



Reskilling 4Employment Software Developer

Acesso móvel a sistemas de informação

Bruno Santos

bruno.santos.mcv@msft.cesae.pt

Tópicos



Centro para o Desenvolvimento
de Competências Digitais

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

Convenções



Centro para o Desenvolvimento
de Competências Digitais

- A utilização de ; é opcional mas desencorajada no Kotlin

Convenções



Centro para o Desenvolvimento
de Competências Digitais

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

Convenções



Centro para o Desenvolvimento
de Competências Digitais

- 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



Centro para o Desenvolvimento
de Competências Digitais

- 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



Centro para o Desenvolvimento
de Competências Digitais

- 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



Centro para o Desenvolvimento
de Competências Digitais

- 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



Centro para o Desenvolvimento
de Competências Digitais

- 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 minuscula_com_underscore

Elemento	Prefixo
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



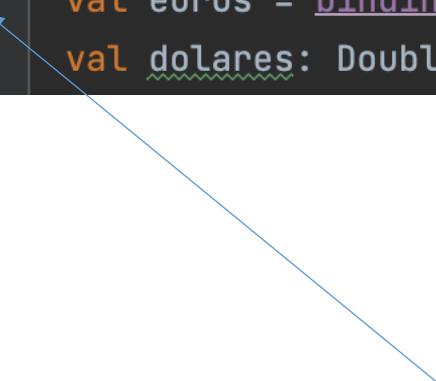
Centro para o Desenvolvimento
de Competências Digitais

- 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.text.toString()  
18     val dolares: Double = String.forma
```

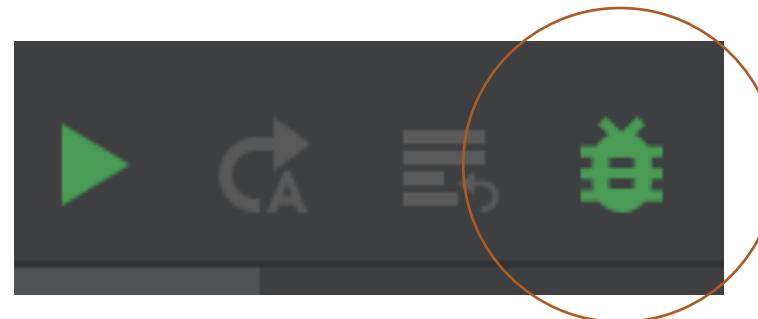
```
16 ding.buttonDolar.setOnClickListener {  
17     val euros = binding.editValorEuros.text.toString()  
18     val dolares: Double = String.forma
```



Clicar aqui

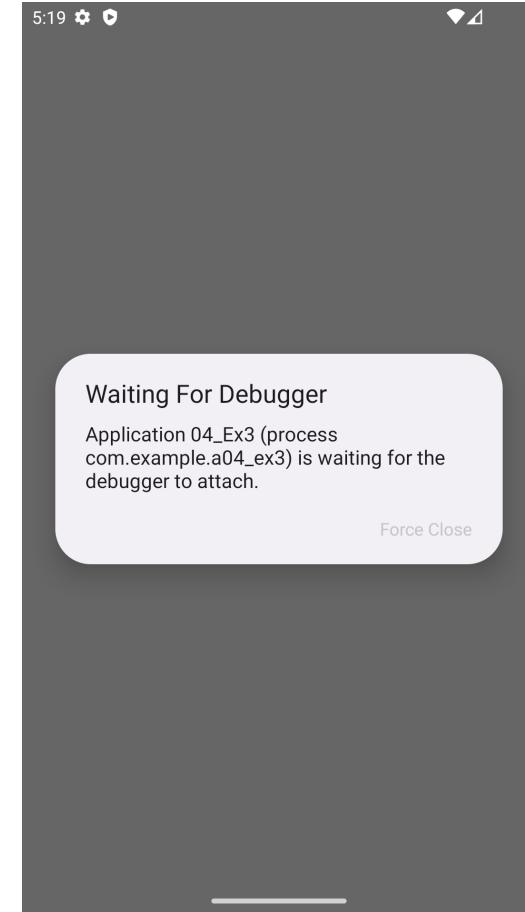
Debug

- 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.



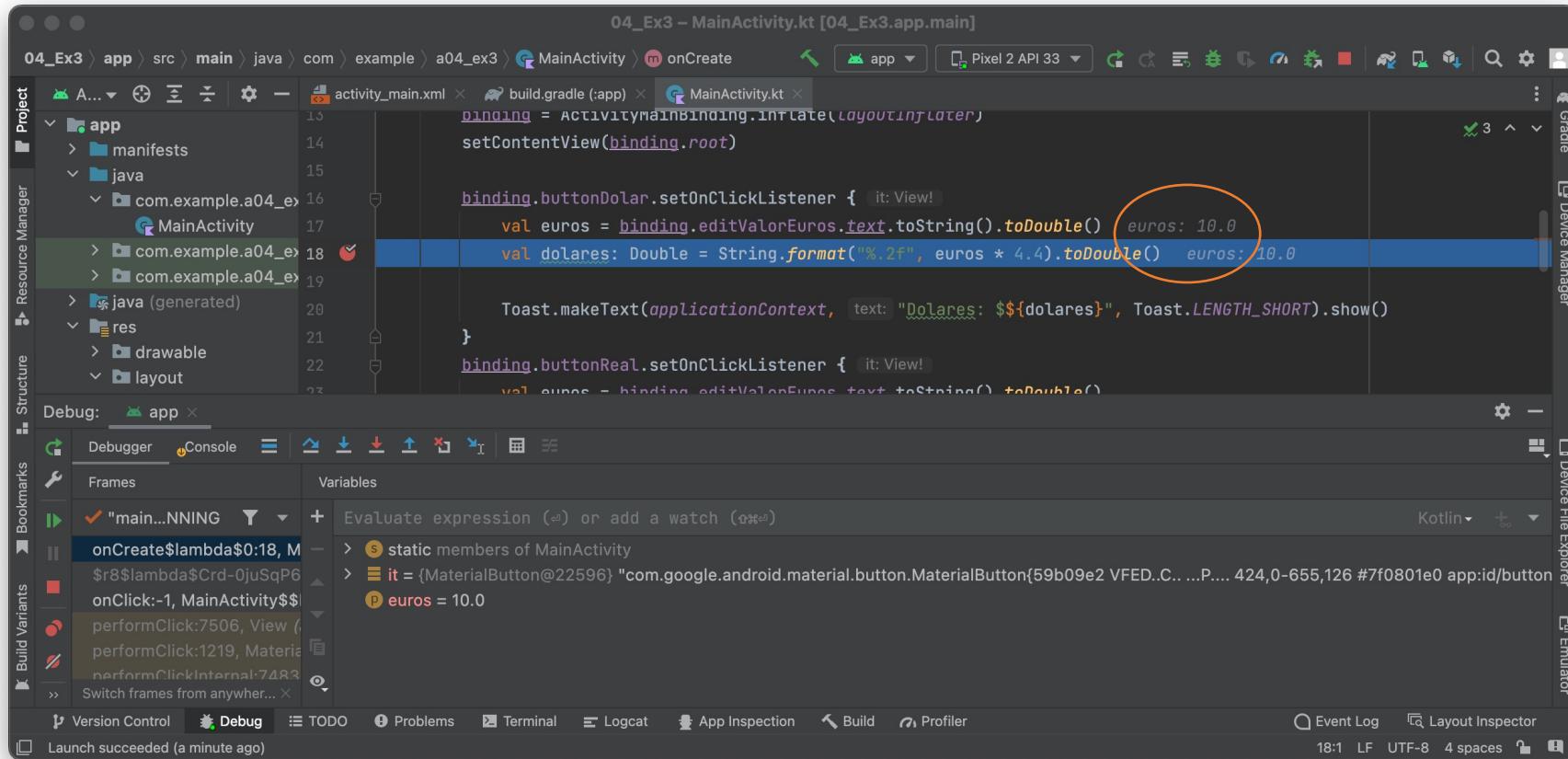
Debug



Centro para o Desenvolvimento
de Competências Digitais

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

Debug



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "04_Ex3". The "app" module contains "manifests", "java", "res", and "layout" directories.
- Code Editor:** The file "MainActivity.kt" is open, showing Kotlin code for an Android application. A variable "euros" is highlighted with a blue selection bar. The status bar at the bottom of the code editor shows "euros: 10.0".
- Debug Bar:** The "Debug" tab is selected, showing the current stack frames. The top frame is "main\$NNING" with a breakpoint. Other frames include "onCreate\$lambda\$0:18", "onClick:-1", and "performClick:7506".
- Variables View:** The "Variables" section shows a list of variables, including "static members of MainActivity", "it" (a MaterialButton), and "euros = 10.0".
- Bottom Status:** The status bar at the bottom indicates "Launch succeeded (a minute ago)" and shows the time as 18:11, encoding as LF, and 4 spaces.

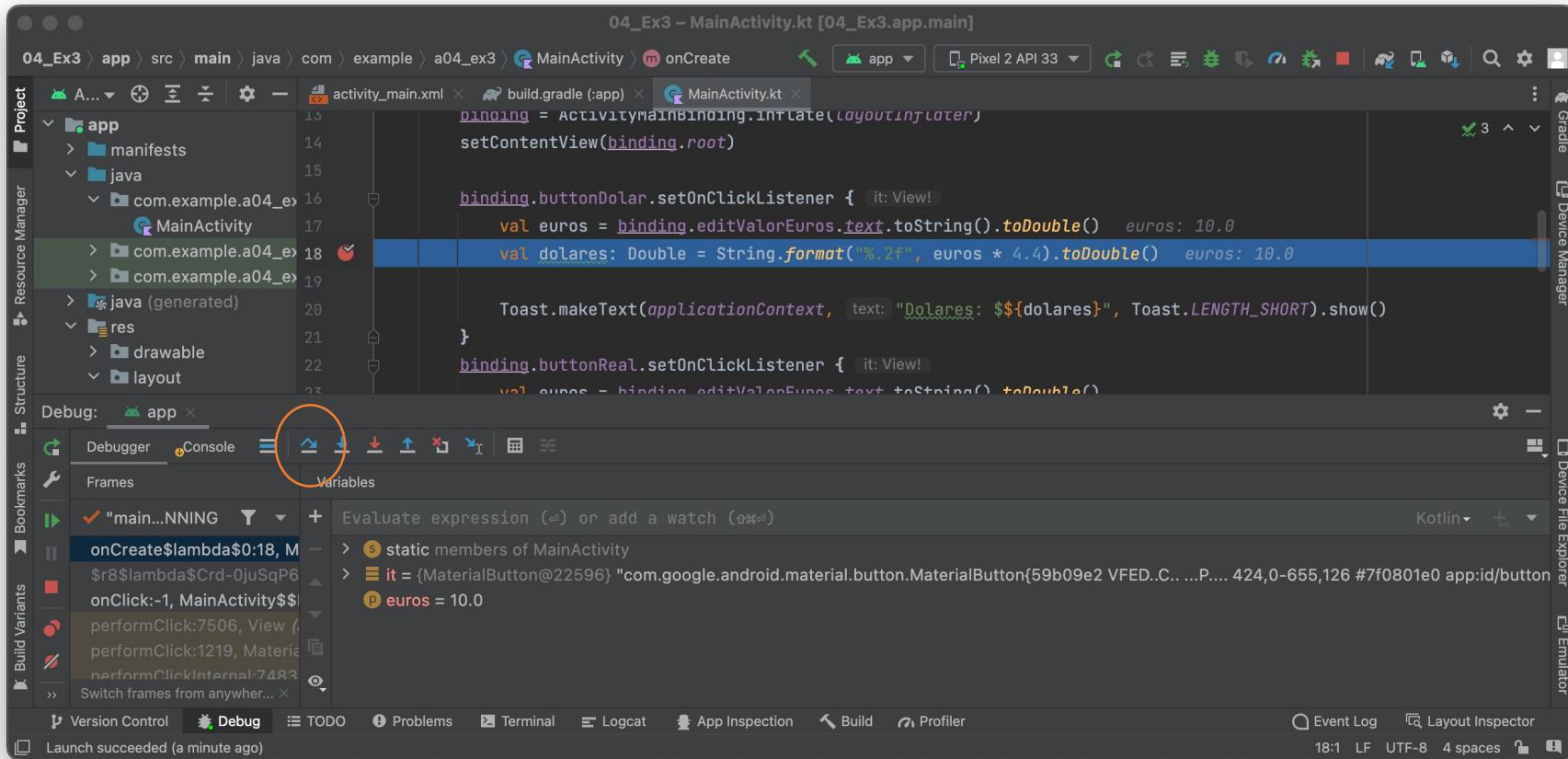
Debug



Centro para o Desenvolvimento
de Competências Digitais

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

Debug



The screenshot shows the Android Studio interface during a debugging session. The code editor displays `MainActivity.kt` with the following code:

```
04_Ex3 - MainActivity.kt [04_Ex3.app.main]
04_Ex3 > app > src > main > java > com > example > a04_ex3 > MainActivity > onCreate
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

    binding.buttonDolar.setOnClickListener { it: View!
        val euros = binding.editValorEuros.text.toString().toDouble() euros: 10.0
        val dolares: Double = String.format("%.2f", euros * 4.4).toDouble() euros: 10.0
        Toast.makeText(applicationContext, text: "Dolares: ${dolares}", Toast.LENGTH_SHORT).show()
    }

    binding.buttonReal.setOnClickListener { it: View!
        val euros = binding.editValorEuros.text.toString().toDouble()
```

The `Variables` tab in the bottom-left corner of the interface is highlighted with a red circle, indicating the current state of the application's variables.

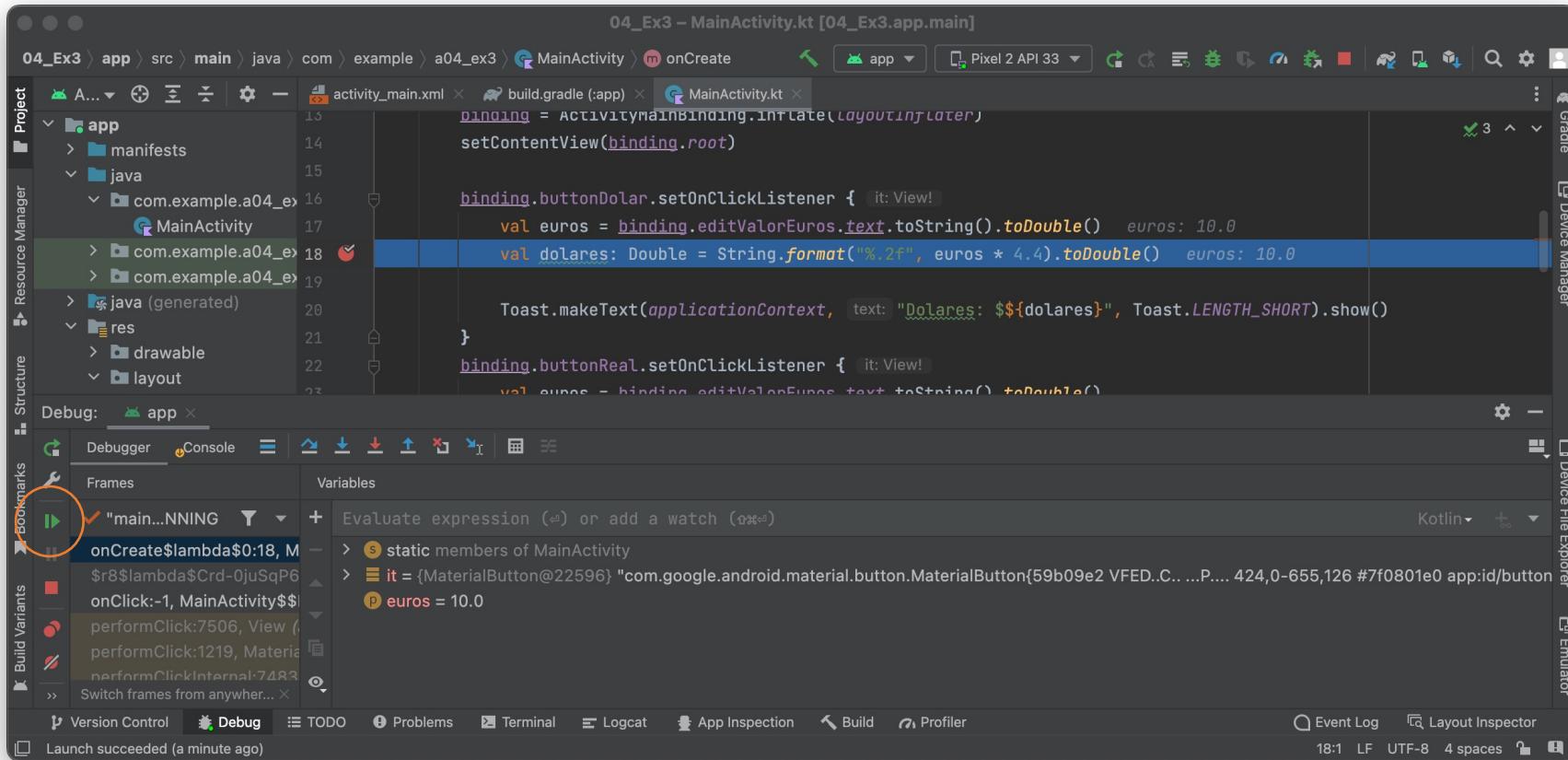
Debug



Centro para o Desenvolvimento
de Competências Digitais

- 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



The screenshot shows the Android Studio interface during a debug session. The top bar displays the project name "04_Ex3" and the file "MainActivity.kt". The code editor shows the MainActivity.kt file with several lines of Kotlin code. A specific line of code is highlighted in blue: `val dolares: Double = String.format("%.2f", euros * 4.4).toDouble()`. The status bar at the bottom indicates "Launch succeeded (a minute ago)".

The left sidebar shows the project structure with the "app" module selected. The "Debug" tab is active in the bottom navigation bar.

The bottom-left corner of the screenshot has a red circle highlighting the "Breakpoints" icon in the toolbar, which is used to set or clear breakpoints in the code.

Toast

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



Desenvolvido por Bruno Santos

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 escolher 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



Centro para o Desenvolvimento
de Competências Digitais

```
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")
    }
}
```

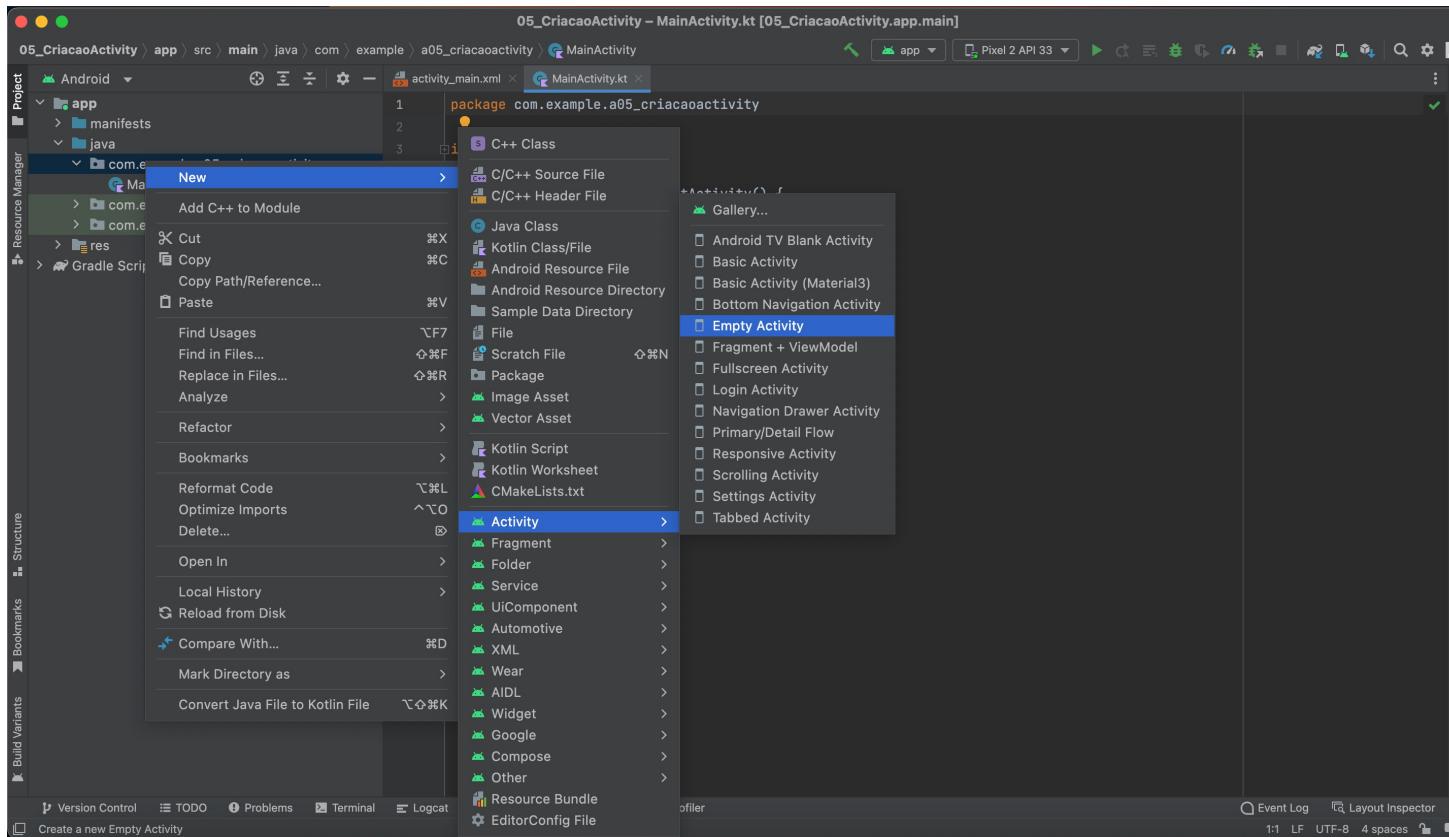
Criação de novas Activity



Centro para o Desenvolvimento
de Competências Digitais

- 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, selecionar a opção New, posteriormente Activity e finalmente o tipo de Activity pretendida.

Criação de novas Activity



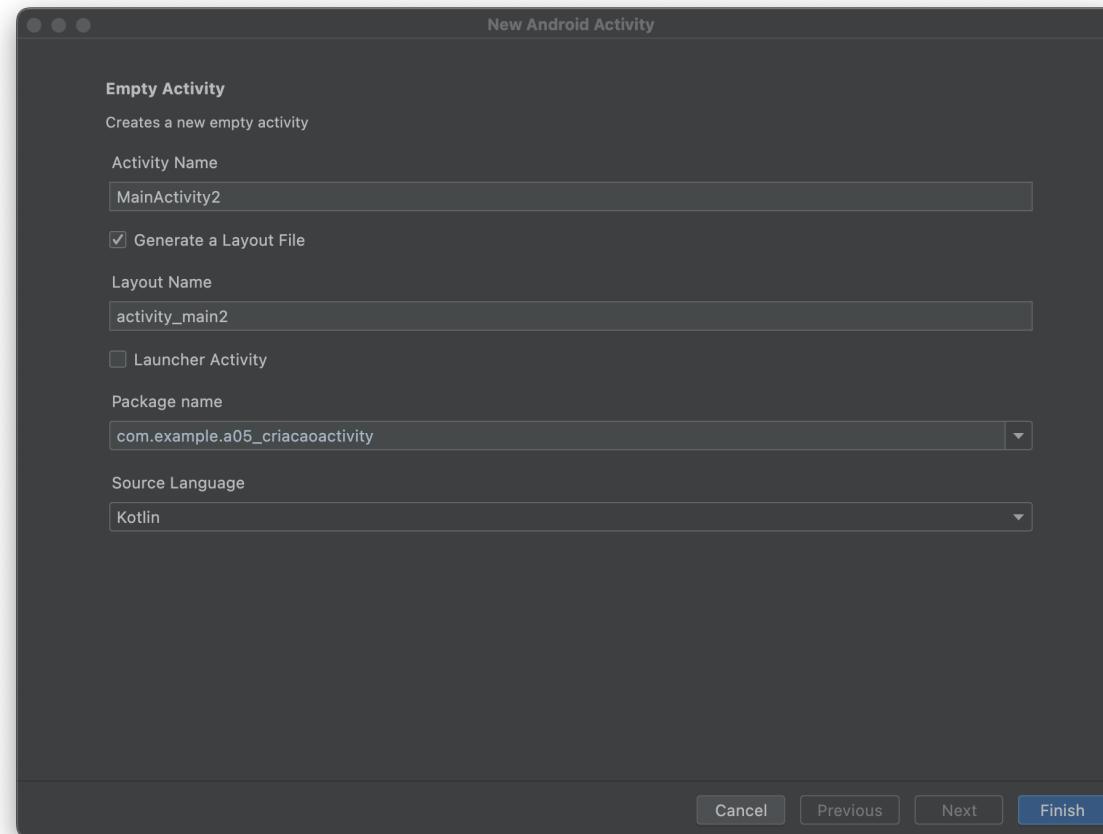
Criação de novas Activity



Centro para o Desenvolvimento
de Competências Digitais

- Selecionamos 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



Centro para o Desenvolvimento
de Competências Digitais

A screenshot of the Android Studio interface. The project navigation bar shows a file named "MainActivity2.kt" is currently selected. The code editor displays the following Java code:

```
05_CriacaoActivity - MainActivity2 [05_CriacaoActivity.app.main]
05_CriacaoActivity app src main java com example a05_criacaoactivity MainActivity2
1 package com.example.a05_criacaoactivity
2
3 import ...
4
5 class MainActivity2 : AppCompatActivity() {
6     override fun onCreate(savedInstanceState: Bundle?) {
7         super.onCreate(savedInstanceState)
8         setContentView(R.layout.activity_main2)
9     }
10 }
11 }
```

The Android Studio interface includes toolbars, a status bar at the bottom, and various panels for resource management and device configuration.

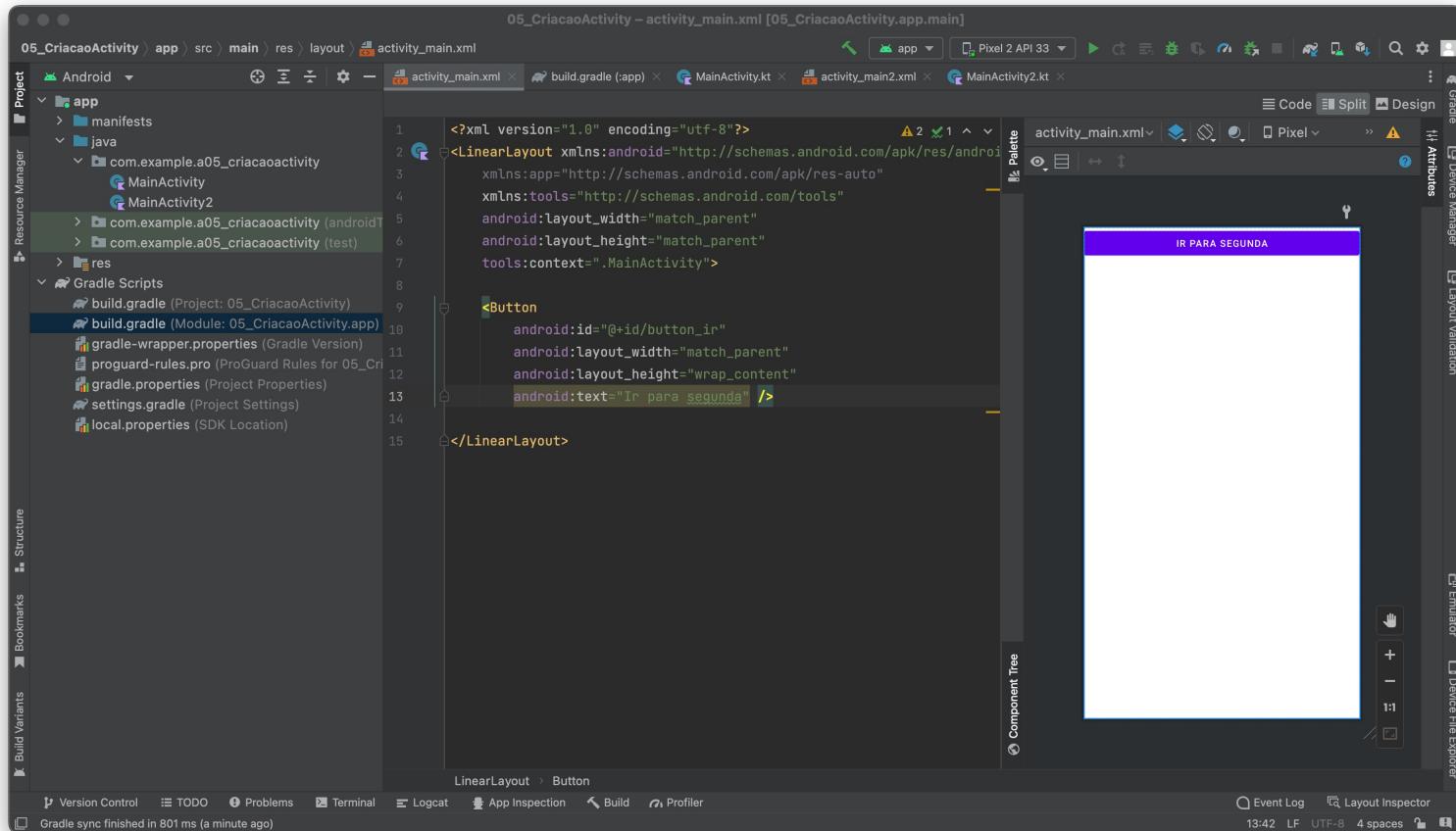
Intent



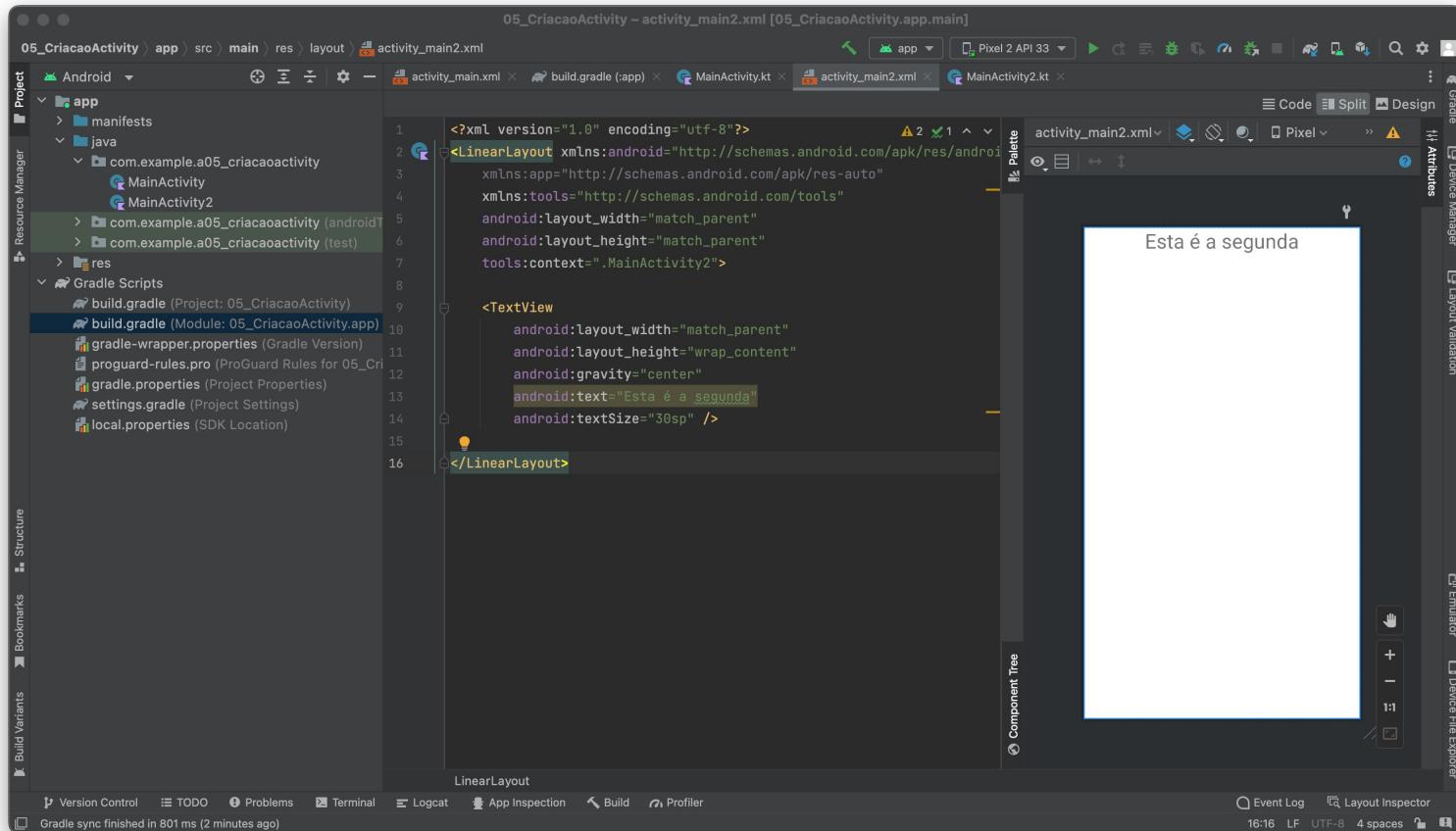
Centro para o Desenvolvimento
de Competências Digitais

- 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



Centro para o Desenvolvimento
de Competências Digitais

- “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



Centro para o Desenvolvimento
de Competências Digitais



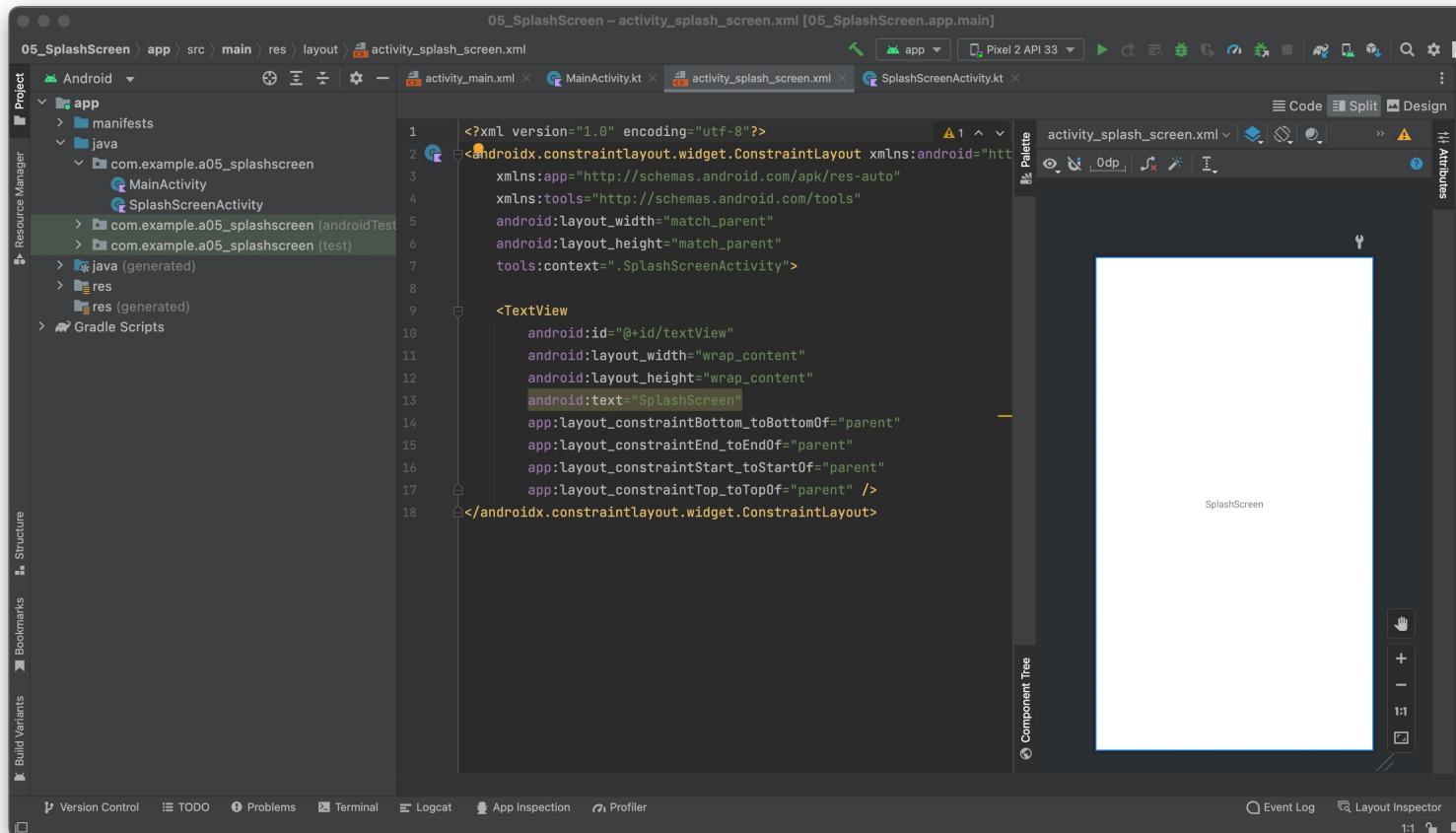
SplashScreen



Centro para o Desenvolvimento
de Competências Digitais

- 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



Centro para o Desenvolvimento
de Competências Digitais

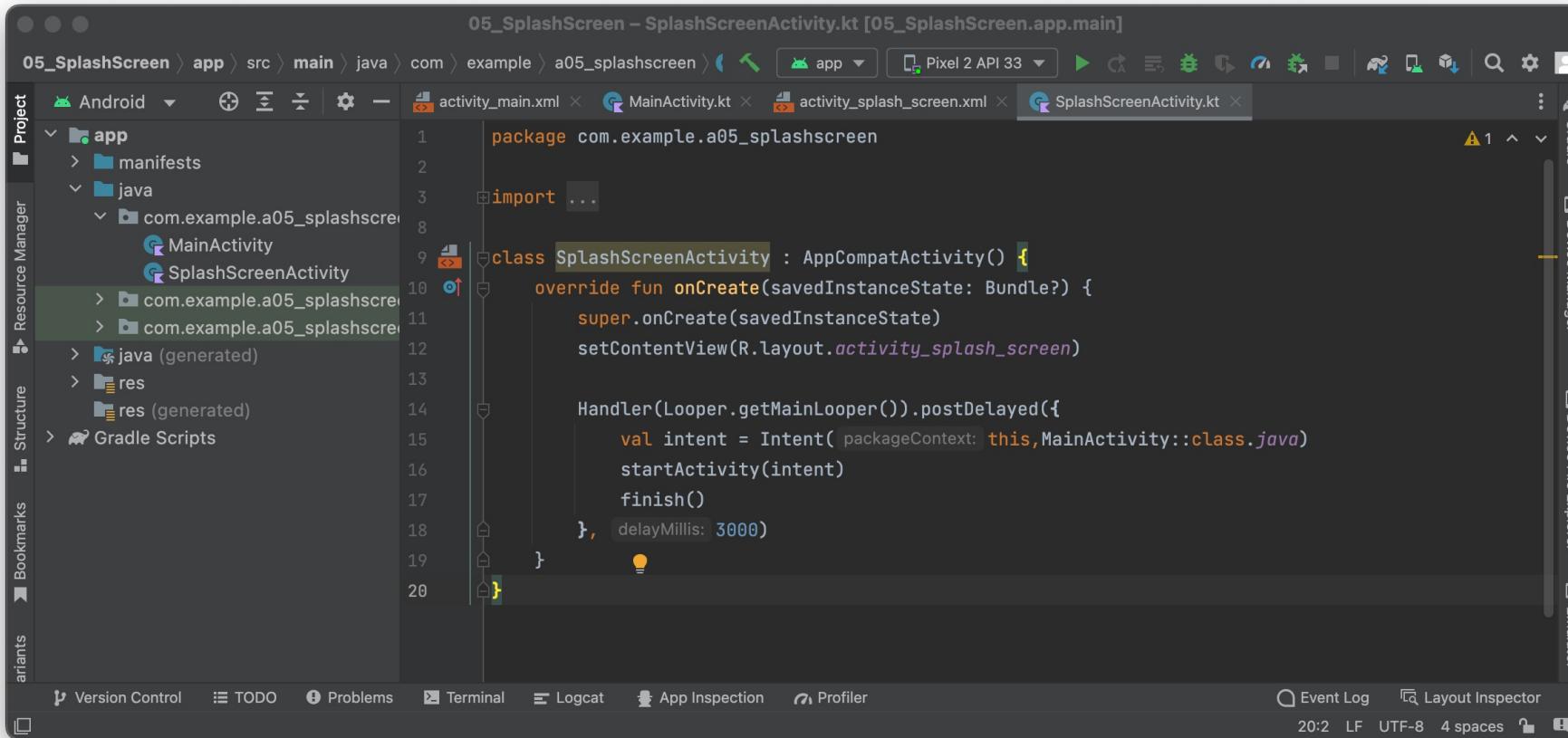
The screenshot shows the Android Studio interface with the project '05_SplashScreen' open. The 'activity_main.xml' file is selected in the layout editor. The XML code defines a ConstraintLayout with a single TextView containing the text 'Primeira Activity'. The preview window on the right shows a white screen with the text 'Primeira Activity' centered. The bottom status bar indicates the screen is at 1:1 scale.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Primeira Activity"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

SplashScreen



The screenshot shows the Android Studio interface with the project '05_SplashScreen' open. The code editor displays the file 'SplashScreenActivity.kt' under the package 'com.example.a05_splashscreen'. The code implements a splash screen logic using a Handler to delay the transition to the main activity:

```
1 package com.example.a05_splashscreen
2
3 import ...
4
5 class SplashScreenActivity : AppCompatActivity() {
6     override fun onCreate(savedInstanceState: Bundle?) {
7         super.onCreate(savedInstanceState)
8         setContentView(R.layout.activity_splash_screen)
9
10        Handler(Looper.getMainLooper()).postDelayed({
11            val intent = Intent(packageContext, MainActivity::class.java)
12            startActivity(intent)
13            finish()
14        }, delayMillis: 3000)
15    }
16}
17
18
19
20}
```

The 'activity_main.xml' and 'activity_splash_screen.xml' files are also visible in the top bar. The bottom navigation bar includes tabs for Version Control, TODO, Problems, Terminal, Logcat, App Inspection, Profiler, Event Log, Layout Inspector, and Emulator.

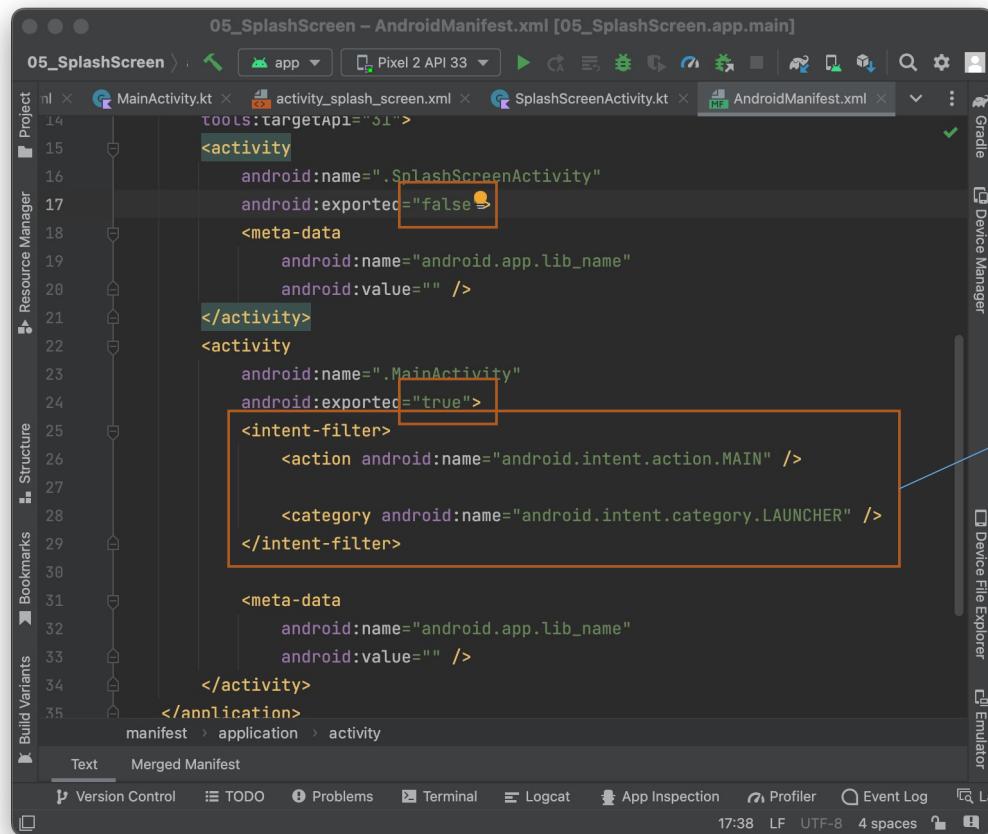
SplashScreen



Centro para o Desenvolvimento
de Competências Digitais

- 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.

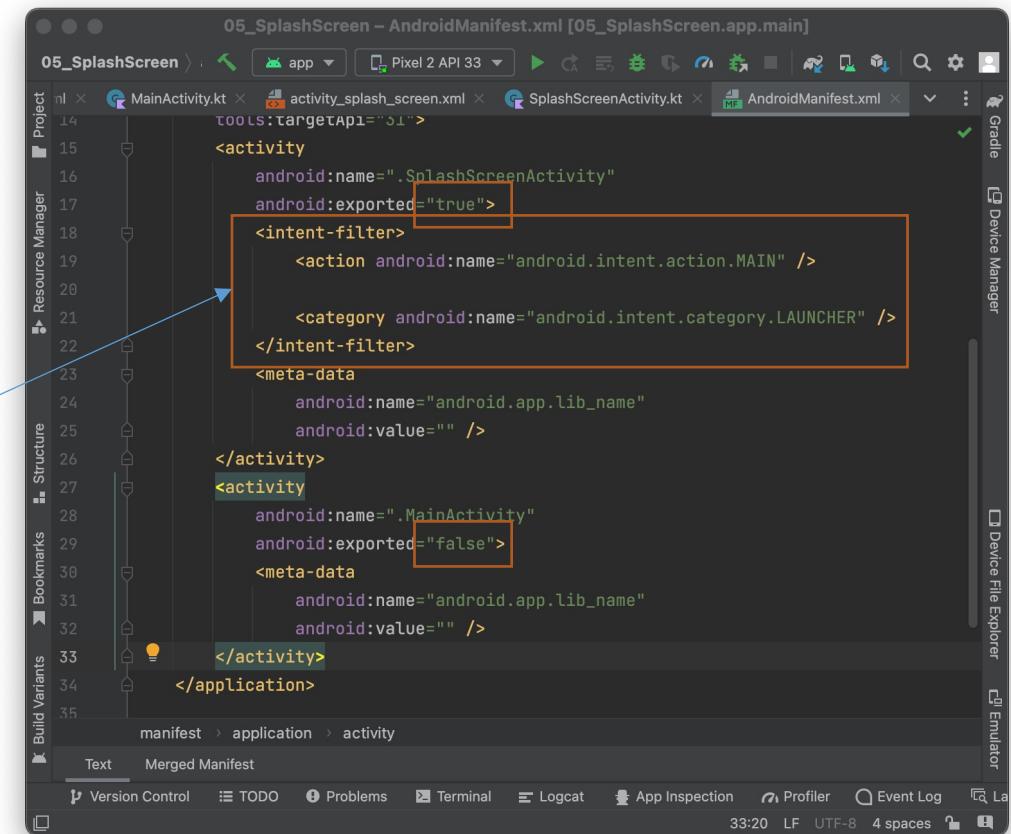
SplashScreen



05_SplashScreen – AndroidManifest.xml [05_SplashScreen.app.main]

```
14
15     tools:targetApi="31">
16     <activity
17         android:name=".SplashScreenActivity"
18         android:exported="false" style="outline: 1px solid orange;">
19             <meta-data
20                 android:name="android.app.lib_name"
21                 android:value="" />
22         </activity>
23         <activity
24             android:name=".MainActivity"
25             android:exported="true" style="outline: 1px solid orange;">
26                 <intent-filter>
27                     <action android:name="android.intent.action.MAIN" />
28
29                     <category android:name="android.intent.category.LAUNCHER" />
30                 </intent-filter>
31
32                 <meta-data
33                     android:name="android.app.lib_name"
34                     android:value="" />
35             </activity>
36         </application>
37     manifest > application > activity
38 
```

This screenshot shows the AndroidManifest.xml file in the Android Studio IDE. It contains two activities: SplashScreenActivity and MainActivity. Both activities have their `android:exported` attribute set. The first activity is set to false, and the second is set to true. A blue arrow points from the first activity's configuration to the second activity's configuration, indicating a comparison or flow between them.



05_SplashScreen – AndroidManifest.xml [05_SplashScreen.app.main]

```
14
15     tools:targetApi="31">
16     <activity
17         android:name=".SplashScreenActivity"
18         android:exported="true" style="outline: 1px solid orange;">
19             <intent-filter>
20                 <action android:name="android.intent.action.MAIN" />
21
22                 <category android:name="android.intent.category.LAUNCHER" />
23             </intent-filter>
24             <meta-data
25                 android:name="android.app.lib_name"
26                 android:value="" />
27         </activity>
28         <activity
29             android:name=".MainActivity"
30             android:exported="false" style="outline: 1px solid orange;">
31                 <meta-data
32                     android:name="android.app.lib_name"
33                     android:value="" />
34             </activity>
35         </application>
36     manifest > application > activity
37 
```

This screenshot shows the same AndroidManifest.xml file after modifications. The `android:exported` attribute for the SplashScreenActivity has been changed to true, while the MainActivity's attribute remains at false. A blue arrow points from the SplashScreenActivity's configuration to the MainActivity's configuration, indicating a comparison or flow between them.

SplashScreen

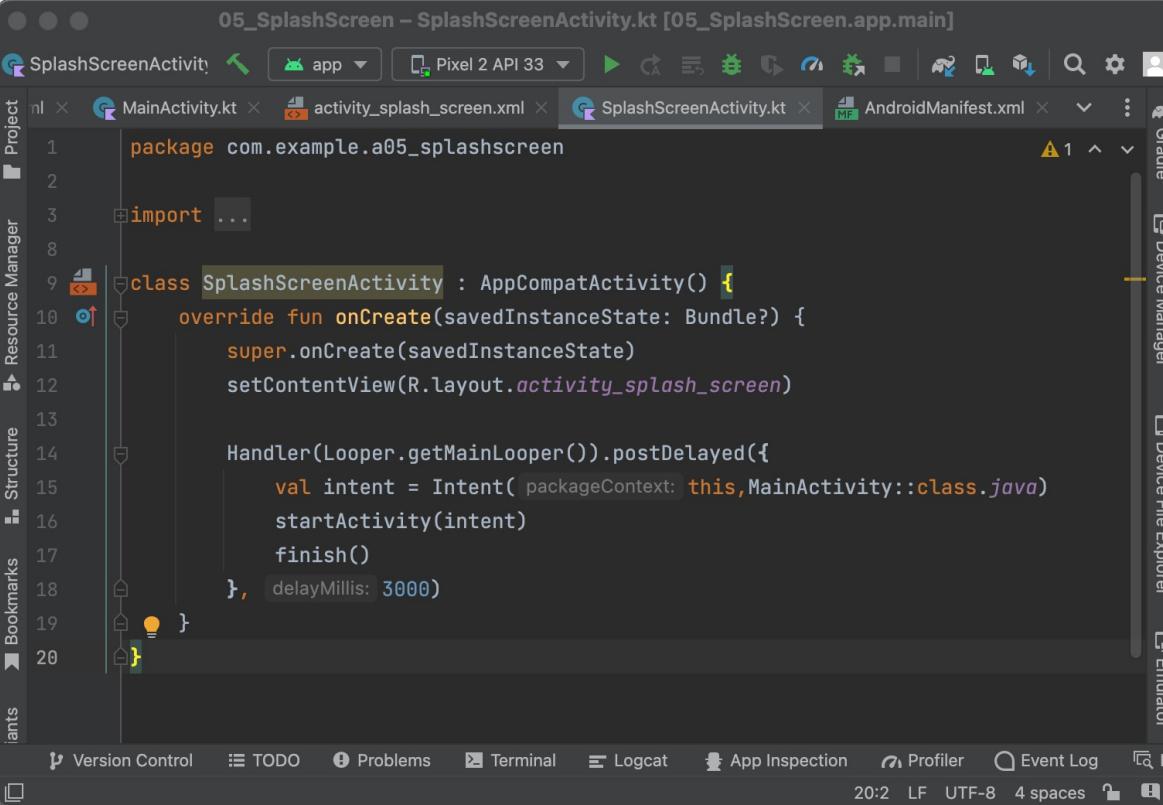


Centro para o Desenvolvimento
de Competências Digitais

- 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



The screenshot shows the Android Studio interface with the project '05_SplashScreen' open. The main editor window displays the code for `SplashScreenActivity.kt`. The code defines a new activity that extends `AppCompatActivity`. It overrides the `onCreate` method to set the content view to `R.layout.activity_splash_screen`. Inside this method, it uses a `Handler` to post a delayed task. This task creates an intent to start the `MainActivity`, then starts the activity, and finally finishes the current activity after a delay of 3000 milliseconds.

```
05_SplashScreen – SplashScreenActivity.kt [05_SplashScreen.app.main]
SplashScreenActivity MainActivity.kt activity_splash_screen.xml SplashScreenActivity.kt AndroidManifest.xml

1 package com.example.a05_splashscreen
2
3 import ...
4
5 class SplashScreenActivity : AppCompatActivity() {
6     override fun onCreate(savedInstanceState: Bundle?) {
7         super.onCreate(savedInstanceState)
8         setContentView(R.layout.activity_splash_screen)
9
10        Handler(Looper.getMainLooper()).postDelayed({
11            val intent = Intent(packageContext, MainActivity::class.java)
12            startActivity(intent)
13            finish()
14        }, delayMillis: 3000)
15    }
16}
17
18
19
20
```

SplashScreen



Centro para o Desenvolvimento
de Competências Digitais

- 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



Centro para o Desenvolvimento
de Competências Digitais

- 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



Centro para o Desenvolvimento
de Competências Digitais

- 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.