



Reskilling 4Employment Software Developer

Acesso móvel a sistemas de informação

Bruno Santos

bruno.santos.mcv@msft.cesae.pt

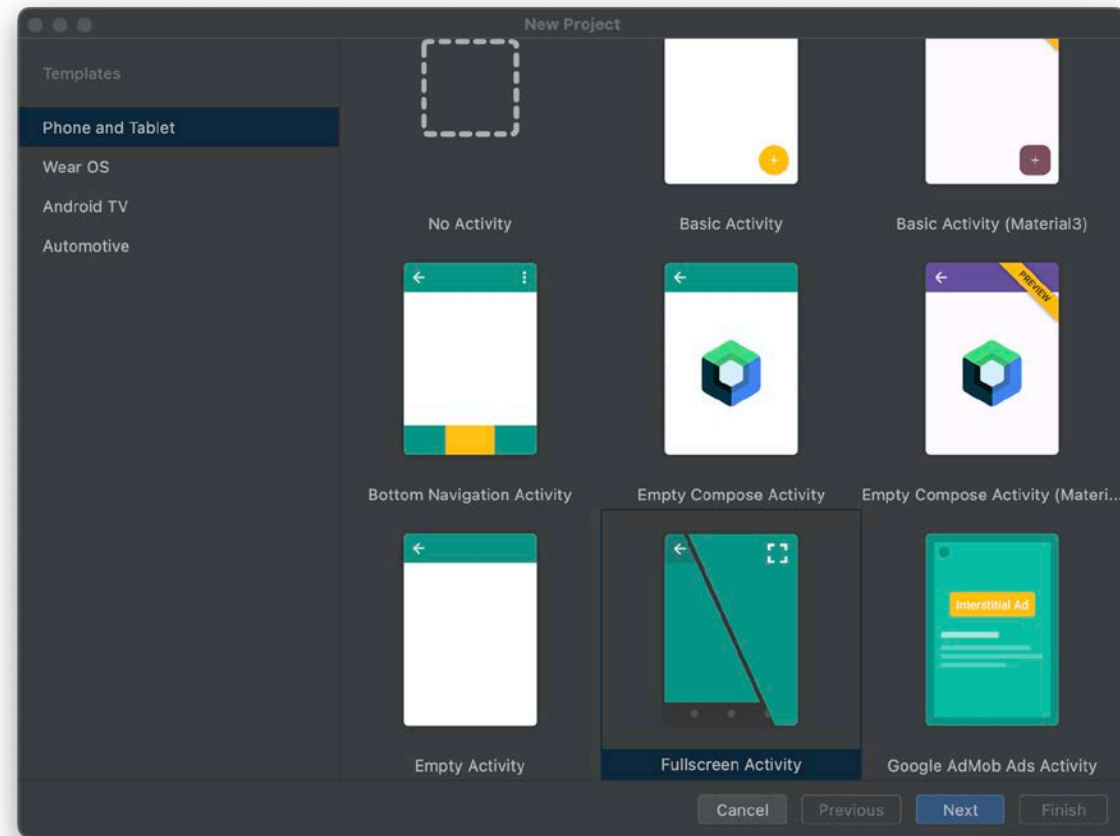
Tópicos

- FullScreen Activity
 - Themes
 - ScreenOrientation
 - App multilingue
 - TextClock
 - Window Flags
 - Broadcast Receiver
 - Android Resources
- Animações

FullScreenActivity

- A FullScreenActivity é um dos templates disponíveis aquando da criação de uma nova Activity.

FullscreenActivity



FullScreenActivity

- No entanto, a FullScreenActivity traz muito código com base no seu template, assim vamos reestruturar tudo e fazer do início



FullscreenActivity

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="?attr/fullscreenBackgroundColor"
    android:theme="@style/ThemeOverlay.12_AppRelogio.FullscreenContainer"
    tools:context=".FullscreenActivity">

</FrameLayout>
```

```
package com.example.a12_apprelogio

import ...

class FullscreenActivity : AppCompatActivity() {

    private lateinit var binding: ActivityFullscreenBinding

    @SuppressWarnings("ClickableViewAccessibility")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityFullscreenBinding.inflate(layoutInflater)
        setContentView(binding.root)
    }
}
```

FullScreenActivity

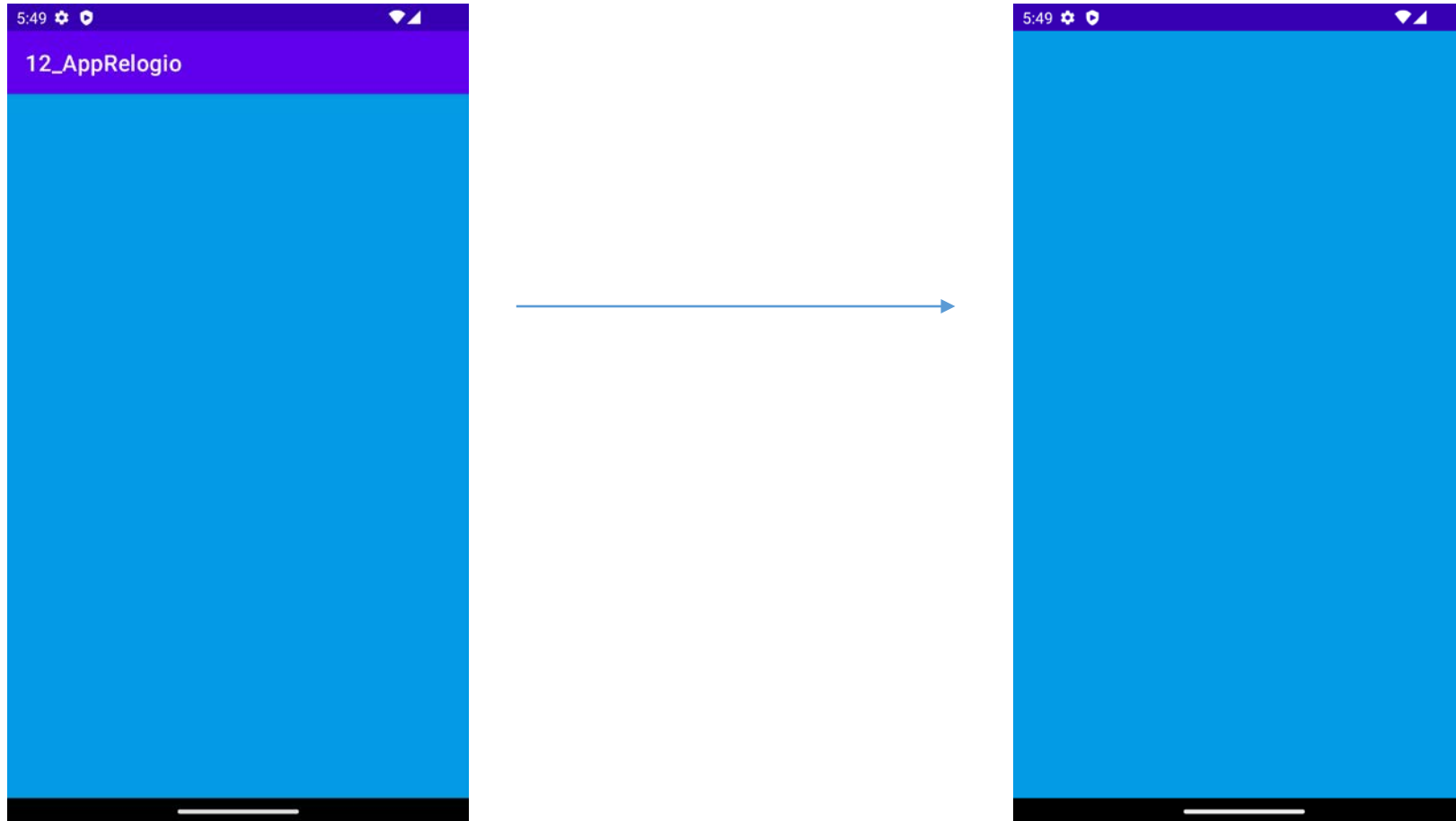
- Como pretendemos que a Activity funcione em FullScreen devemos remover a ActionBar, para isso acedemos ao ficheiro themes.xml, dentro da pasta res/values/themes e substituímos:

`parent="Theme.MaterialComponents.DayNight.DarkActionBar"`

por

`parent="Theme.MaterialComponents.DayNight.NoActionBar"`

FullScreenActivity

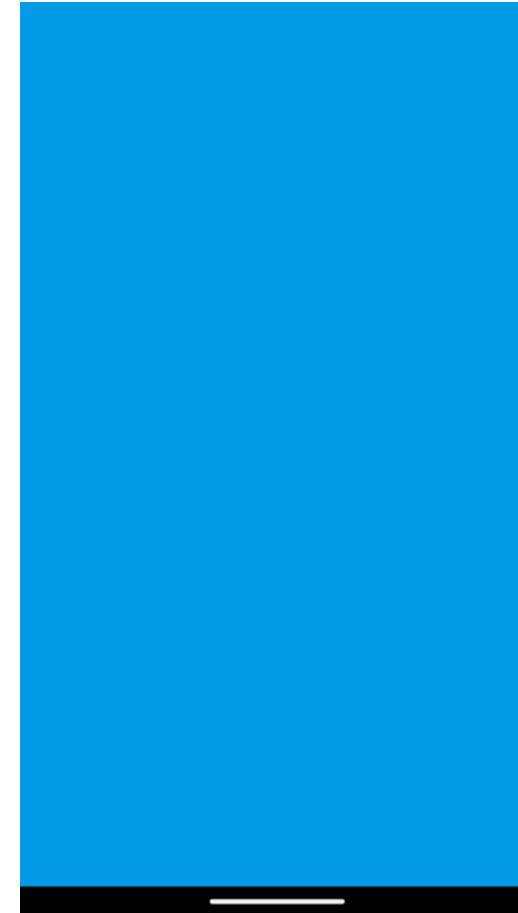


FullscreenActivity

- Podemos ainda retirar a barra azul no topo do ecrã com o código seguinte.
- De notar que a `FLAG_FULLSCREEN` foi descontinuada mas que pode fazer sentido validar a versão que estamos a usar para perceber qual o código mais adequado.

FullScreenActivity

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(binding.root)  
  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {  
        window.insetsController?.hide(WindowInsets.Type.statusBars())  
    } else {  
        window.addFlags(  
            WindowManager.LayoutParams.FLAG_FULLSCREEN  
        )  
    }  
}
```



FullScreenActivity

- Aproveitamos também para adicionar uma nova FLAG para manter a janela sempre ligada

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {  
    window.insetsController?.hide(WindowInsets.Type.statusBars())  
} else {  
    window.setFlags(  
        WindowManager.LayoutParams.FLAG_FULLSCREEN,  
        WindowManager.LayoutParams.FLAG_FULLSCREEN  
    )  
}  
  
window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON)
```

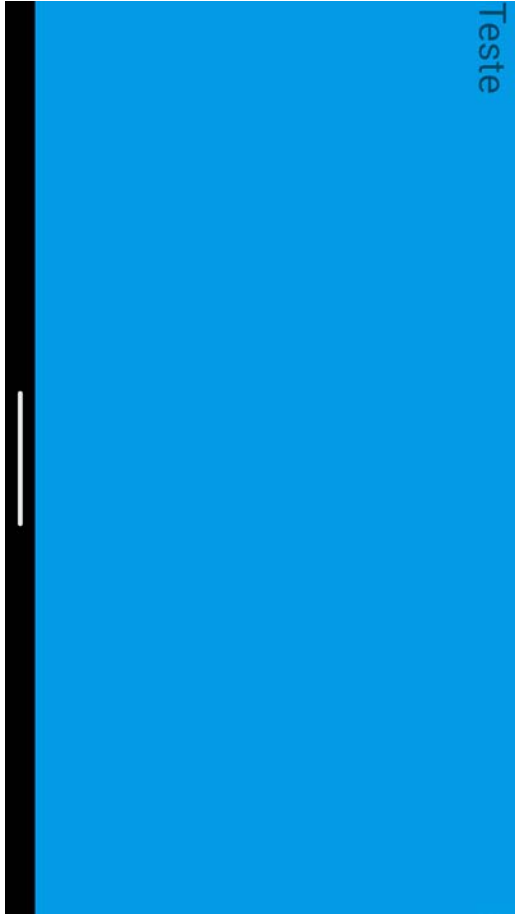
ScreenOrientation

- Vamos alterar a orientação do ecrã de forma a utilizarmos a aplicação na horizontal, para isso acedemos ao ficheiro `AndroidManifest.xml` e dentro da `Activity` que estamos a aceder acrescentamos a propriedade:

`android:screenOrientation`

- Aqui podemos definir vários parâmetros, entre eles:
 - `landscape`: roda a `Activity` quando o dispositivo é rodado para a direita
 - `reverseLandscape`: roda a `Activity` quando o dispositivo é rodado para a esquerda
 - `sensorLandscape`: roda a `Activity` ou para a esquerda ou para a direita consoante a rotação do ecrã.

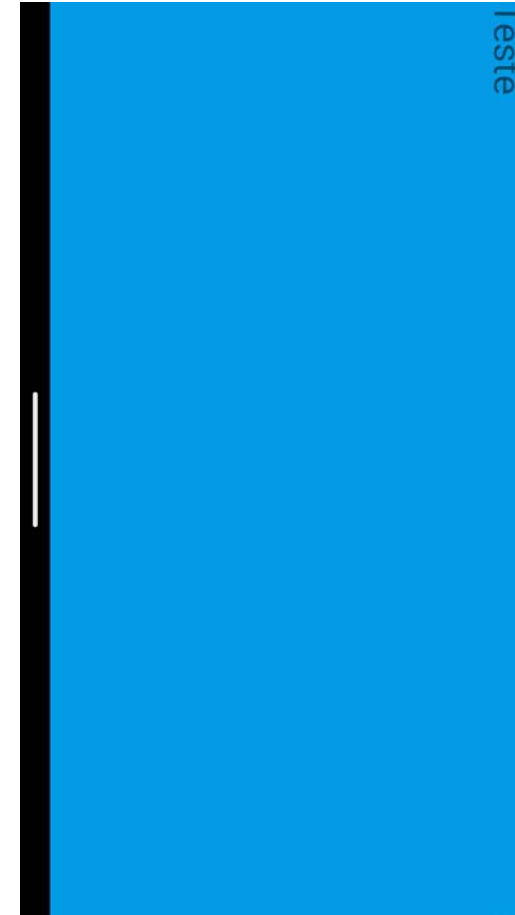
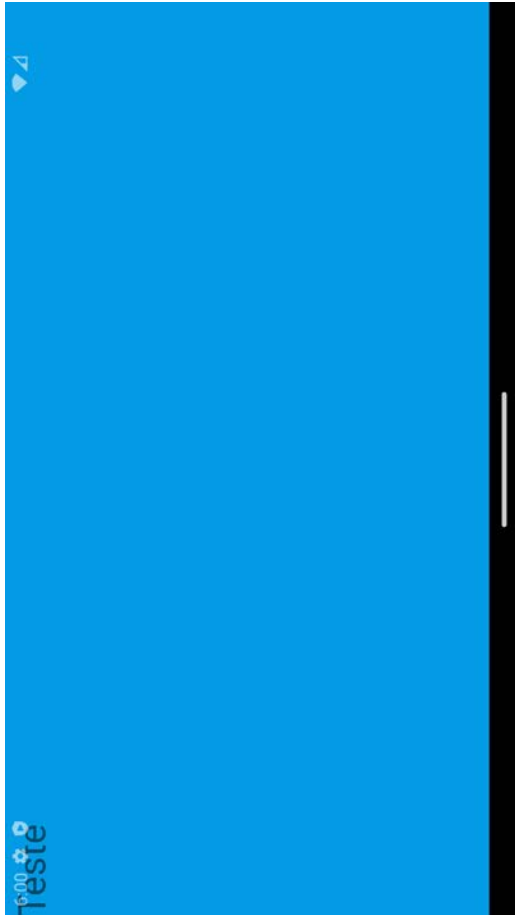
ScreenOrientation (landscape)



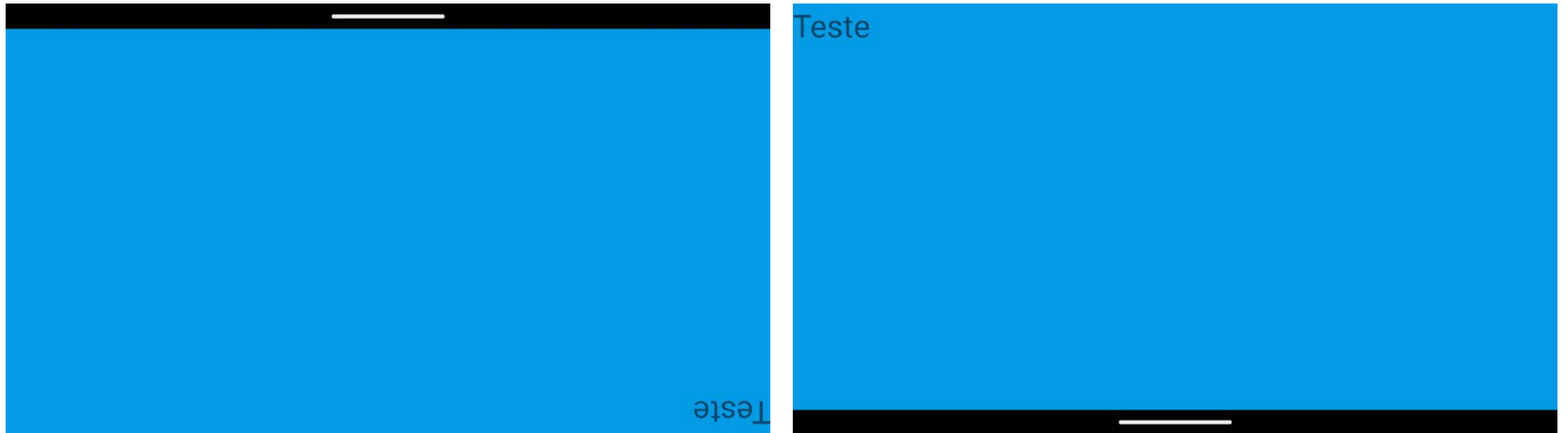
ScreenOrientation (landscape)



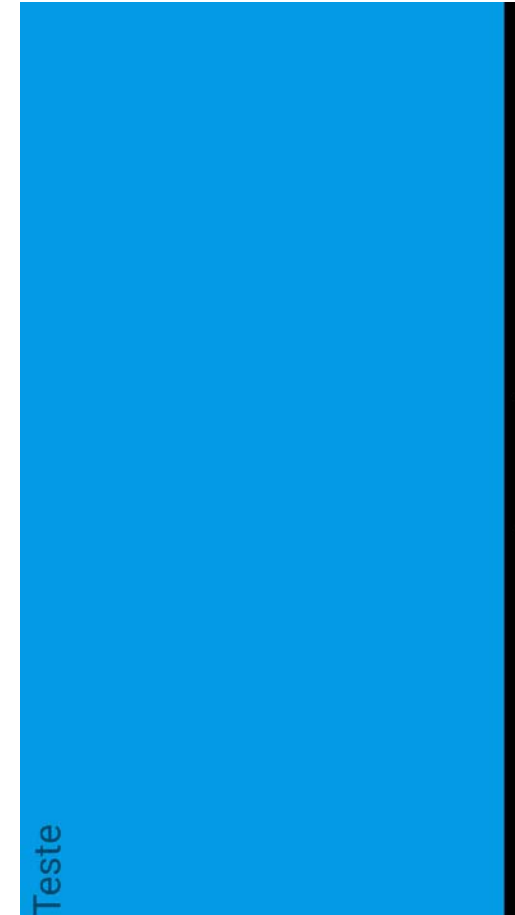
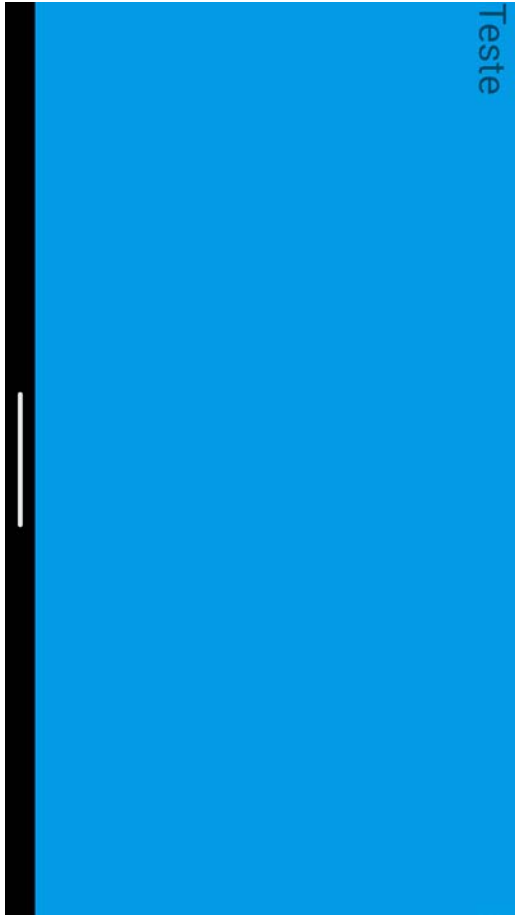
ScreenOrientation (reverseLandscape)



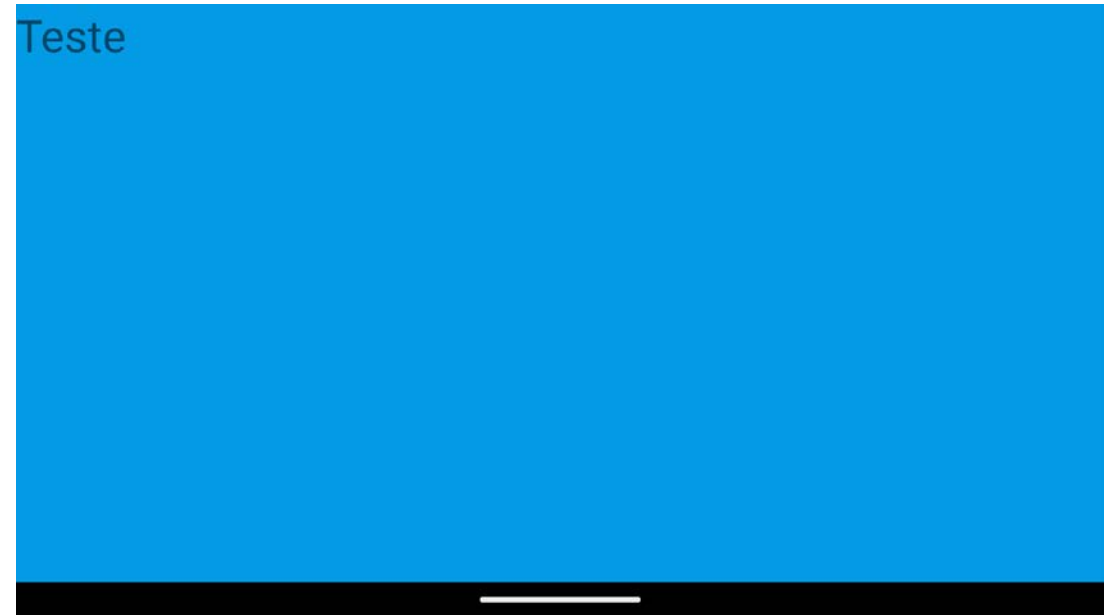
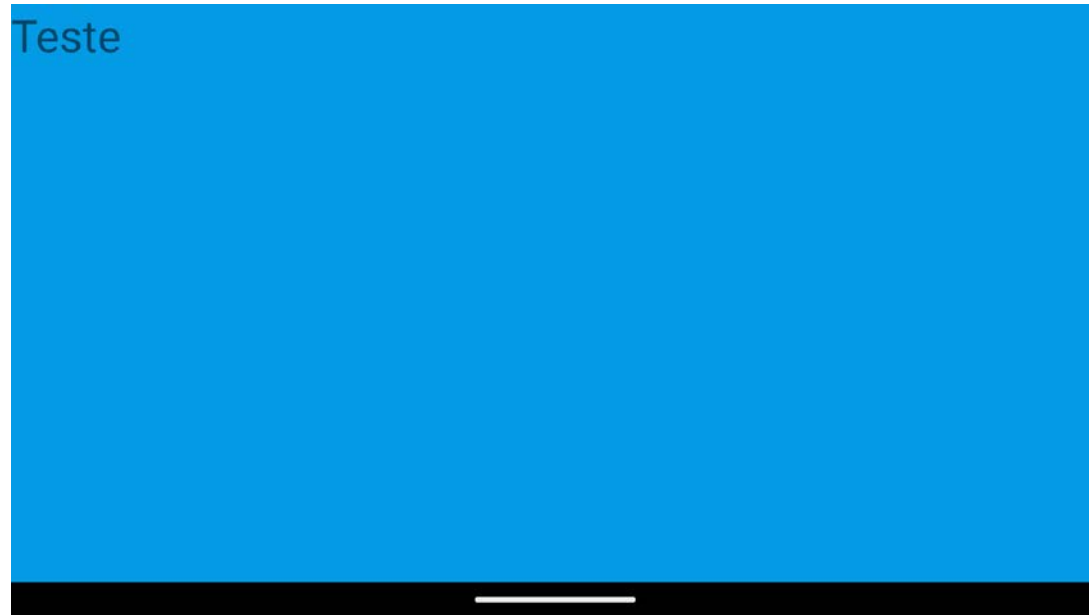
ScreenOrientation (reverseLandscape)



ScreenOrientation (sensorLandscape)



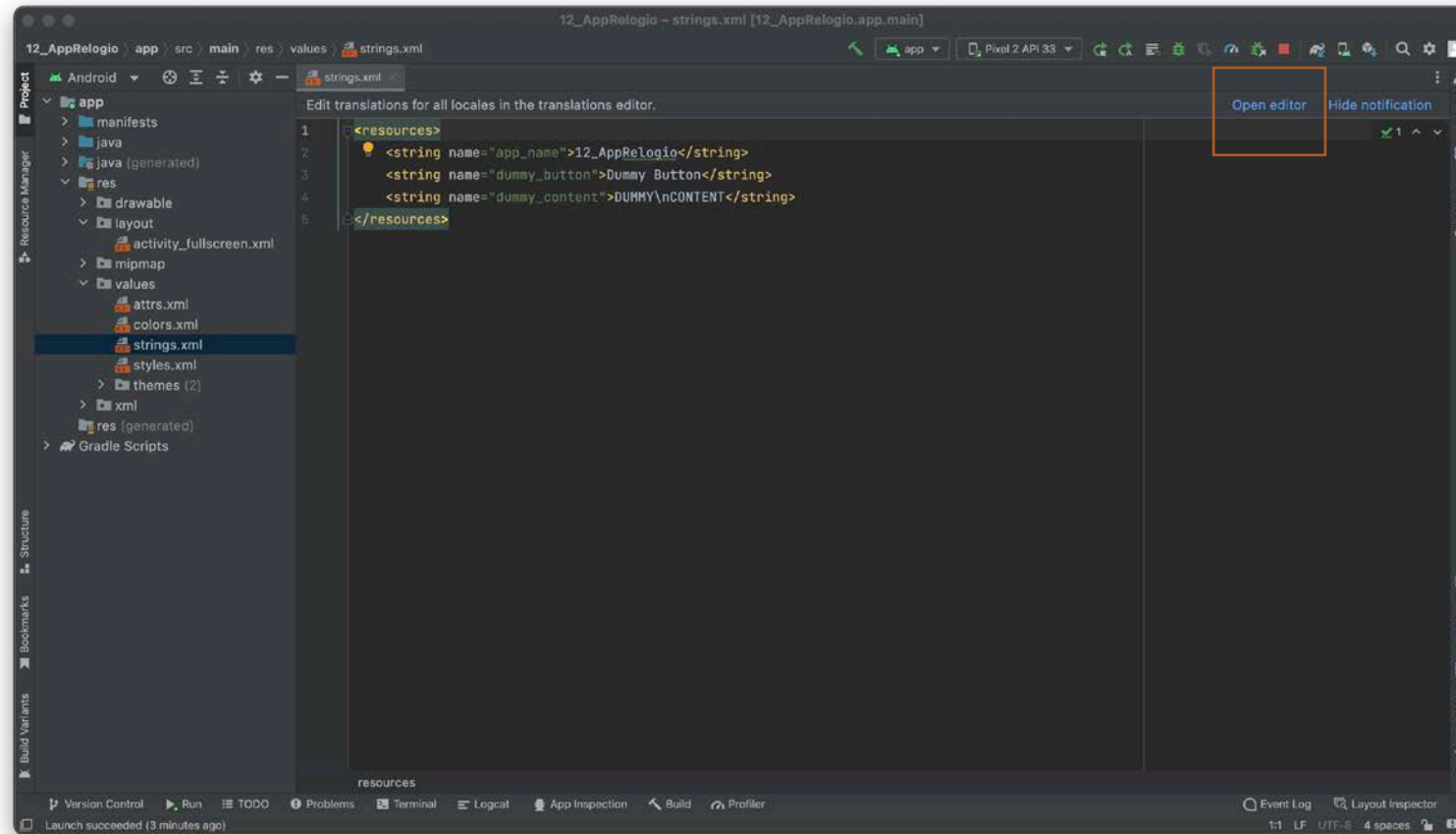
ScreenOrientation (sensorLandscape)



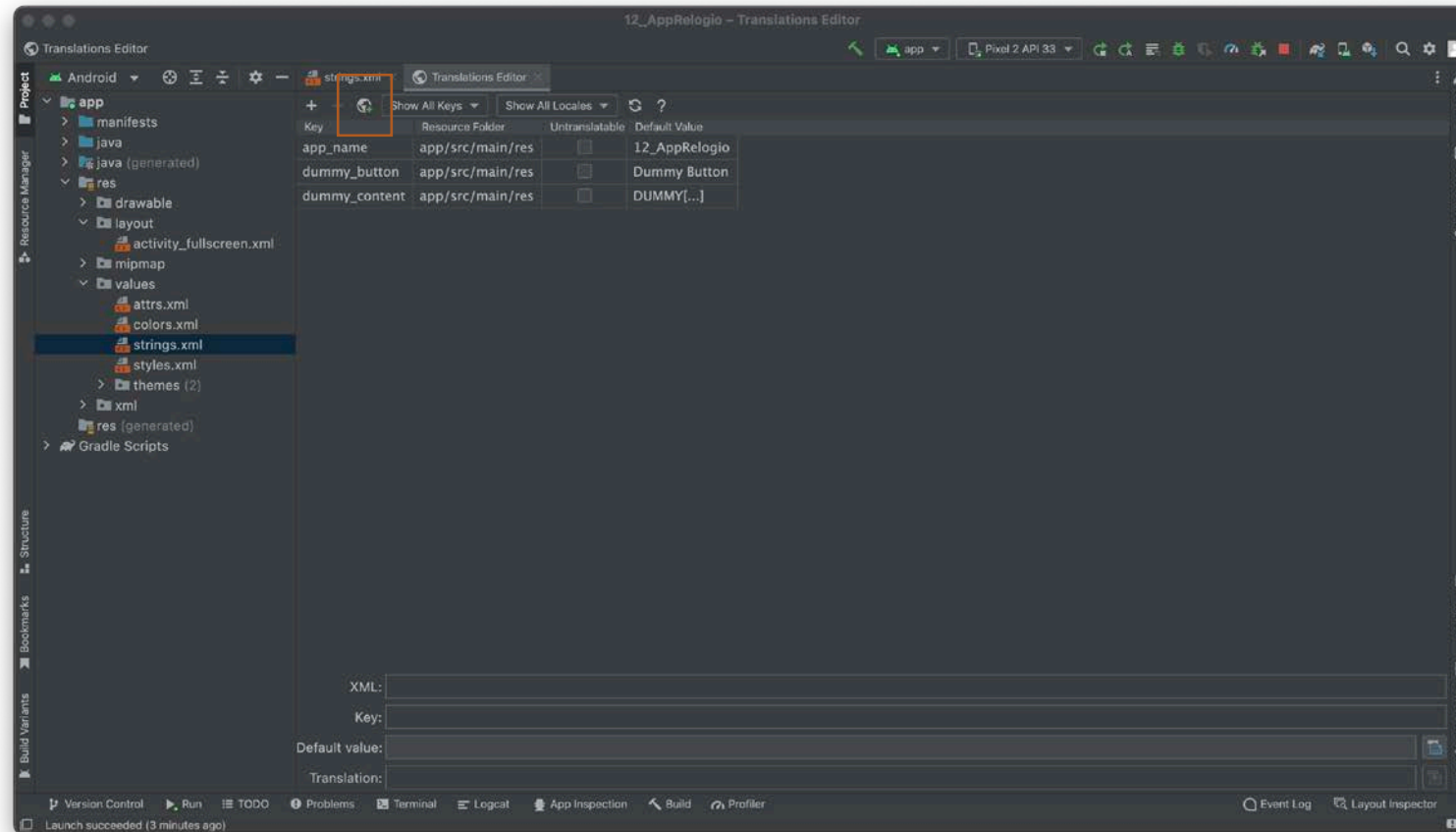
App multilingue

- Para a criação de aplicações em várias línguas podemos abrir o editor de strings apresentado no ficheiro strings.xml (pasta res/values).
- Para isso clicamos no link Open Editor nesse ficheiro e depois no globo + para adicionar novo idioma

App multilingue



App multilingue



App multilingue

- Neste exemplo adicionamos duas línguas (PT e ES) e necessitamos traduzir todas as strings existentes ou marcar como Untranslatable. Este último caso vamos fazer apenas com o nome da APP.

App multilingue

Show All Keys

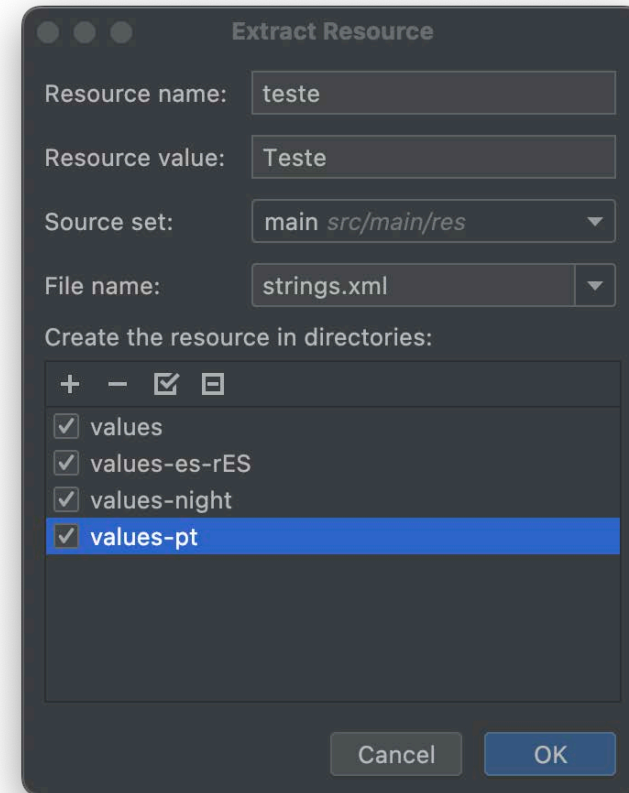
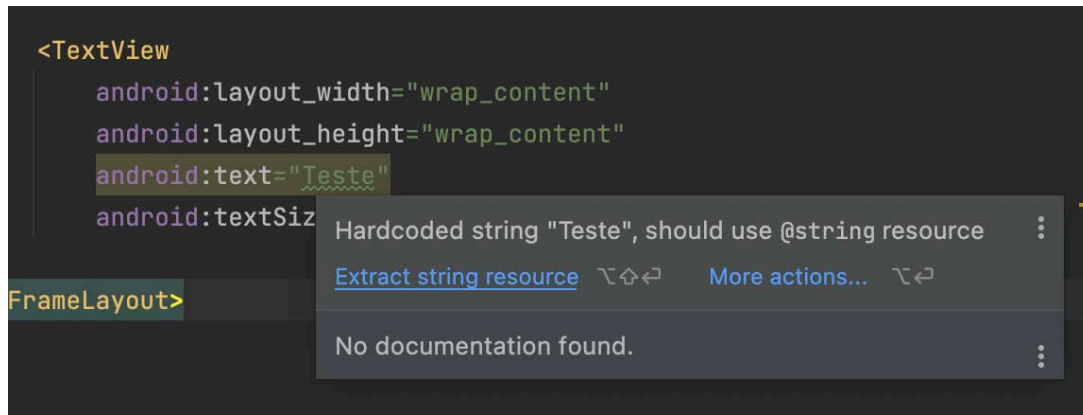
Show All Locales

Key	Resource Folder	Untranslatable	Default Value	Portuguese...	Spanish (e...
app_name	app/src/main/res	<input checked="" type="checkbox"/>	12_AppRelogio		
dummy_button	app/src/main/res	<input type="checkbox"/>	Dummy Button	Botão Fictício	Botón
dummy_content	app/src/main/res	<input type="checkbox"/>	DUMMY[...]	Fictício	Ficticio

App multilingue

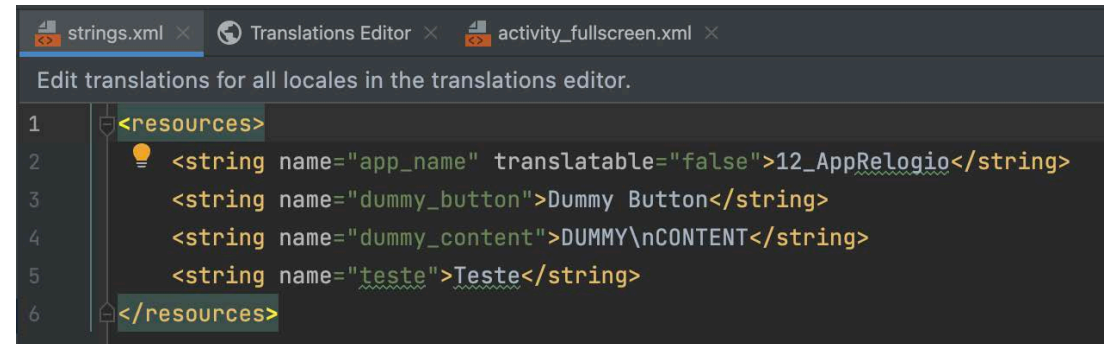
- No XML da Activity foi criada uma TextView com o valor “Teste”. Vamos extrair o valor para os arquivo de strings

App multilingue



App multilingue

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/teste"
    android:textSize="30sp" />
```



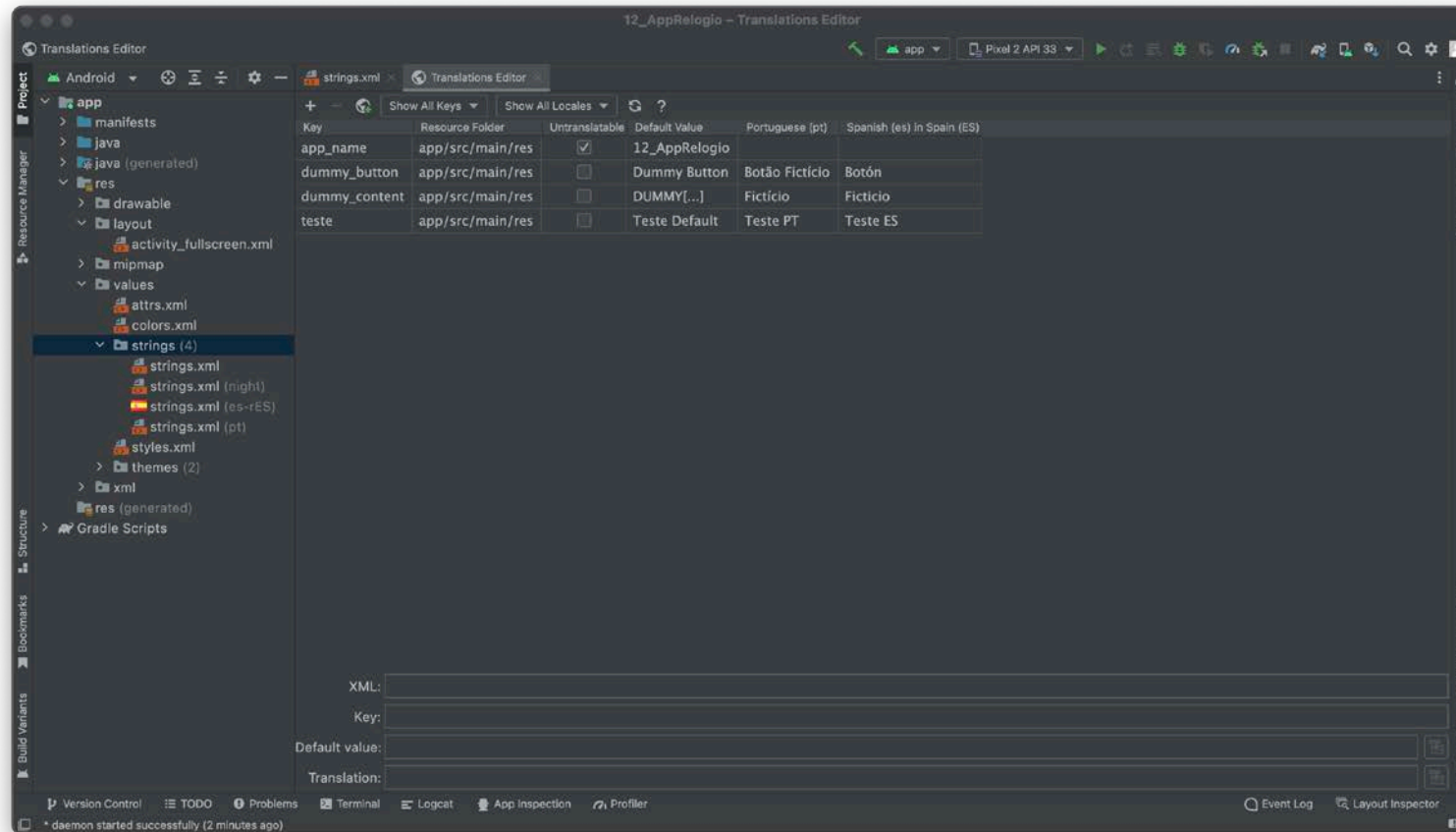
The screenshot shows the Translations Editor in Android Studio. The top bar has tabs for 'strings.xml', 'Translations Editor', and 'activity_fullscreen.xml'. Below the tabs, a message says 'Edit translations for all locales in the translations editor.' The main area shows the XML code for strings.xml with line numbers 1 through 6 on the left. The code is as follows:

```
1 <resources>
2   <string name="app_name" translatable="false">12_AppRelogio</string>
3   <string name="dummy_button">Dummy Button</string>
4   <string name="dummy_content">DUMMY\nCONTENT</string>
5   <string name="teste">Teste</string>
6 </resources>
```

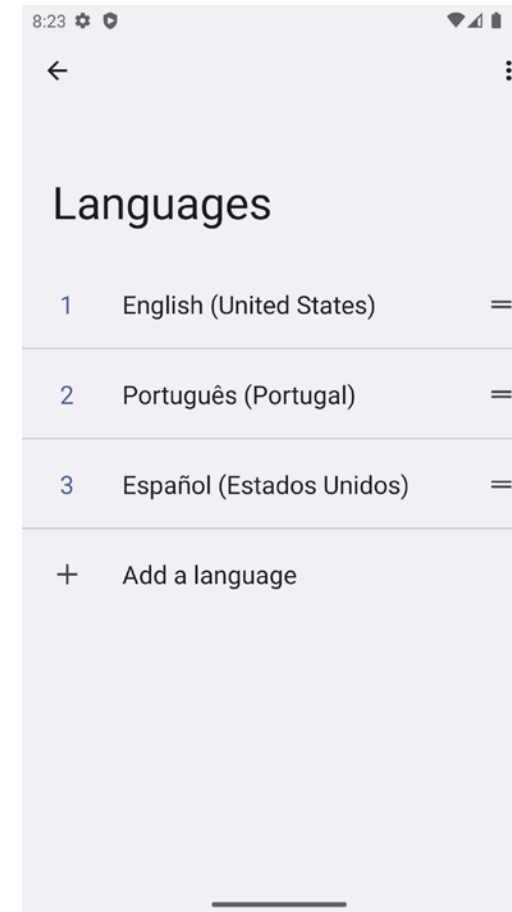
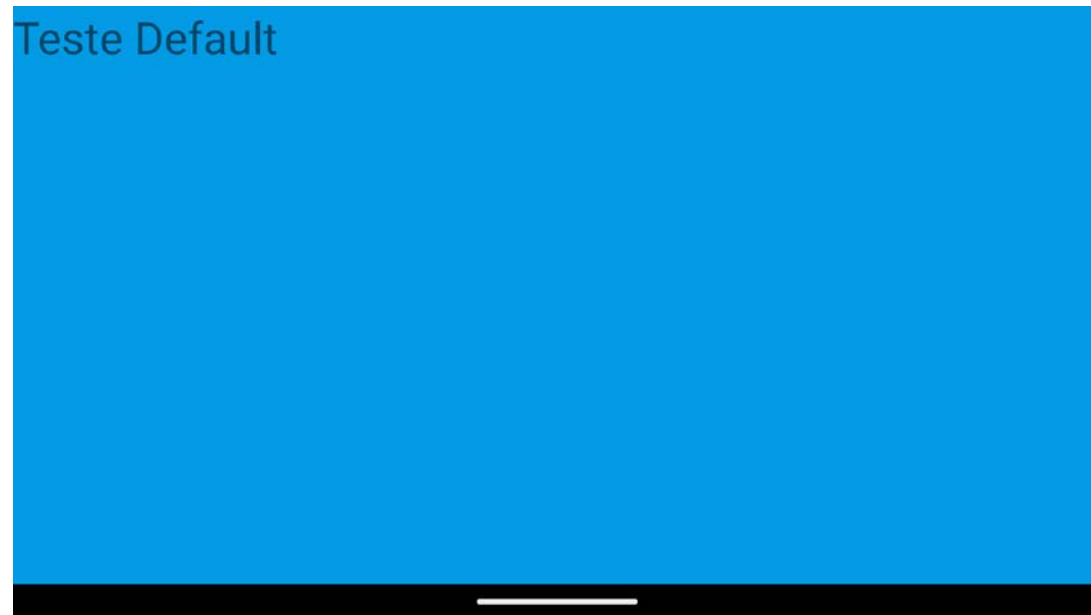
App multilingue

- No editor de idiomas vamos alteramos o valor nas outras línguas
- Dependendo do idioma do emulador, será carregada a tradução correspondente.
- Após executar a aplicação, altere a linguagem do emulador para testar o comportamento da aplicação

App multilingue



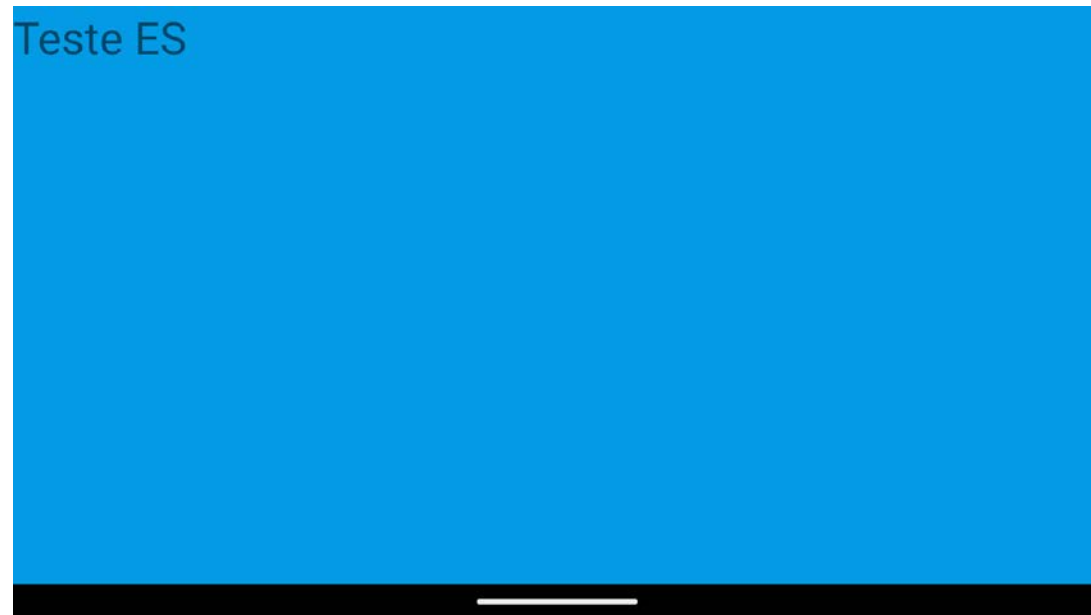
App multilingue



App multilingue

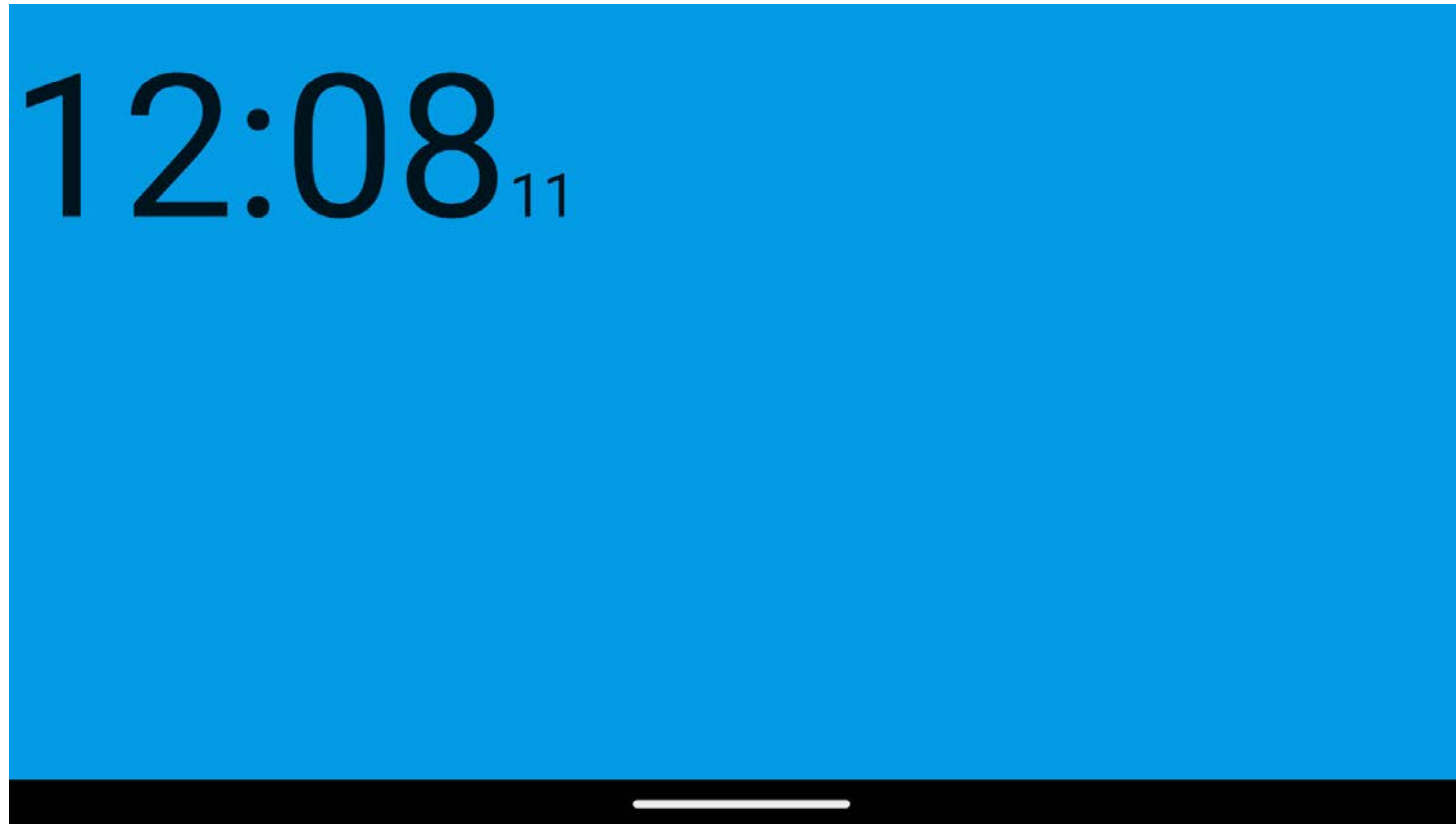


App multilingue



TextClock

- Vamos criar uma aplicação que funcione como relógio



TextClock

- Para isso necessitamos de colocar no layout o elemento TextClock que vai ler a hora do equipamento que estamos a utilizar.
- No caso vamos utilizar dois elementos, um para as horas e minutos e outro para os segundos

```
<TextClock
    android:id="@+id/clock_horas_minutos"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:format12Hour="hh:mm"
    android:format24Hour="hh:mm"
    android:text="00:00"
    android:textSize="100sp" />

<TextClock
    android:id="@+id/clock_segundos"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:format12Hour="ss"
    android:format24Hour="ss"
    android:text="00"
    android:textSize="30sp" />
```

BroadcastReceiver

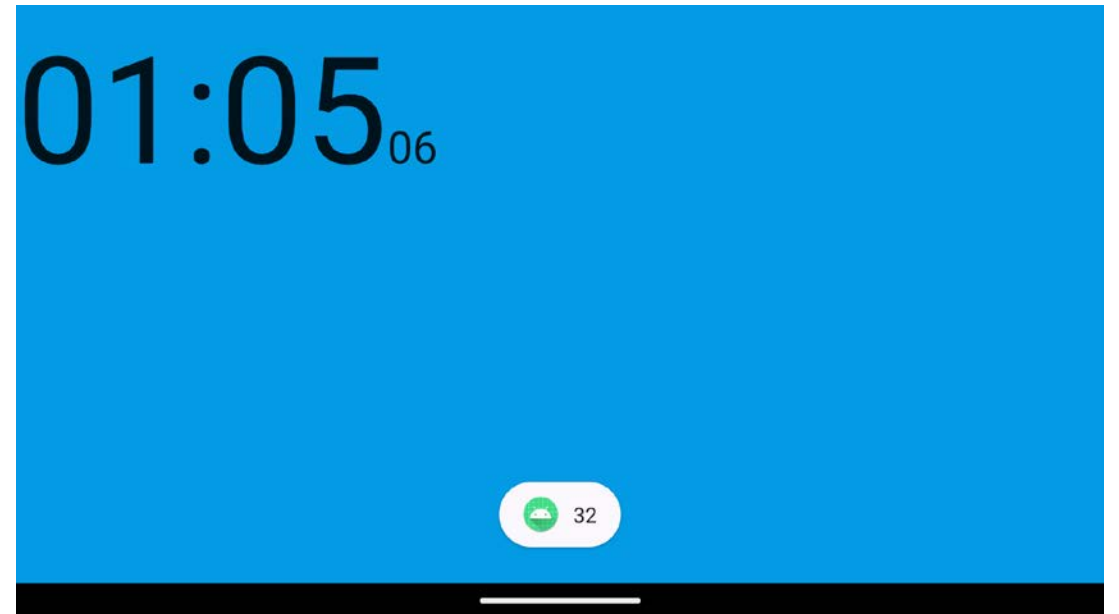
- Vamos acrescentar novas funcionalidades à nossa aplicação relógio. Para começar adicionamos o nível de bateria aos elementos visíveis ao relógio, usando um BroadcastReceiver.

BroadcastReceiver

- Com o BroadcastReceiver vamos conseguir receber um “alerta” sempre que o nível da bateria é alterado, assim necessitamos de aceder ao registerReceiver.
- O IntentFilter permite filtrarmos o tipo de evento que pretendemos analisar, neste caso apenas a alteração do nível de bateria.
- Neste exemplo começamos por colocar o valor num Toast e depois vamos passar para a interface

BroadcastReceiver

```
val bateriaReceiver: BroadcastReceiver = object : BroadcastReceiver() {  
    override fun onReceive(context: Context?, intent: Intent?) {  
        if (intent != null) {  
            val nivel: Int = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, defaultValue: 0)  
            Toast.makeText(applicationContext, nivel.toString(), Toast.LENGTH_SHORT).show()  
        }  
    }  
}  
  
registerReceiver(bateriaReceiver, IntentFilter(Intent.ACTION_BATTERY_CHANGED))
```



BroadcastReceiver

- Vamos reformular o layout para incluir o valor da bateria

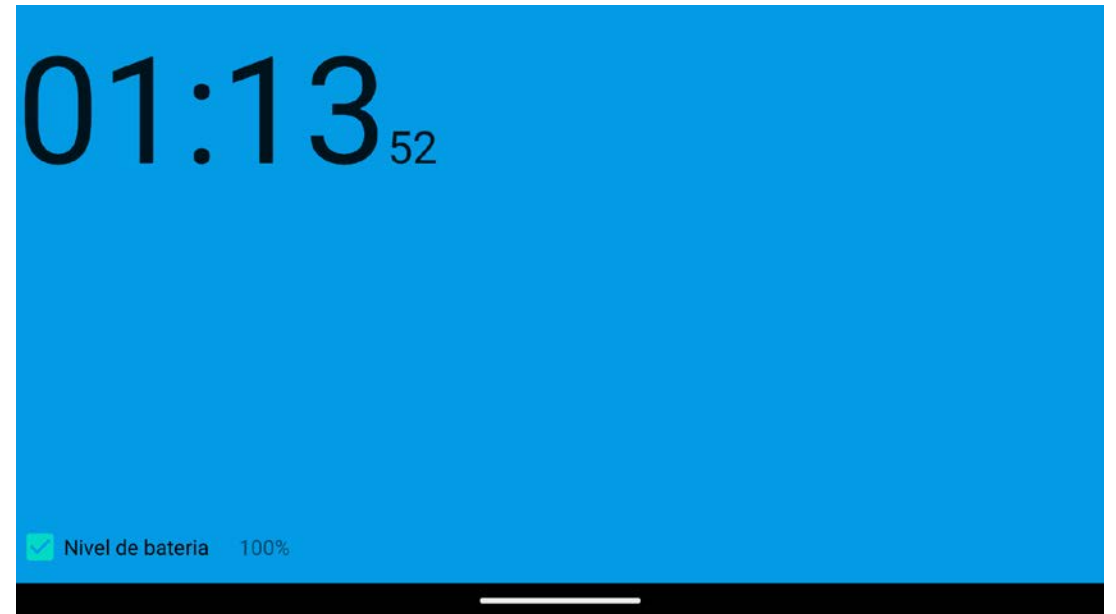
BroadcastReceiver

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:background="?attr/fullscreenBackgroundColor"
7      android:theme="@style/ThemeOverlay.12_AppRelogio.FullscreenContainer"
8      tools:context=".FullscreenActivity">
9
10     <LinearLayout
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:orientation="horizontal">
14
15         <TextClock
16             android:id="@+id/clock_horas_minutos"
17             android:layout_width="wrap_content"
18             android:layout_height="wrap_content"
19             android:format12Hour="hh:mm"
20             android:format24Hour="hh:mm"
21             android:text="00:00"
22             android:textSize="100sp" />
23
```

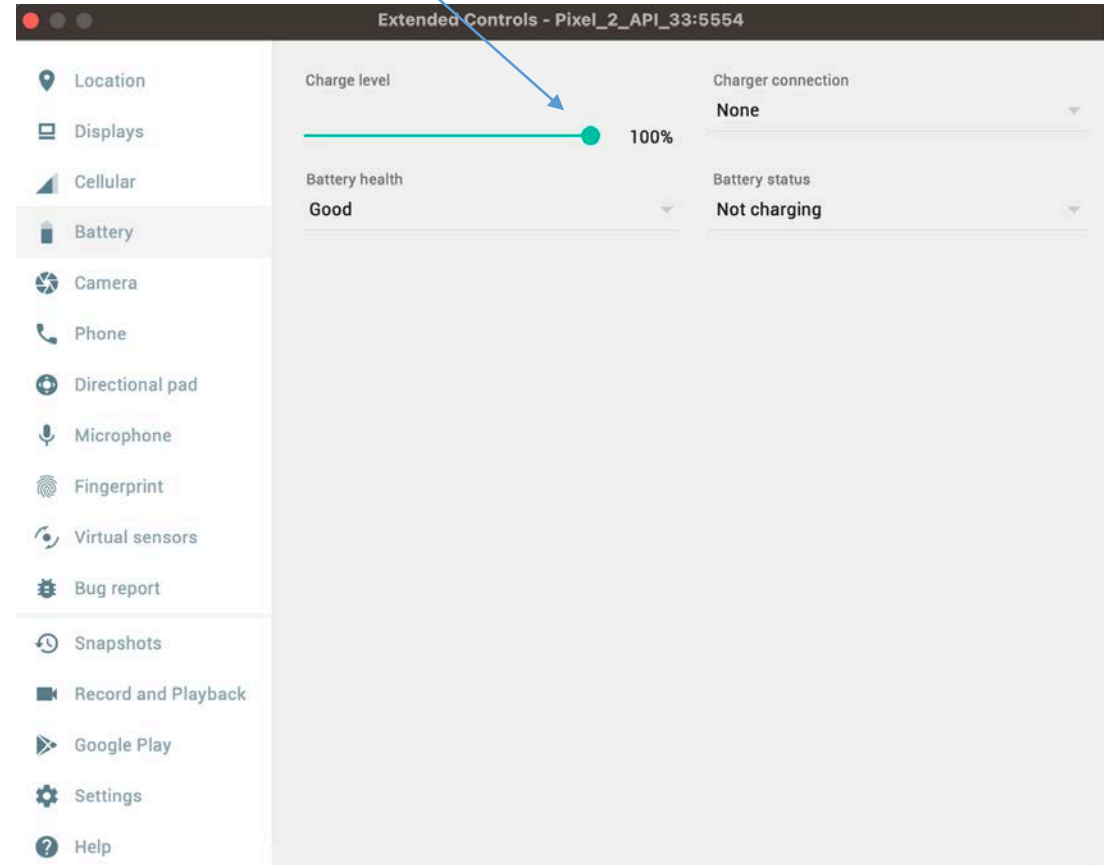
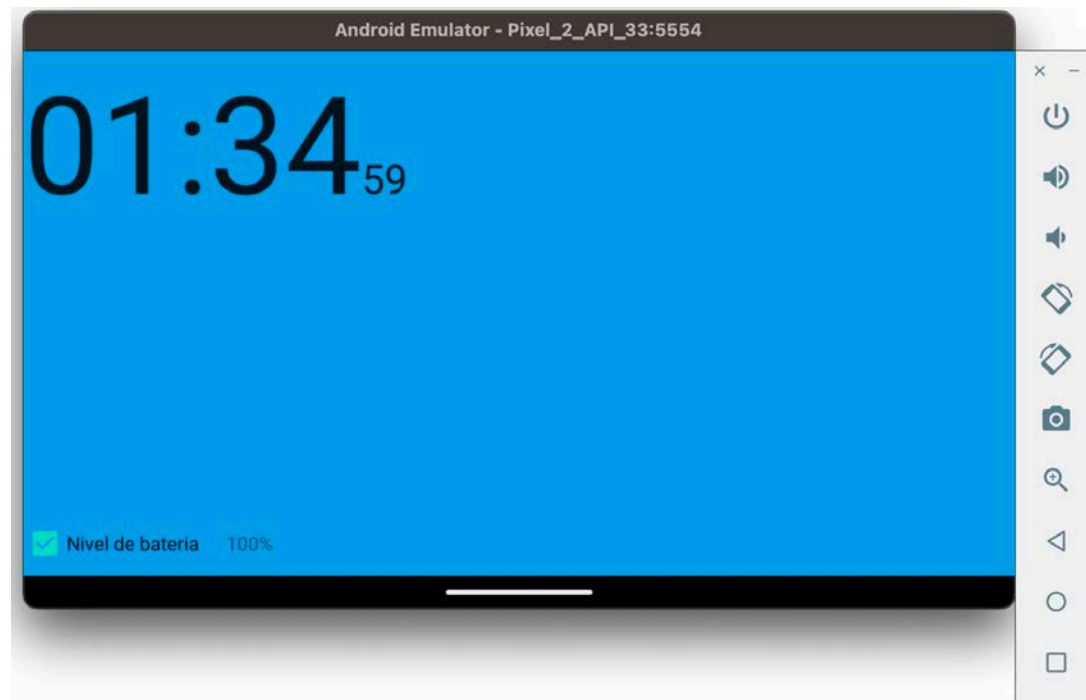
```
24     <TextClock
25         android:id="@+id/clock_segundos"
26         android:layout_width="wrap_content"
27         android:layout_height="wrap_content"
28         android:format12Hour="ss"
29         android:format24Hour="ss"
30         android:text="00"
31         android:textSize="30sp" />
32     </LinearLayout>
33     <LinearLayout
34         android:layout_width="match_parent"
35         android:layout_height="wrap_content"
36         android:layout_alignParentBottom="true">
37
38         <CheckBox
39             android:id="@+id/check_nivel_bateria"
40             android:layout_width="wrap_content"
41             android:layout_height="wrap_content"
42             android:checked="true"
43             android:text="@string/nivel_de_bateria" />
44
45         <TextView
46             android:id="@+id/text_nivel_bateria"
47             android:layout_width="wrap_content"
48             android:layout_height="wrap_content"
49             android:layout_marginStart="20dp"
50             android:text="@string/nivel_de_bateria" />
51     </LinearLayout>
52 </RelativeLayout>
```

BroadcastReceiver

```
val bateriaReceiver: BroadcastReceiver = object : BroadcastReceiver() {  
    override fun onReceive(context: Context?, intent: Intent?) {  
        if (intent != null) {  
            val nivel: Int = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, defaultValue: 0)  
            //Toast.makeText(applicationContext, nivel.toString(), Toast.LENGTH_SHORT).show()  
            binding.textNivelBateria.text = nivel.toString()  
        }  
    }  
}
```



BroadcastReceiver



Nível da bateria

- Agora queremos dar a opção ao utilizador de mostrar ou não o nível da bateria marcando ou desmarcando o CheckBox criado anteriormente

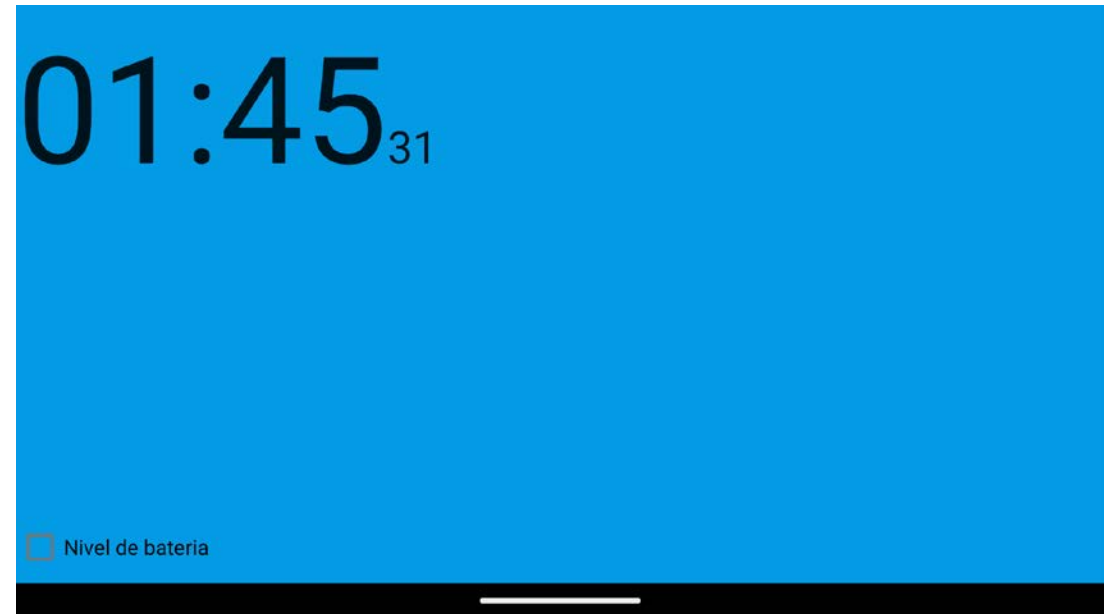
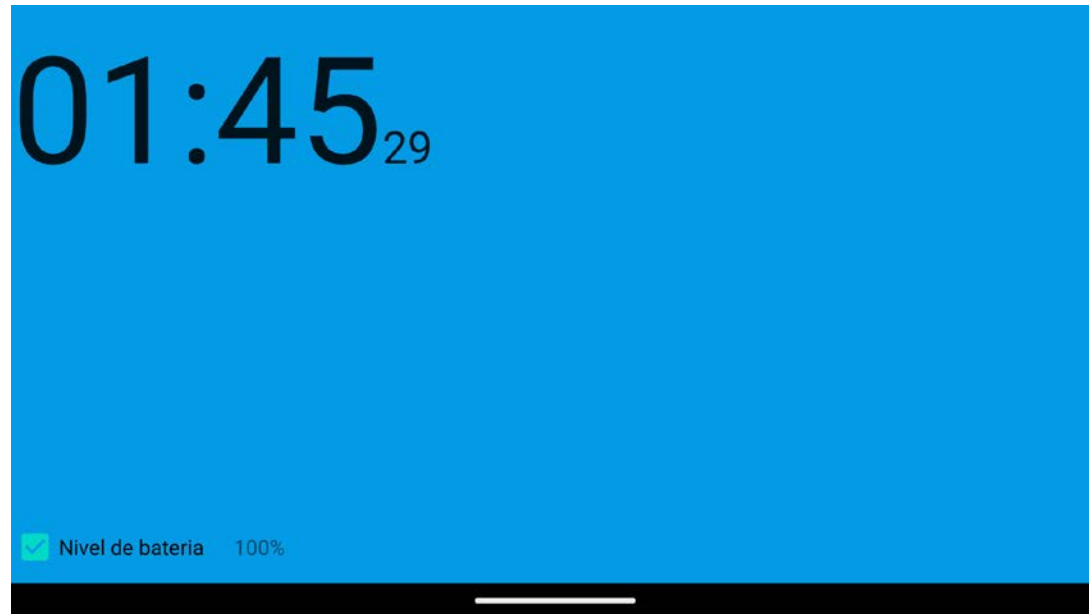
Nível da bateria

```
private lateinit var binding: ActivityFullscreenBinding
private var isChecked = true
```

```
registerReceiver(bateriaReceiver, IntentFilter(Intent.ACTION_BATTERY_CHANGED))

binding.checkNivelBateria.setOnClickListener { it: View!
    if (isChecked) {
        isChecked = false
        binding.checkNivelBateria.isChecked = false
        binding.textNivelBateria.visibility = View.GONE
    } else {
        isChecked = true
        binding.checkNivelBateria.isChecked = true
        binding.textNivelBateria.visibility = View.VISIBLE
    }
}
```

Nível da bateria



Layout

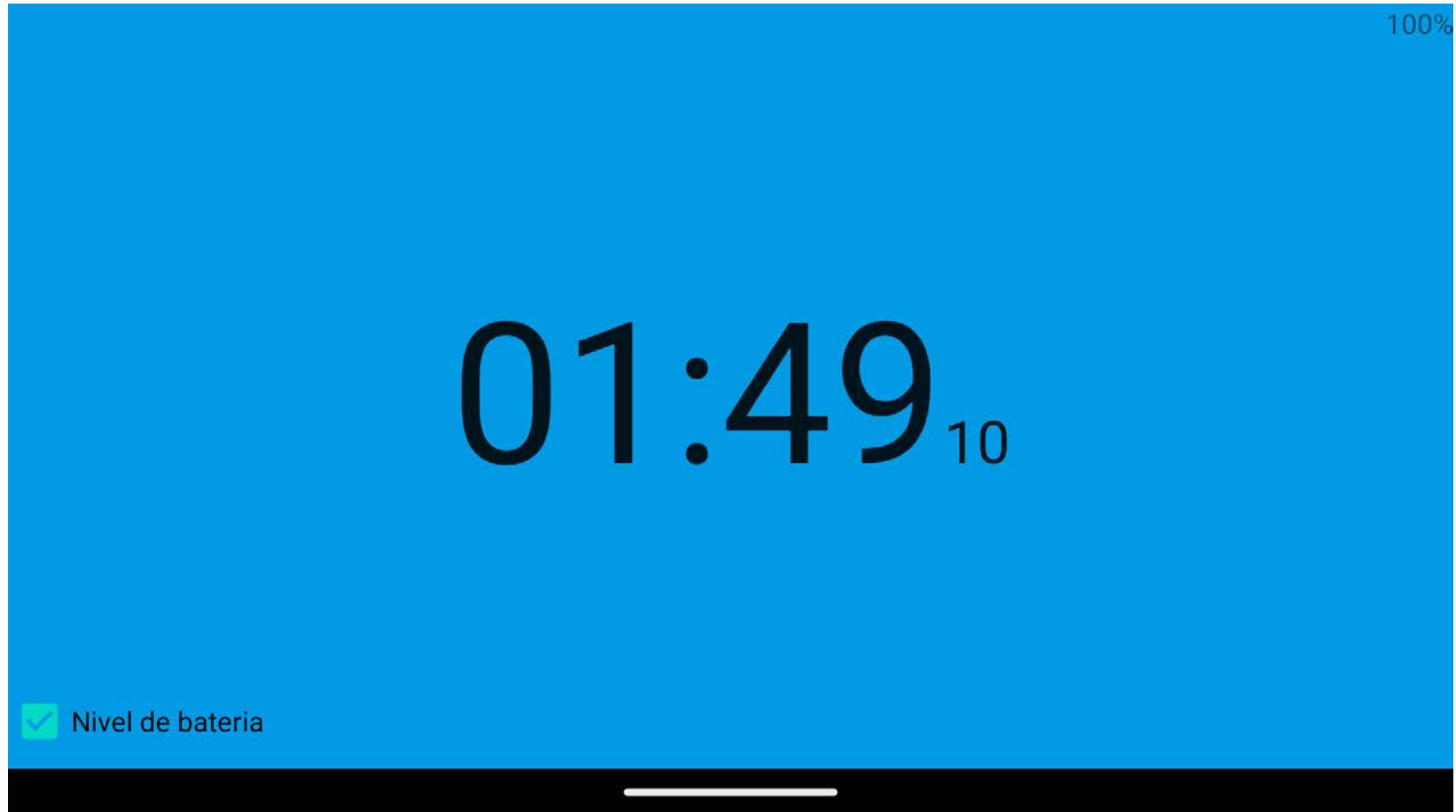
- Vamos agora reajustar novamente o layout para separar os vários elementos e colocar o CheckBox como sendo um menu

Layout

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:background="?attr/fullscreenBackgroundColor"
7      android:theme="@style/ThemeOverlay.12_AppRelogio.FullscreenContainer"
8      tools:context=".FullscreenActivity">
9
10     <TextView
11         android:id="@+id/text_nivel_bateria"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:layout_alignParentTop="true"
15         android:layout_alignParentEnd="true"
16         android:layout_marginStart="20dp"
17         android:text="@string/nivel_de_bateria" />
18
19     <LinearLayout
20         android:layout_width="match_parent"
21         android:layout_height="wrap_content"
22         android:layout_centerInParent="true"
23         android:gravity="center|bottom"
24         android:orientation="horizontal">
25
26         <TextClock
27             android:id="@+id/clock_horas_minutos"
28             android:layout_width="wrap_content"
```

```
28         android:layout_width="wrap_content"
29         android:layout_height="wrap_content"
30         android:format12Hour="hh:mm"
31         android:format24Hour="hh:mm"
32         android:text="00:00"
33         android:textSize="100sp" />
34
35     <TextClock
36         android:id="@+id/clock_segundos"
37         android:layout_width="wrap_content"
38         android:layout_height="wrap_content"
39         android:format12Hour="ss"
40         android:format24Hour="ss"
41         android:text="00"
42         android:textSize="30sp" />
43 </LinearLayout>
44 <LinearLayout
45     android:layout_width="match_parent"
46     android:layout_height="wrap_content"
47     android:layout_alignParentBottom="true">
48
49     <CheckBox
50         android:id="@+id/check_nivel_bateria"
51         android:layout_width="wrap_content"
52         android:layout_height="wrap_content"
53         android:checked="true"
54         android:text="@string/nivel_de_bateria" />
55 </LinearLayout>
56 </RelativeLayout>
```

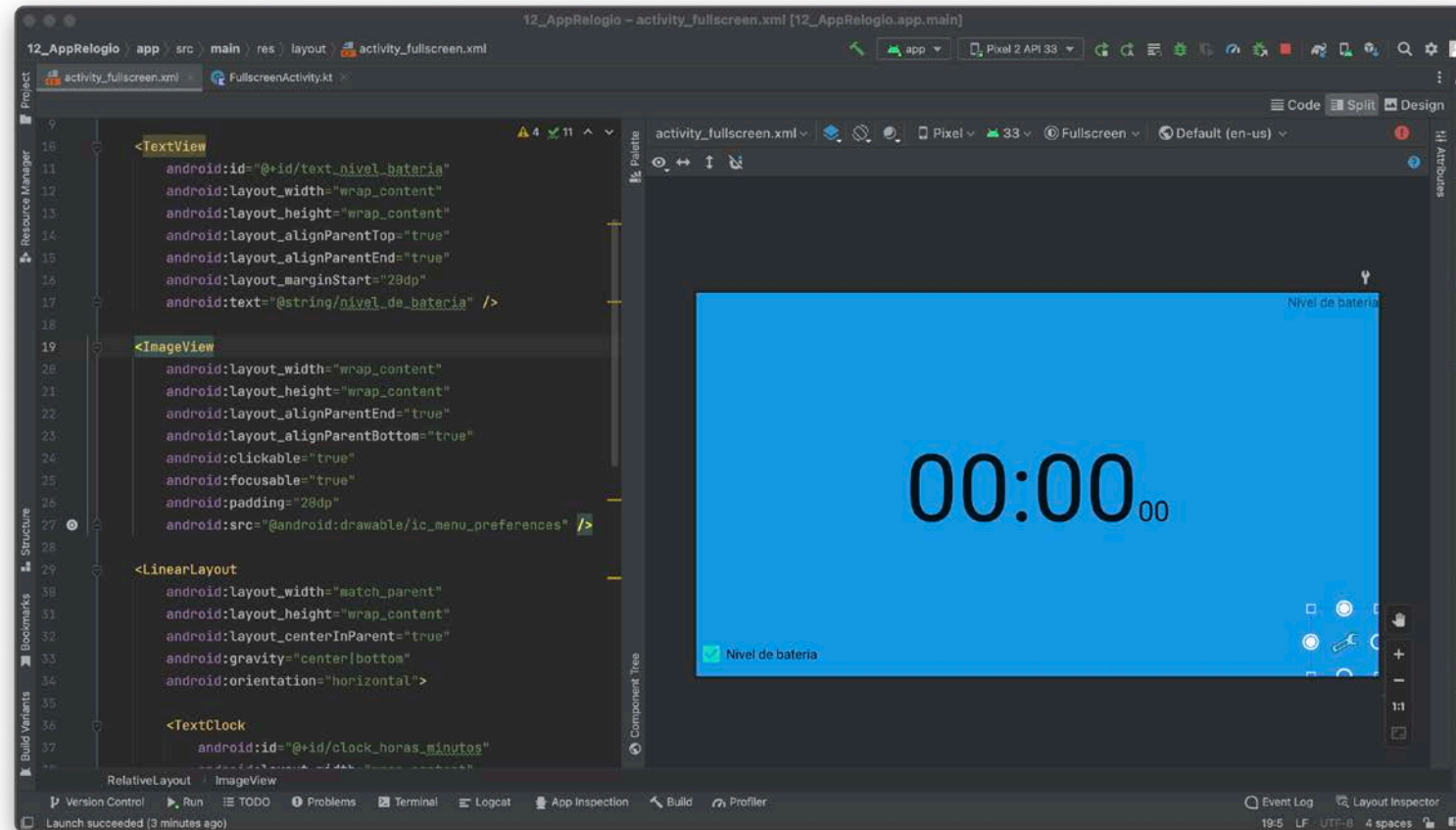
Layout



Recursos Android

- Vamos agora alterar a barra inferior. Pretendemos agora que o CheckBox nem sempre esteja visível mas sim que ao clicar num botão de preferências seja apresentado o CheckBox.
- Começamos por acrescentar uma ImageView com um botão já existente no Android, o `ic_menu_preferences`

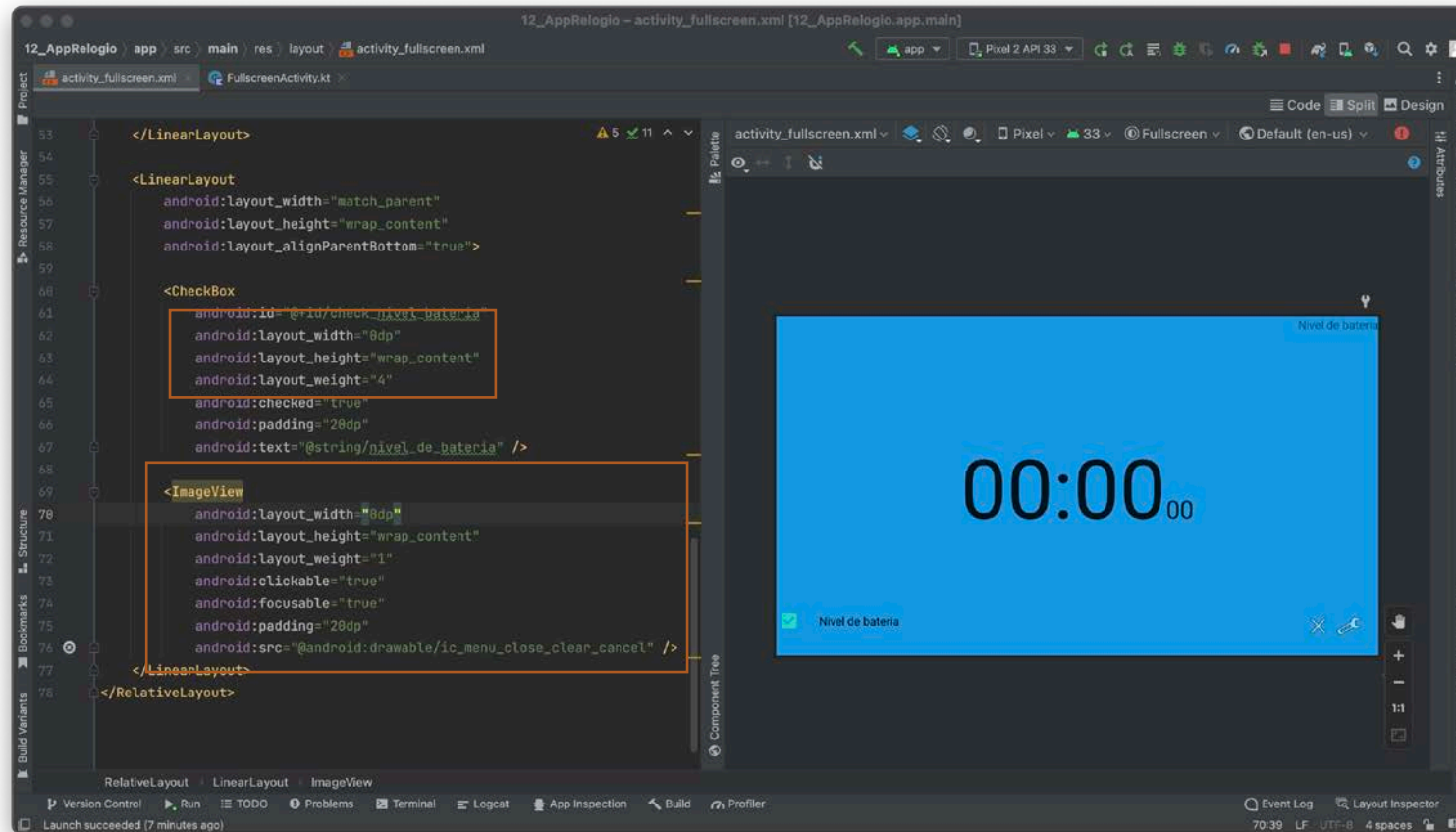
Recursos Android



Animação

- Agora vamos criar uma animação para que ao clicar no ImageView de preferências apareça um menu com a CheckBox e que dentro desse menu tenha também um botão de fechar para esconder o menu. Iremos implementar esse botão de forma semelhante ao botão de preferências.
- Para dividirmos um LinearLayout em partes e conseguirmos que um elemento ocupe uma parte do todo podemos usar o atributo weight.

Animação



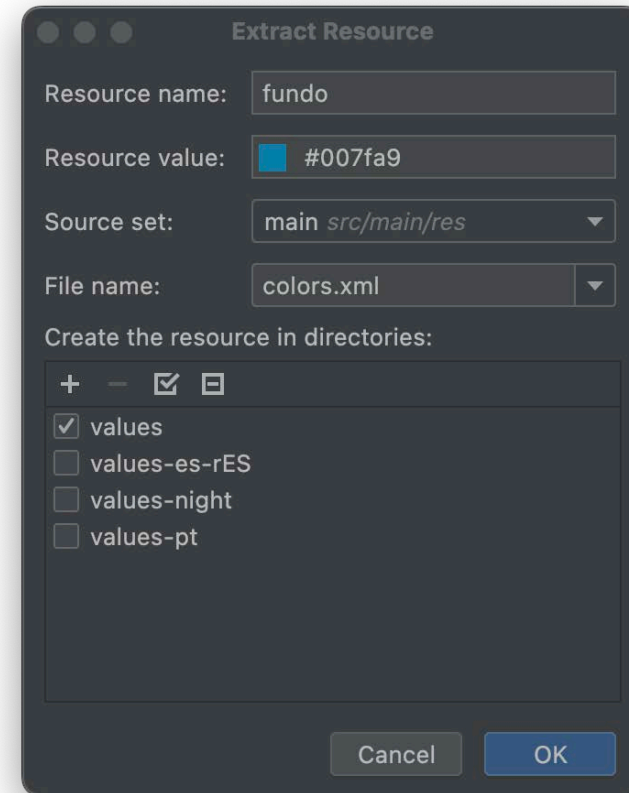
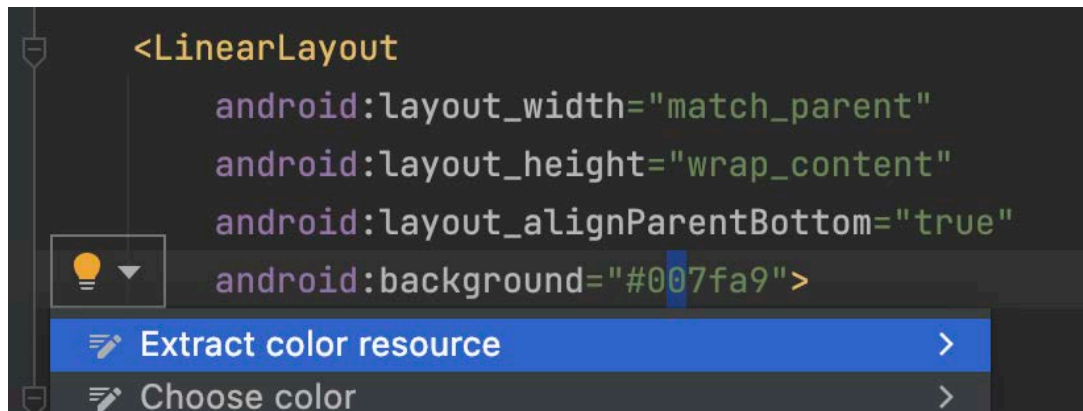
Animação

- Assim estamos a dizer que dividimos a o width do LinearLayout em 5 partes iguais, 4 para o CheckBox e uma para a ImageView. Como estamos a dividir o tamanho pelo Layout faz sentido que o parâmetro width de cada elemento fique a 0.
- Foi adicionado também um padding de 30dp na CheckBox.
- Com o ic_menu_close_clear_cancel conseguimos colocar o botão X na posição pretendida.

Animação

- Vamos ainda adicionar uma cor de fundo ao `LinearLayout` para separar melhor o menu de todo o layout.
- Este valor deve ser extraído (à semelhança das strings) para o ficheiro `colors.xml`

Animação



Animação

- De seguida atribuímos IDs para os dois ImageView e para o LinearLayout que irá conter o menu

```
<ImageView  
    android:id="@+id/image_view_preferencias"
```

```
<LinearLayout  
    android:id="@+id/layout_menu"
```

```
<ImageView  
    android:id="@+id/image_view_fechar"
```

Animação

- Agora os eventos de clique nas duas ImageView

```
binding.imageViewPreferencias.setOnClickListener { it: View!  
  
}  
  
binding.imageViewFechar.setOnClickListener { it: View!  
  
}
```

Animação

- Vamos colocar o menu como escondido quando é iniciado a aplicação fazendo uma translação em Y de um valor suficientemente alto para que fique escondido.

```
binding.layoutMenu.animate().translationY( value: 500F)
```


Animação

- De seguida implementamos as animações nos eventos de clique.

```
binding.imageViewPreferencias.setOnClickListener { it: View!  
    binding.layoutMenu.visibility = View.VISIBLE  
    binding.layoutMenu.animate().translationY( value: 0F).setDuration(  
        resources.getInteger(android.R.integer.config_mediumAnimTime)  
        .toLong()  
    )  
}
```

- Neste caso a translação irá para o ponto 0 que é o ponto onde está neste momento.

Animação

- Com o método `getMeasuredHeight` conseguimos que a translação seja de apenas o height do layout.

```
binding.imageViewFechar.setOnClickListener { it: View!
    binding.layoutMenu.animate().translationY(binding.layoutMenu.measuredHeight.toFloat())
        .setDuration(
            resources.getInteger(android.R.integer.config_mediumAnimTime)
                .toLong()
        )
}
```

Animação

```
binding.layoutMenu.animate().translationY( value: 500F)

binding.imageViewPreferencias.setOnClickListener { it: View!
    binding.layoutMenu.visibility = View.VISIBLE
    binding.layoutMenu.animate().translationY( value: 0F).setDuration(
        resources.getInteger(android.R.integer.config_mediumAnimTime)
        .toLong()
    )
}

binding.imageViewFechar.setOnClickListener { it: View!
    binding.layoutMenu.animate().translationY(binding.layoutMenu.measuredHeight.toFloat())
        .setDuration(
            resources.getInteger(android.R.integer.config_mediumAnimTime)
            .toLong()
        )
}
```

Projeto final

