



Reskilling 4Employment Software Developer

Acesso móvel a sistemas de informação

Bruno Santos

bruno.santos.mcv@msft.cesae.pt

Tópicos

- Convenções de Código
- Ferramenta de Debug
- Toast
- Ciclos e condições em Java
- Criação de novas Activity
- Intent
- SplashScreen

Convenções

- A utilização de ; é opcional mas desencorajada no Kotlin

Convenções

- O uso de camelCase deve ser utilizado em
 - Variáveis
 - Métodos
 - Atributos de classes
 - Parâmetros

Convenções

- Para constantes (const) ou variáveis definidas fora de classes deve escritas em maiúsculas separadas por _

```
const val ITEM_KEY = "TYX"  
val MAX_ITEMS: Int = 8  
  
class Course (val rating: Float) {  
    private val courseName = "Kotlin"
```

Convenções

- Nas classes deve ser utilizado o padrão PascalCase com a primeira letra maiúscula.
- Para construtores com poucos parâmetros podem ser escritos em apenas uma linha, caso contrário deve ser colocado um parâmetro por linha.

```
class Person(val name: String)
```

```
class Course(  
    val title: String,  
    val students: Int,  
    val rating: Float  
)
```

Convenções

- A organização de uma classe deve ser:
 - Variáveis de contexto da classe e inicializações
 - Construtores
 - Overrides
 - Métodos com sobrecarga

Convenções

- Chavetas devem ser usadas sempre em
 - if
 - else
 - while
 - for
 - do...while
 - try...catch
 - Funções
 - Classes

```
if (condicao) {  
} else {  
}
```

```
try {  
} catch (e: Exception) {  
} finally {  
}
```

```
while (condicao) {  
}
```

```
for (l in listOf(1, 2)) {  
}
```


Convenções

- Funções que não retornam valor devem ser apresentadas sem informação de retorno.
- Funções que retornam valor devem apresentar o tipo de dados retornado

```
fun noReturn() {  
    println("Sem retorno")  
}
```

```
fun sayMyName(name: String): String {  
    return "Hello, $name"  
}
```

Convenções

- Nomes de ficheiros de layout devem ser prefixados com o elemento que representam

<i>Componente</i>	<i>Classe</i>	<i>Nome do layout</i>
Activity	UserProfileActivity	activity_user_profile
Fragment	SignUpFragment	fragment_sign_up
Dialog	ChagePasswordDialog	dialog_change_password

Convenções

- Quando um elemento não tiver nenhum conteúdo dentro das tags deve-se usar self closing tags.

```
<TextView android:textSize="32sp" />
```

Convenções

- Identificadores de elementos são escritos em minúscula_com_underline

<i>Elemento</i>	<i>Prefixo</i>
TextView	text_
EditText	edit_
Button	button_
ImageView	image_
Menu	menu_

```
<TextView android:id="@+id/text_name" />  
  
<EditText android:id="@+id/edit_cupom_price" />  
  
<ImageView android:id="@+id/image_pets" />
```

Debug

- A ferramenta de Debug permite ao programador pausar a execução do projeto em tempo real e perceber o estado da mesma.
- Esta ferramenta permite, por exemplo, verificar o valor de uma variável em tempo de execução.
- Para utilizar a ferramenta de Debug é necessário criar breakpoints ao longo do código clicando imediatamente à direita do número da linha.

Debug

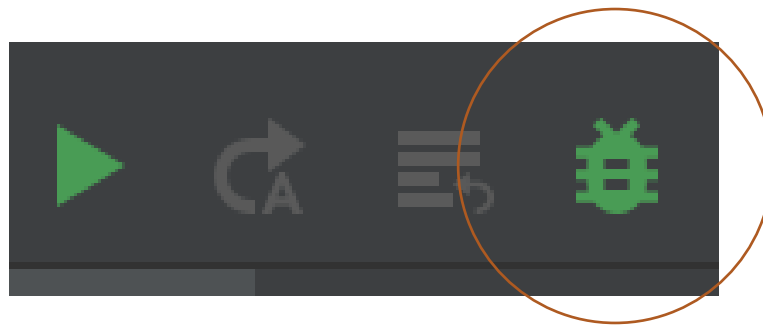
```
16  ding.buttonDolar.setOnClickListener
17      val euros = binding.editValorEuros
18      val dolares: Double = String.forma
```

```
16  ding.buttonDolar.setOnClickListener
17  ●  val euros = binding.editValorEuros
18      val dolares: Double = String.fo
```

Clicar aqui

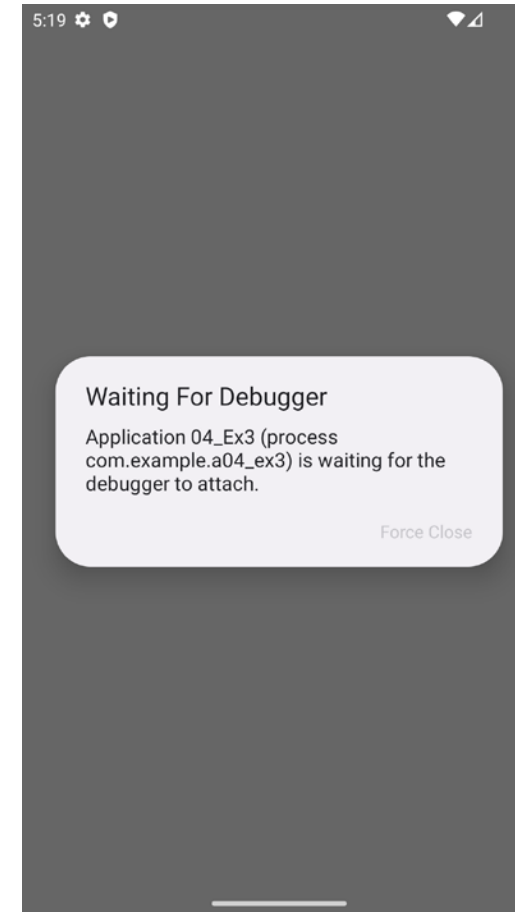
Debug

- Ao executar a aplicação quando o programa chegar à linha de código selecionada irá parar.
- Para correr a aplicação em modo de Debug é necessário clicar no botão respetivo na barra de tarefas:



Debug

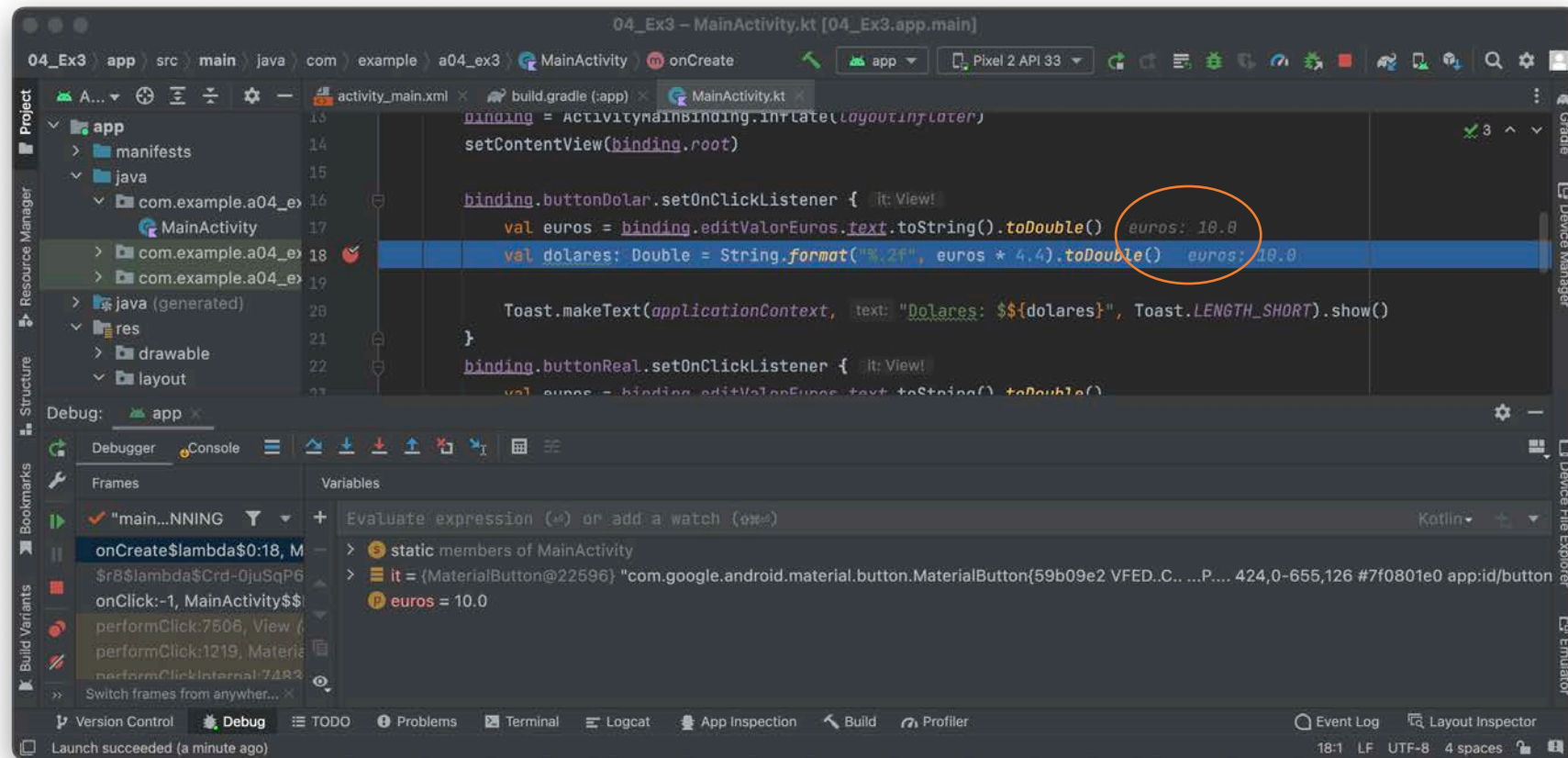
- Ao executar a aplicação em modo Debug recebemos a mensagem de início.



Debug

- Quando chegamos à linha em que está o primeiro breakpoint o programa é pausado e podemos ver os valores das variáveis colocando o rato em cima das mesmas

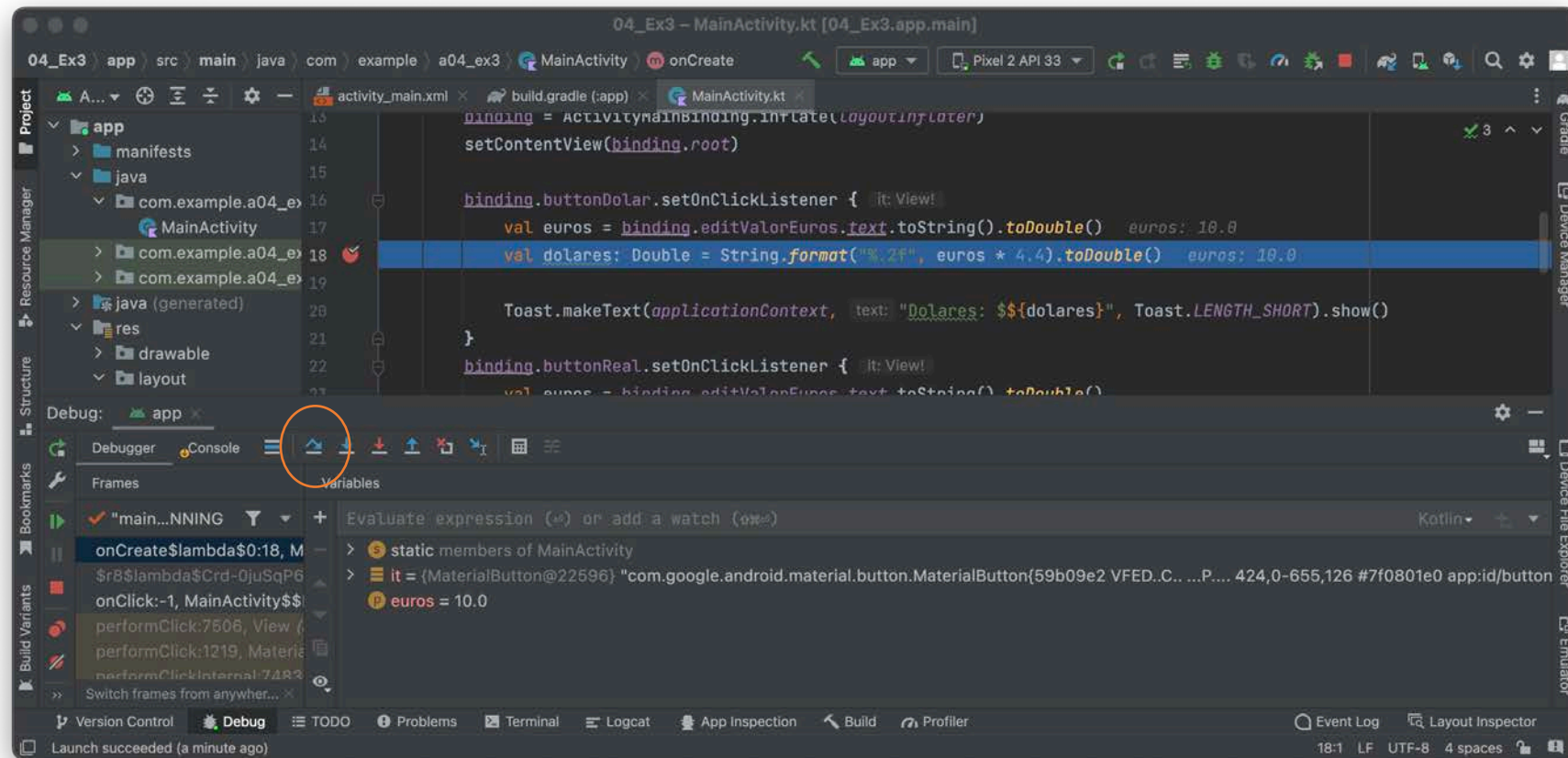
Debug



Debug

- Clicando no botão de Step Over (F8) avançamos o programa linha a linha.

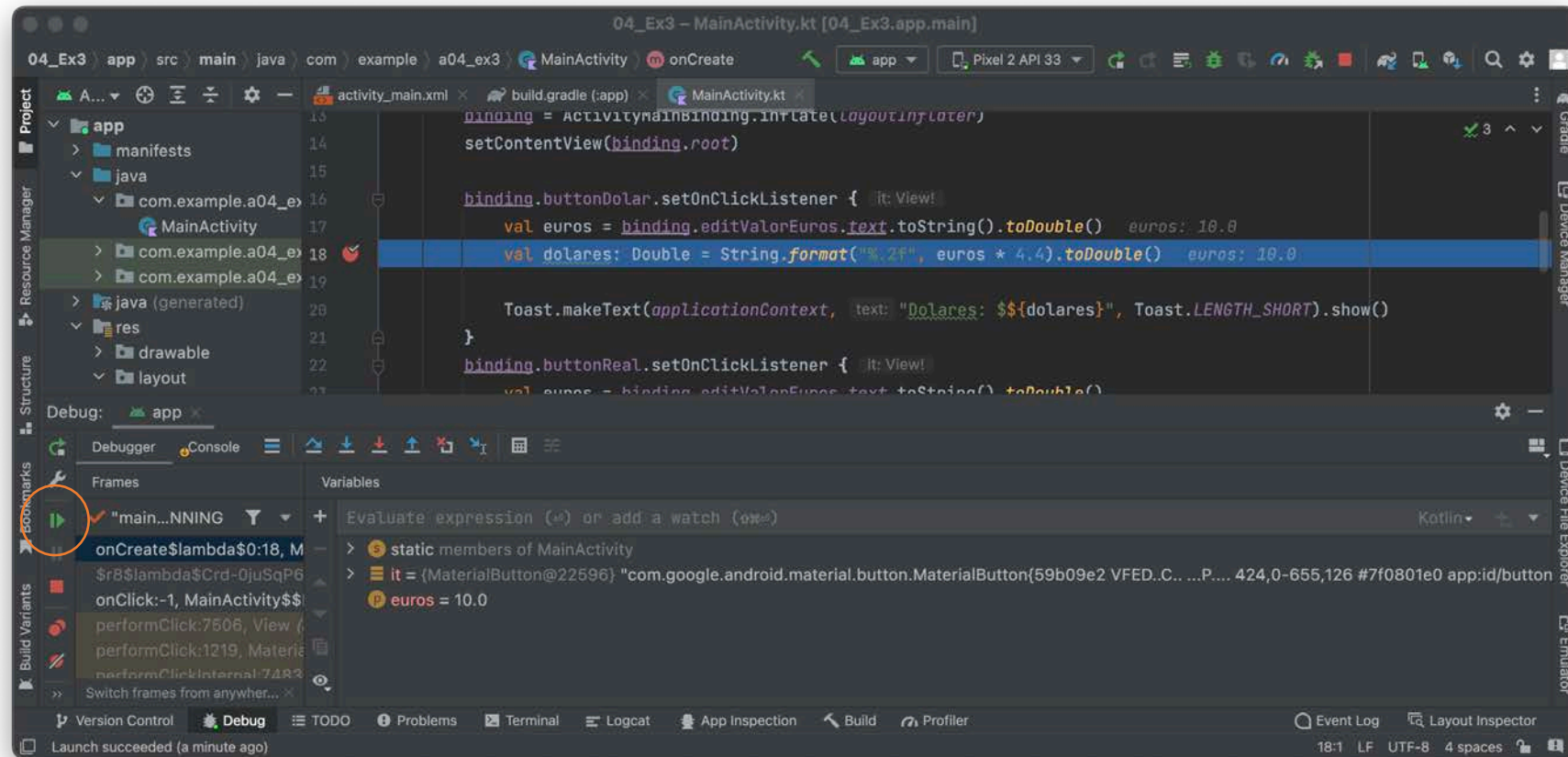
Debug



Debug

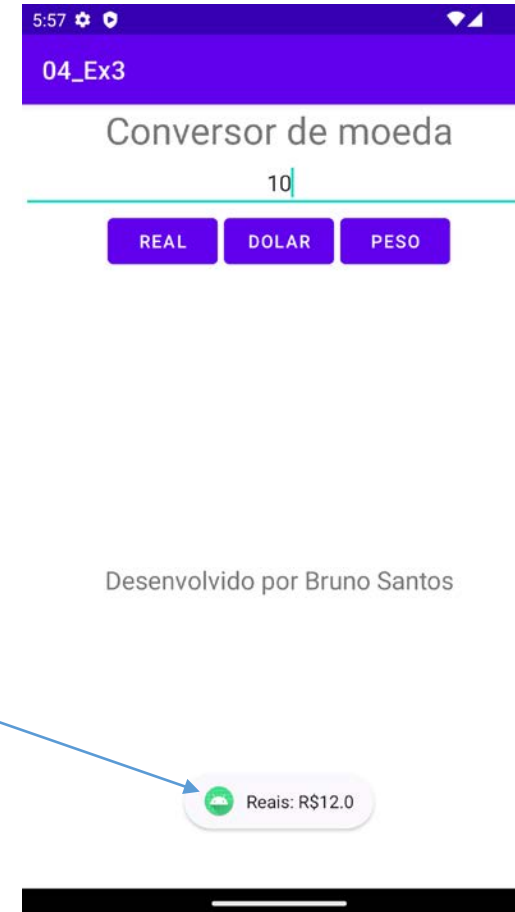
- Clicando no botão Resume Program (F9) a execução do programa é retomada normalmente até ao próximo ponto de Debug encontrado.
- Clicando no botão Stop (Ctrl + F2) o programa é parado.

Debug



Toast

- O Toast é um balão que surge como pop-up temporário na parte inferior do ecrã com uma mensagem definida pelo utilizador.



Toast

- Para criar um Toast é utilizado o seguinte código:

```
Toast.makeText(applicationContext, "Isto é um Toast", Toast.LENGTH_SHORT).show()
```

- Importante:
 - applicationContext informa a Activity onde está a ser executado o código;
 - “Isto é um Toast” é a mensagem que vai ser apresentada;
 - Toast.LENGTH_SHORT é o tempo que irá ser apresentada, podemos escolar entre:
 - Toast.LENGTH_LONG
 - Toast.LENGTH_SHORT

Ciclos

```
for (i in 1 ≤ .. ≤ 3) {  
    println(i)  
}  
  
for (i in 6 ≥ downTo ≥ 0 step 2) {  
    println(i)  
}  
  
var ints = arrayOf(1,2,3)  
for (item: Int in ints) {  
    // ...  
}  
  
var nomes = arrayOf("Maria", "José", "Filipa", "Carlos")  
for (nome in nomes) print(nome)
```

Ciclos

```
var i = 5
while (i < 10) {
    i++
}
```

```
var i = 5
do {
    i++
} while (i < 10)
```

Condições

```
var valor = 10
var max: Int

if (valor > 10) max = valor

if (valor > 10) {

} else {

}

if (valor > 10) {

} else if (valor < 10) {

} else {

}
```

Condições

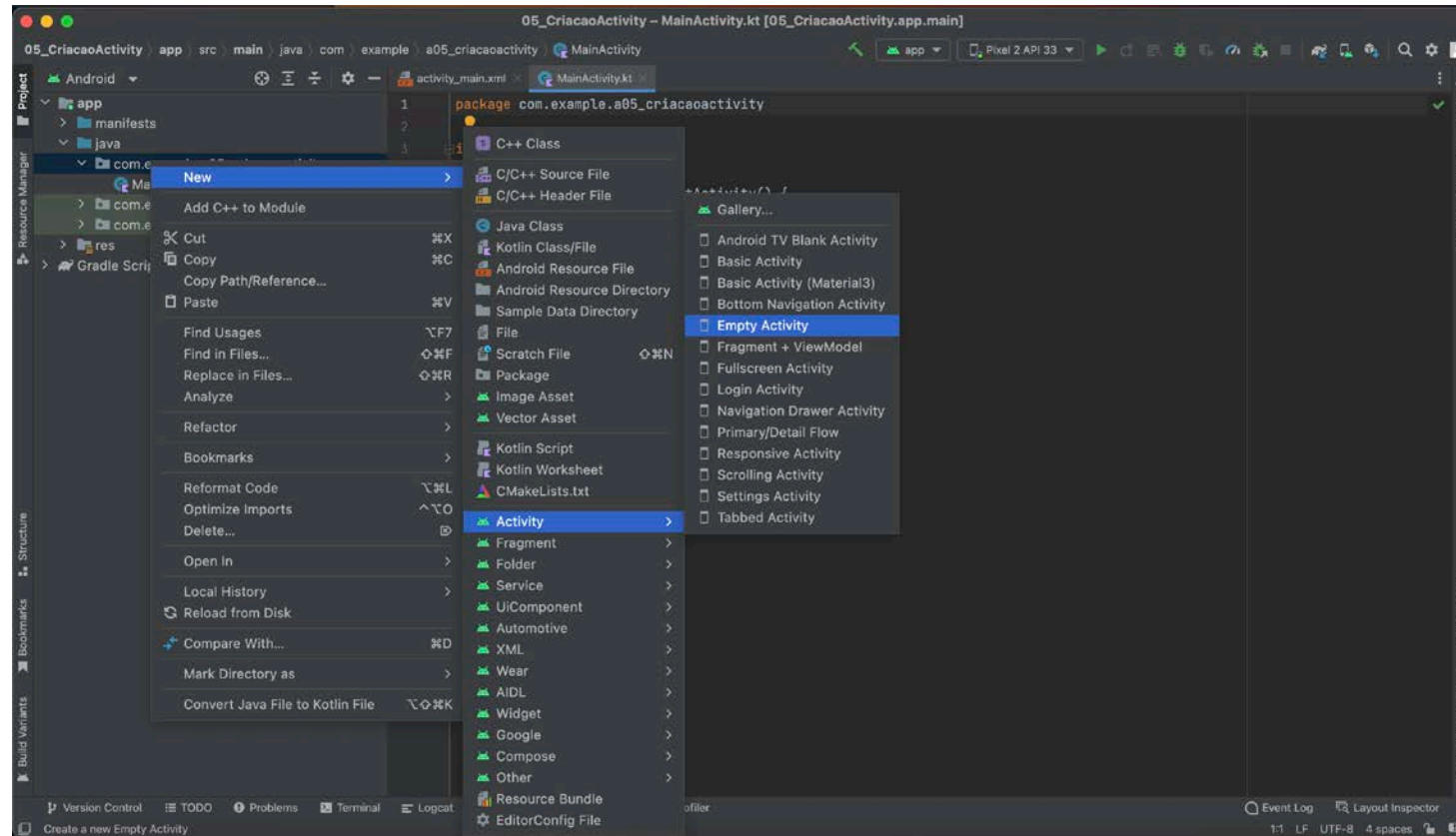
```
var x = 10

when (x) {
    1 -> print("x == 1")
    2 -> print("x == 2")
    else -> {
        print("x não é nem 1 nem 2")
    }
}
```

Criação de novas Activity

- Para criar novas Activity dentro da mesma aplicação devemos dentro da janela de Project, clicar com o botão direito do rato em cima de app, seleccionar a opção New, posteriormente Activity e finalmente o tipo de Activity pretendida.

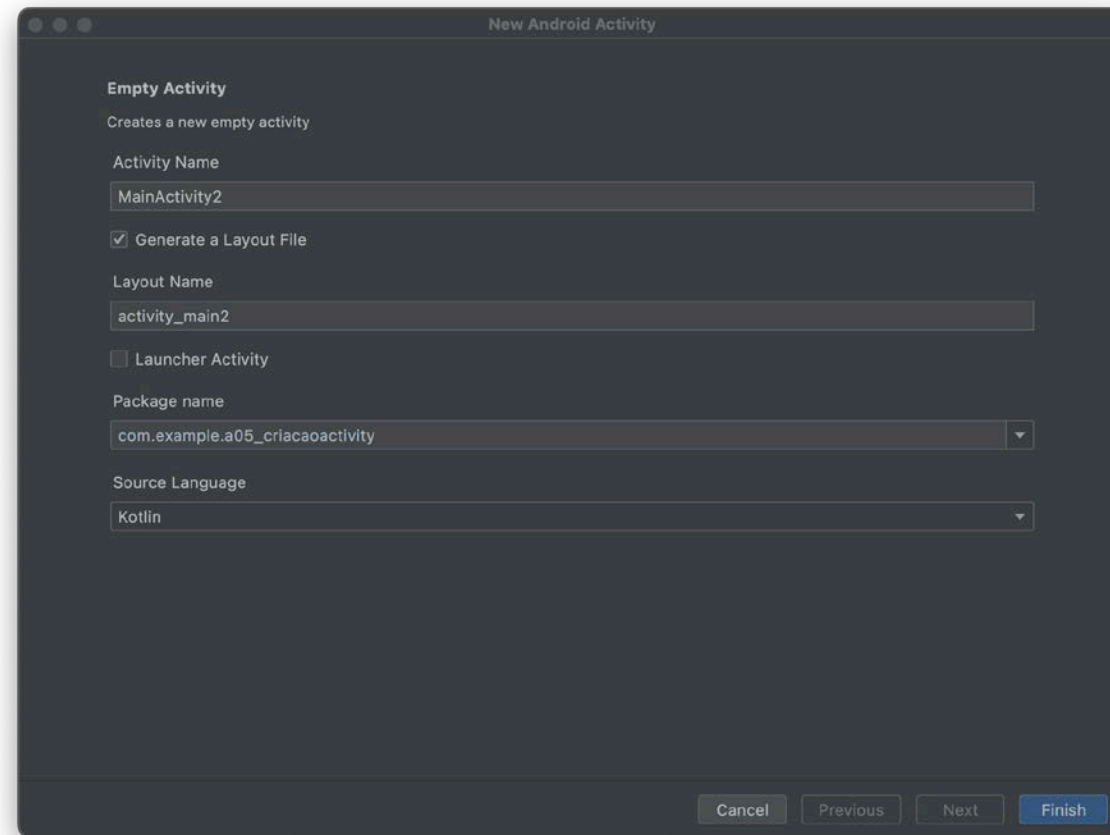
Criação de novas Activity



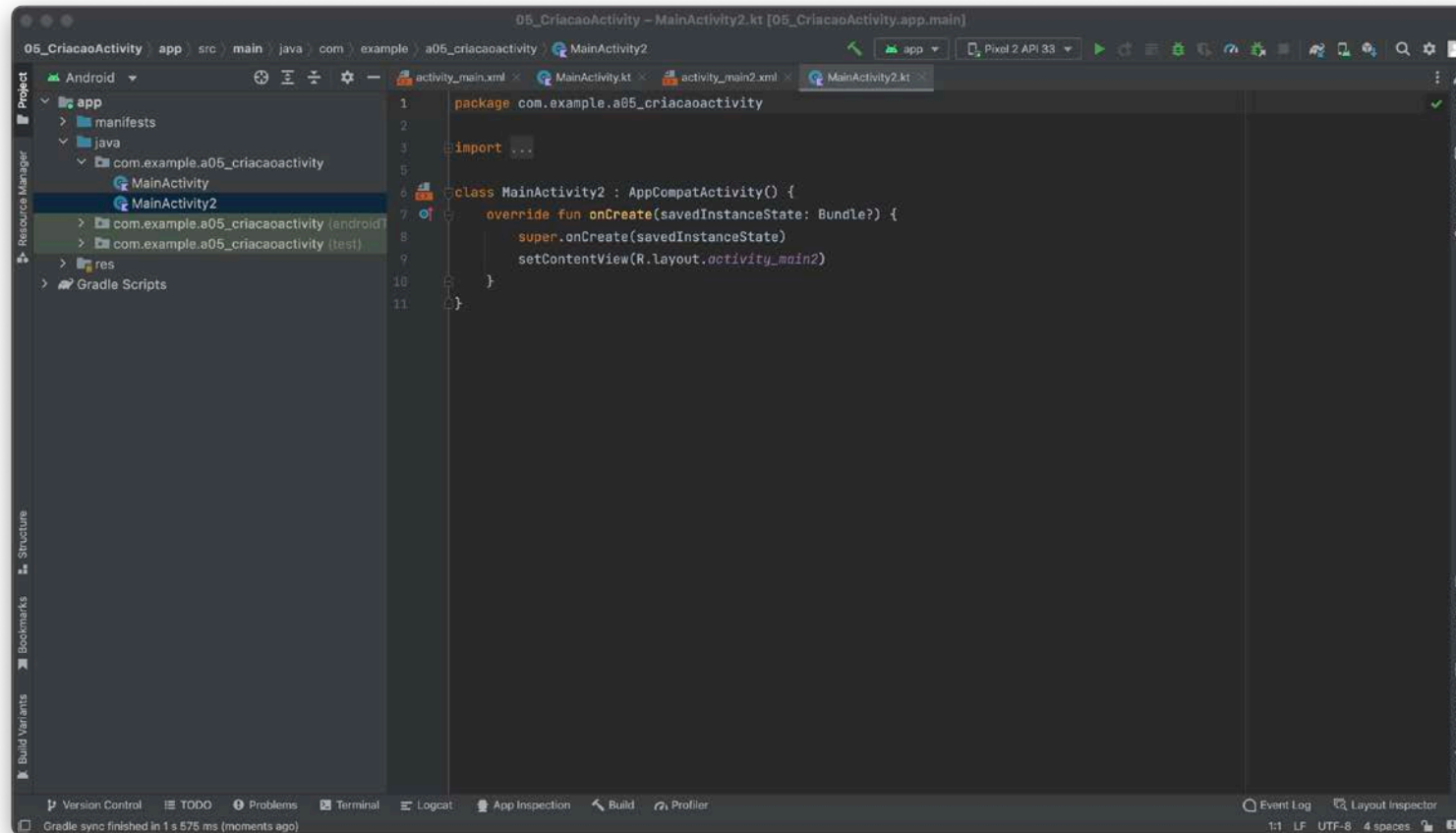
Criação de novas Activity

- Seleccionamos Empty Activity, renomeamos a nova Activity e clicamos em Finish. A nova Activity é criada.
- Juntamente com o ficheiro Kotlin é criado o ficheiro XML de layout.
- No exemplo seguinte foi dado o nome MainActivity2 à Activity:
 - Ficheiro Java: MainActivity2.kt
 - Layout: activity_main2.xml

Criação de novas Activity



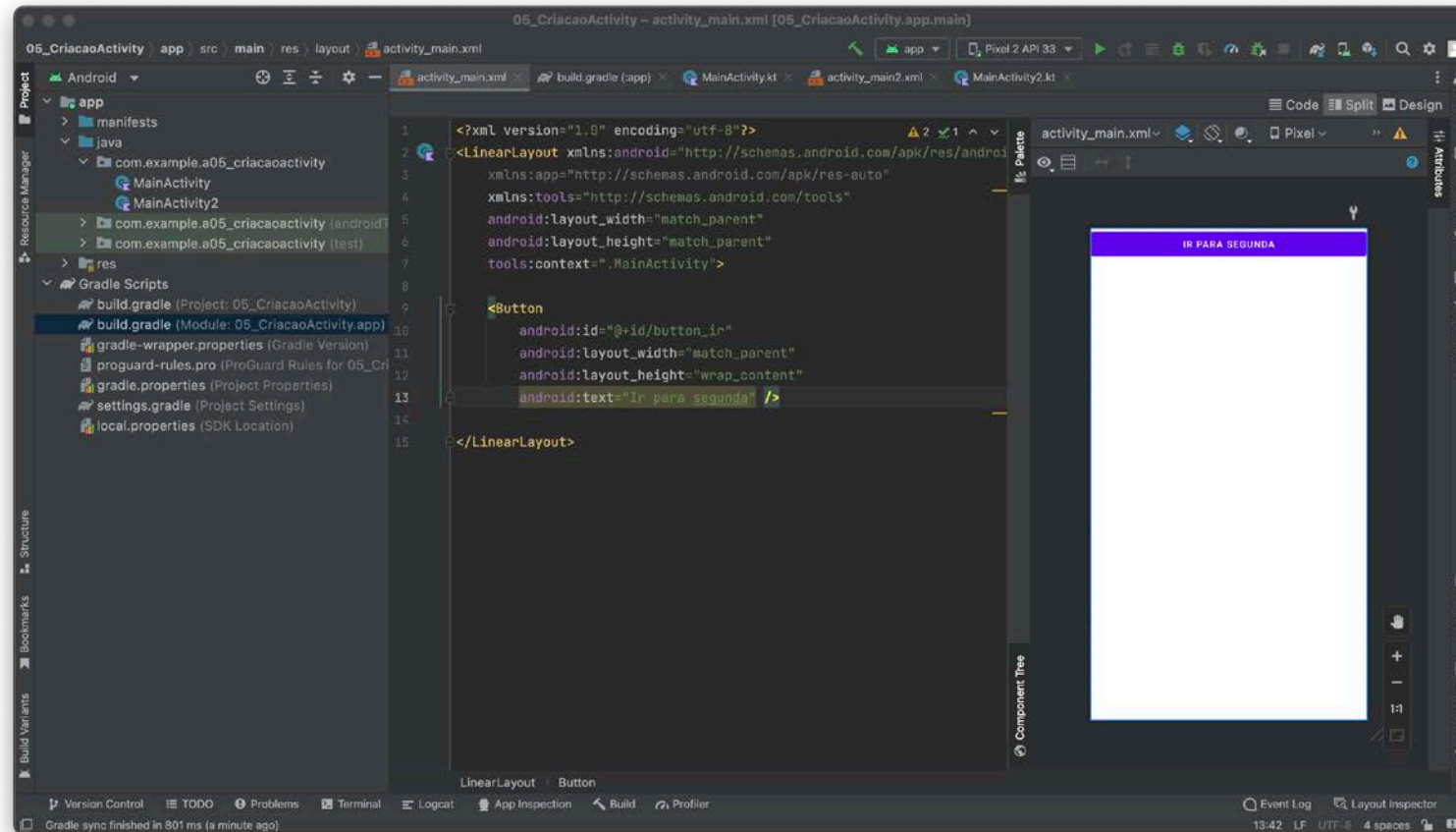
Criação de novas Activity



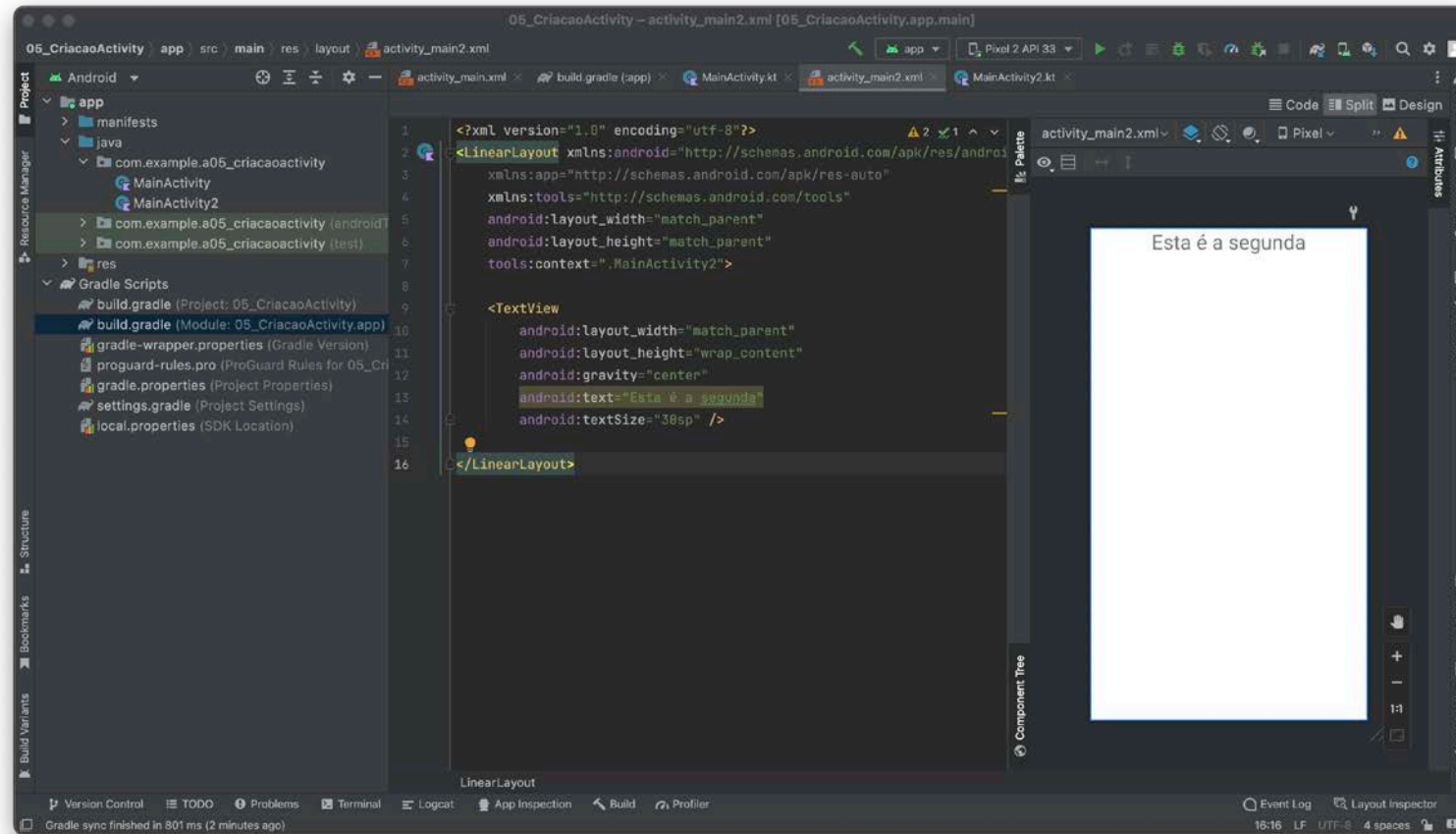
Intent

- O Intent é um elemento que permite navegar entre Activity.
- Após a criação de duas Activity vamos criar um botão na primeira Activity que quando clicado irá redirecionar para a segunda Activity.
- Na segunda Activity apenas terá uma TextView a indicar que esse é a segunda Activity

Intent



Intent



Intent

- Vamos programar o evento de clique no botão da MainActivity.
- Dentro do evento de clique vamos criar um objeto Intent e dizer-lhe que queremos ir da MainActivity para a MainActivity2.

```
val intent = Intent(this, MainActivity2::class.java)  
startActivity(intent)
```

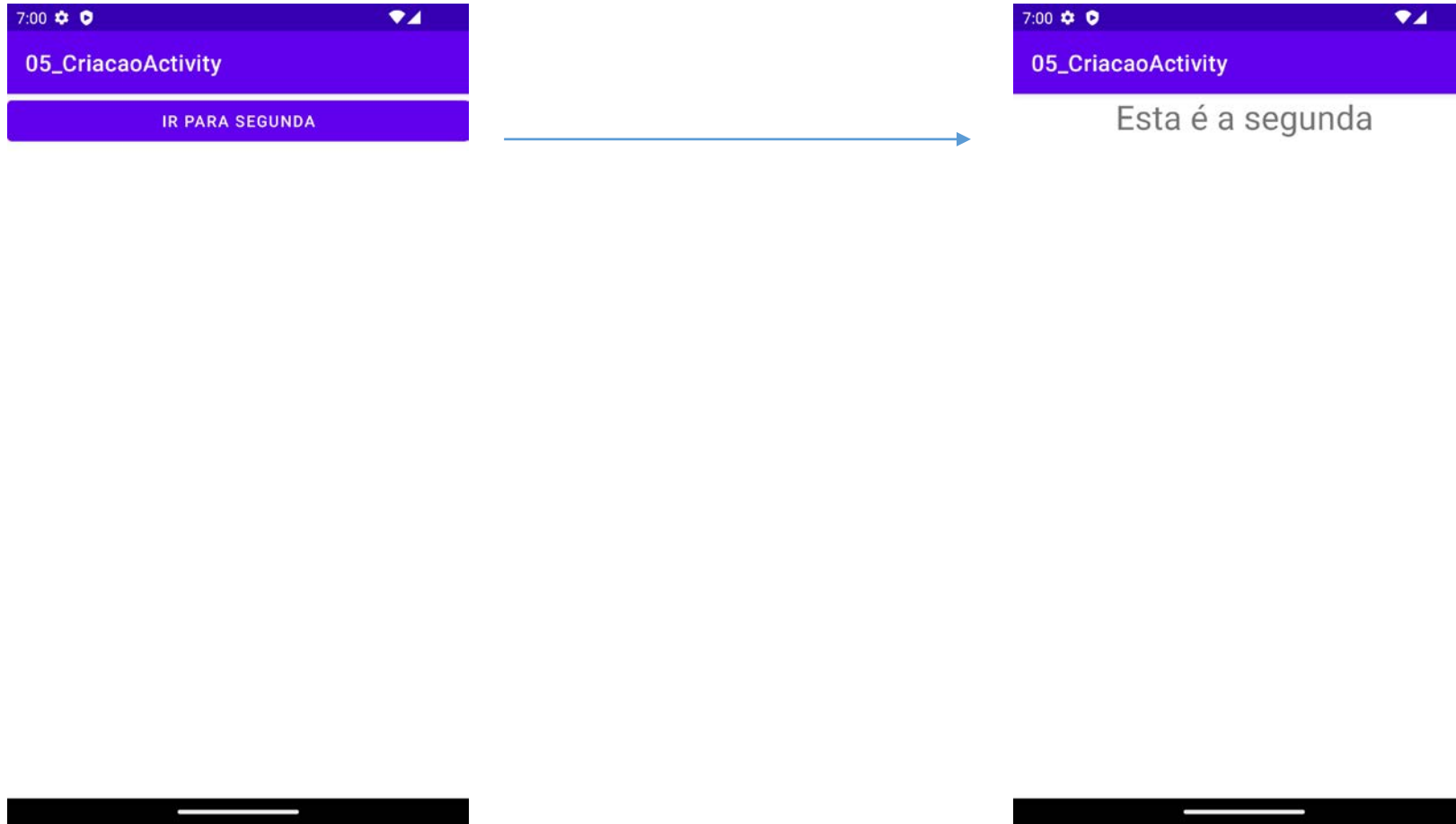
- Ou

```
startActivity(Intent(this, MainActivity2::class.java))
```

Intent

- “`val intent`” é a criação do Intent ao qual chamamos “`intent`”;
- “`this`” é a Activity onde nos encontramos;
- “`MainActivity2::class.java`” é a Activity para onde queremos ir;
- “`startActivity(intent)`” é para iniciar o Intent criado anteriormente.

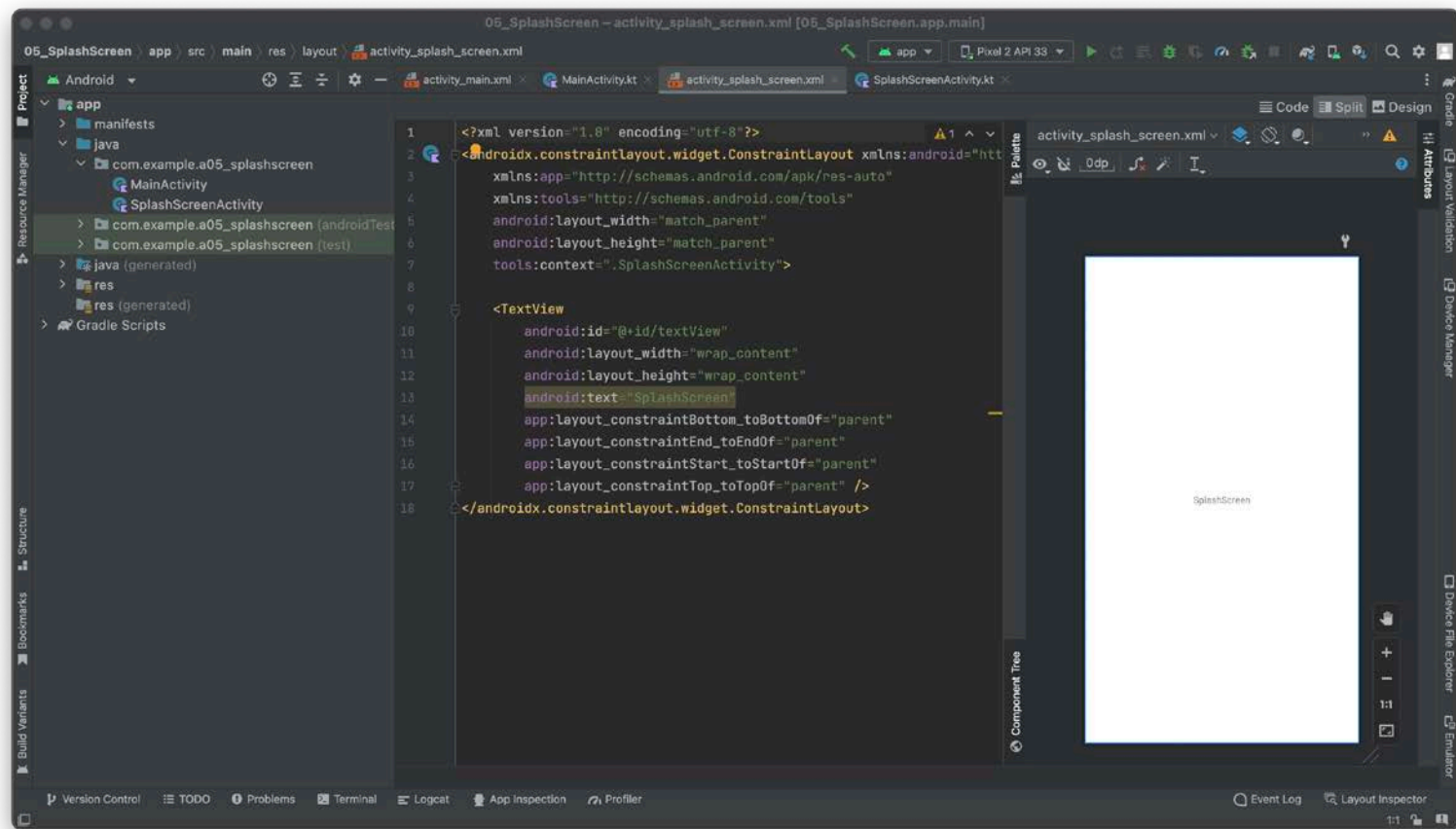
Intent



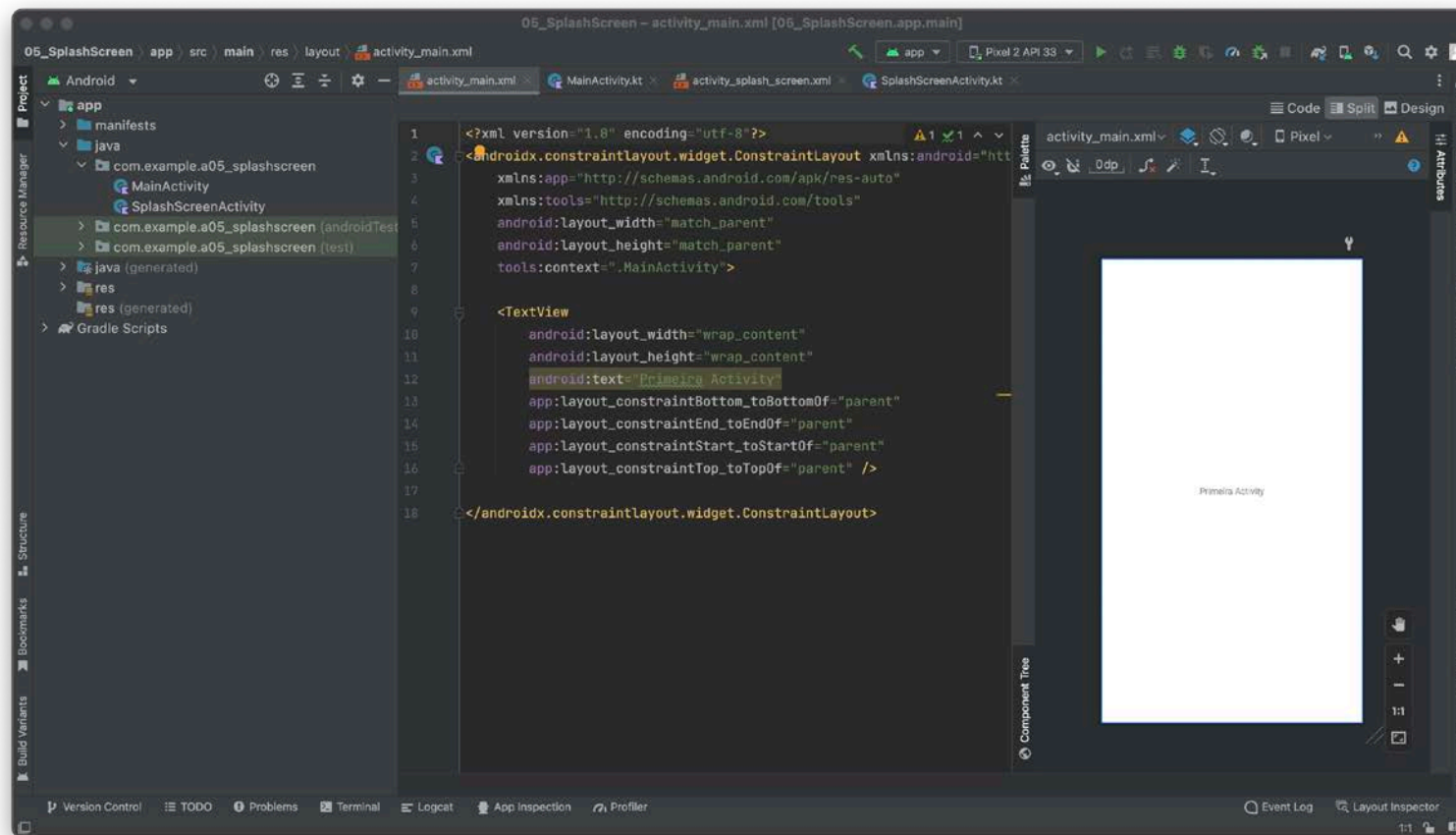
SplashScreen

- O SplashScreen é um ecrã, normalmente apresentado no início da execução da aplicação, onde é apresentado ao utilizador uma Activity e após um valor temporal definido pelo utilizador é redirecionado para uma outra Activity.
- Vamos definir 2 Activity, a primeira (MainActivity) tem uma TextView com a mensagem: “SplashScreen”; a segunda (MainActivity2) tem uma TextView com a mensagem: “Primeira Activity”.

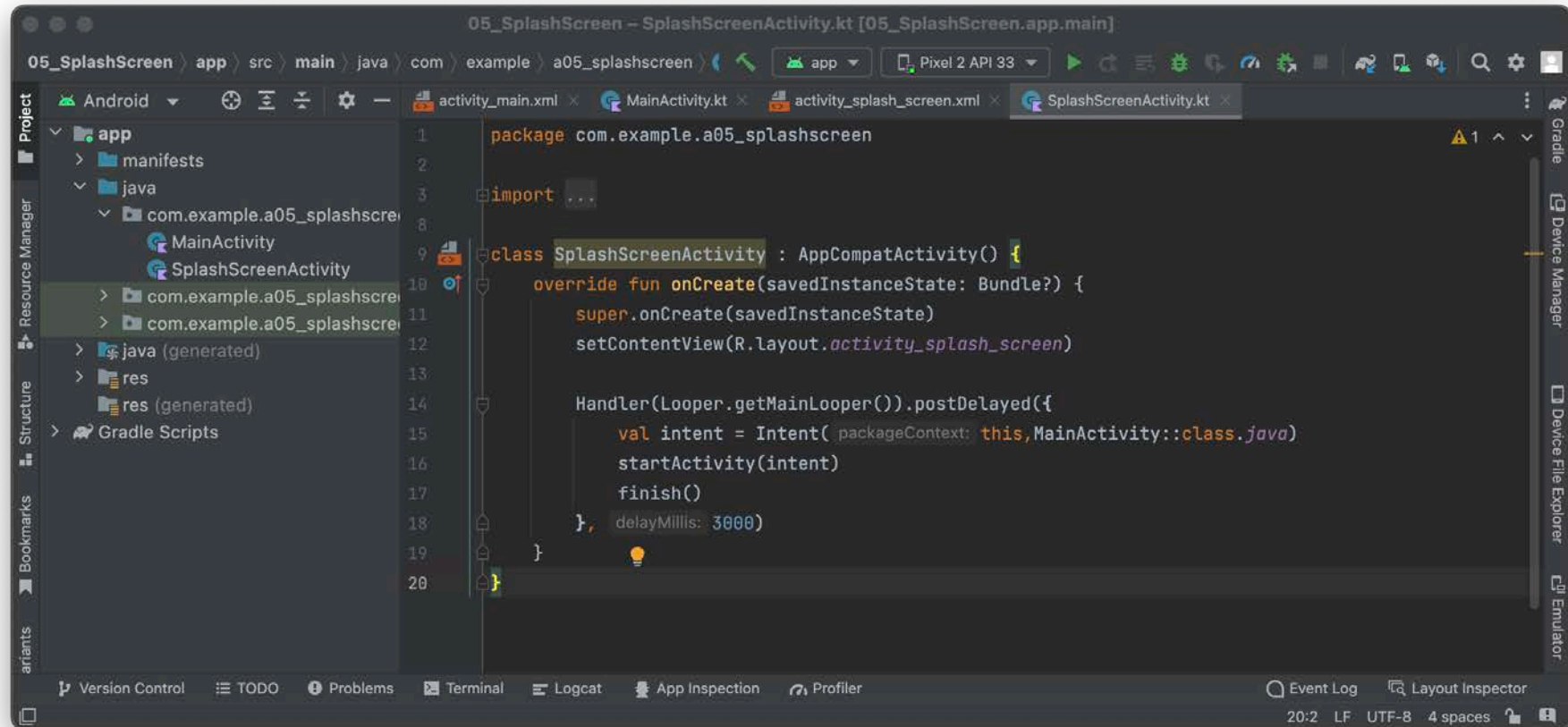
SplashScreen



SplashScreen



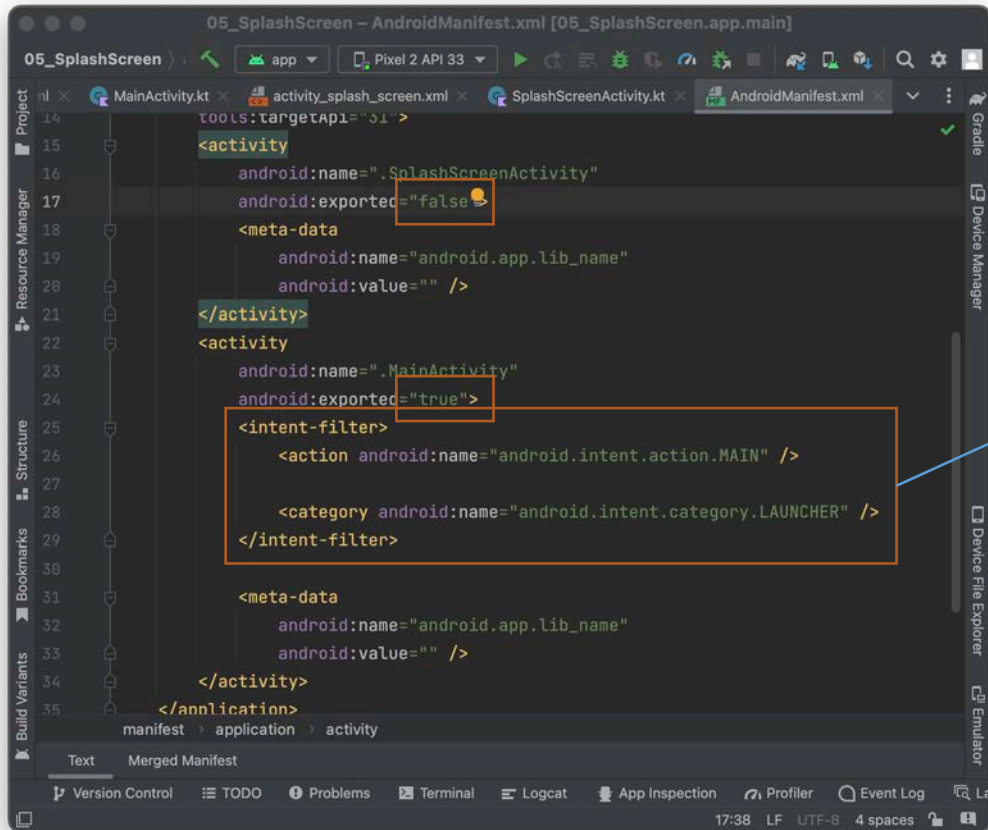
SplashScreen



SplashScreen

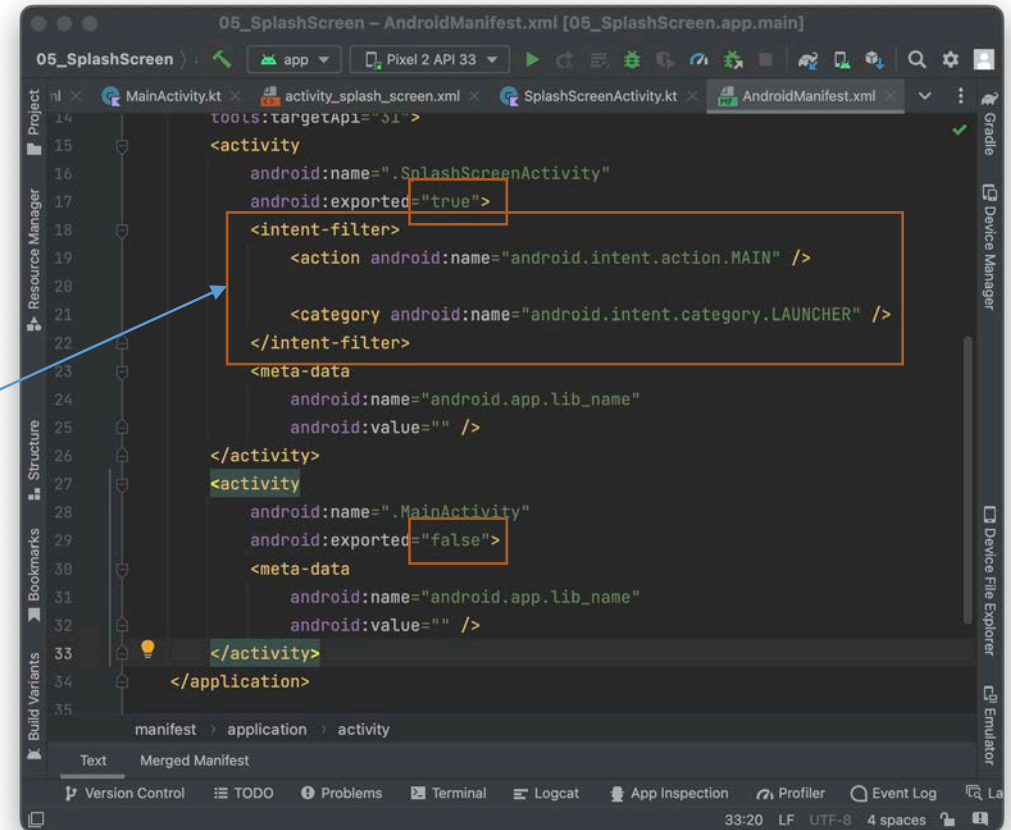
- Para garantir que a SplashScreenActivity é a primeira a ser iniciada temos de abrir o AndroidManifest e alterar a Activity de arranque.
- Passamos o conteúdo de intent-filter para a Activity de arranque e o parâmetro android:exported para true na SplashScreen e para false na MainActivity.

SplashScreen



This screenshot shows the initial configuration of the `SplashScreenActivity` in the `AndroidManifest.xml` file. The activity is named `.SplashScreenActivity` and has `android:exported="false"`. It includes a `<meta-data>` element with `android:name="android.app.lib_name"` and `android:value=""`. Below it, the `.MainActivity` is configured with `android:exported="true"` and an `<intent-filter>` for the `android.intent.action.MAIN` category `android.intent.category.LAUNCHER`. A blue arrow points from the `<intent-filter>` of `.MainActivity` to the `<intent-filter>` of `.SplashScreenActivity` in the next screenshot.

```
14  tools:targetApi="31">
15  <activity
16      android:name=".SplashScreenActivity"
17      android:exported="false">
18      <meta-data
19          android:name="android.app.lib_name"
20          android:value="" />
21  </activity>
22  <activity
23      android:name=".MainActivity"
24      android:exported="true">
25      <intent-filter>
26          <action android:name="android.intent.action.MAIN" />
27
28          <category android:name="android.intent.category.LAUNCHER" />
29      </intent-filter>
30
31      <meta-data
32          android:name="android.app.lib_name"
33          android:value="" />
34  </activity>
35  </application>
```



This screenshot shows the updated configuration of the `SplashScreenActivity` in the `AndroidManifest.xml` file. The activity is now named `.SplashScreenActivity` and has `android:exported="true"`. It now includes an `<intent-filter>` with `<action android:name="android.intent.action.MAIN" />` and `<category android:name="android.intent.category.LAUNCHER" />`. Below it, the `.MainActivity` is configured with `android:exported="false"` and a `<meta-data>` element with `android:name="android.app.lib_name"` and `android:value=""`. The `</application>` tag is at the bottom.

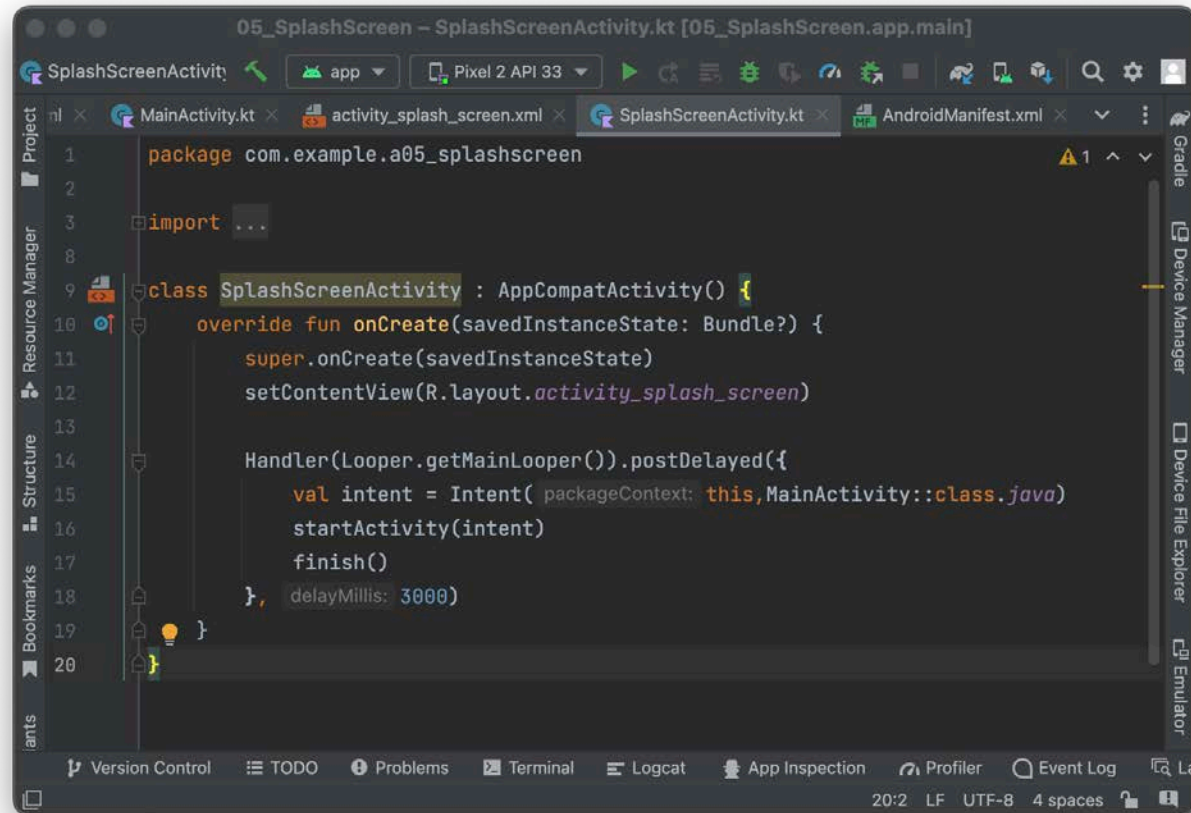
```
14  tools:targetApi="31">
15  <activity
16      android:name=".SplashScreenActivity"
17      android:exported="true">
18      <intent-filter>
19          <action android:name="android.intent.action.MAIN" />
20
21          <category android:name="android.intent.category.LAUNCHER" />
22      </intent-filter>
23      <meta-data
24          android:name="android.app.lib_name"
25          android:value="" />
26  </activity>
27  <activity
28      android:name=".MainActivity"
29      android:exported="false">
30      <meta-data
31          android:name="android.app.lib_name"
32          android:value="" />
33  </activity>
34  </application>
```

SplashScreen

- No MainActivity.java vamos adicionar o seguinte código após a linha setContentView(...)

```
Handler(Looper.getMainLooper()).postDelayed({  
    val intent = Intent( packageContext: this, MainActivity::class.java)  
    startActivity(intent)  
    finish()  
}, delayMillis: 3000)
```

SplashScreen



SplashScreen

- Importante verificar o valor que aparece no final do código: 3000, este é o valor em milissegundos que irá demorar a passar da primeira para a segunda Activity.
- Para indicar a passagem entre a MainActivity e a MainActivity2 usamos o Intent como anteriormente.

Exercício 1

- Crie uma aplicação que tenha 2 botões:
 - O primeiro, quando clicado, apresenta um Toast com a mensagem “Botão clicado”
 - O segundo, quando clicado, redireciona para uma segunda Activity na qual tem uma TextView com a mensagem “Bem-vindo à nova Activity”.

Exercício 2

- Crie uma aplicação com 3 Activity:
 - MainActivity: deve conter um formulário de Login (username e password), quando o utilizador inserir username = user e password = pass, deve ser redirecionado para a LoginOkActivity, caso os dados de login estejam errados deve redirecionar para a LoginErradoActivity.
 - LoginOkActivity: deve apresentar ao utilizador uma mensagem de boas vindas numa TextView;
 - LoginErradoActivity: deve apresentar ao utilizador uma mensagem num Toast de login errado e um botão que quando clicado volta à MainActivity.

Exercício 3

- Crie uma aplicação que utilize um SplashScreen. O conteúdo da aplicação deve ficar ao seu critério.

Exercício 4

- Crie uma aplicação que pede ao utilizador para inserir um número e apresente:
 - A indicação se o número é par ou ímpar (utilize um TextView).
 - A indicação se o número é ou não primo (utilize um TextView).
- NOTA: Um número é primo se for divisível apenas por 1 e por ele mesmo. Exemplos: 2, 3, 5, 7,...

Exercício 5

- Pretende-se determinar o número de semanas, dias e horas a que corresponde um dado valor de horas pedido ao utilizador. Exemplo: 300h são 1 semana, 5 dias e 12 horas.
- Crie os elementos de layout necessários para apresentar os valores de semanas, dias e horas em elementos separados.