

Desenvolvimento Mobile

Aula 6

Estudo de Caso

Nesta aula vamos estudar como estilizar as **Views** da aplicação. Para isso, criaremos uma tela estática conforme o protótipo ao lado.

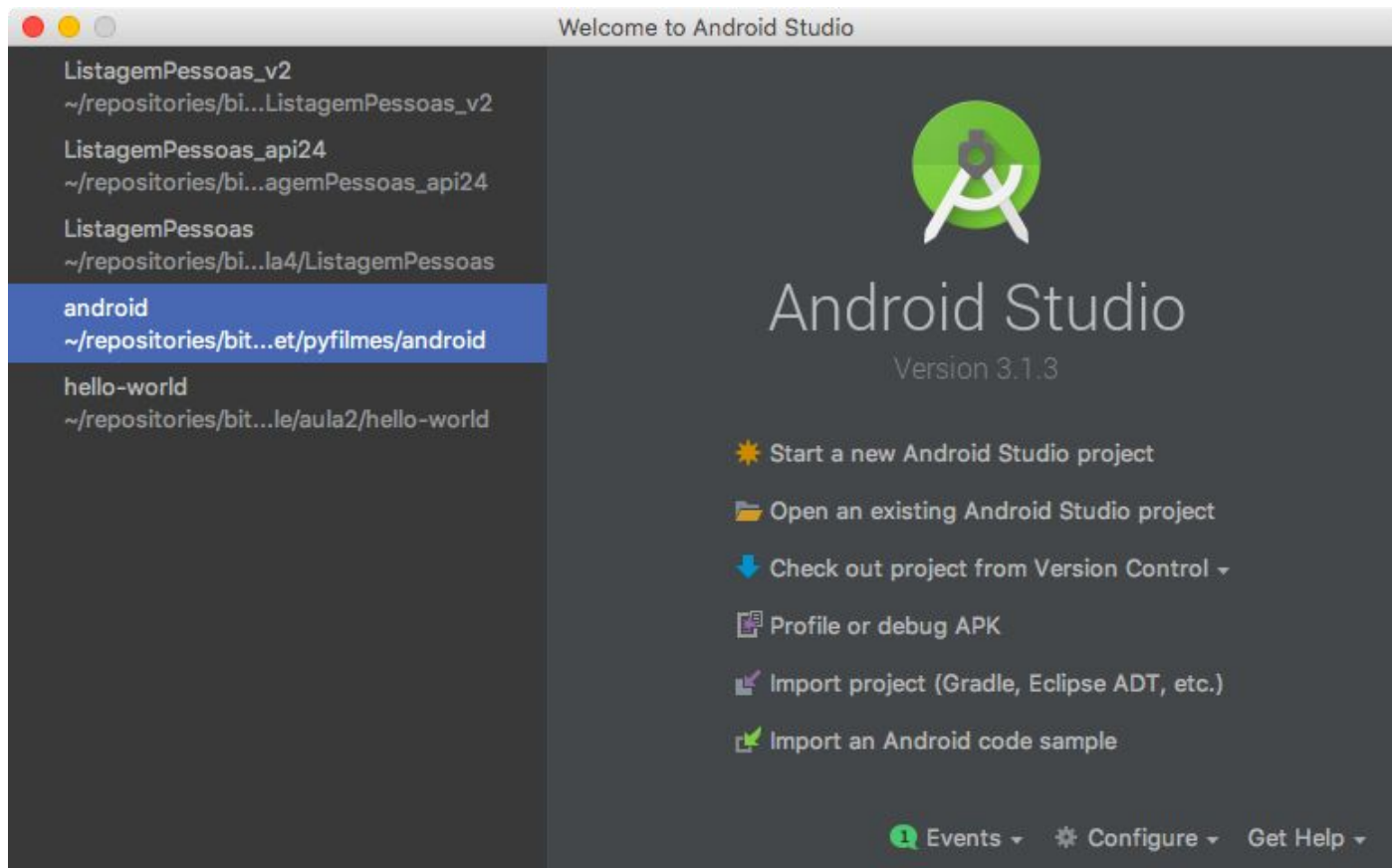
Itens necessários:

- **Retirar *ActionBar***
- Trabalhar com cores e gradiente
- Imagens
- Layouts mais sofisticados e aninhados
- Criação de shapes circulares



Criando um novo Projeto

Crie um novo projeto chamado *AppComEstilo*



Crie um novo projeto chamado *AppComEstilo*

Application name

AppComEstilo

Company domain

lgapontes.com

Project location

/Users/lgapontes/repositories/bitbucket/aulas/desenvolvimento-mobile/aula5/AppComEstilo_v1

...

Package name

com.lgapontes.appcomestilo_v1

Done

Crie um novo projeto chamado *AppComEstilo*

☒ Phone and Tablet

API 15: Android 4.0.3 (IceCreamSandwich) ▼

By targeting **API 15 and later**, your app will run on approximately **100%** of devices. [Help me choose](#)

☐ Include Android Instant App support

☐ Wear

API 21: Android 5.0 (Lollipop) ▼

☐ TV

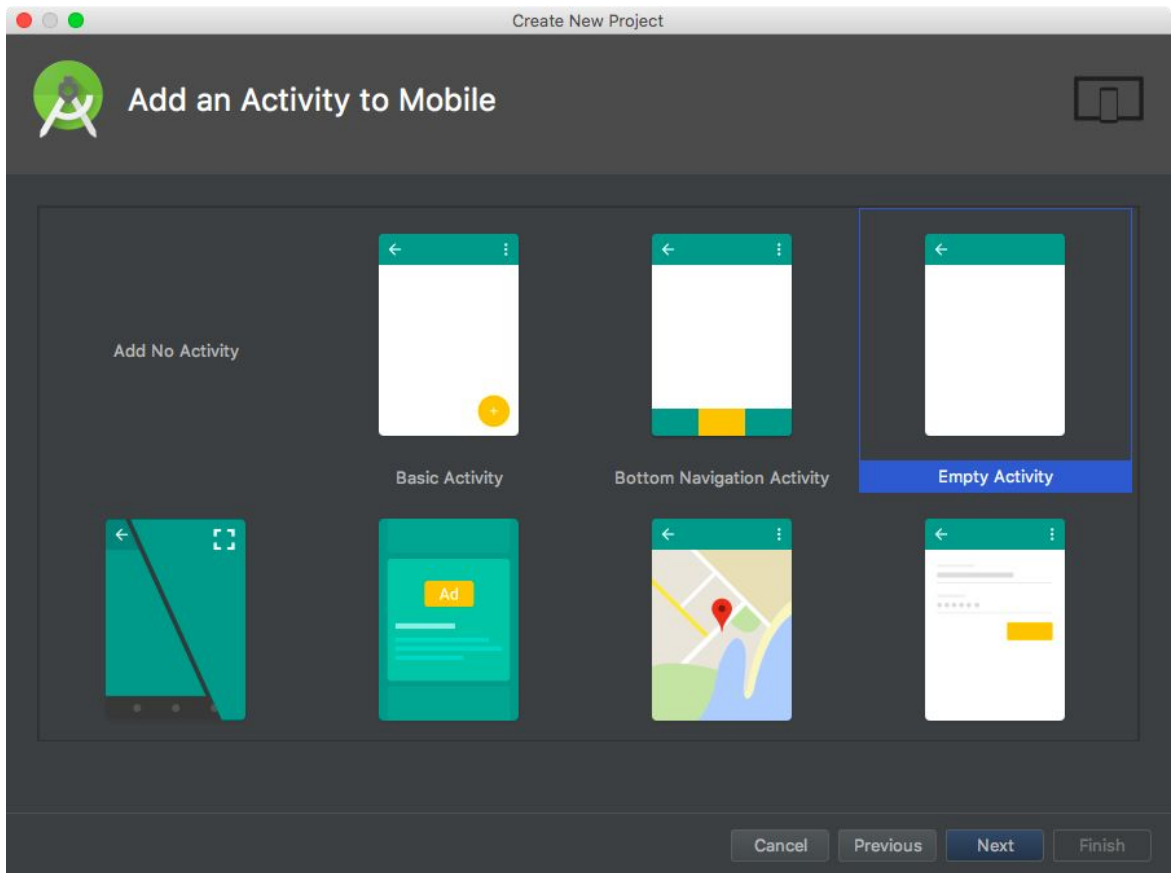
API 21: Android 5.0 (Lollipop) ▼

☐ Android Auto

☐ Android Things

API 24: Android 7.0 (Nougat) ▼

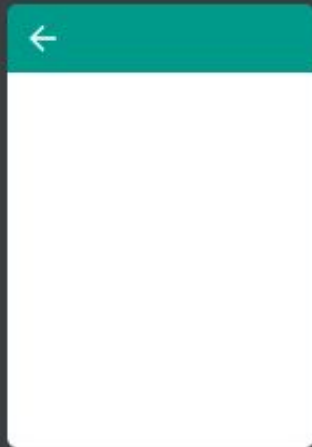
Crie um novo projeto chamado *AppComEstilo*



Escolha a **Empty Activity**

Crie um novo projeto chamado *AppComEstilo*

Creates a new empty activity



Activity Name:

MainActivity

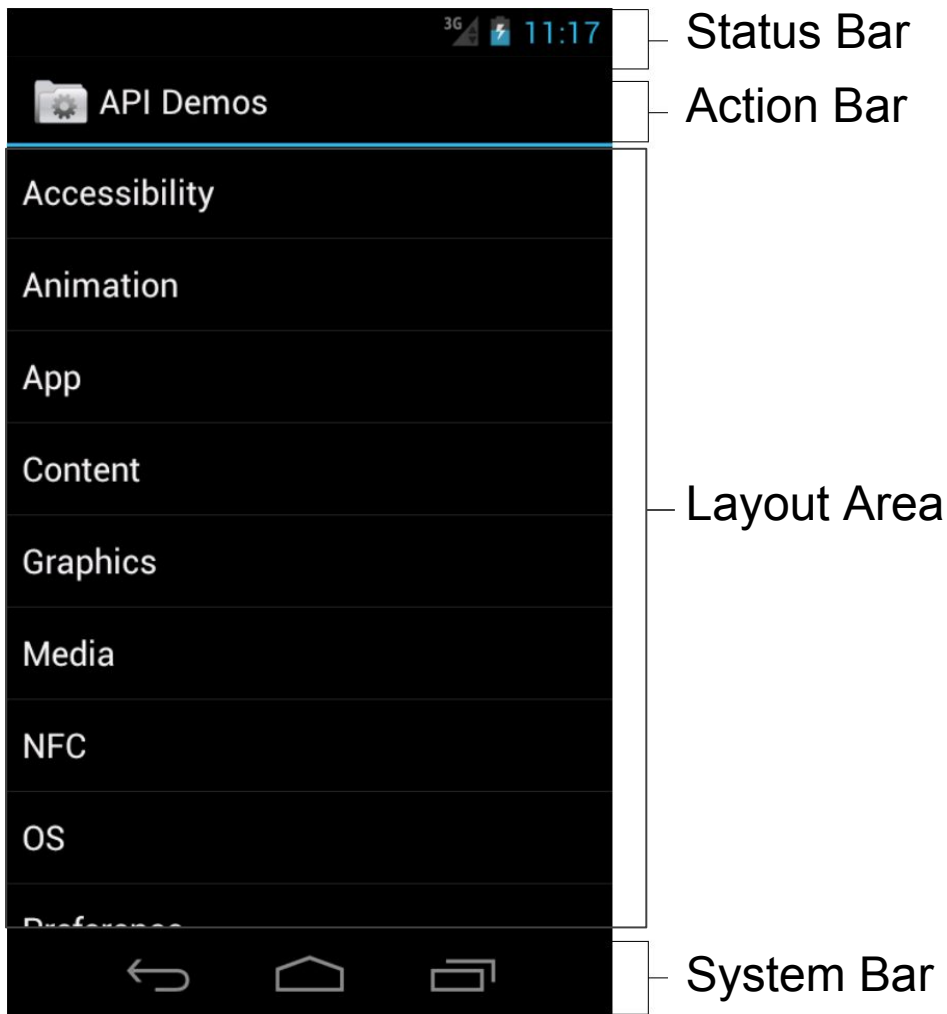
☒ Generate Layout File

Layout Name:

activity_main

☒ Backwards Compatibility (AppCompat)

Removendo a ActionBar



Estrutura de uma aplicação Android

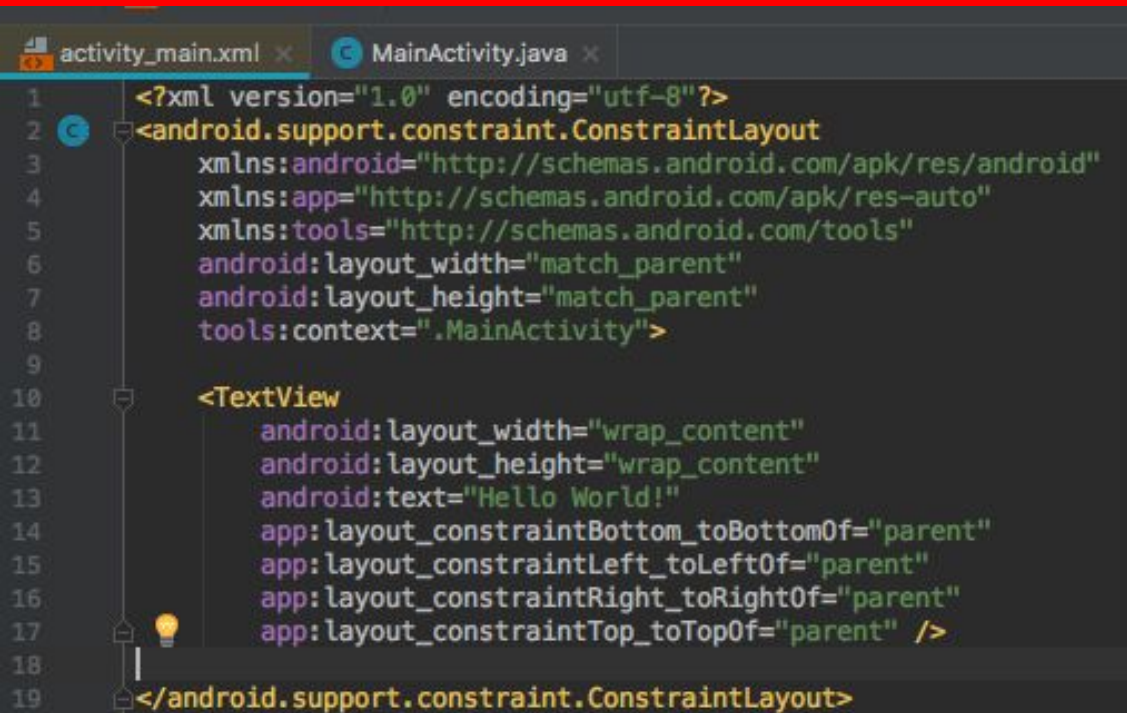
Status Bar: contém informações gerais do sistema Android, cujo conteúdo não é manipulado pela aplicação.

Action Bar: barra de ação geralmente formada pelo nome da aplicação e ícones de navegação. É exibida por default.

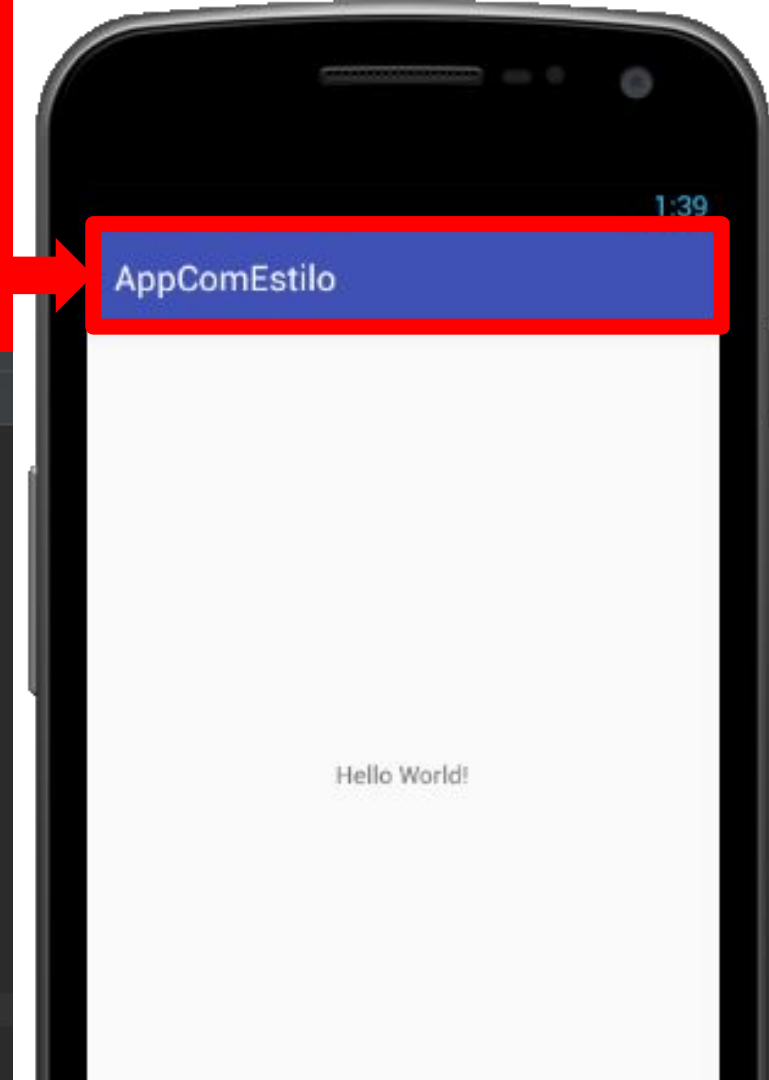
Layout Area: local onde as *Views* da aplicação são exibidas.

System Bar: barra do sistema que contém o *Back Button*, *Home Button* e o *Overview Button*.

Mas enfim, por que a Action Bar é exibida se ela não consta no XML de Layout?



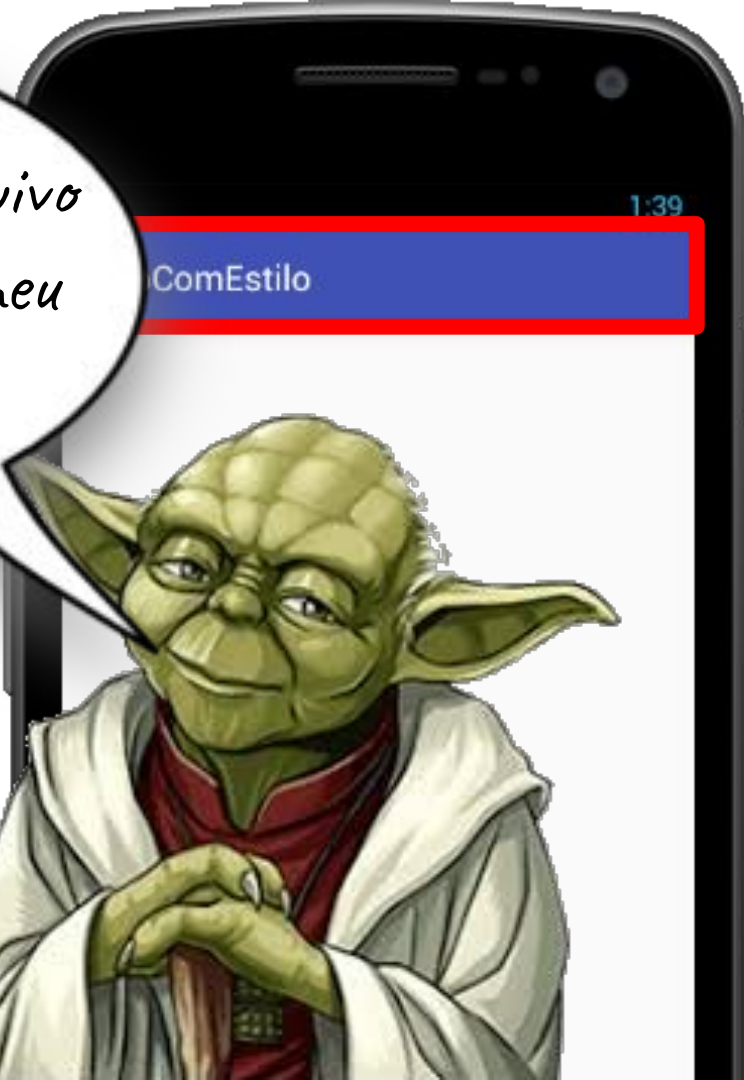
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9
10    <TextView
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:text="Hello World!"
14        app:layout_constraintBottom_toBottomOf="parent"
15        app:layout_constraintLeft_toLeftOf="parent"
16        app:layout_constraintRight_toRightOf="parent"
17        app:layout_constraintTop_toTopOf="parent" />
18
19 </android.support.constraint.ConstraintLayout>
```



Mas enfim, por que
exibida se ela n

*A resposta está no arquivo
AndroidManifest.xml meu
jovem padawan.*

```
activity_main.xml x MainActivi
1 <?xml version="1.0" encoding="utf-8"
2 <android.support.constraint.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res-auto"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9
10    <TextView
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:text="Hello World!"
14        app:layout_constraintBottom_toBottomOf="parent"
15        app:layout_constraintLeft_toLeftOf="parent"
16        app:layout_constraintRight_toRightOf="parent"
17        app:layout_constraintTop_toTopOf="parent" />
18
19    </android.support.constraint.ConstraintLayout>
```

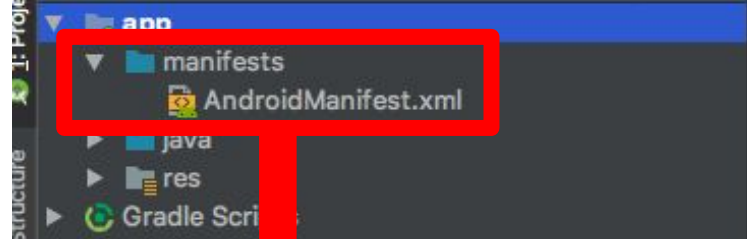


Arquivo de Manifesto

Arquivo que o sistema Android utiliza para inicializar a aplicação.

Todo projeto Android precisar deste arquivo, exatamente com o nome *AndroidManifest.xml*.

O APK gerado pelo Android Studio salva este arquivo na raiz do projeto.



```
activity_main.xml x AndroidManifest.xml x MainActivity.java x
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lgapontes.appcomestilo_v1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AppComEstilo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Dica Extra

O arquivo APK (Android Package) gerado pelo Android Studio, que serve para publicar a aplicação no Play Store, nada mais é do que um arquivo ZIP com tudo que o Android precisa para executar seu projeto.



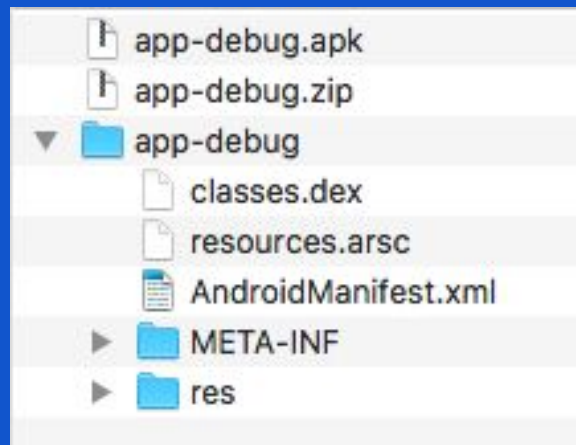
Arquivo APK

=

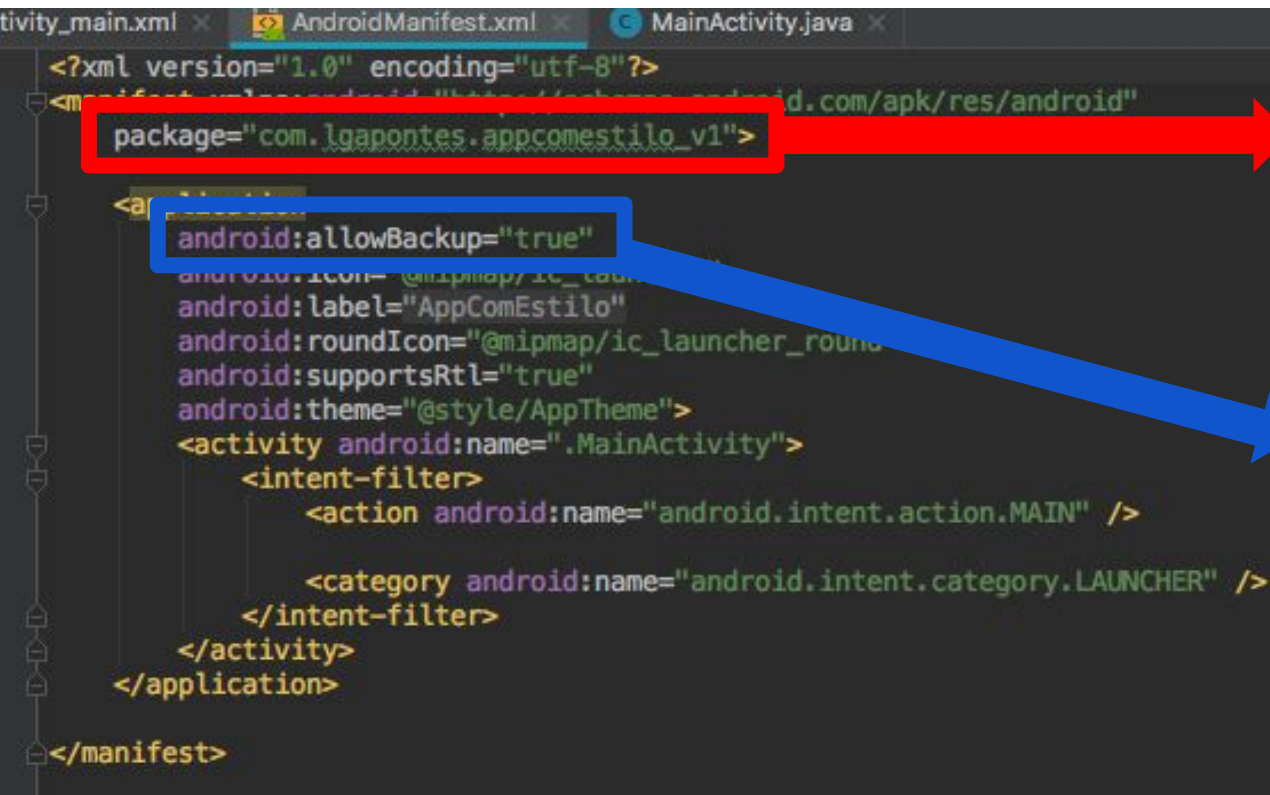


Arquivo ZIP

=



Entendendo o arquivo AndroidManifest.xml



The image shows a screenshot of an IDE with the AndroidManifest.xml file open. The file content is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lgapontes.appcomestilo_v1">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AppComEstilo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Annotations in the image:

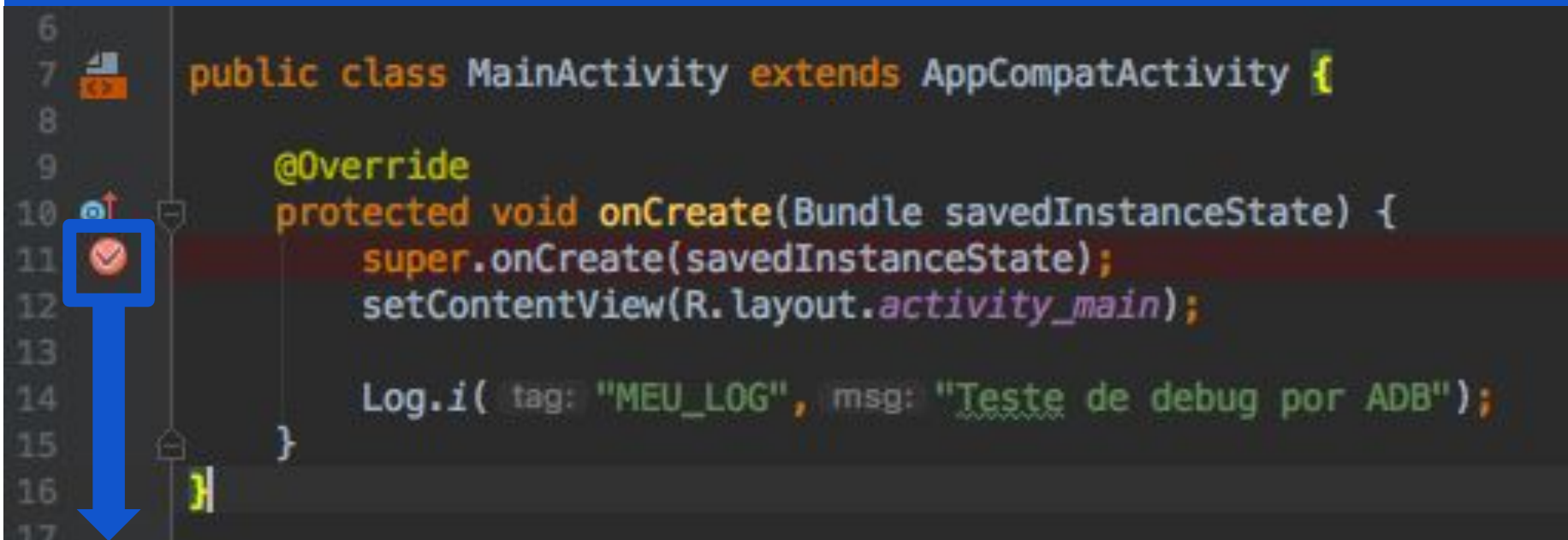
- A red box highlights the `package="com.lgapontes.appcomestilo_v1">` attribute. A red arrow points from this box to the red text block on the right.
- A blue box highlights the `android:allowBackup="true"` attribute. A blue arrow points from this box to the blue text block on the right.

O pacote serve como identificador exclusivo para o aplicativo no Play Store.

Permite usar o ADB (Android Debug Bridge) para realizar debug do app rodando no celular pelo Android Studio. Permite também fazer backup dos dados do app no computador.

Debugando no Android Studio

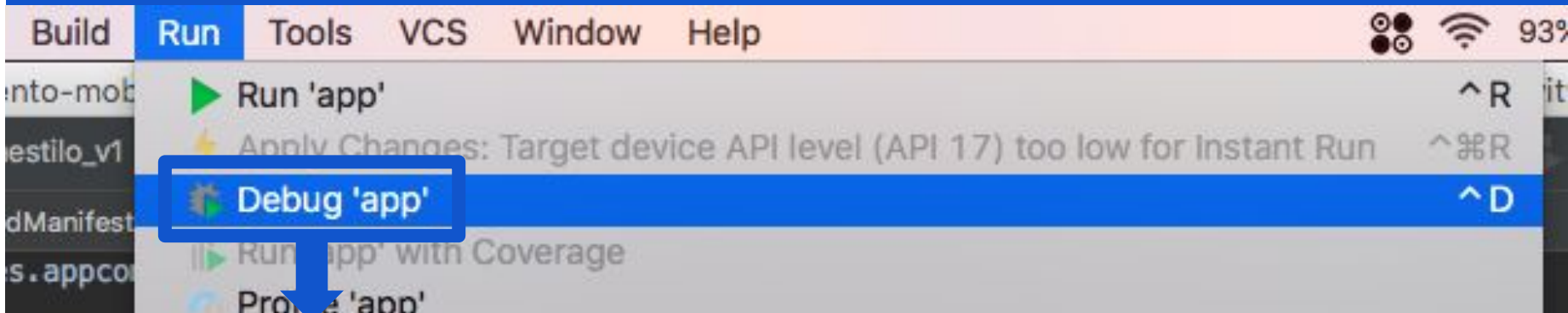
Através Android Debug Bridge podemos debugar o app. Altere o método `onCreate()` do `AppComEstilo`.



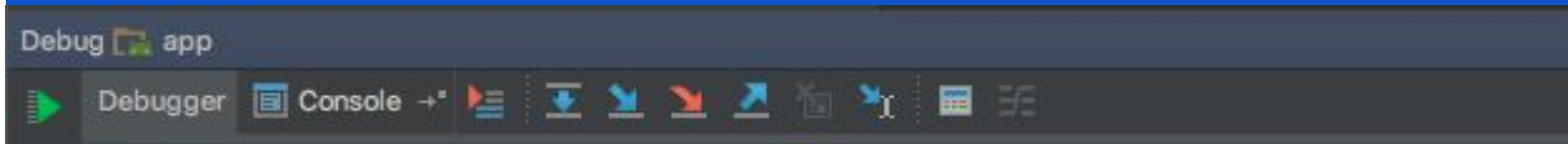
Clique neste ponto para adicionar um *breakpoint* de debug.

Debugando no Android Studio

Execute o app em modo de debug.



Neste momento o aplicativo no celular ficará suspenso. A partir daí é possível controlar o avanço da execução com os botões da aba *Debugger*.



<https://developer.android.com/studio/debug/?hl=pt-br>

Entendendo o arquivo AndroidManifest.xml

```
activity_main.xml x AndroidManifest.xml x MainActivity.java x
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lgapontes.appcomestilo_v1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

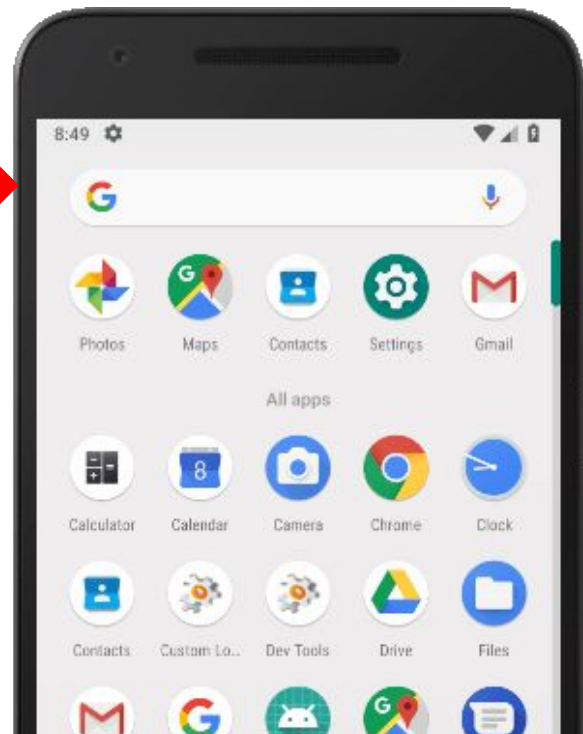
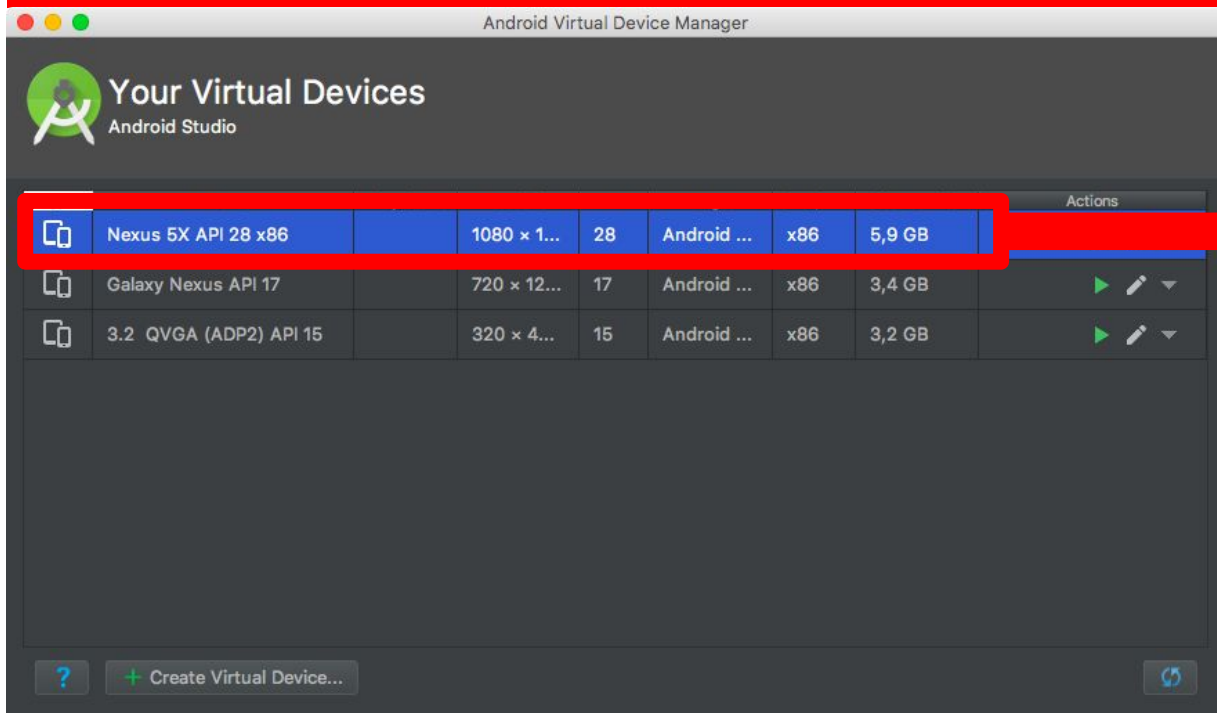
</manifest>
```

Lembre-se que na pasta *res/mipmap* ficam guardados os ícones da aplicação.

Por que temos ícones e ícones arredondados?
A partir da API 25 o Android fornece recursos nativos para visualizar os ícones arredondados. Nós precisamos suportá-lo.

Visualizando ícones arredondados no Android

Execute um emulador com API 25 (Android 7.1), ou superior, e veja a exibição dos ícones arredondados.



Entendendo o arquivo AndroidManifest.xml

```
activity_main.xml x AndroidManifest.xml x MainActivity.java x
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lgapontes.appcomestilo_v1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AppComEstilo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

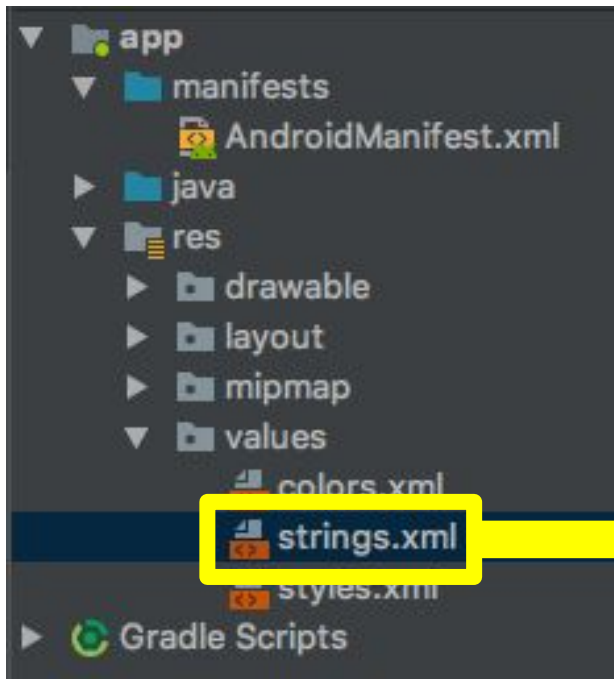
</manifest>
```

Note que apesar de estar escrito *AppComEstilo*, quando você clicar sobre esta label o Android Studio vai exibir a sintaxe:

@string/app_name

Trata-se de uma referência à variável *app_name* disponível no arquivo *res/values/strings.xml*

Definindo *strings* para uso na aplicação



Neste arquivo nós podemos guardar todas as labels utilizadas na aplicação. Através deste arquivo podemos também trabalhar com recursos de **i18n** (internacionalização). *Depois voltaremos a este arquivo para definir a string "Que a força esteja com você!"*.

```
<resources>  
  <string name="app_name">AppComEstilo</string>  
</resources>
```

Mais detalhes sobre i18n (internacionalização):

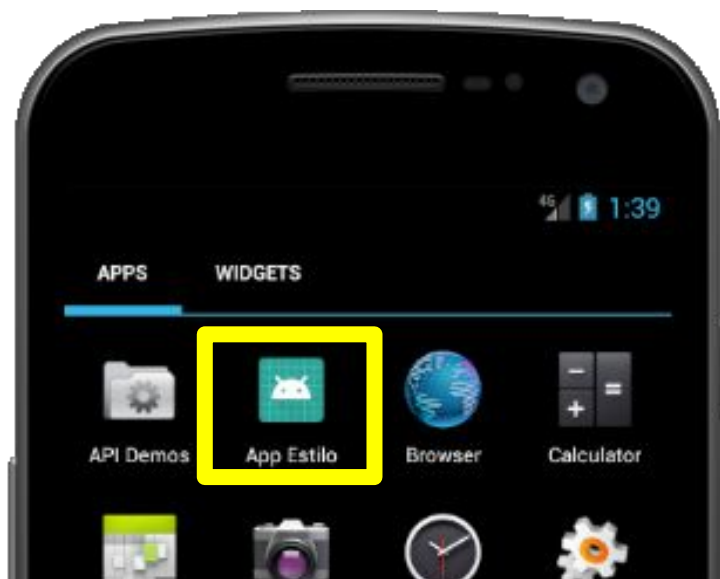
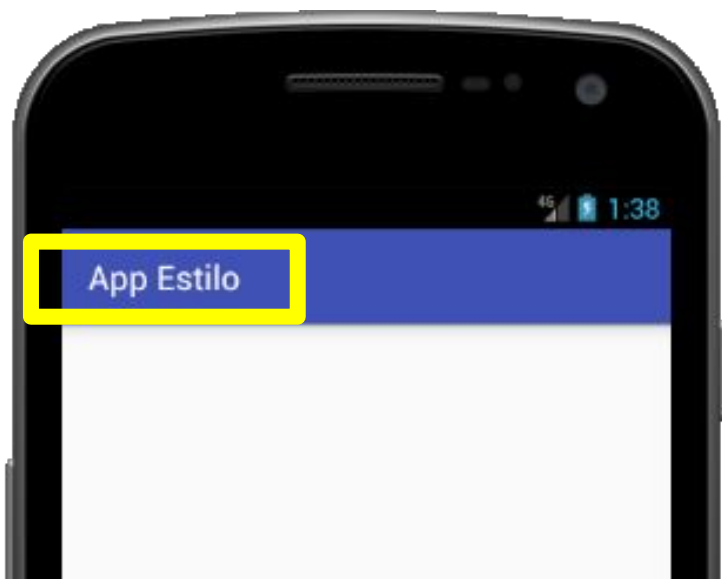
<https://developer.android.com/training/basics/supporting-devices/languages?hl=pt-br>

Alterando o nome da aplicação

```
<resources>  
  <string name="app_name">App Estilo</string>  
</resources>
```

Altere a variável *app_name* para App Estilo e execute a aplicação.

Como esta variável está aplicada ao atributo *label* do *AndroidManifest.xml*, o nome exibido na *ActionBar* e no *Launch Icon* da app agora será "App Estilo".



Entendendo o arquivo AndroidManifest.xml

```
activity_main.xml x AndroidManifest.xml x MainActivity.java x
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lgapontes.appcomestilo_v1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AppComEstilo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        <!-- ... -->

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Suporte ao layout RTL (right-to-left), útil em idiomas cuja escrita se dá da direita para esquerda (ex. Árabe, Persa e Hebraico). **API 17+**

Referência à *activity* de nossa aplicação. Todas as *activities* precisam estar declaradas aqui. Veremos mais detalhes no futuro.

Entendendo o arquivo AndroidManifest.xml

```
activity_main.xml x AndroidManifest.xml x MainActivity.java x
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lgapontes.appcomestilo_v1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AppComEstilo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Finalmente, o atributo *theme* aponta para um tema definido no arquivo *res/values/styles.xml*

Vamos investigar o que existe dentro deste arquivo.

Arquivo *res/values/styles.xml*

```
<resources>

  <!-- Base application theme. -->
  <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
  </style>

</resources>
```

O que é isso?



Android Support Library

O Android possui várias bibliotecas que podem ser incluídas no aplicativo. No geral elas são organizadas em versões e cada uma dessas versões possuem recursos diferentes.



v4 Support
Libraries



v7 Support
Libraries



v8 Support
Libraries



v13 Support
Libraries

...

<https://developer.android.com/topic/libraries/support-library/features>

V7 Support Libraries



v7 Support
Libraries



v7 appcompat



v7 cardview



v7 gridlayout



v7 mediarouter



v7 palette



v7 recyclerview



Trabalha com os padrões de layout com **ActionBar** e implementa as práticas definidas no **Material Design**.

```
Theme.AppCompat.Light
Theme.AppCompat
Theme.AppCompat.Light.NoActionBar
Theme.AppCompat.NoActionBar
Theme.AppCompat.Light.DarkActionBar
Theme.AppCompat.CompactMenu
Theme.AppCompat.DayNight
Theme.AppCompat.DayNight.DarkActionBar
Theme.AppCompat.DayNight.Dialog
Theme.AppCompat.DayNight.Dialog.Alert
Theme.AppCompat.DayNight.Dialog.MidWidth
Use → to overwrite the current identifier with the chosen v
```

Customizando o tema

```
<resources>

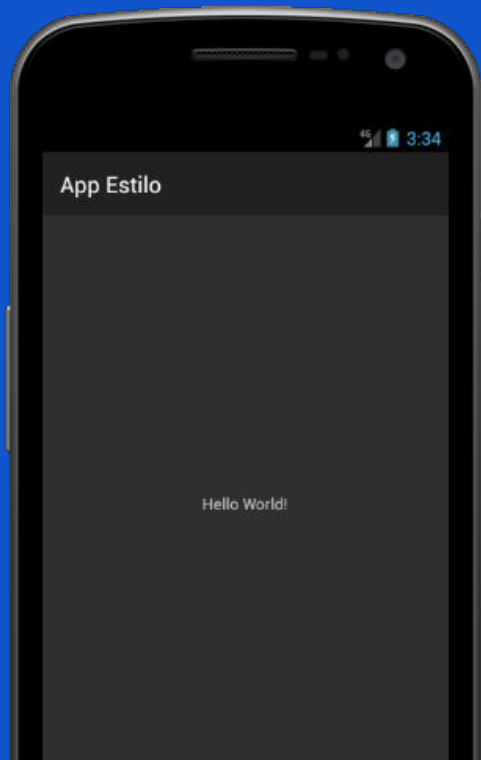
  <!-- Base application theme. -->
  <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here.
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    -->
  </style>

</resources>
```

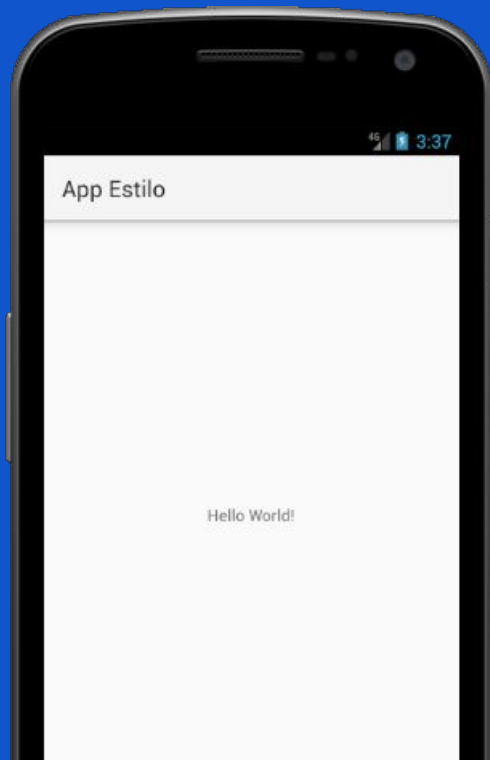
Vamos fazer algumas experiências com o tema da aplicação.
Primeiramente comente a definição das cores primárias e acentuadas.

Customizando o tema

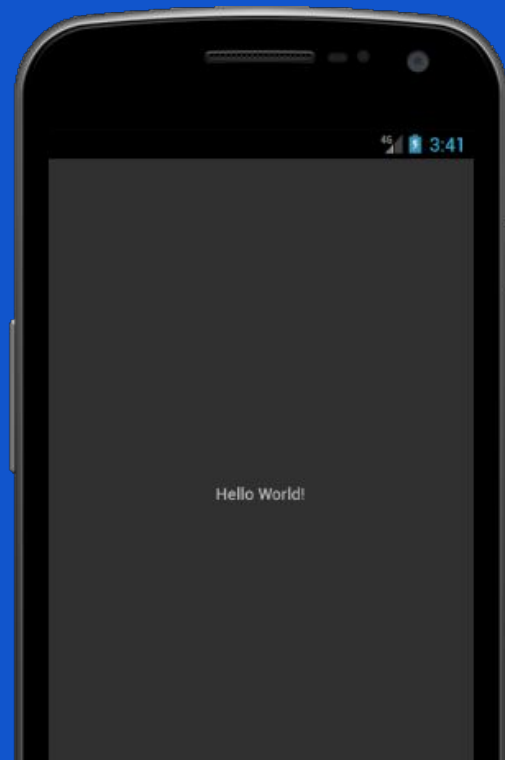
`Theme.AppCompat`



`Theme.AppCompat.Light`



`Theme.AppCompat.NoActionBar`

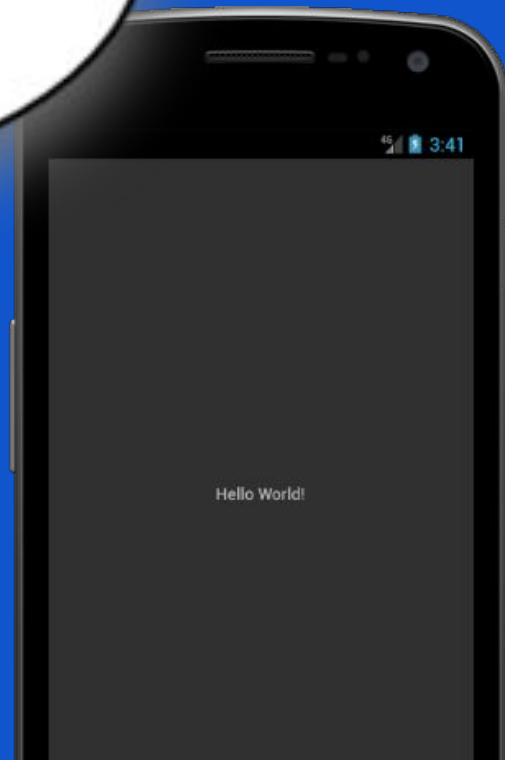
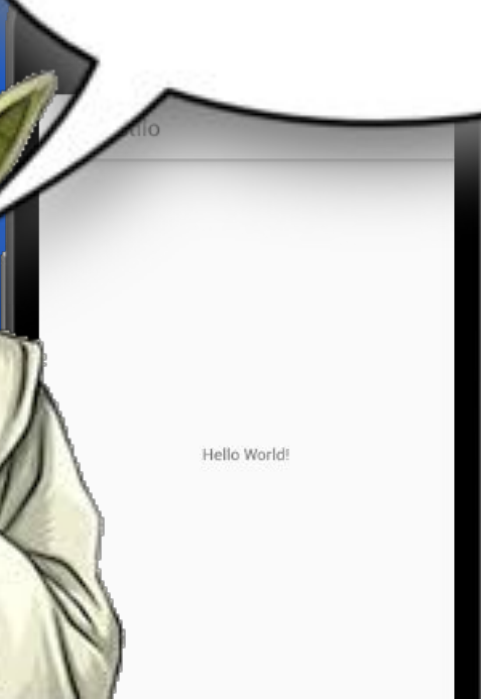


Customizando o tema

`Theme.AppCompat`

*A ActionBar foi removida
pelo poder da força!*

`Theme.AppCompat.NoActionBar`

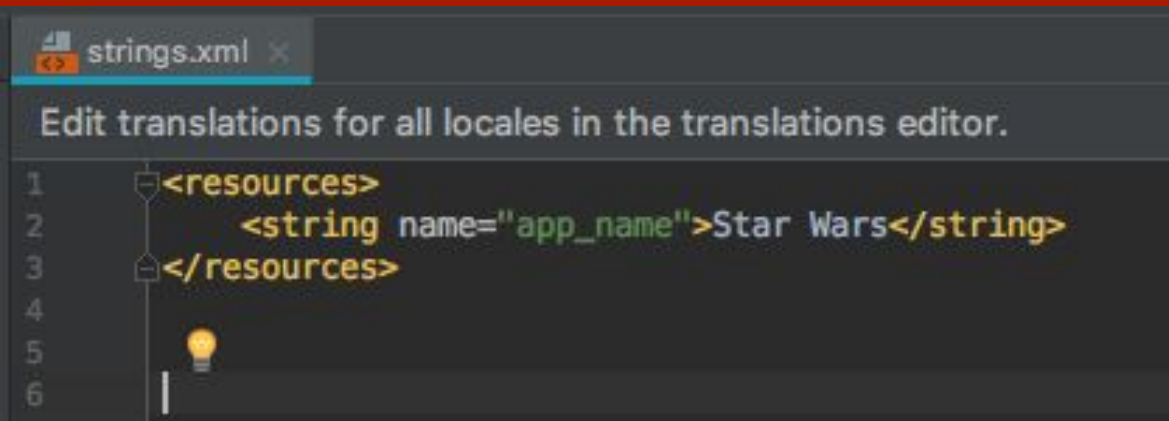


Exercício em Sala

Crie a aplicação *AppComEstilo* observando os detalhes destacados a seguir:

- Compatível com a *API 15* e com uma *Empty Activity*
- Altere o nome da aplicação para "*Star Wars*"
- Defina um layout claro que não exiba a *ActionBar*

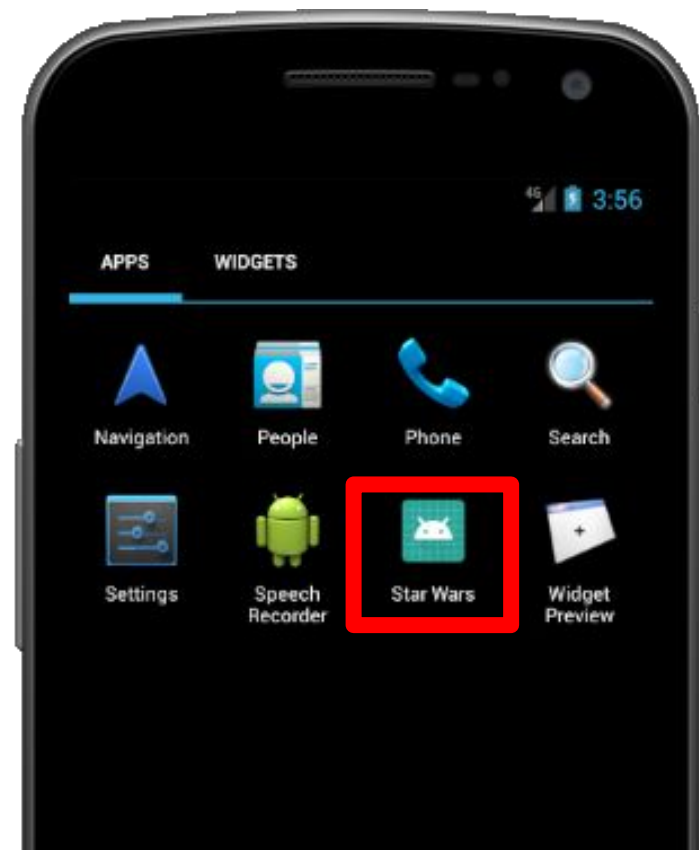
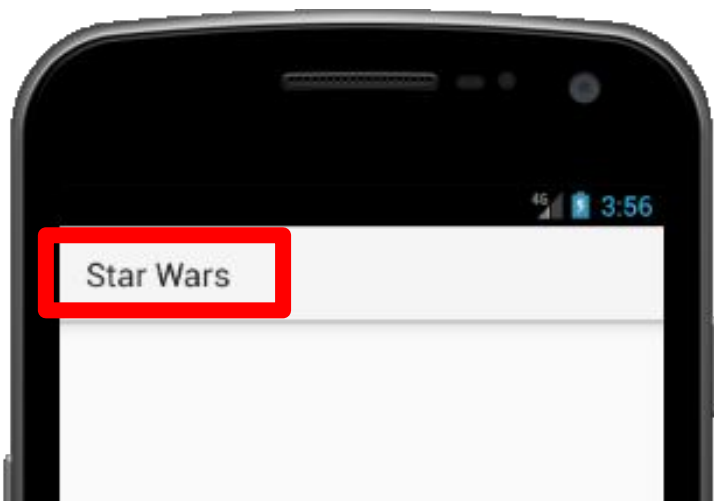
Resolvendo o Exercício



The screenshot shows an IDE window titled 'strings.xml'. Below the title bar, there is a message: 'Edit translations for all locales in the translations editor.' The main area displays XML code with line numbers 1 through 6 on the left. The code is as follows:

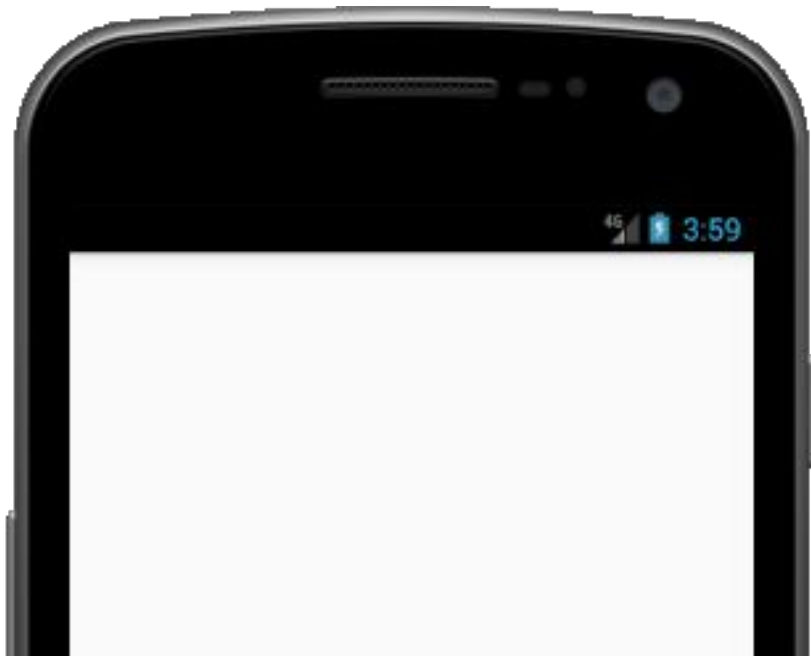
```
1 <resources>
2   <string name="app_name">Star Wars</string>
3 </resources>
```

A lightbulb icon is visible at the bottom left of the code editor area.



Resolvendo o Exercício

```
<!-- Base application theme. -->  
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">  
    <!-- Customize your theme here. -->  
    <item name="colorPrimary">@color/colorPrimary</item>  
</style>
```



Estudo de Caso

Nesta aula vamos estudar como estilizar as **Views** da aplicação. Para isso, criaremos uma tela estática conforme o protótipo ao lado.

Itens necessários:



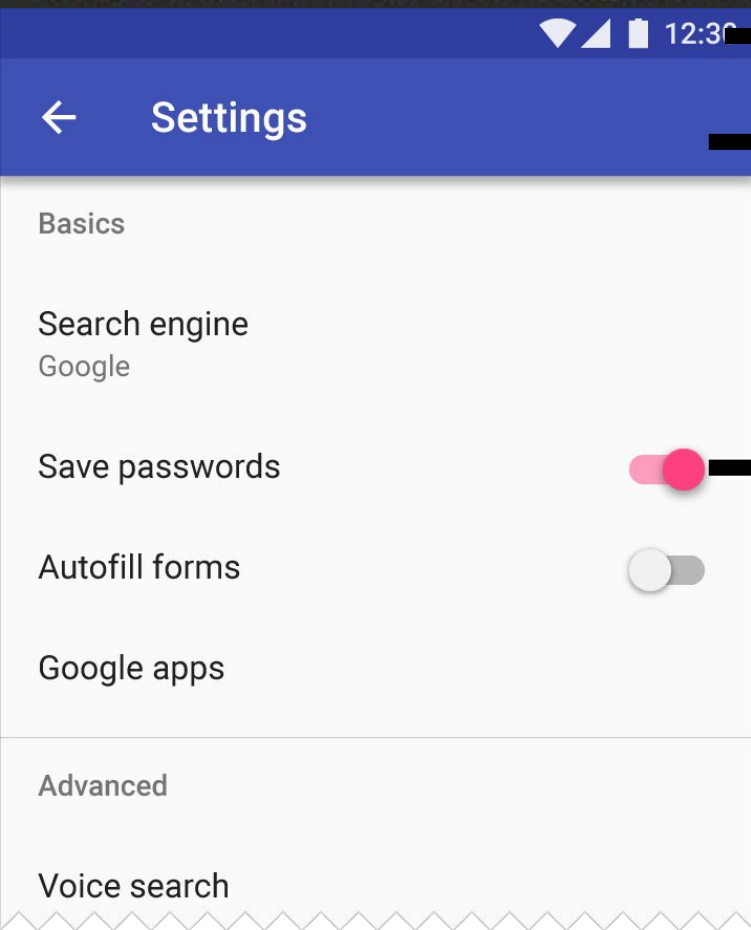
Retirar ActionBar

- **Trabalhar com cores e gradiente**
- Imagens
- Layouts mais sofisticados e aninhados
- Criação de shapes circulares



styles.xml

```
<item name="colorPrimary">@color/colorPrimary</item>
<item name="colorPrimaryDark">@color/colorPrimaryDark</item>
<item name="colorAccent">@color/colorAccent</item>
```



`colorPrimaryDark`

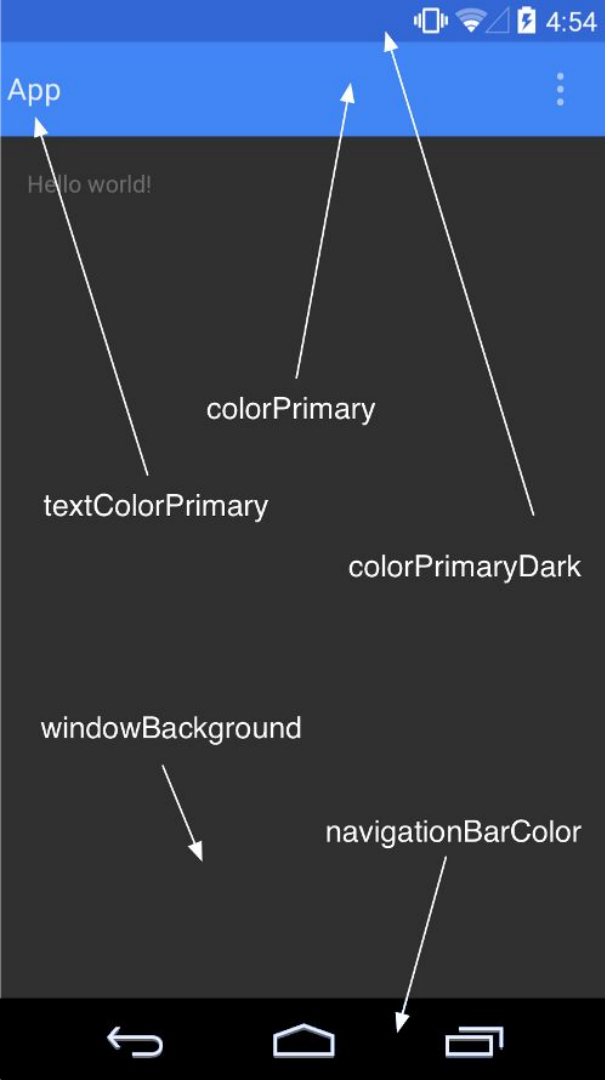
Cor primária mais escura usada na *StatusBar* disponível do Android 5.0 (API 21) para frente.

`colorPrimary`

Cor primária para a *ActionBar*, botões primários e o FAB (Floating Action Button).

`colorAccent`

Cor secundária (ou acentuada) que será utilizada nos elementos secundários (Radiobutton, Checkbox, Switch, etc).

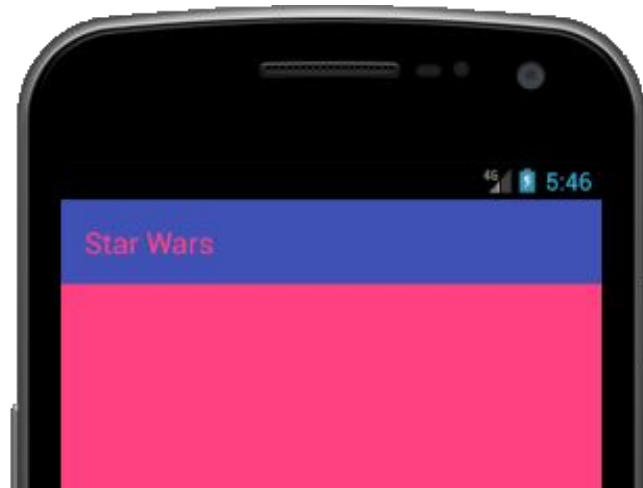


```
<resources>

<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>

    <item name="android:windowBackground">@color/colorAccent</item>
    <item name="android:textColorPrimary">@color/colorAccent</item>
</style>

</resources>
```



Ok, mas de onde vem essas cores?

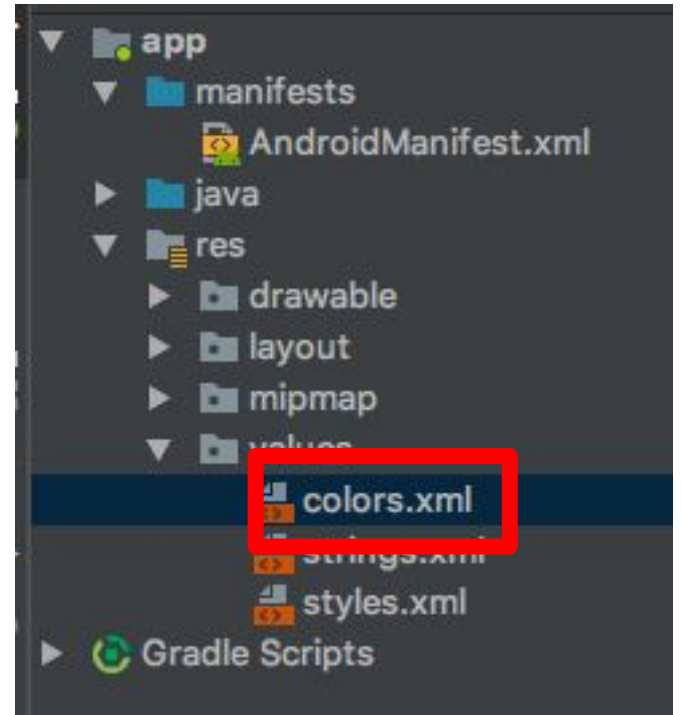
```
<resources>

  <!-- Base application theme. -->
  <style name="AppTheme" parent="Theme.AppCompat">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>

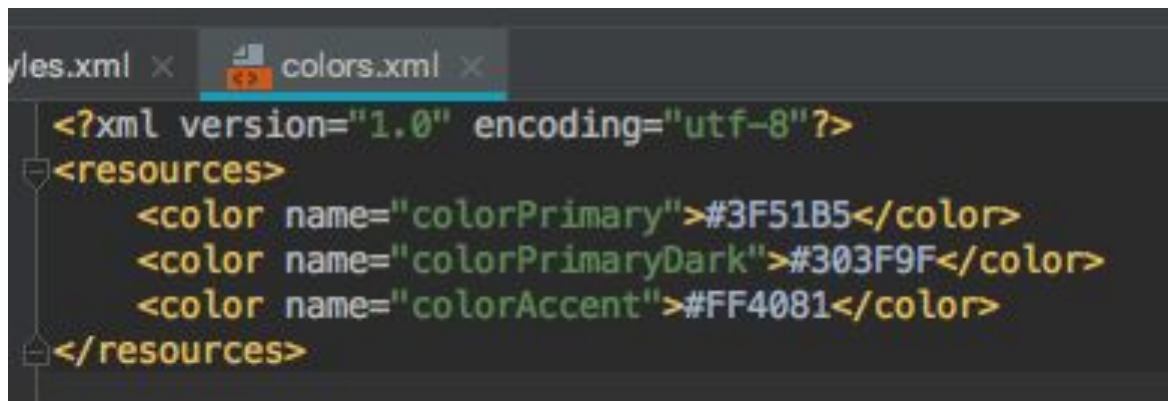
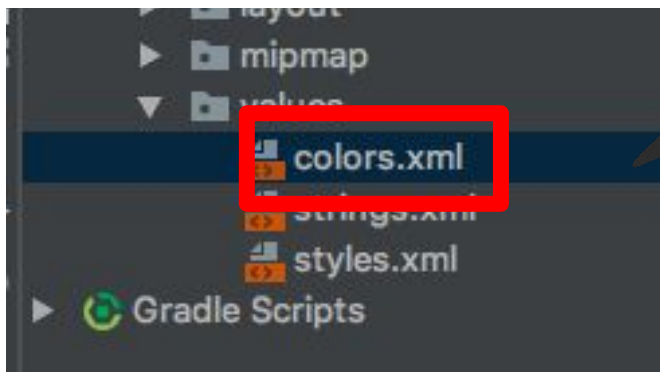
    <item name="android:windowBackground">@color/colorAccent</item>
    <item name="android:textColorPrimary">@color/colorAccent</item>
  </style>
</resources>
```

Ok, mas de onde vem essas cores?

```
theme. —>  
parent="Theme.AppCompat">  
  <item>@color/colorPrimary</item>  
  <item>@color/colorPrimaryDark</item>  
  <item>@color/colorAccent</item>  
  <item>@color/windowBackground</item>  
  <item>@color/textColorPrimary</item>
```



Ok, mas de onde vem essas cores?



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="colorPrimary">#3F51B5</color>
  <color name="colorPrimaryDark">#303F9F</color>
  <color name="colorAccent">#FF4081</color>
</resources>
```

Neste arquivo nós podemos criar quaisquer cores, definindo um nome e um valor RGB em hexadecimal (como no CSS de aplicações web).

Definindo as cores do estudo de caso

```
<resources>
  <color name="roxo">#A60053</color>
  <color name="amarelo">#FFF4C3</color>
  <color name="laranja">#FF8000</color>
  <color name="branco">#FFFFFF</color>
</resources>
```

No nosso estudo de caso nós vamos precisar de quatro cores.

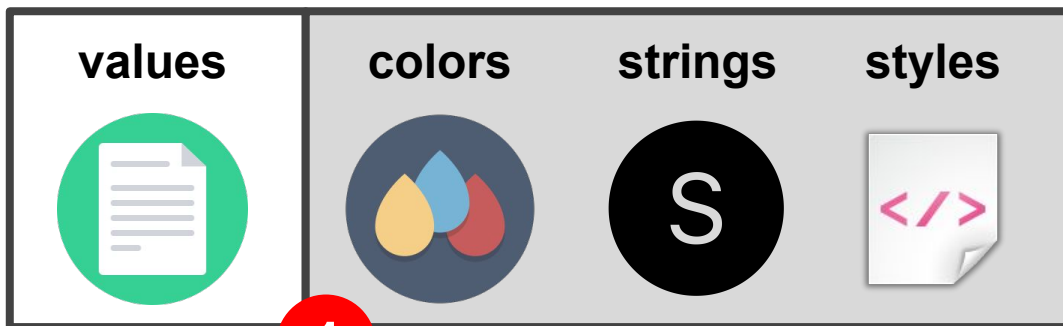
Cores utilizadas no tema da aplicação.

```
<resources>
  <style name="AppTheme" parent="Theme.AppCompat.NoActionBar">
    <item name="android:windowBackground">@color/roxo</item>
    <item name="android:textColor">@color/laranja</item>
  </style>
</resources>
```

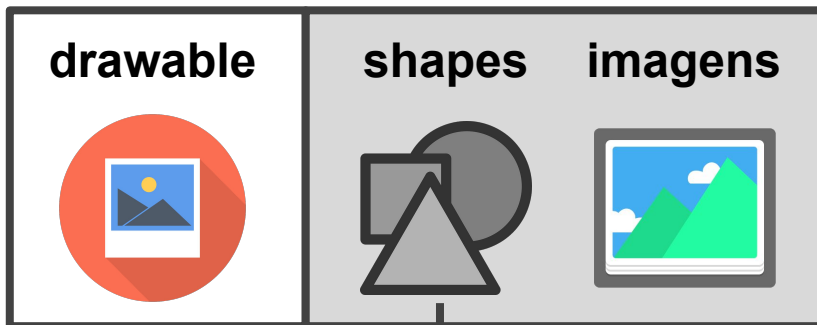


Criando o efeito gradiente

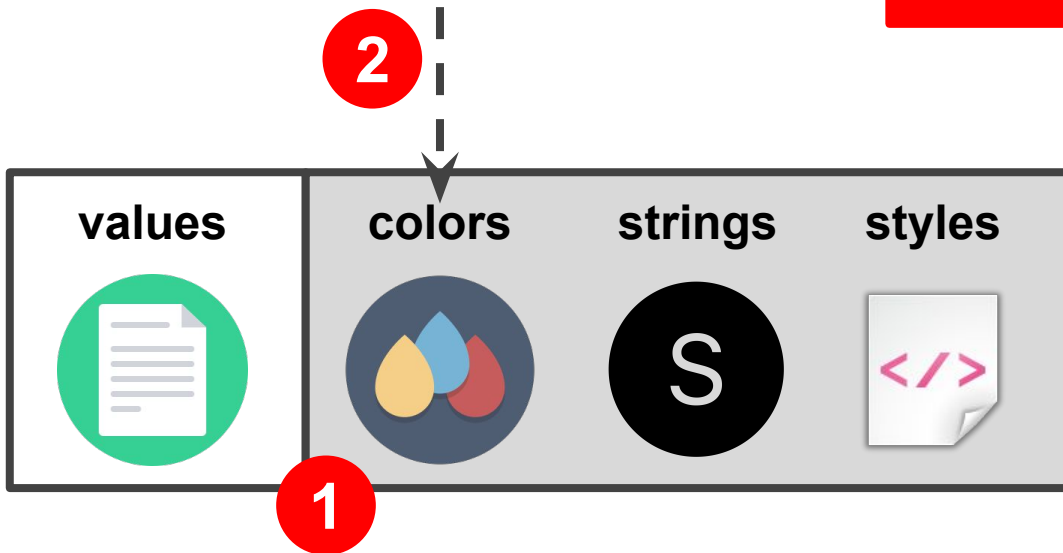
1- Definir as cores



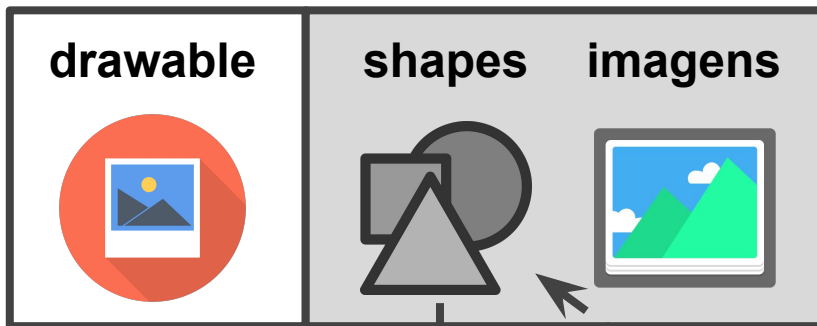
Criando o efeito gradiente



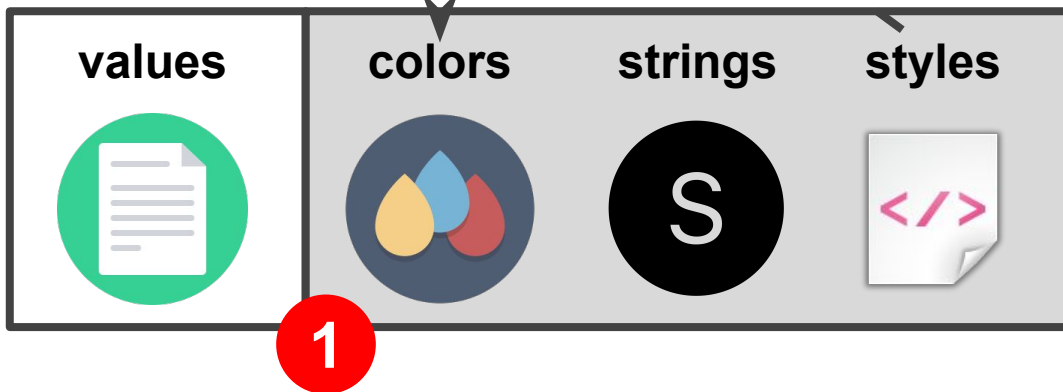
- 1- Definir as cores
- 2- Criar um shape



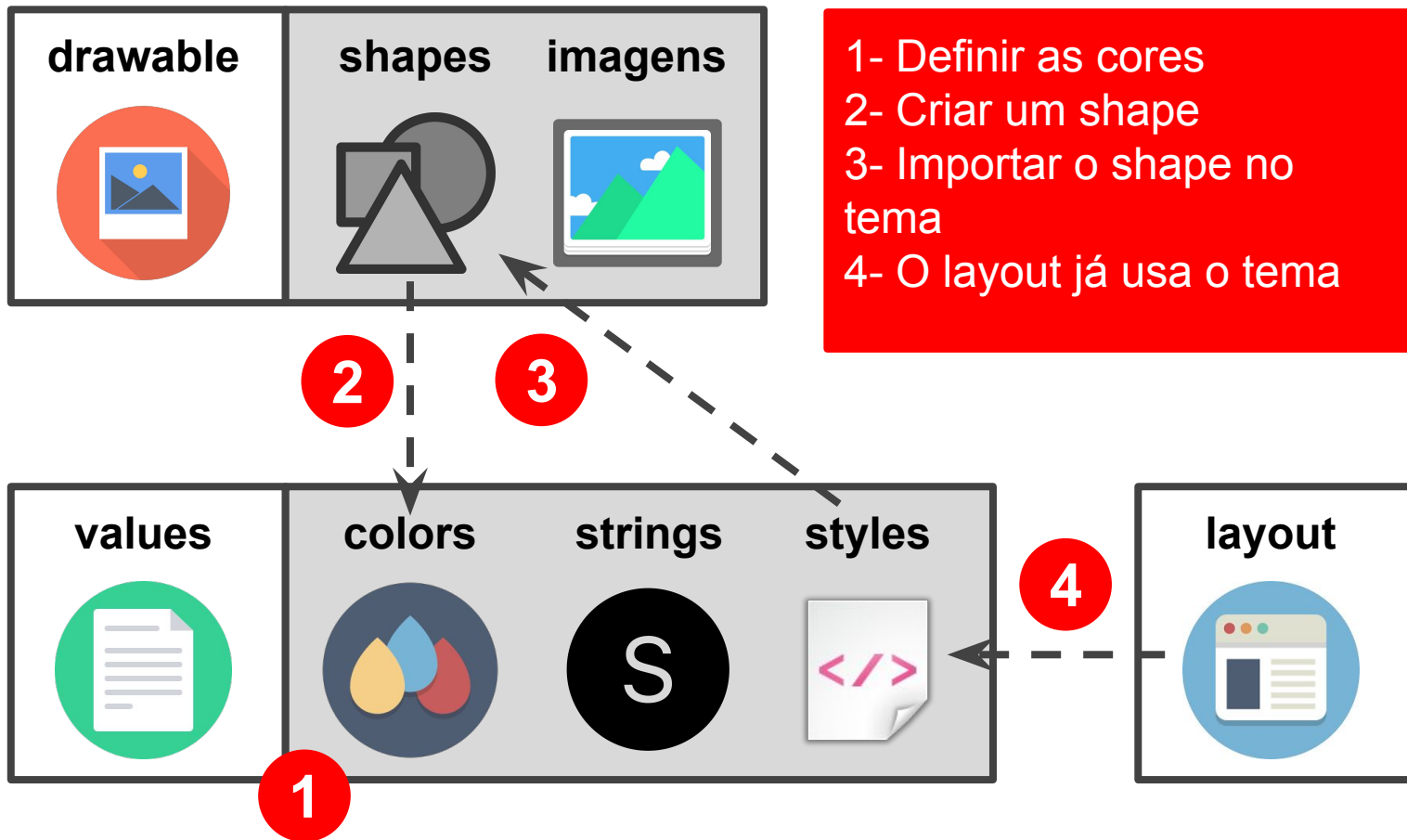
Criando o efeito gradiente



- 1- Definir as cores
- 2- Criar um shape
- 3- Importar o shape no tema



Criando o efeito gradiente



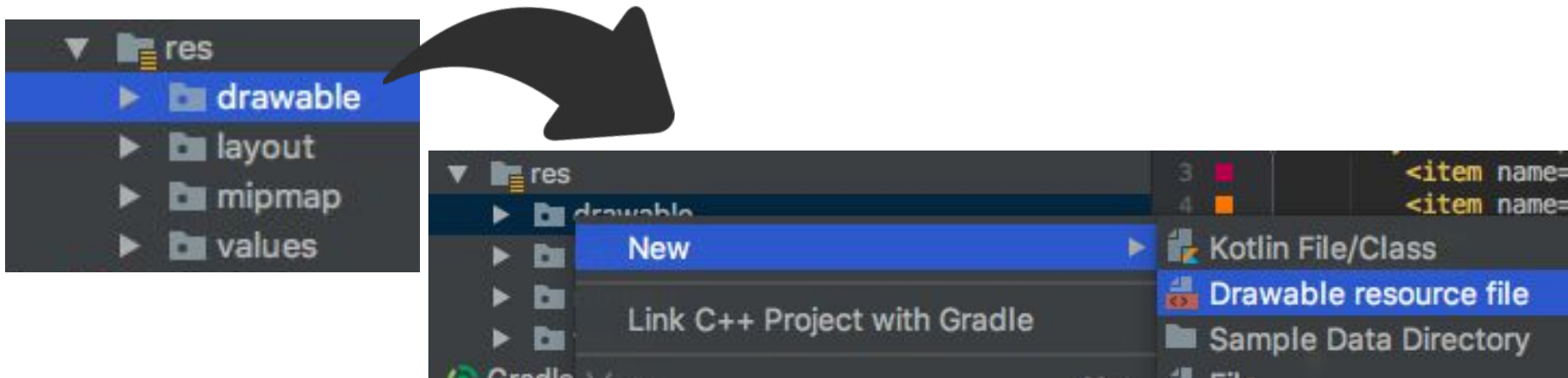
Criando o efeito gradiente

1- Definir as cores

```
<resources>  
  <color name="roxo">#A60053</color>  
  <color name="amarelo">#FFF4C3</color>  
  <color name="laranja">#FF8000</color>  
  <color name="branco">#FFFFFF</color>  
</resources>
```

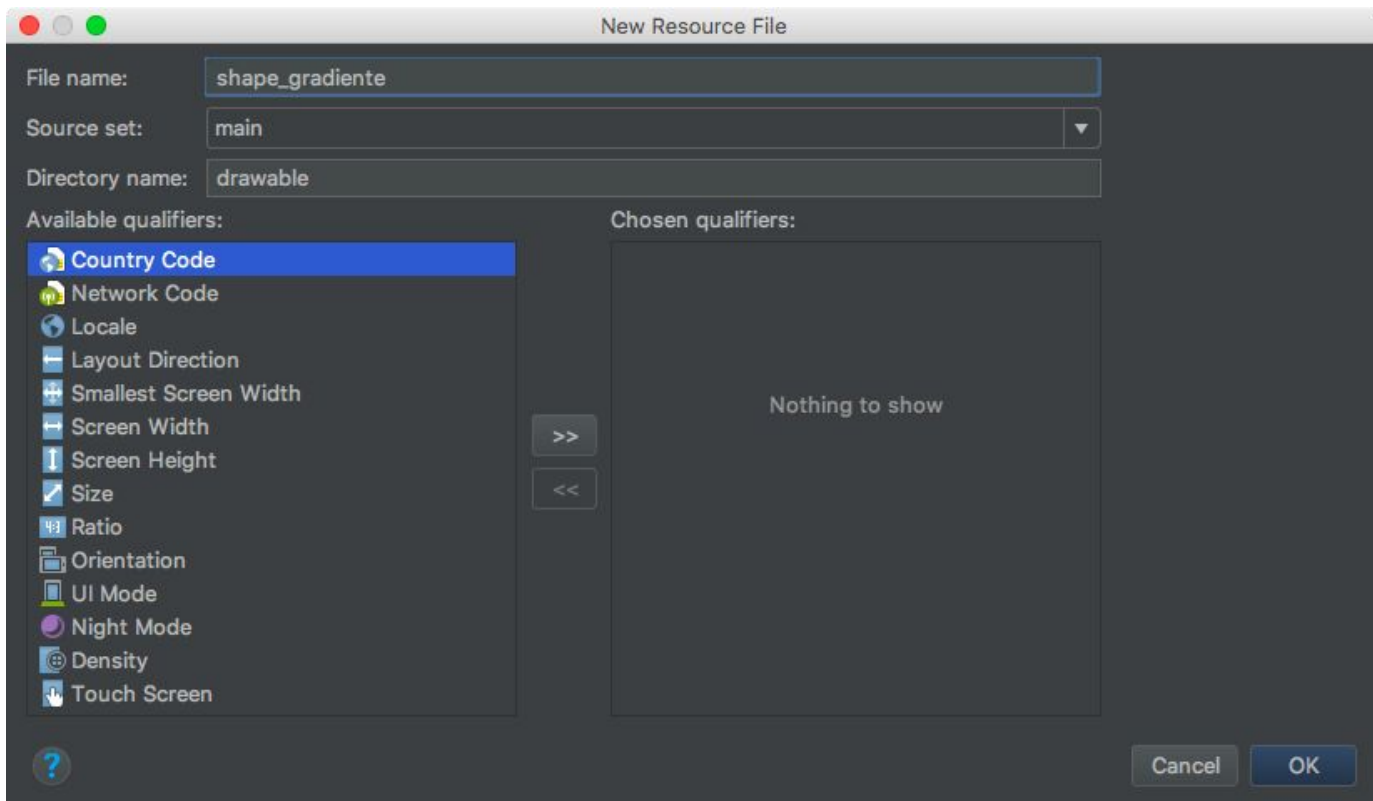
Criando o efeito gradiente

2- Criar um shape



Criando o efeito gradiente

2- Criar um shape



Criando o efeito gradiente

2- Criar um shape

```
activity_main.xml × styles.xml × shape_gradiente.xml × colors.xml ×
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
<item>
    <shape>
        <gradient
            android:angle="270"
            android:startColor="@color/roxo"
            android:endColor="@color/amarelo"
            android:type="linear" />
        </shape>
    </item>
</selector>
```


Criando o efeito gradiente

3- Importar o shape no tema

```
<resources>
  <style name="AppTheme" parent="Theme.AppCompat.NoActionBar">
    <item name="android:windowBackground">@drawable/shape_gradiente</item>
    <item name="android:textColor">@color/branco</item>
  </style>
</resources>
```

Note que alguns atributos devem ser precedidos pelo namespace *android*. Os atributos *textColor* e *windowBackground* são exemplos deste caso. Caso você não coloque este namespace o app não vai executar!

Criando o efeito gradiente

4- O layout já usa o tema

```
activity_main.xml x AndroidManifest.xml x MainActivity.java x
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lgapontes.appcomestilo_v1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AppComEstilo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```



Outros recursos *drawable*

ClipDrawable: XML que permite cortar uma imagem	NinePatchDrawable: um tipo de .png que permite redimensionamento
LayerDrawable: matriz de outros drawables	BitmapDrawable: imagem .png, .jpg ou .gif
LevelListDrawable: XML que organiza várias imagens representadas por um nível (exemplo: barra de progresso)	TransitionDrawable: XML que organiza uma transição entre duas imagens.
InsetDrawable: XML que define uma imagem envolvendo outra (útil para fazer bordas 3D, por exemplo)	StateListDrawable: XML que referencia imagens diferentes (para trocar uma imagem quando um botão é pressionado, por exemplo)
ScaleDrawable: XML que muda o tamanho de outra imagem.	ShapeDrawable: XML que define um formato geométrico (com cores ou gradientes)

Veja: <https://developer.android.com/guide/topics/resources/drawable-resource>

Exercício em Sala

Crie uma aplicação *MundoVerde* com os seguintes detalhes:

- Com API 21 (ou superior) e *Empty Layout*
- Crie as cores verde1 (#1b5e20), verde2 (#2e7d32), verde3 (#00c853) e verde 4 (#b9f6ca)
- Utilize a cor verde1 no *StatusBar* e verde2 no *ActionBar*
- Crie um shape chamado *mundo_verde*, com gradiente radial e as cores verde3 e verde4, e coloque-o como background da activity

Dica: gradientes radiais devem conter o atributo *gradientRadius*. A medida *%p* significa percentual em relação ao elemento pai.

```
android:gradientRadius="100%p"
```

Resolvendo o Exercício

Application name

MundoVerde

Company domain

lgapontes.com

Project location

/Users/lgapontes/repositories/bitbucket/aulas/desenvolvimento-mobile/aula6/MundoVerde_v1

...

Package name

com.lgapontes.mundoverde_v1

Done

Resolvendo o Exercício

☒ Phone and Tablet

API 21: Android 5.0 (Lollipop)

By targeting **API 21 and later**, your app will run on approximately **71,3%** of devices. [Help me choose](#)

☐ Include Android Instant App support

☐ Wear

API 21: Android 5.0 (Lollipop)

☐ TV

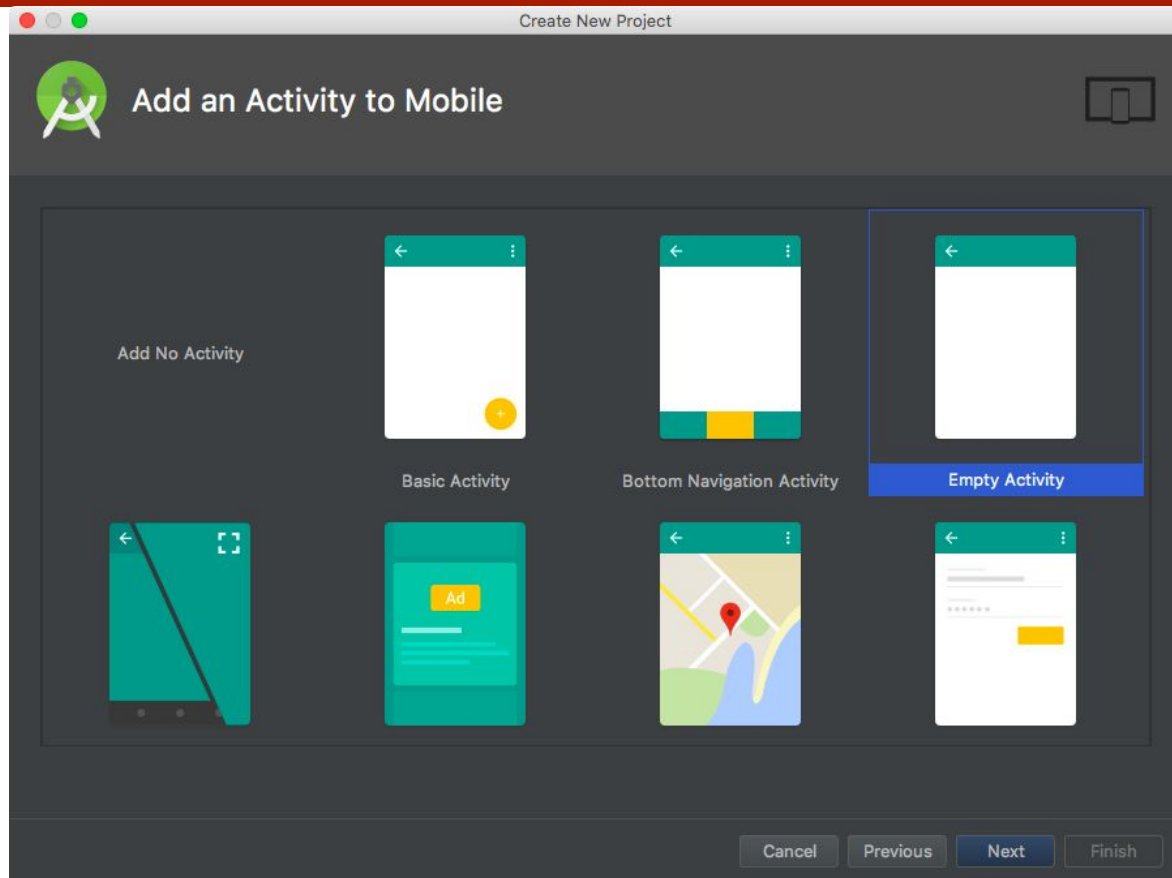
API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Android Things



API 24: Android 7.0 (Nougat)

Resolvendo o Exercício



Resolvendo o Exercício

Create New Project

 **Configure Activity** 


Creates a new empty activity

Activity Name:

☒ Generate Layout File

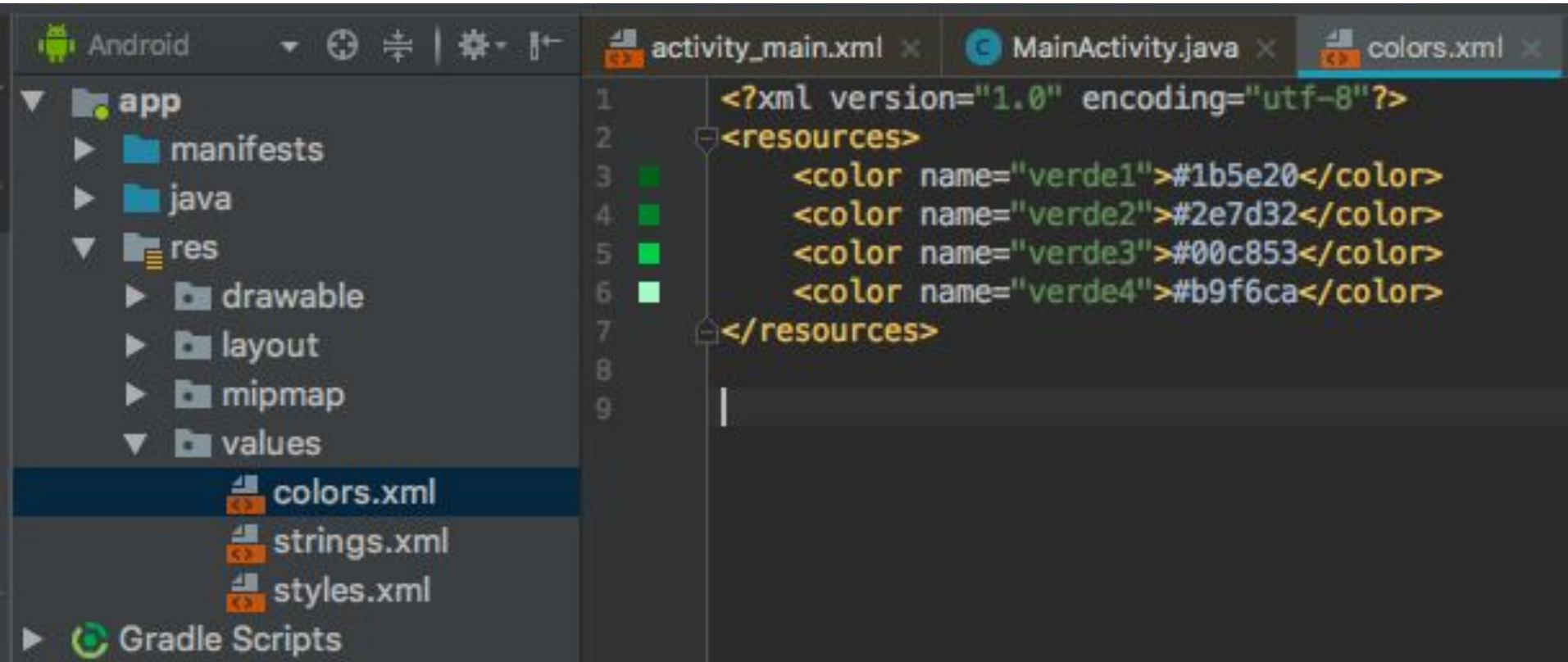
Layout Name:

☒ Backwards Compatibility (AppCompat)

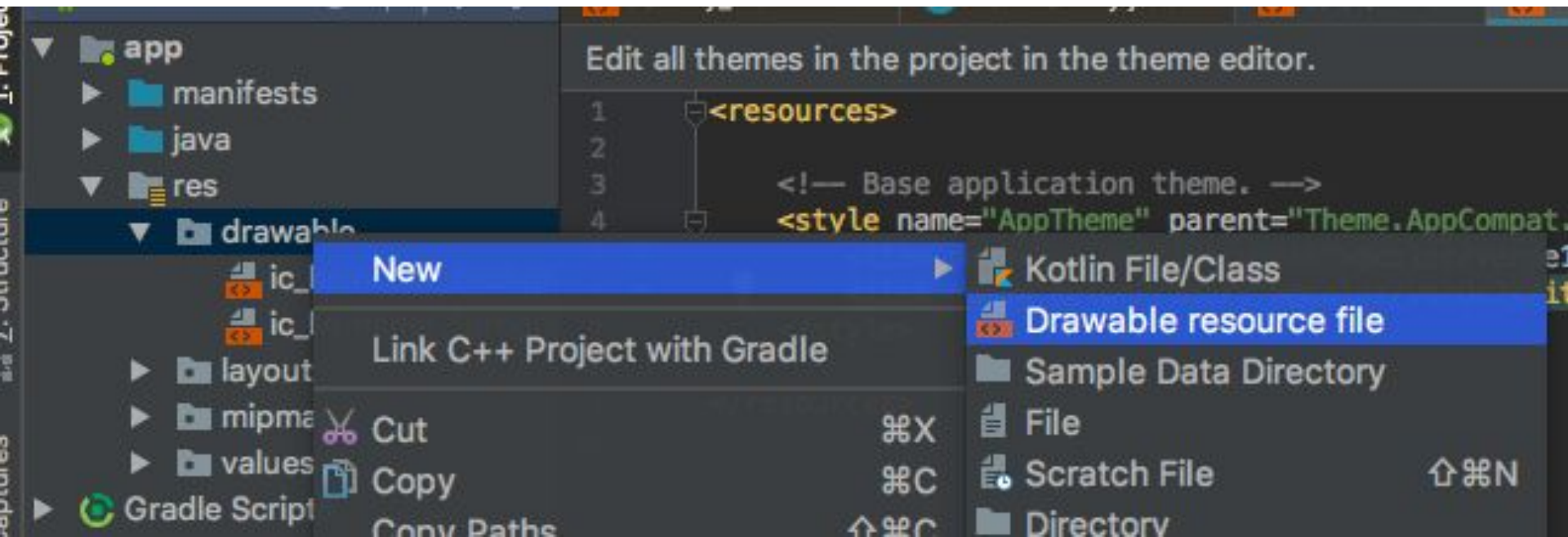


The name of the activity class to create

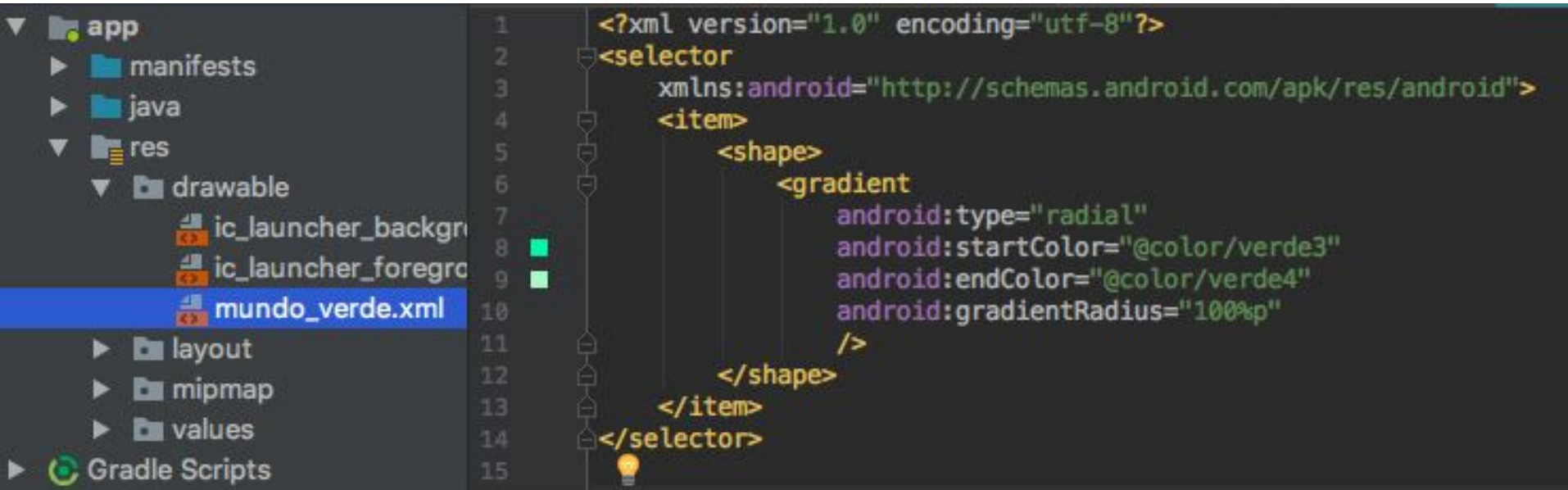
Resolvendo o Exercício



Resolvendo o Exercício

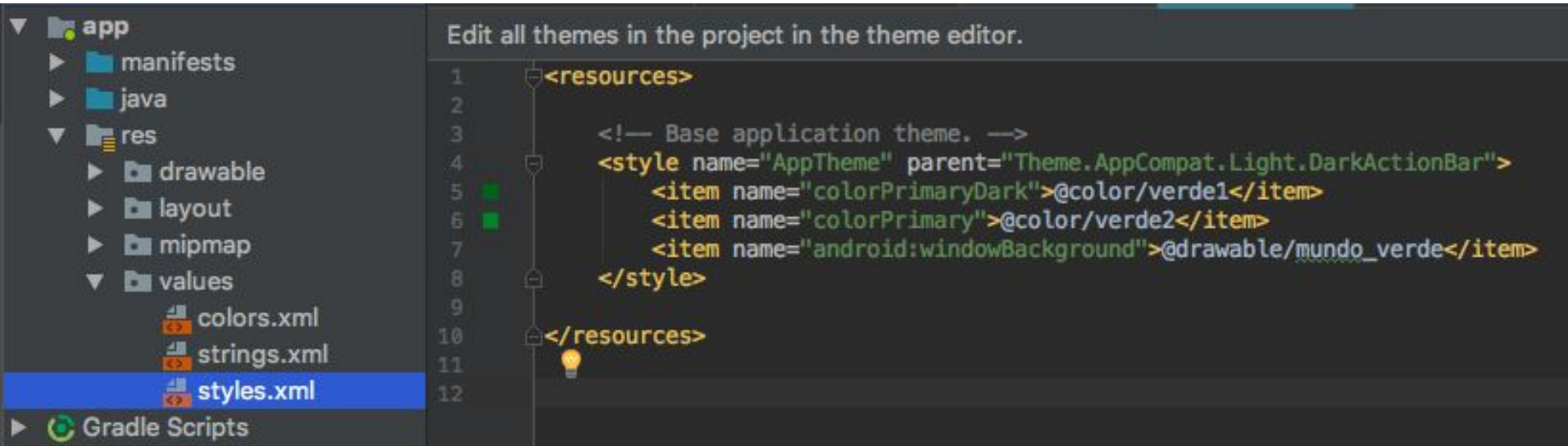


Resolvendo o Exercício



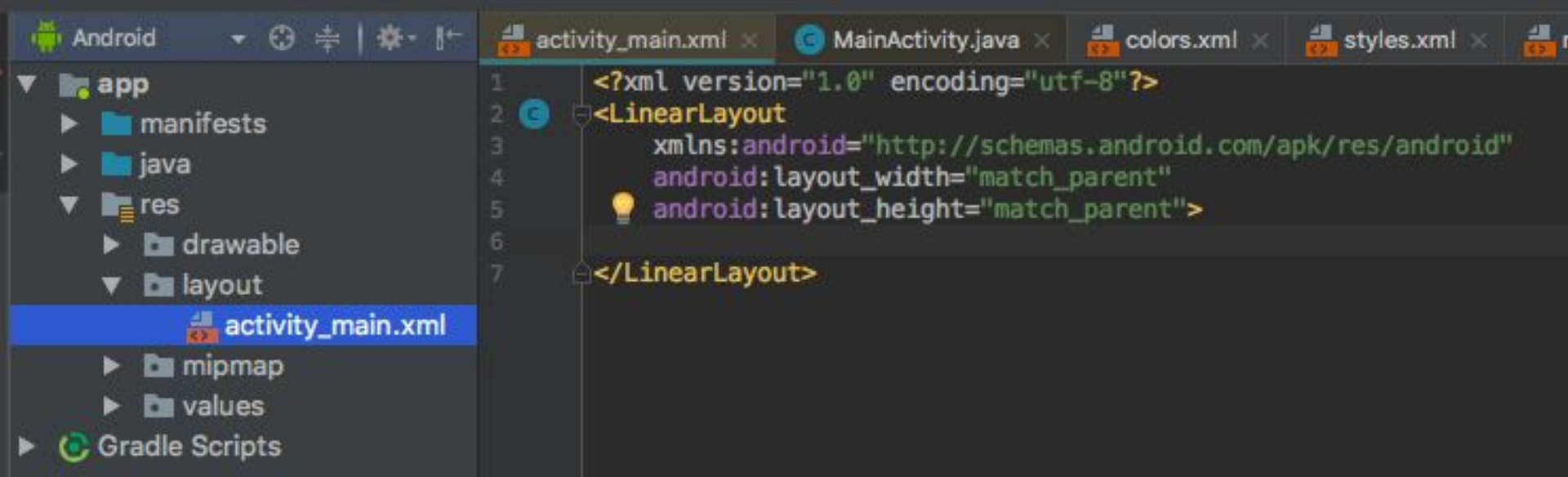
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <selector
3      xmlns:android="http://schemas.android.com/apk/res/android">
4      <item>
5          <shape>
6              <gradient
7                  android:type="radial"
8                  android:startColor="@color/verde3"
9                  android:endColor="@color/verde4"
10                 android:gradientRadius="100%p"
11             />
12          </shape>
13      </item>
14  </selector>
15
```

Resolvendo o Exercício

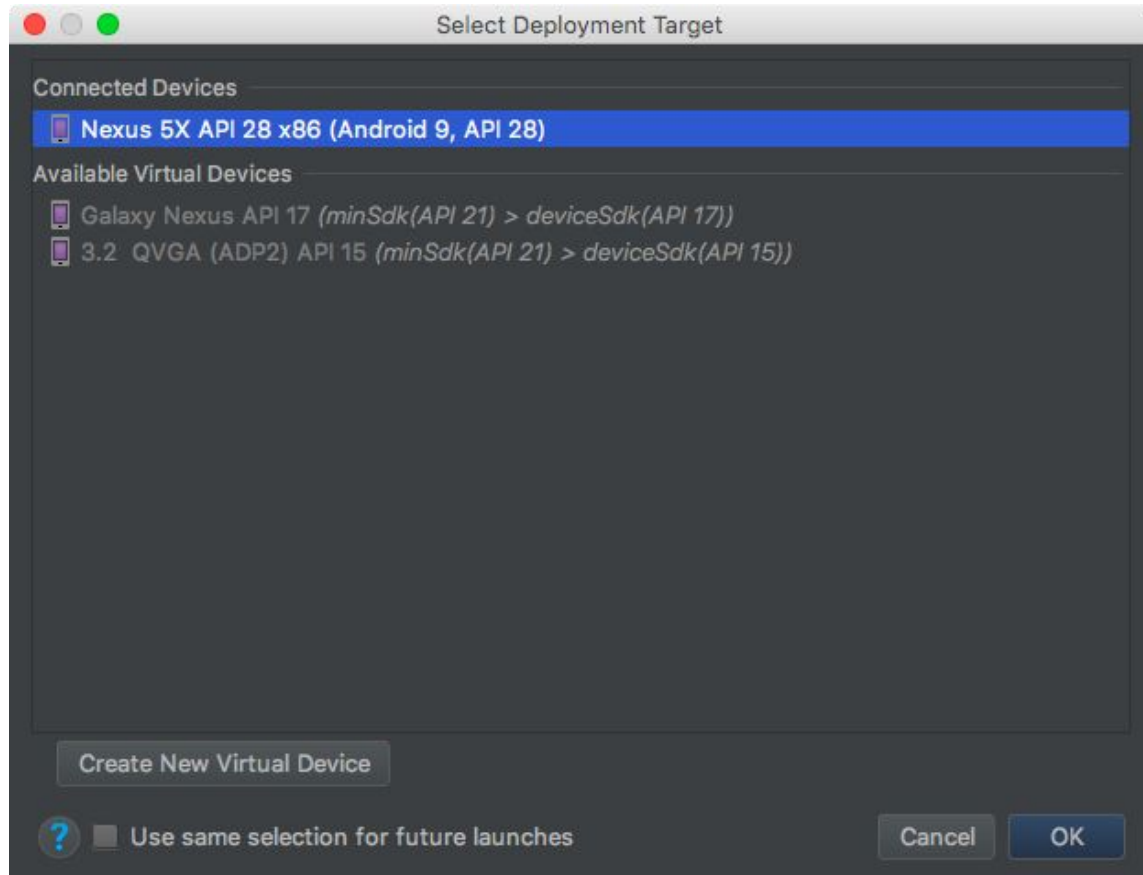


Dica: não se esqueça que o atributo *windowBackground* deve ser precedido pelo namespace *android*.

Resolvendo o Exercício

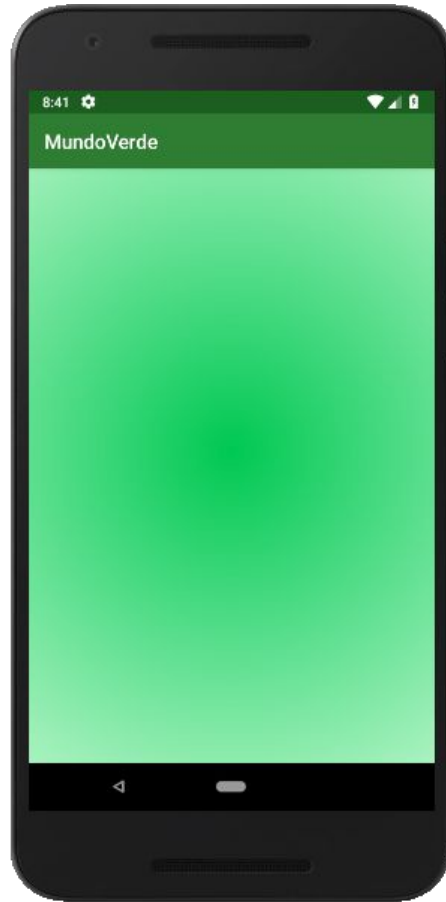


Resolvendo o Exercício



Não se esqueça de executar o projeto em emulador (ou celular) igual ou posterior à API 21 (Android 5.0).

Resolvendo o Exercício



Estudo de Caso

Nesta aula vamos estudar como estilizar as **Views** da aplicação. Para isso, criaremos uma tela estática conforme o protótipo ao lado.

Itens necessários:

- ✓ Retirar *ActionBar*
- ✓ Trabalhar com cores e gradiente
 - **Imagens**
 - Layouts mais sofisticados e aninhados
 - Criação de shapes circulares



*Na próxima aula continuar
nós vamos...*



Obrigado!

