

Desenvolvimento Mobile

Plataforma Android

Prof. Dr. Marcelo Otone Aguiar

Universidade Federal do Espírito Santo - UFES

5 de Novembro de 2024

Conteúdo

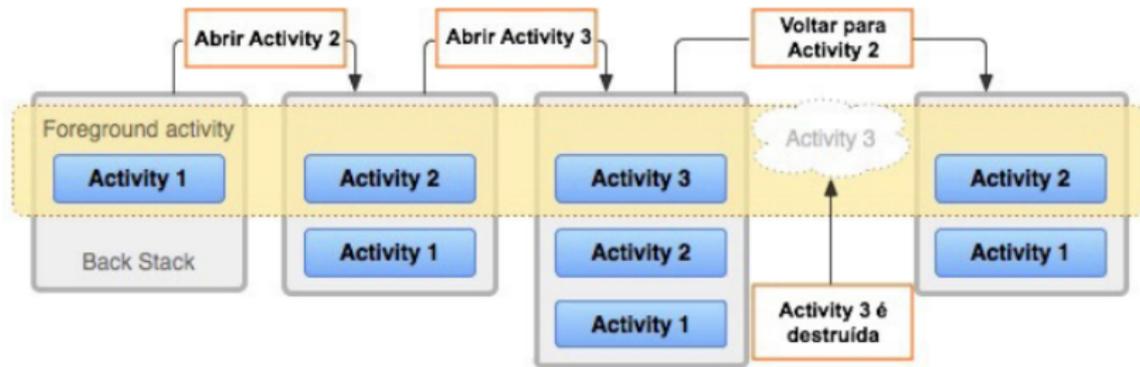
- Activities
- Preparando a Activity
- Referências

Activity

- A **Activity** é responsável pela representação gráfica da tela do **app**
- É nela que adicionamos os componentes visuais de interação com usuário, como botões, caixas de texto, de seleção, entre outros.
- É comum um **app** possuir várias **Activities**, mas nada impede de possuir apenas uma, vai depender do problema abordado
- Uma **Activity** pode iniciar outras **Activities**

Activity

- Quando uma **Activity** inicia outra, a **Activity** iniciada é empilhada sobre a principal
- A medida que o botão voltar é pressionado, as **Activities** vão sendo desempilhadas
- A imagem a seguir ilustra esse fluxo de funcionamento:



Fonte: Resende (2023)

Activity

- Um erro muito comum entre desenvolvedores que não estão habituados com esse conceito, é, ao invés de desempilhar a **Activity** atual para voltar a anterior, abrir novamente a **Activity anterior**
- E, embora, na prática isso possa até funcionar, é algo indesejado, pois, criará uma situação em que a memória RAM do dispositivo poderá sobrecarregar, uma vez que, as **Activities** não são removidas da pilha de navegação

Ativando o Recurso *View Binding*

- Um recurso disponível desde a versão **Android Studio 3.6**, denominado como *View Binding* facilita o acesso a elementos da interface gráfica (**views**) no código, evitando a necessidade de usar o método *findViewById()*
- Quando o *View Binding* está ativado, o **Android Studio** gera automaticamente uma classe de *binding* para cada arquivo de layout XML, associando-o aos elementos que ele contém
- Assim, em vez de localizar manualmente cada **view**, você pode acessar os elementos da interface com segurança por meio das propriedades da classe de *binding*
- Isso reduz a probabilidade de erros, como tentativas de acessar **views** nulas, e torna o código mais claro e seguro

Ativando o Recurso *View Binding*

Exemplo usando *findViewById()*

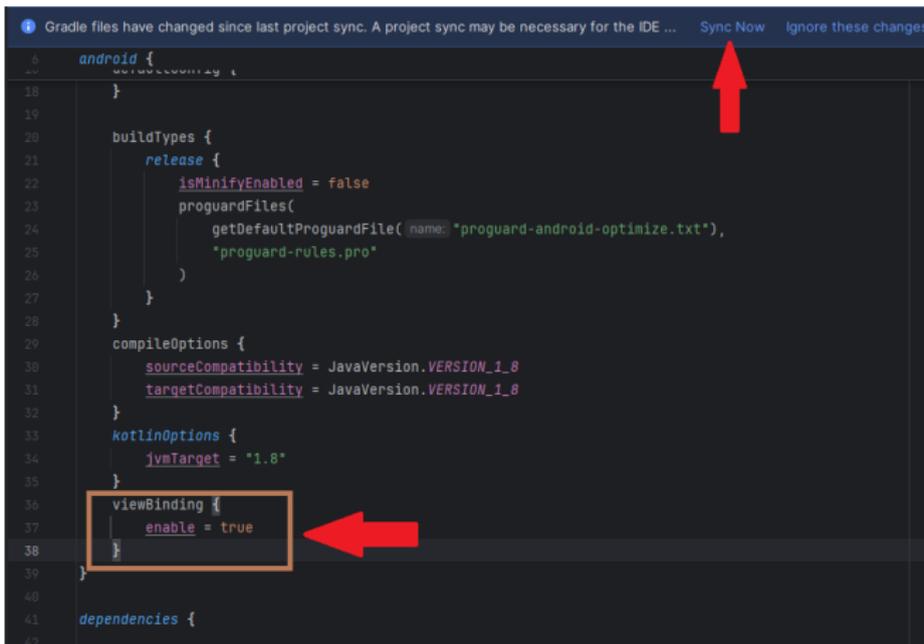
```
1  override fun onCreate(savedInstanceState: Bundle?) {  
2      super.onCreate(savedInstanceState)  
3      setContentView(R.layout.activity_main)  
4  
5      val textView: TextView = findViewById(R.id.textView)  
6      val button: Button = findViewById(R.id.button)  
7  
8      textView.text = "Hello , World!"
```

Exemplo usando *View Binding*

```
1  private lateinit var binding: ActivityMainBinding  
2  
3  override fun onCreate(savedInstanceState: Bundle?) {  
4      super.onCreate(savedInstanceState)  
5      binding = ActivityMainBinding.inflate(layoutInflater)  
6      setContentView(binding.root)  
7  
8      binding.textView.text = "Hello , World!"
```

Ativando o Recurso *View Binding*

- Para ativar, abra o arquivo: `build.gradle.kts` (Module:app)
- Adicione a configuração conforme a imagem a seguir e clique em **Sync Now**



The screenshot shows the `build.gradle.kts` file for the app module. The `viewBinding` block is highlighted with an orange box, and the `enable = true` line is pointed to by a red arrow. Another red arrow points to the `Sync Now` button in the top right corner of the IDE interface.

```
6  android {
7  }
8  }
9
10
11
12
13
14
15
16
17
18 }
19
20 buildTypes {
21     release {
22         isMinifyEnabled = false
23         proguardFiles(
24             getDefaultProguardFile("proguard-android-optimize.txt"),
25             "proguard-rules.pro"
26         )
27     }
28 }
29
30 compileOptions {
31     sourceCompatibility = JavaVersion.VERSION_1_8
32     targetCompatibility = JavaVersion.VERSION_1_8
33 }
34
35 kotlinOptions {
36     jvmTarget = "1.8"
37 }
38
39 viewBinding {
40     enable = true
41 }
42
43
44
45
46
47
48
49
50
51 dependencies {
```

Referências

- RESENDE, K. **Kotlin com Android**: Crie aplicativos de maneira fácil e divertida. São Paulo: Casa do Código, 2023.