qualquer momento, clique no botão "Atualizar" na parte de cima da janela delas.

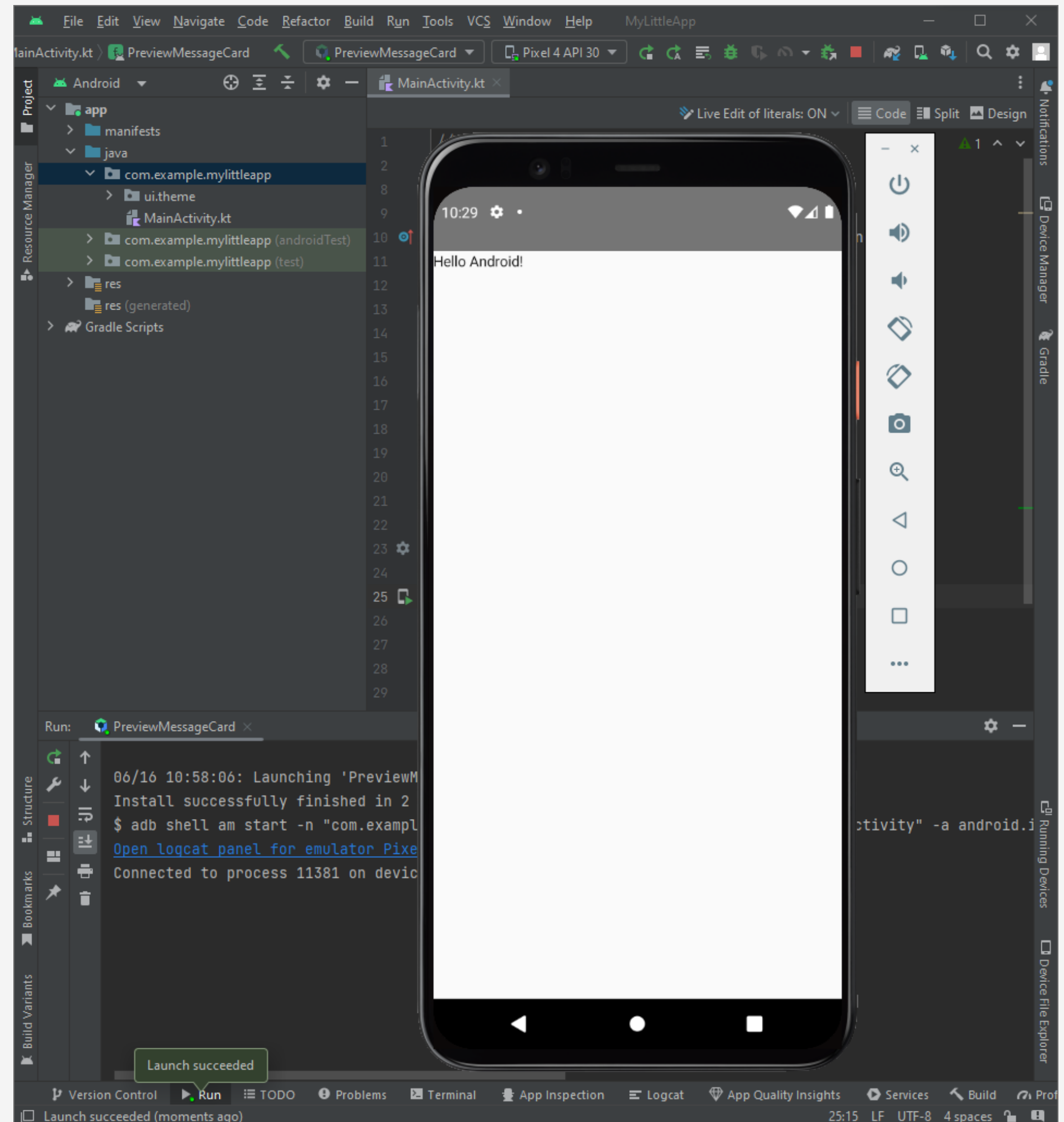
Visualizar a função no Android Studio



Executar no Emulador



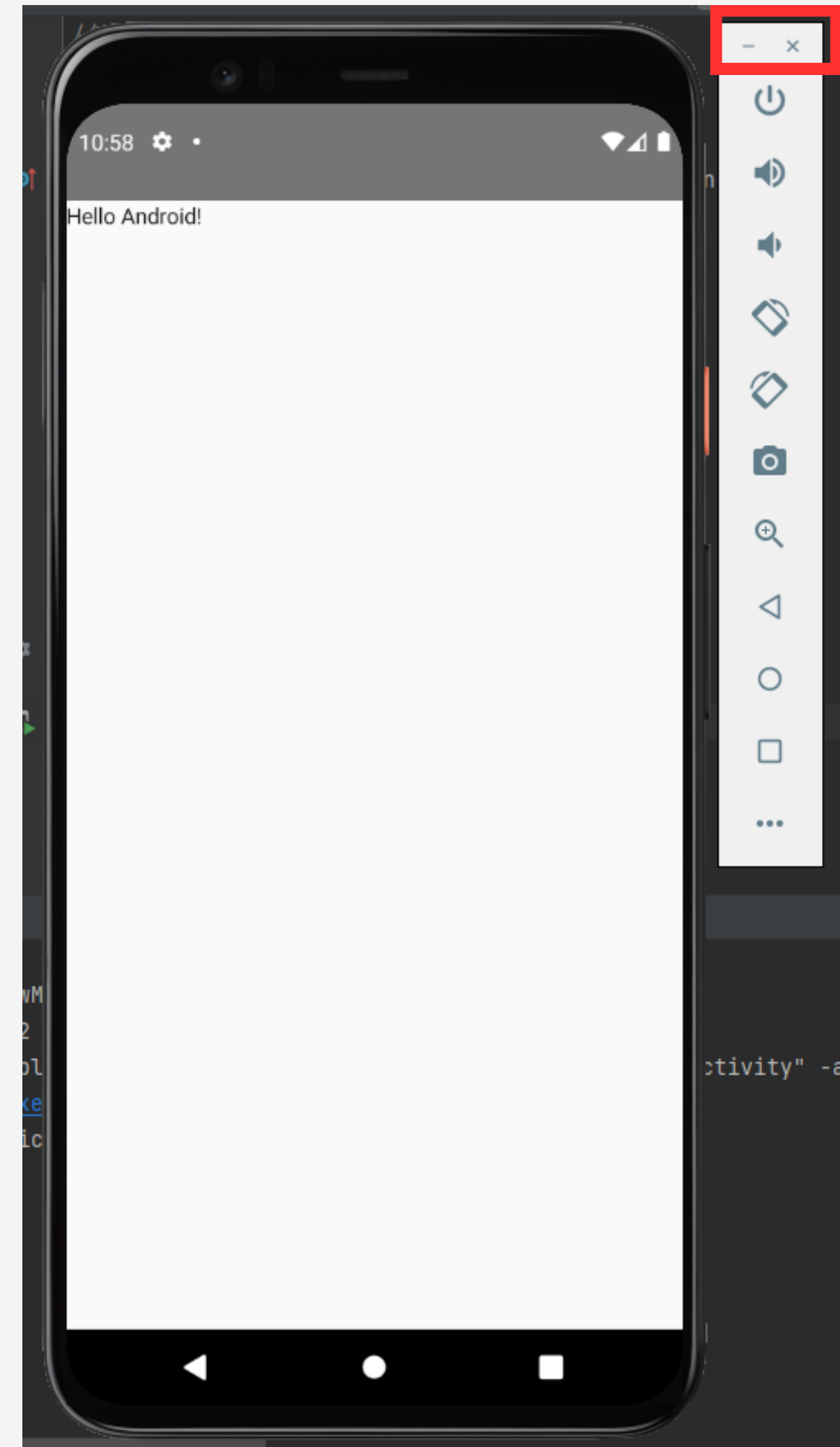
Executar no Emulador



Parar o Emulador

Por favor, evitar erros com a paragem absoluta no Emulador (Shutdown).

Usar o x no canto superior direito, não o botão Turn Off.





#R4E

Software Developer

Android

Primeiro Projeto

