Unprojeção
CENTRO UNIVERSITÁRIO

Linguagem e Técnica de Programação Mobile

AULA 3 – Começando a programar no Google Android

Prof. João Paulo Pimentel
joao.pimentel@projecao.br

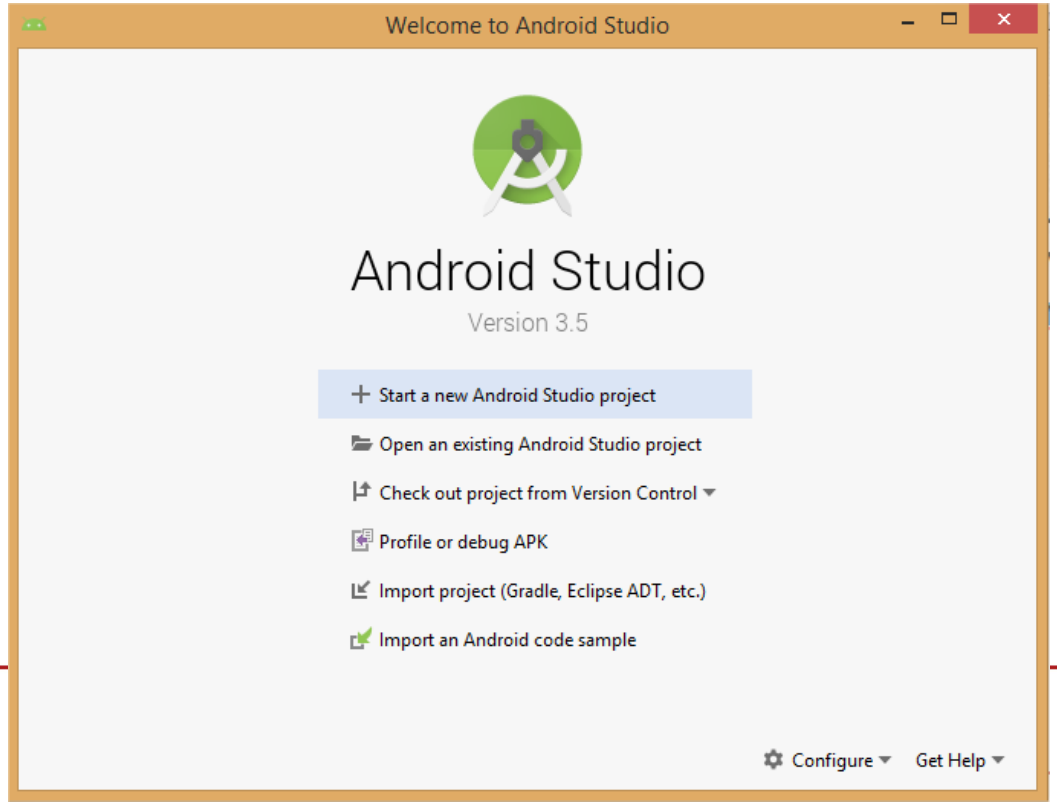
Roteiro da Aula

- Começando a programar no Google Android;
- Conhecendo a estrutura geral de um projeto no Android Studio:
  - O diretório “app” (application);
  - O diretório “res” (resources);
  - O diretório “drawable”;
  - O diretório “layout”;
  - O diretório “values”;
  - O diretório “mipmap”;
- Visão geral da ferramenta de desenvolvimento;
- Executando a nossa aplicação.

Começando a programar no Google Android

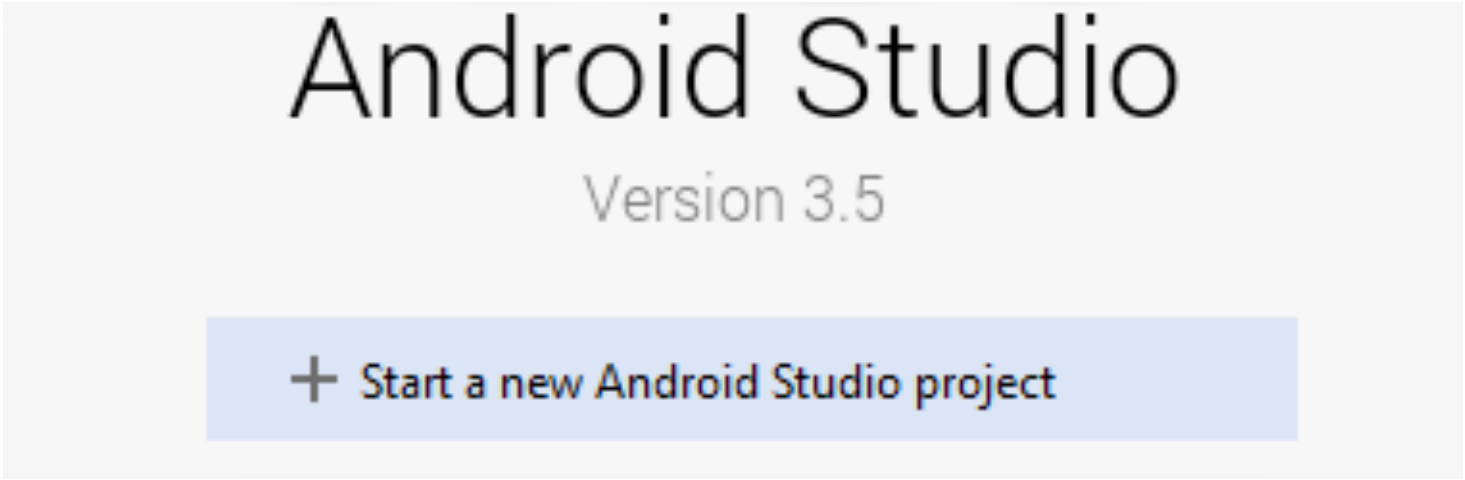
Começando a programar no Google Android

- Depois de tudo configurado e instalado (Aula 2), vamos a partir de agora começar a desenvolver nossas aplicações para a plataforma Android usando o **Android Studio**.



Começando a programar no Google Android

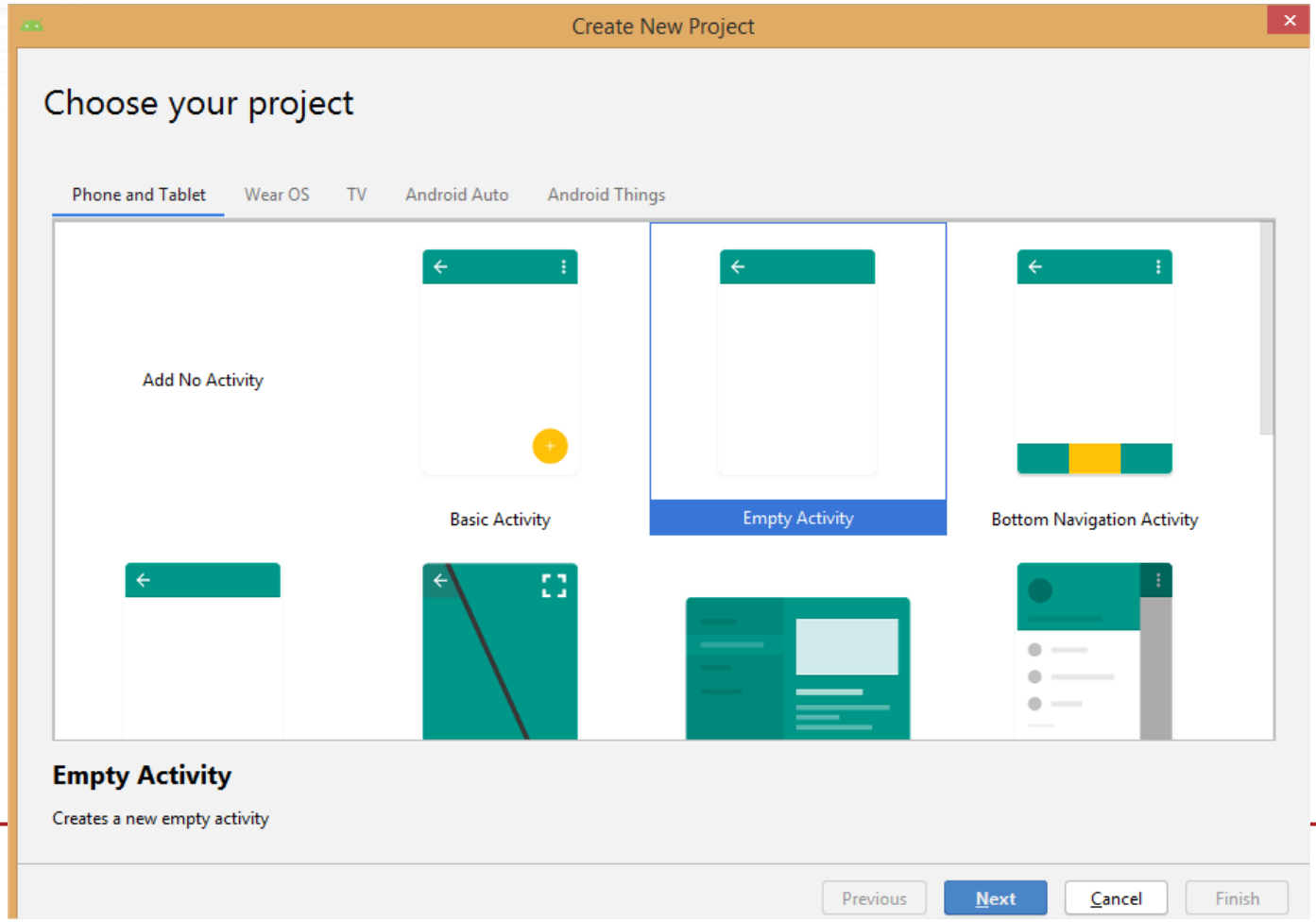
- Na janela Welcome to Android Studio, clique em ***Start a new Android Studio project***.



- Ou, se tiver um projeto aberto, selecione ***File > New > New Project.***

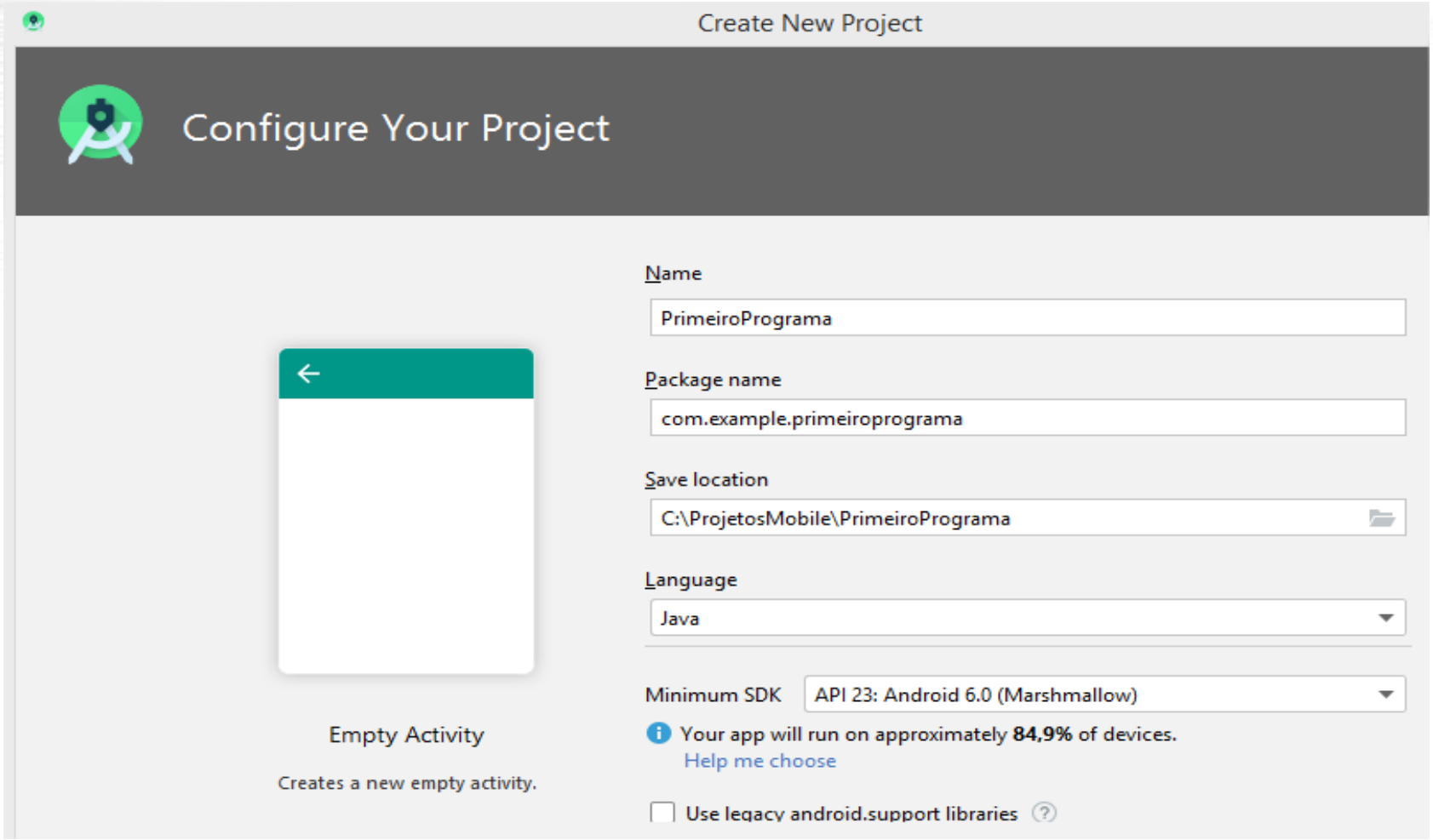
Começando a programar no Google Android

- Na janela ***Select a Project Template***, selecione ***Empty Activity***. Clique em ***Next***.



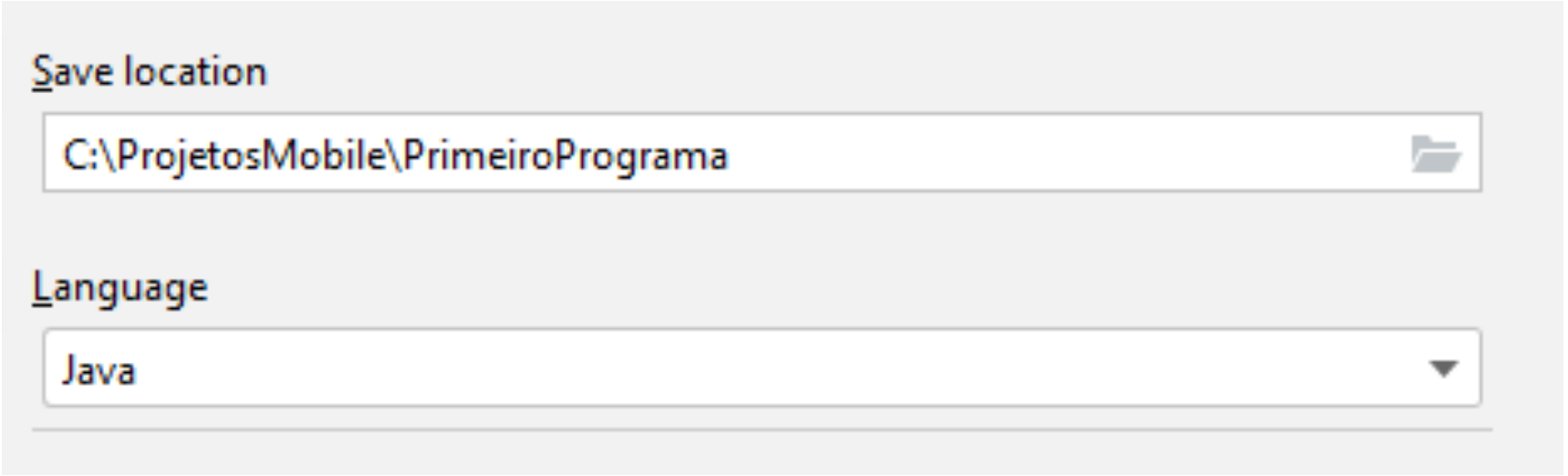
Começando a programar no Google Android

- Na janela ***Configure Your Project***, insira os seguintes valores (mude para para **Java**, se estiver em **Kotlin**):



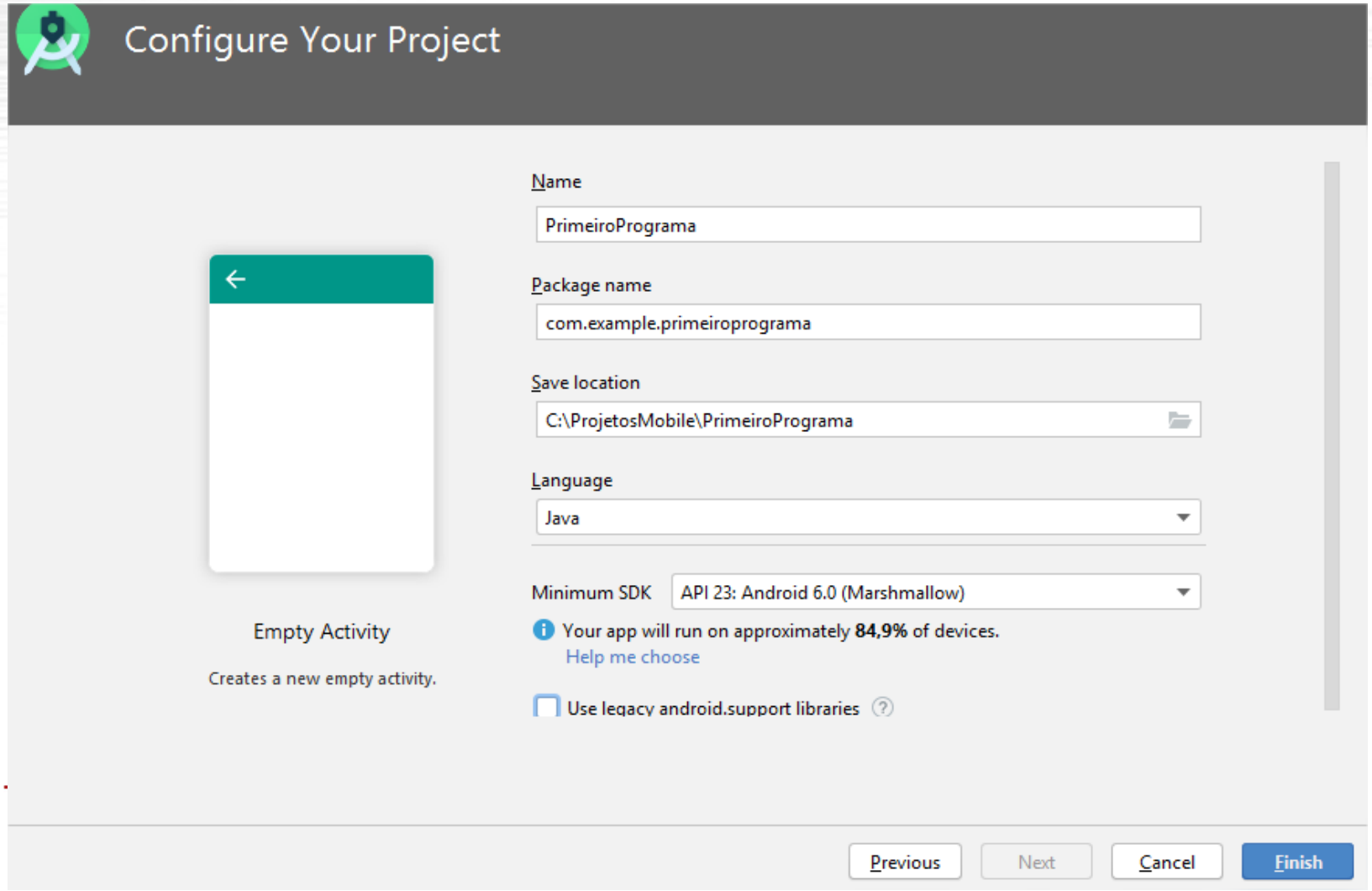
Começando a programar no Google Android

- Agora em “**Save location**” , que indica onde o projeto será criado, fica por sua escolha, **PORÉM**, o diretório (**caminho**) que você escolher não deverá conter espaços em branco, **fica a dica** . Vejamos como ficou na figura abaixo :



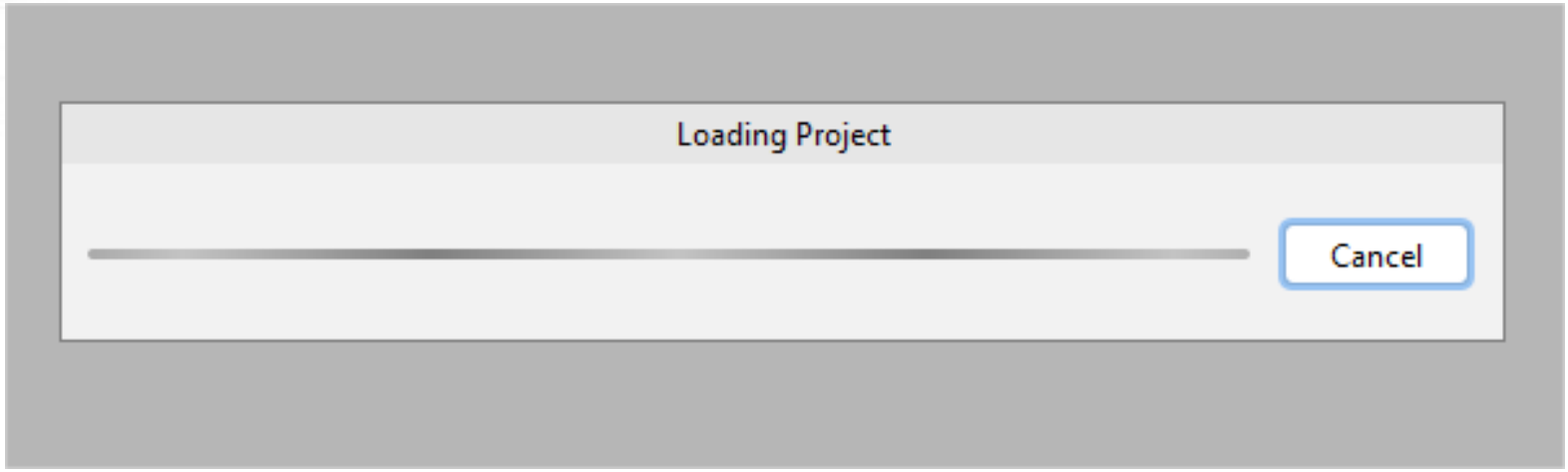
Começando a programar no Google Android

- Após tudo configurado, clique em ***Finish***:

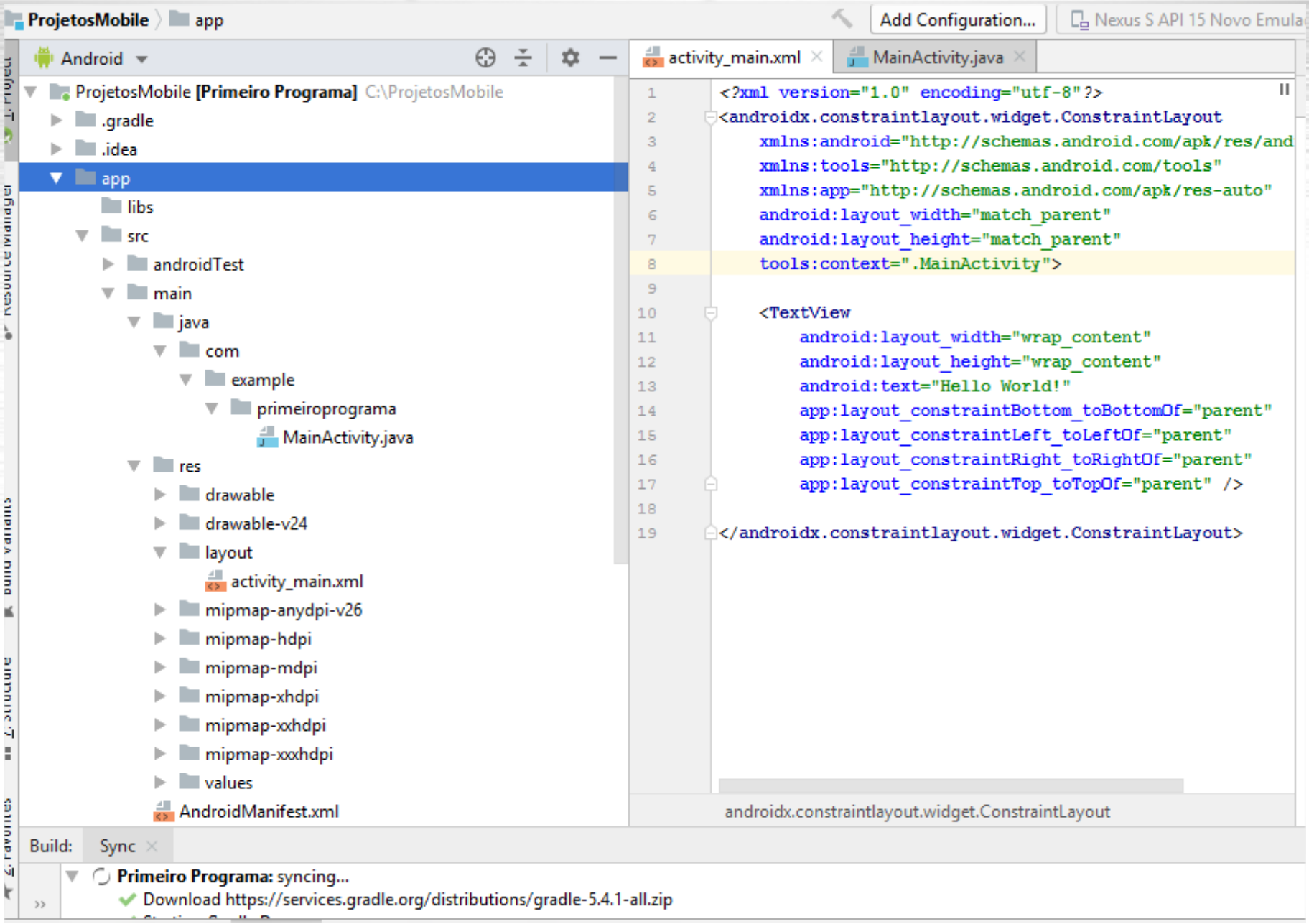


Começando a programar no Google Android

- Criando e carregando o Projeto...



Arquivos Importantes



Arquivos Importantes

activity_main.xml | MainActivity.java | AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    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="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

Arquivos Importantes

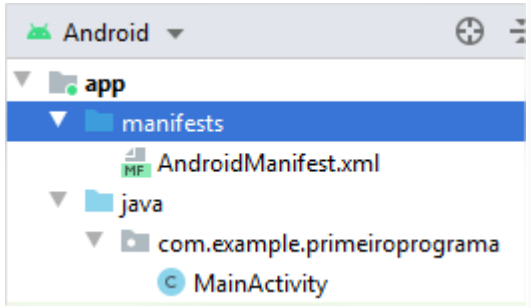
activity_main.xml | MainActivity.java

```
1  package com.example.primeiroprograma;
2
3  import androidx.appcompat.app.AppCompatActivity;
4  import android.os.Bundle;
5
6  public class MainActivity extends AppCompatActivity {
7
8      @Override
9      protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
14
```

Arquivos Importantes

activity_main.xml | MainActivity.java | AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.primeiroprograma">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="PrimeiroPrograma"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportsRtl="true"
11         android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15
16                 <category android:name="android.intent.category.LAUNCHER" />
17             </intent-filter>
18         </activity>
19     </application>
20
21 </manifest>
```



Sobre o AndroidManifest

- É obrigatório todo aplicativo **Android** ter um arquivo **AndroidManifest.xml** no diretório raiz do projeto.
- O arquivo **AndroidManifest** é responsável por apresentar informações essenciais e importantes sobre o aplicativo ao **sistema operacional Android**.
- Essas informações são utilizadas para o sistema operacional Android **executar o aplicativo com suas devidas permissões e configurações.**

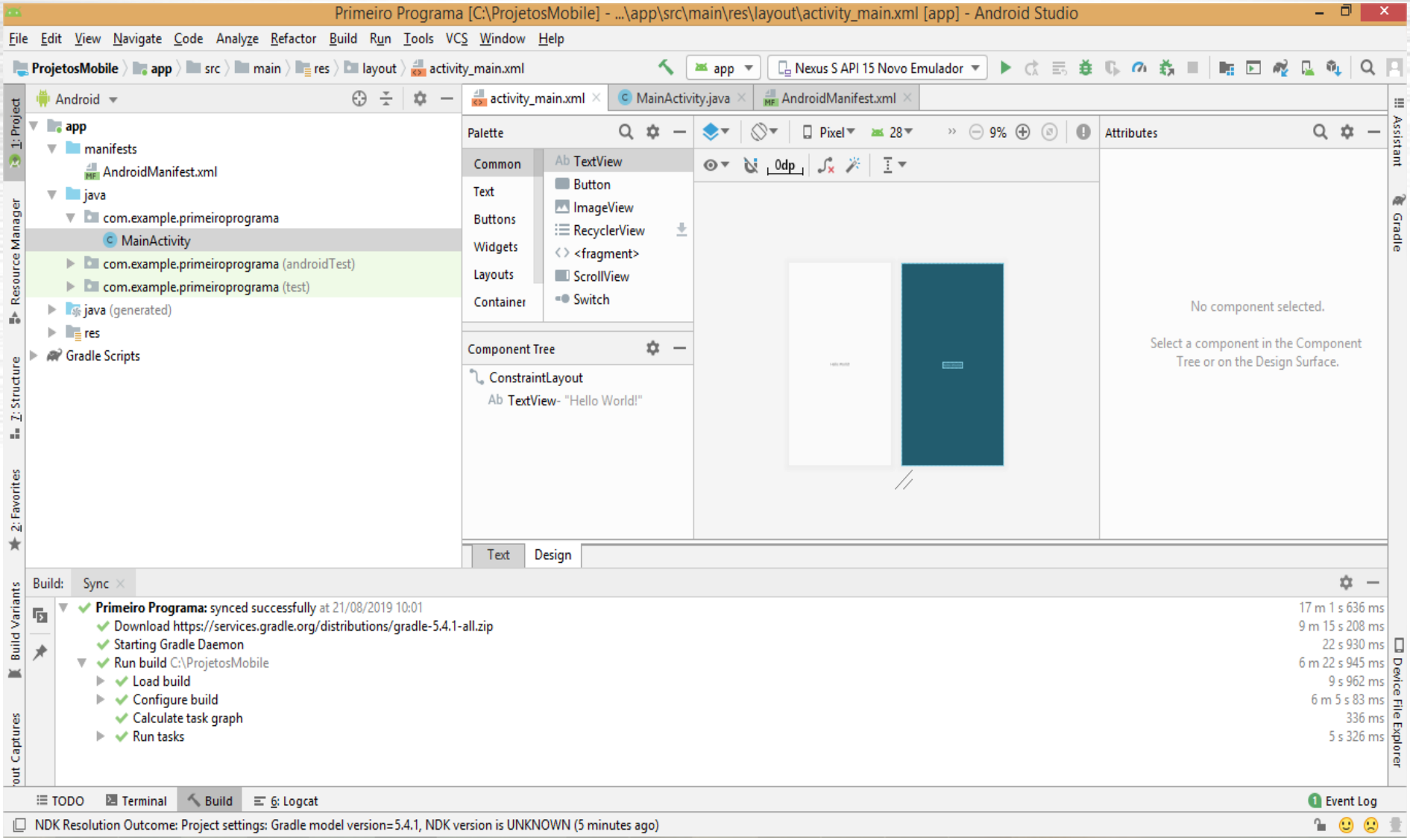
Começando a programar no Google Android

- Toda classe (**arquivos ".java")** na plataforma Android, que representa uma aplicação, é considerada uma "atividade" (**Activity**).

activity_main.xml | MainActivity.java | AndroidManifest.xml

```
1   package com.example.primeiroprograma;
2
3   import androidx.appcompat.app.AppCompatActivity;
4
5   import android.os.Bundle;
6
7   public class MainActivity extends AppCompatActivity {
8
9       @Override
10      protected void onCreate(Bundle savedInstanceState) {
11          super.onCreate(savedInstanceState);
12          setContentView(R.layout.activity_main);
13      }
14  }
15
```

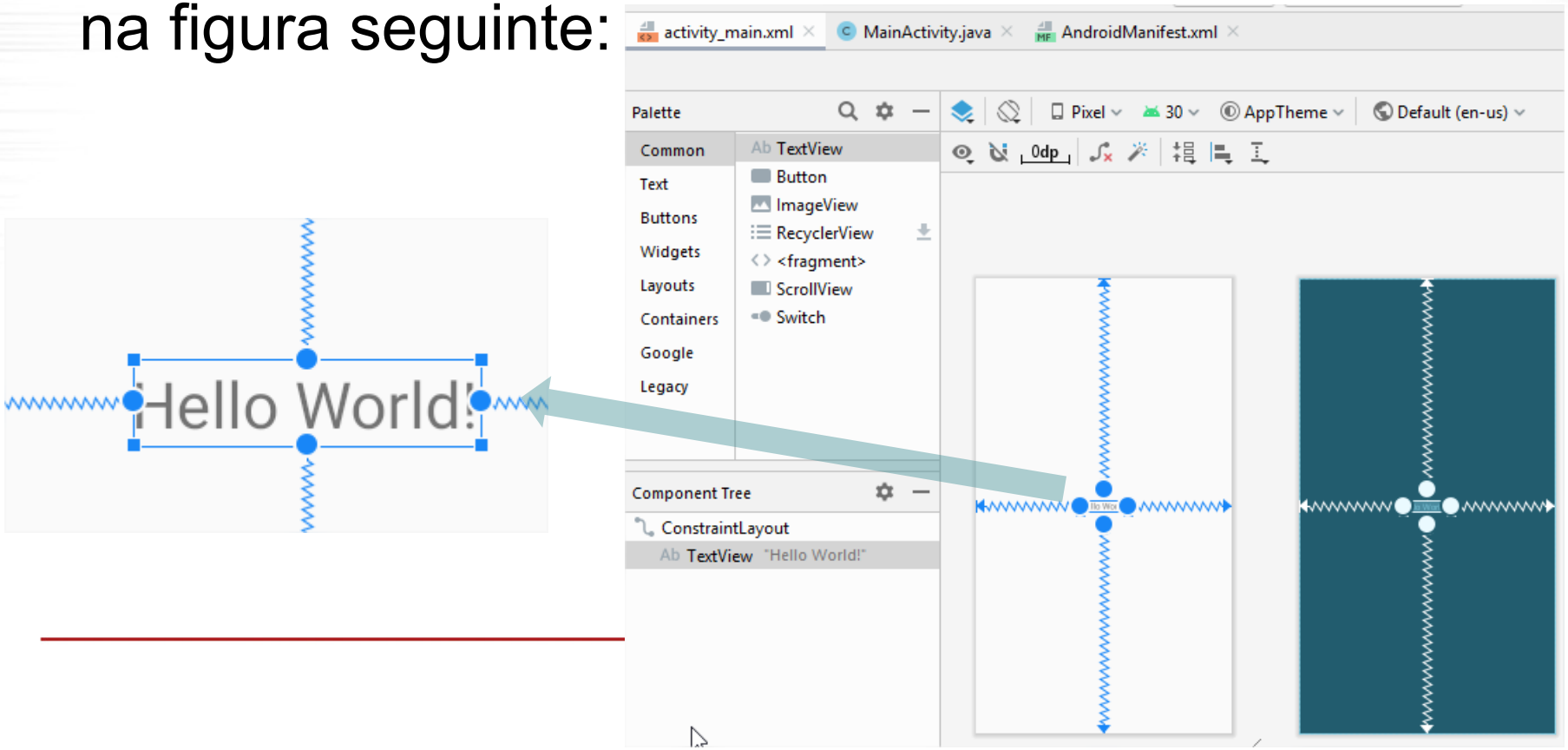
Começando a programar no Google Android



Conhecendo a estrutura geral de um projeto no Android Studio

Conhecendo a estrutura geral de um projeto no Android Studio

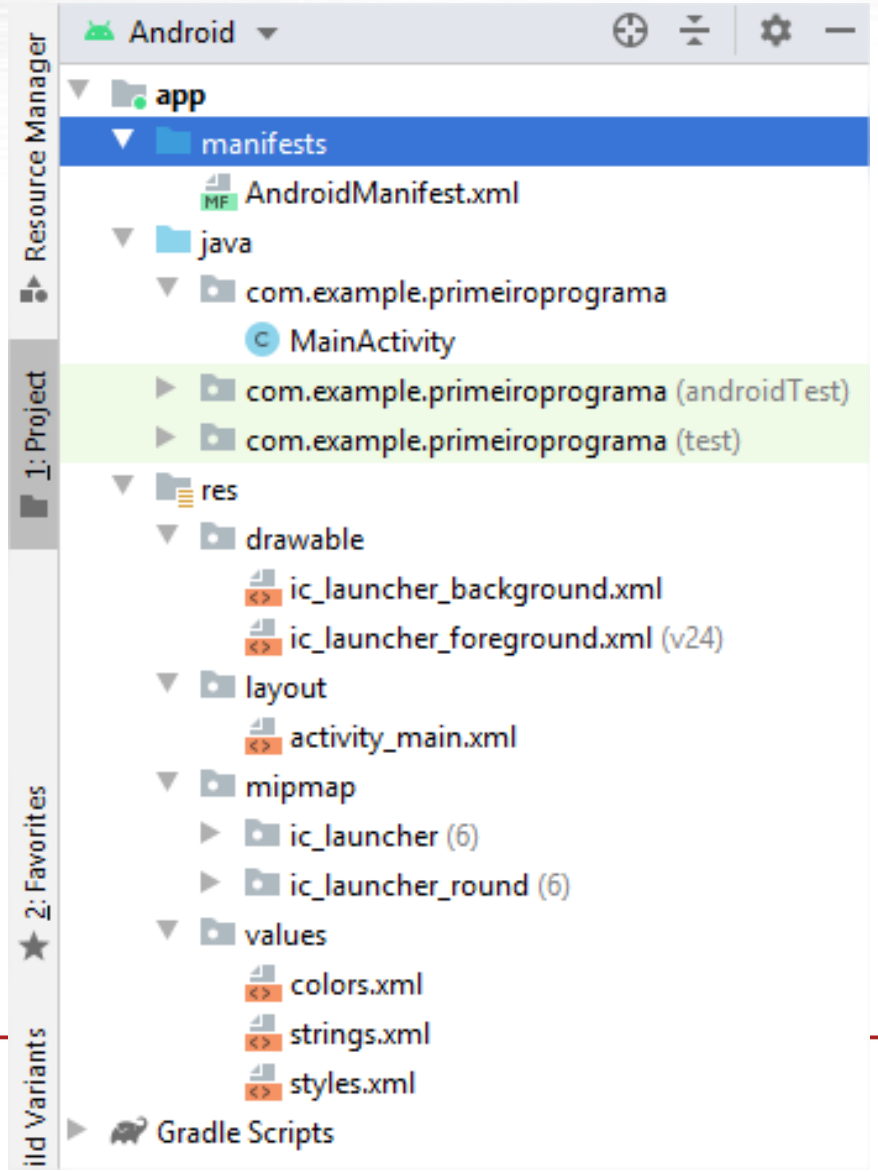
- A primeira coisa que visualizamos que nos chama atenção é a tela do dispositivo com a frase “**Hello world**”, conforme podemos conferir na figura seguinte:



Conhecendo a estrutura geral de um projeto no Android Studio

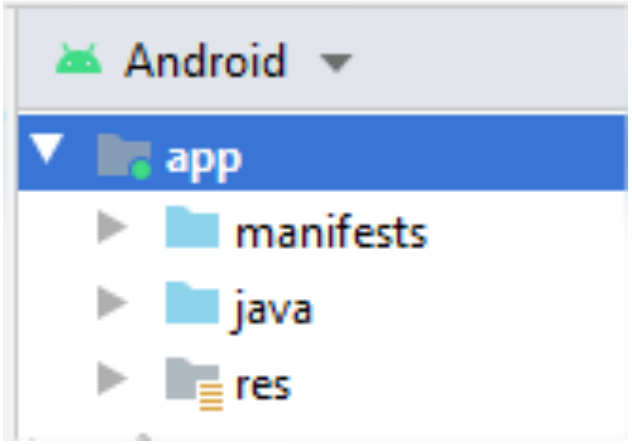
- Se observamos a figura no slide anterior, ao lado da tela do dispositivo temos uma **paleta de componentes** disponíveis que podemos utilizar para construir as nossas aplicações.
- Se visualizarmos **o lado esquerdo**, existe uma seção chamada "**Project**", onde nela existe uma pasta chamada "**app**" (que é o nosso projeto), constituído por vários subdiretórios , que por sua vez, possui seus respectivos arquivos, conforme demonstra a figura no próximo slide:

Conhecendo a estrutura geral de um projeto no Android Studio



Conhecendo a estrutura geral de um projeto no Android Studio

- Vamos conhecer agora toda a estrutura de um projeto Android.
- Ele é composto por várias pastas e arquivos, cada um com uma finalidade em especifico.



Conhecendo a estrutura geral de um projeto no Android Studio

- O diretório “**app**” (application)

- Dentro desse diretório temos todos os arquivos principais de nossa aplicação, distribuído pelos seus diretórios.
- Ao expandirmos o diretório “**app**” temos as pastas “**manifest**” e “**java**”.
- Dentro da pasta “**manifest**” temos o arquivo “**AndroidManifest.xml**”, com o seguinte código apresentado no próximo slide:

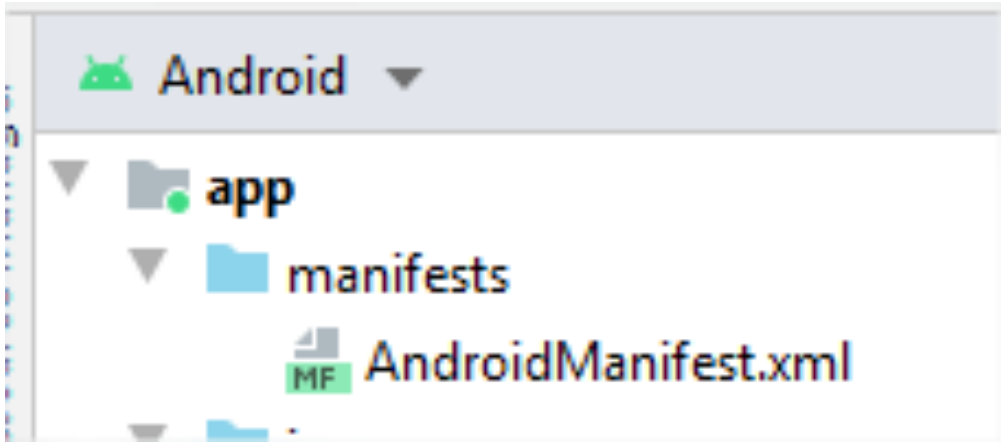
Conhecendo a estrutura geral de um projeto no Android Studio

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.primeiroprograma">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="PrimeiroPrograma"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportsRtl="true"
11         android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15
16                 <category android:name="android.intent.category.LAUNCHER" />
17             </intent-filter>
18         </activity>
19     </application>
20
21 </manifest>
```

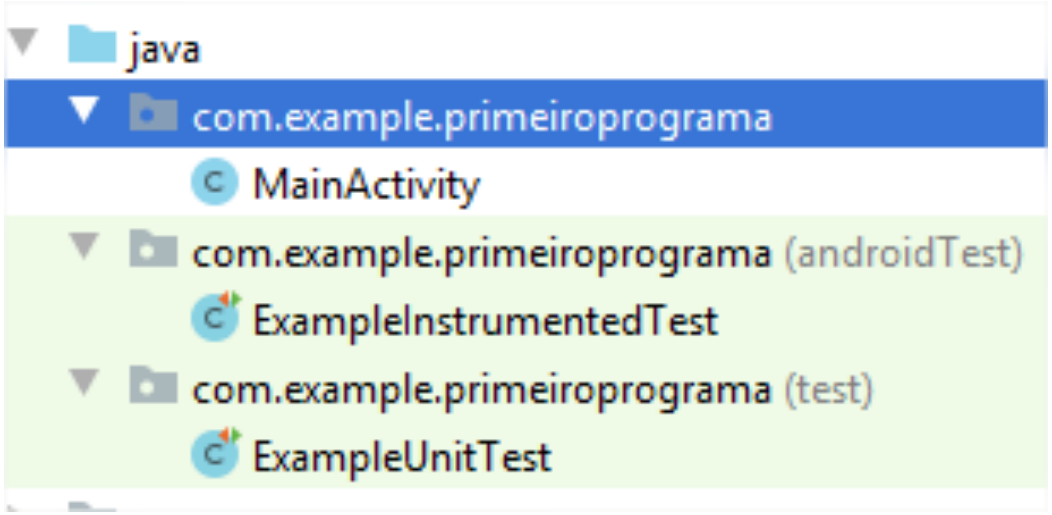
Conhecendo a estrutura geral de um projeto no Android Studio

- Esse arquivo é considerado o "**sistema nervoso**" do Android, é através dele que definimos todas as permissões e configurações primordiais de nossa aplicação para que a mesma seja executada (**não iremos fazer nenhuma modificação nesse arquivo**).



Conhecendo a estrutura geral de um projeto no Android Studio

- Dentro da pasta "**java**" temos as nossas classes (**arquivos ".java"**) que fazem parte da nossa aplicação. Observe que dentro da nossa pasta existem **três pacotes** de mesmo nome ("***com.example.primeiroprograma***"), sendo os dois últimos voltado para testes.



Conhecendo a estrutura geral de um projeto no Android Studio

- No primeiro pacote existe um arquivo chamado "**MainActivity.java**" , que é a **classe principal** do programa, que é através dela que definimos todos os ações da nossa aplicação:

Conhecendo a estrutura geral de um projeto no Android Studio

- Diferentemente das aplicações comuns em Java (**J2SE**), toda classe que representa uma aplicação Android deve ser derivada da classe **Activity** (atividade), conforme você pode ver através do seguinte trecho de código:

```
public class HelloActivity extends Activity {
      :
```

Conhecendo a estrutura geral de um projeto no Android Studio

- A classe "**MainActivity**" possui como método principal o **onCreate**. A primeira linha de código presente neste método faz chamada ao método **onCreate** da classe base (a classe **Activity**), conforme podemos conferir abaixo:

```
super.onCreate(savedInstanceState);
```

Conhecendo a estrutura geral de um projeto no Android Studio

- Logo após esse método vem o método **setContentView**, responsável por exibir a tela da minha aplicação baseado nos layouts “**xml**".
- Por padrão ele chama o arquivo “**activity_main.xml**”.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

Conhecendo a estrutura geral de um projeto no Android Studio

- O diretório “**res**” (resources)

- Dentro da estrutura de um projeto Android existe um diretório chamado “**res**” (resources), onde dentro dele colocamos todos os recursos que podemos utilizar em um programa Android (como **imagens**, **sons**, **músicas** e etc).
- Vamos conhecer os subdiretórios existentes dentro dessa pasta e a finalidade de cada um deles nos próximos slides.

Conhecendo a estrutura geral de um projeto no Android Studio

- O diretório “**drawable**”

- O diretório “**drawable**” presente dentro da pasta “**res**” do nosso projeto possui uma única finalidade: armazenar imagens que serão visualizadas dentro de uma aplicação Android.
- Uma coisa que precisa ser reforçada aqui é que os arquivos de imagens presentes nessa pasta não podem estar escritos de qualquer forma. Os arquivos a serem copiados dentro desse diretório devem possuir somente **letras minúsculas** (de “a” até “z”), **números** (de “0” a “9”) e **underline** (“_”).

Conhecendo a estrutura geral de um projeto no Android Studio

- O diretório “**layout**”

- O diretório “**layout**” armazena todos os arquivos referentes as telas de uma aplicação Android, que normalmente são arquivos “**.xml**”.
- Para aqueles que conhecem a programação de **HTML com JavaScript** para a construção de páginas Web, no Android é similar, é a combinação de **Java com XML** para a **construção de aplicativos Mobile**.

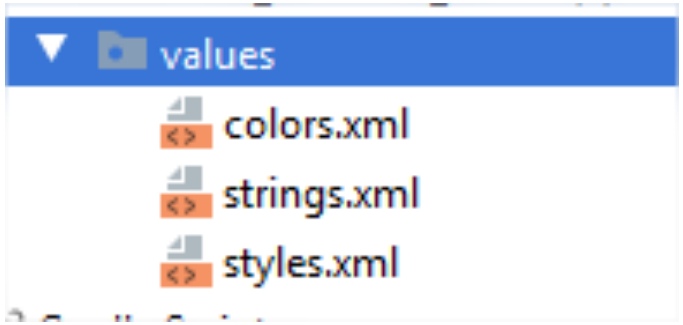
- O diretório “**values**”

- Um dos recursos que o Android permite que usemos na construção das telas de nossa aplicação são “**constantes**”. Como bom programador que todos nós somos, sabemos que uma “**constante**” nada mais é do que um endereço de memória que vai armazenar um determinador valor, que será único até o fim da execução do programa. **Mas, o que isso tem a ver com as telas da nossa aplicação?**

- Normalmente usamos **constantes** para armazenar **valores fixos** que, naturalmente, estarão presentes na maior parte do seu programa. É muito comum criarmos um título como nome de uma aplicação (como por exemplo : "**JPP Software**"), que vai estar presente na maioria das telas (ou em todas) da sua aplicação. Imagine você digitar "manualmente" o nome de seu **software** em todas as telas da sua aplicação, seria um processo trabalhoso você não acha?

Conhecendo a estrutura geral de um projeto no Android Studio

- Agora imagine "**futuramente**" que você precise "**mudar**" o nome de sua aplicação? Você terá novamente o mesmo trabalho para digitar em cada tela o nome da nova aplicação, sem dúvida. Usando **constantes**, você não terá esse trabalho.
- Dentro dessa pasta temos os seguintes arquivos:
  - **colors.xml**
  - **strings.xml**
  - **styles.xml**



Conhecendo a estrutura geral de um projeto no Android Studio

- “**colors.xml**” : esse arquivo guarda constantes relacionadas cores básicas que podem ser utilizadas na aplicação.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="colorPrimary">#6200EE</color>
4      <color name="colorPrimaryDark">#3700B3</color>
5      <color name="colorAccent">#03DAC5</color>
6  </resources>
```

Conhecendo a estrutura geral de um projeto no Android Studio

- “**strings.xml**” : esse arquivo guarda constantes relacionadas à nossa aplicação em geral (como o nome da aplicação, o título que vai aparecer na aplicação e etc.).

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

Conhecendo a estrutura geral de um projeto no Android Studio

- “**styles.xml**” : esse arquivo guarda constantes relacionadas à estilos que podemos utilizar em nossa aplicação.

```
1  <resources>
2      <!-- Base application theme. -->
3      <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
4          <!-- Customize your theme here. -->
5          <item name="colorPrimary">@color/colorPrimary</item>
6          <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
7          <item name="colorAccent">@color/colorAccent</item>
8      </style>
9
10 </resources>
```

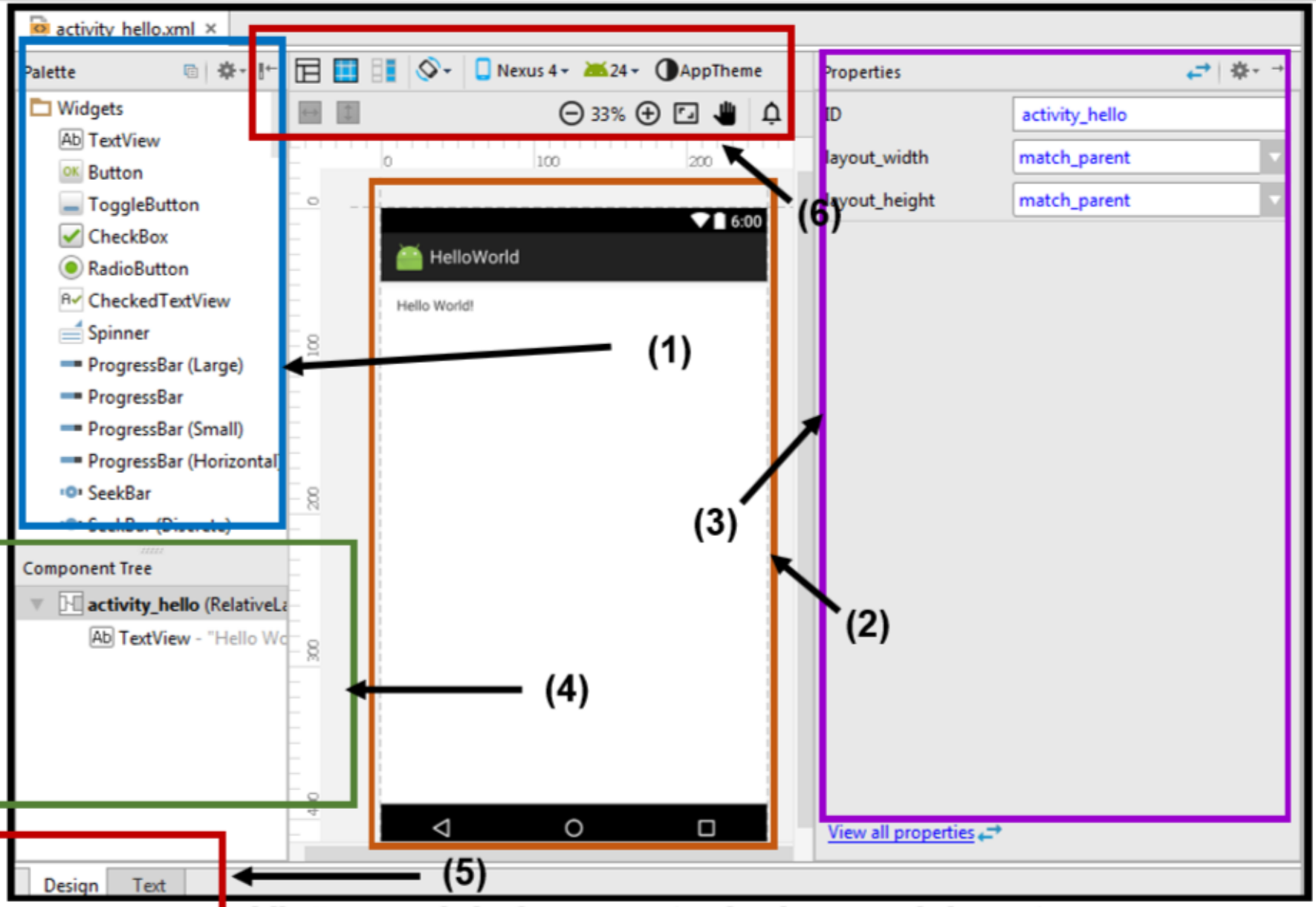
- O diretório “**mipmap**”
- O diretório “**mipmap**” possui as mesmas características do diretório “**drawable**” (armazenar imagens), porém, o mesmo foi destinado a armazenar nesta pasta somente imagens referentes ao **ícone** da nossa aplicação Android, que possa se comportar em várias resoluções de tela.

Visão geral da ferramenta de desenvolvimento

- Acabamos de entender e conhecer nos slides anteriores toda a estrutura de diretórios de um projeto Android no Android Studio.

- Agora vamos dar uma visão geral na nossa ferramenta de desenvolvimento (IDE) para a plataforma Android.

Visão Geral do Android Studio



Visão geral da ferramenta de desenvolvimento

Visão geral da ferramenta de desenvolvimento

- Vamos entender as numerações indicadas na figura anterior:
  - Na **indicação (1)** temos a paleta de componentes onde os mesmos estão separados por seções.
  - As seções são : **Widgets** , **Text Fields**, **Layouts** , **Containers**, **Date e Time** , **Expert** e **Custom** (Componentes personalizados, normalmente criados pelo usuário e entre outros recursos).

Visão geral da ferramenta de desenvolvimento

- Na **indicação (2)** temos um **preview** de como ficará a interface da nossa aplicação quando a mesma for executada, nessa interface podemos arrastar e soltar os componentes (indicados por **(1)**) , construindo assim a nossa aplicação.
- Na **indicação (3)** temos as propriedades do componente (Properties). Quando selecionamos um componente, as suas propriedades são visualizadas e assim, conforme nossa necessidade, podemos alterá-las.

Visão geral da ferramenta de desenvolvimento

- Na **indicação (4)** temos uma seção chamada "**Component Tree**". Nessa seção podemos ver , de forma hierárquica, todos os componentes que estão presentes na tela da nossa aplicação.
- Na **indicação (5)** podemos alternar entre o **modo gráfico** ("**Design**", onde temos a visualização da tela da aplicação) e o modo **XML** ("**Text**", onde visualizamos o código **XML** do arquivo, que corresponde a estrutura que forma a tela de nossa aplicação).

Visão geral da ferramenta de desenvolvimento

- Na **indicação (6)** podemos alterar os temas que podemos utilizar em nossa aplicação **Android**, orientação da tela do dispositivo (**retrato ou paisagem**), resolução da tela do dispositivo e etc.
- A maioria das opções disponíveis nessa seção só terá efeito em tempo de projeto. É necessário configurar as mesmas opções para que elas aconteçam em **tempo de execução** (quando você executa o emulador ou dispositivo real), de acordo com a necessidade.

Executando a nossa aplicação

- Agora que já tivemos uma visão geral da ferramenta de desenvolvimento e do projeto em Android (com toda a sua estrutura de diretórios e arquivos), podemos a partir de agora executarmos a nossa aplicação.

- Mas antes de executarmos a nossa aplicação, vamos fazer alguns comentários.

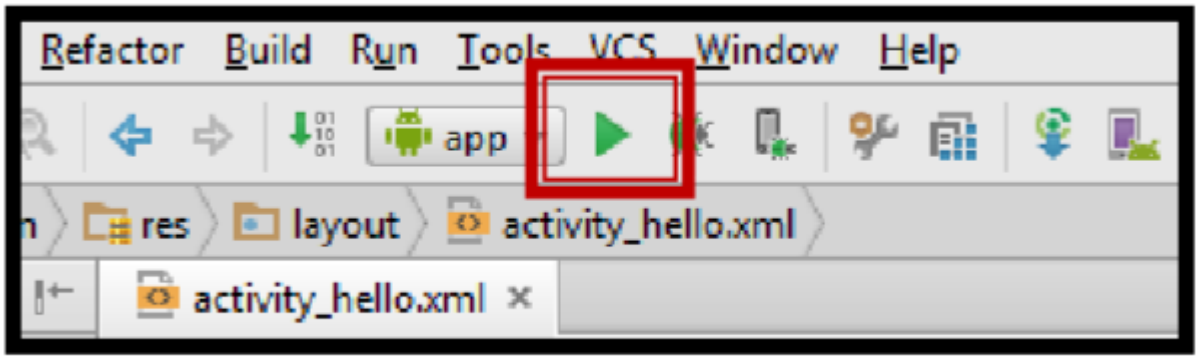
Executando a nossa aplicação

- Observe que quando **criamos** um projeto em **Android**, na tela do dispositivo já tem um componente **TextView** (situado na paleta de componentes , dentro da seção "**Widgets**", nas versões mais atuais fica em "**Common**" ou na "**Text**") onde nele é exibida a frase "**Hello world**".

- Quando executarmos a nossa aplicação através do **emulador**, ela sairá idêntica como está sendo exibida no projeto (uma tela "em branco" com a frase "**Hello world**").

Executando a nossa aplicação

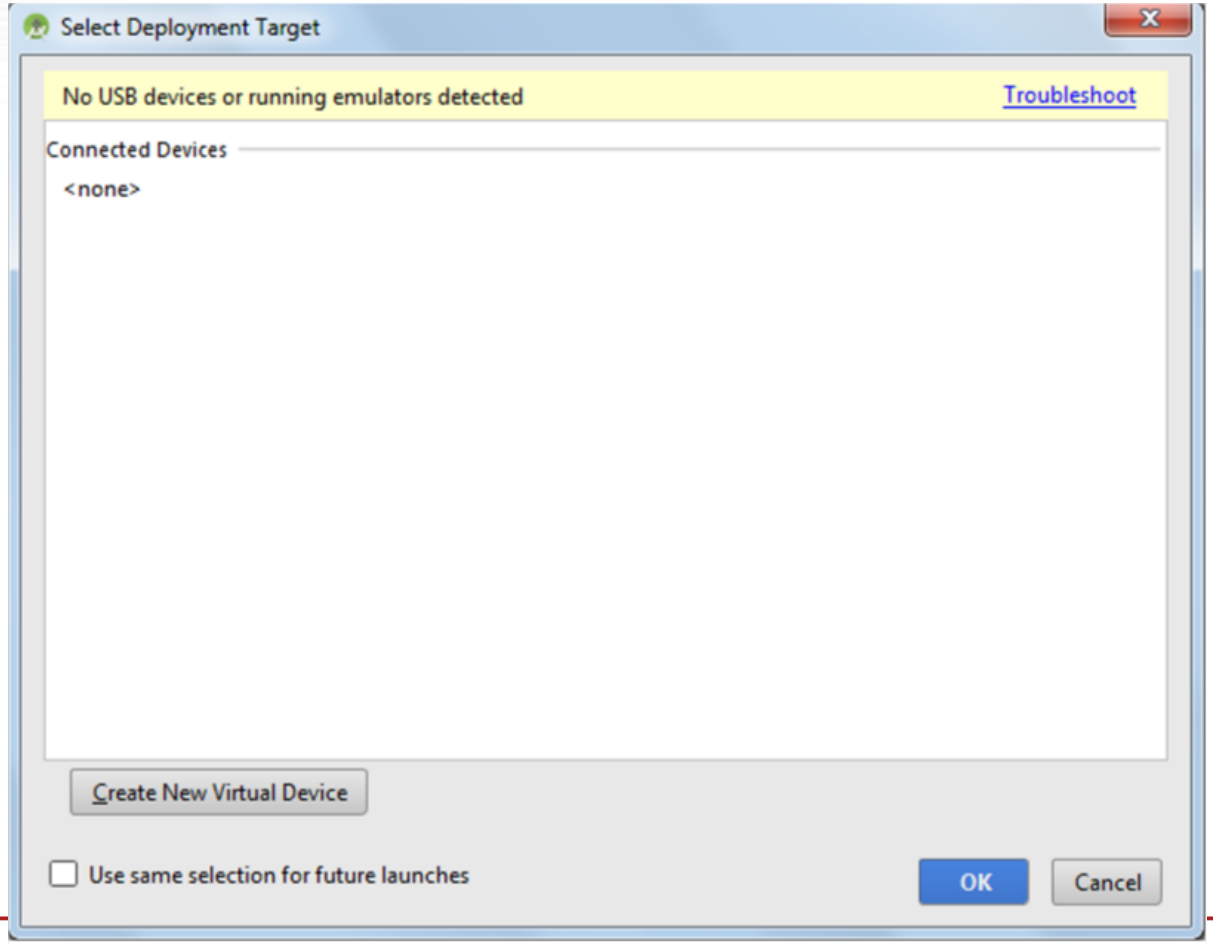
- Como fazer para executarmos a nossa aplicação?
- Para executarmos a nossa aplicação basta irmos no menu "**Run**" / "**Run 'app'** (ou pressionar as teclas **SHIFT+F10**) .
- Uma outra forma também é clicando no botão pressente na barra de ferramentas:



Botão "Run" do Android Studio

Executando a nossa aplicação

- Feito isso será aberta a seguinte caixa de diálogo em seguida:



Caixa de diálogo – Select Deployment Target

Executando a nossa aplicação

- Para executarmos a nossa aplicação iremos fazer uso de um dispositivo virtual (conhecido como "**Android Virtual Device**"), porém, esse dispositivo virtual não está criado.

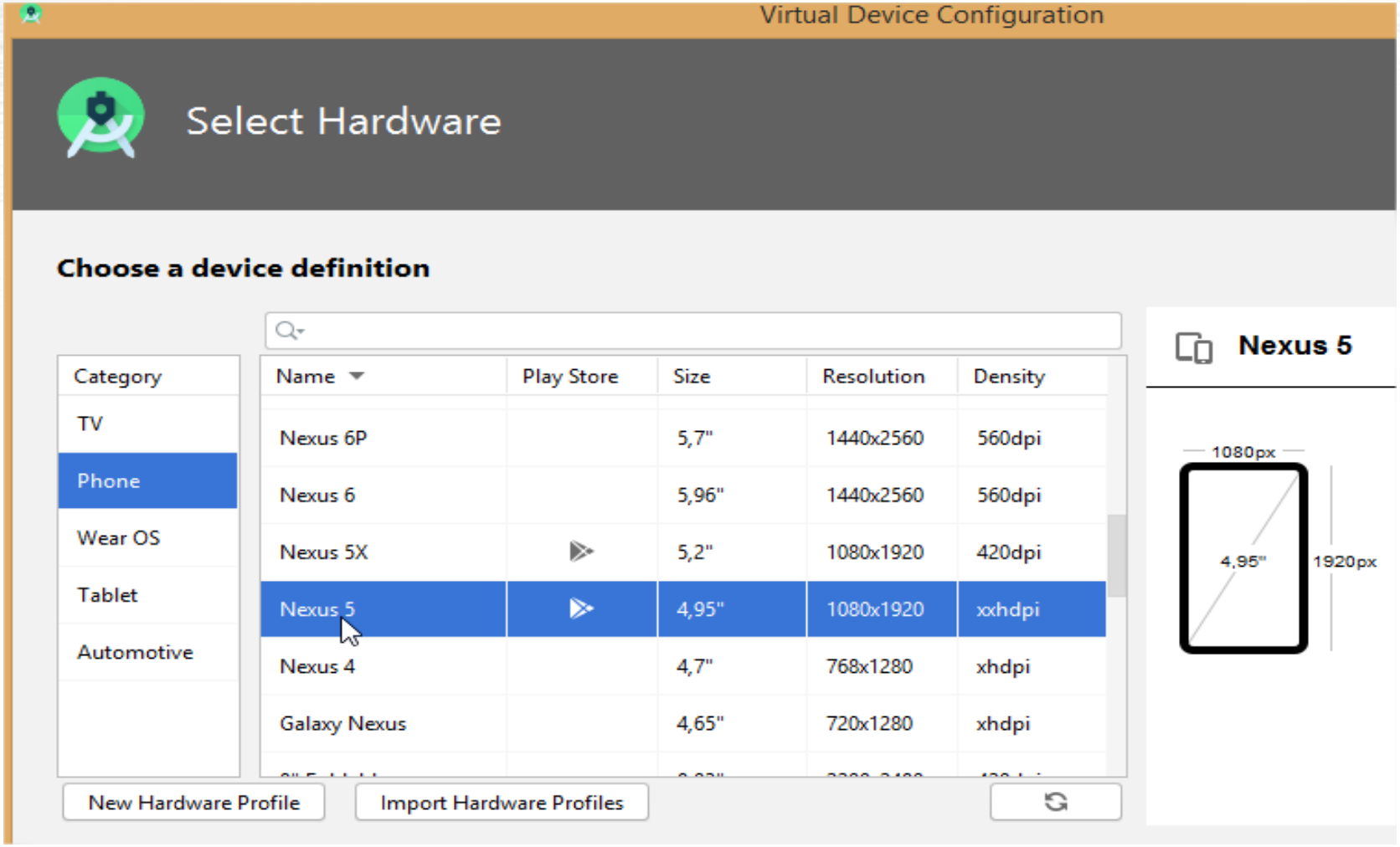
- Vamos nesse exato momento criar um dispositivo virtual que irá executar as nossas aplicações Android. Para criarmos um dispositivo basta clicarmos no botão "**Create New Virtual Device**", conforme indicado na figura a seguir no próximo slide:

Executando a nossa aplicação

Criando um novo dispositivo

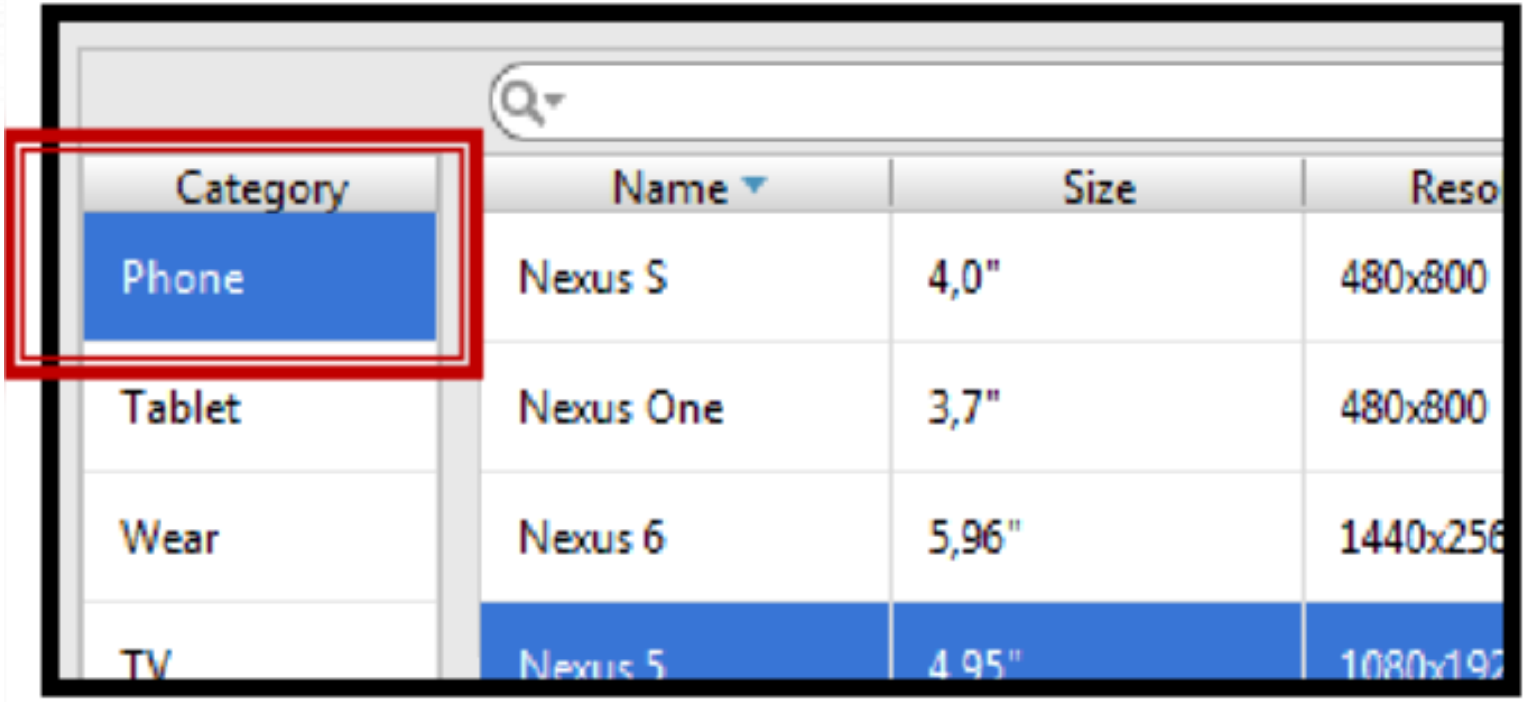
Executando a nossa aplicação

- Feito isso será aberta a seguinte caixa de diálogo abaixo:



Executando a nossa aplicação

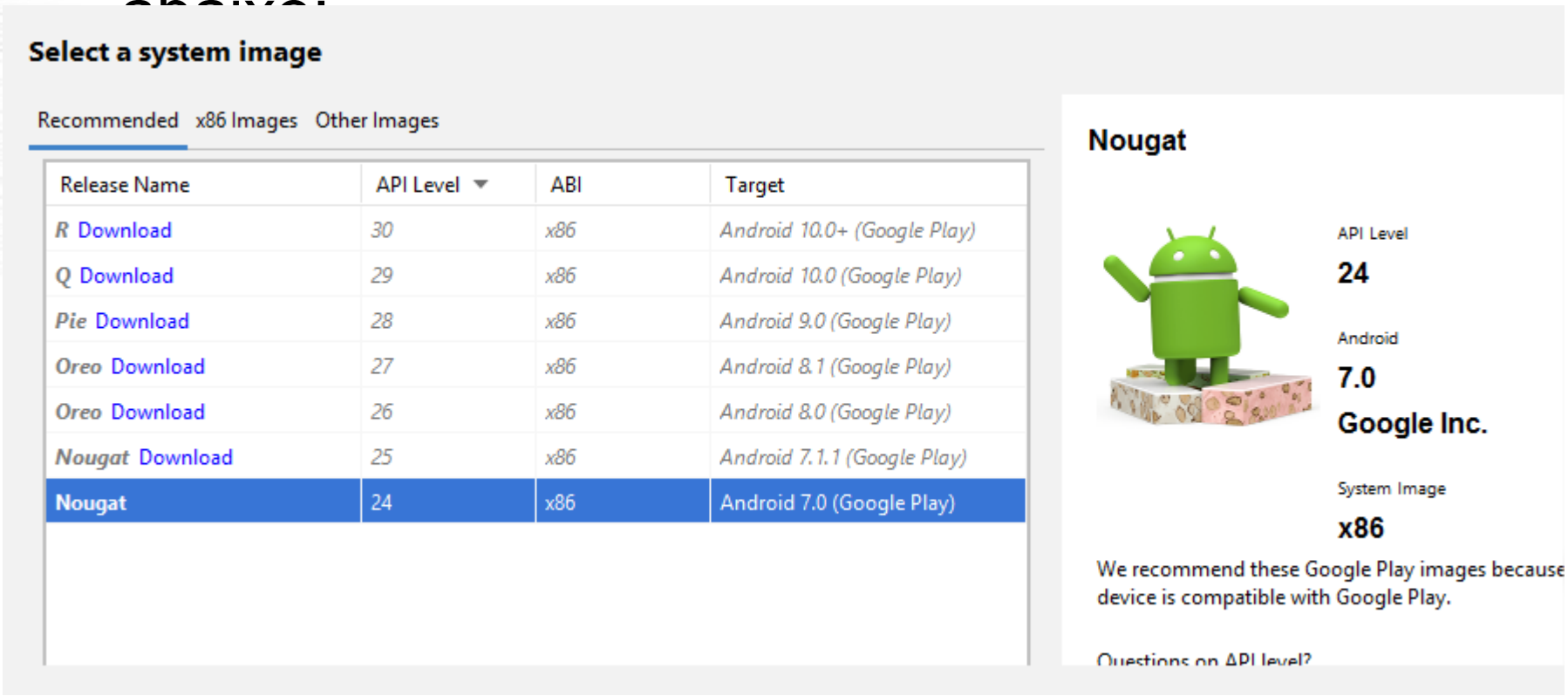
- Na guia "**Category**" vamos deixar marcado a opção "**Phone**", conforme é mostrado na figura a seguir:



Escolhendo a categoria

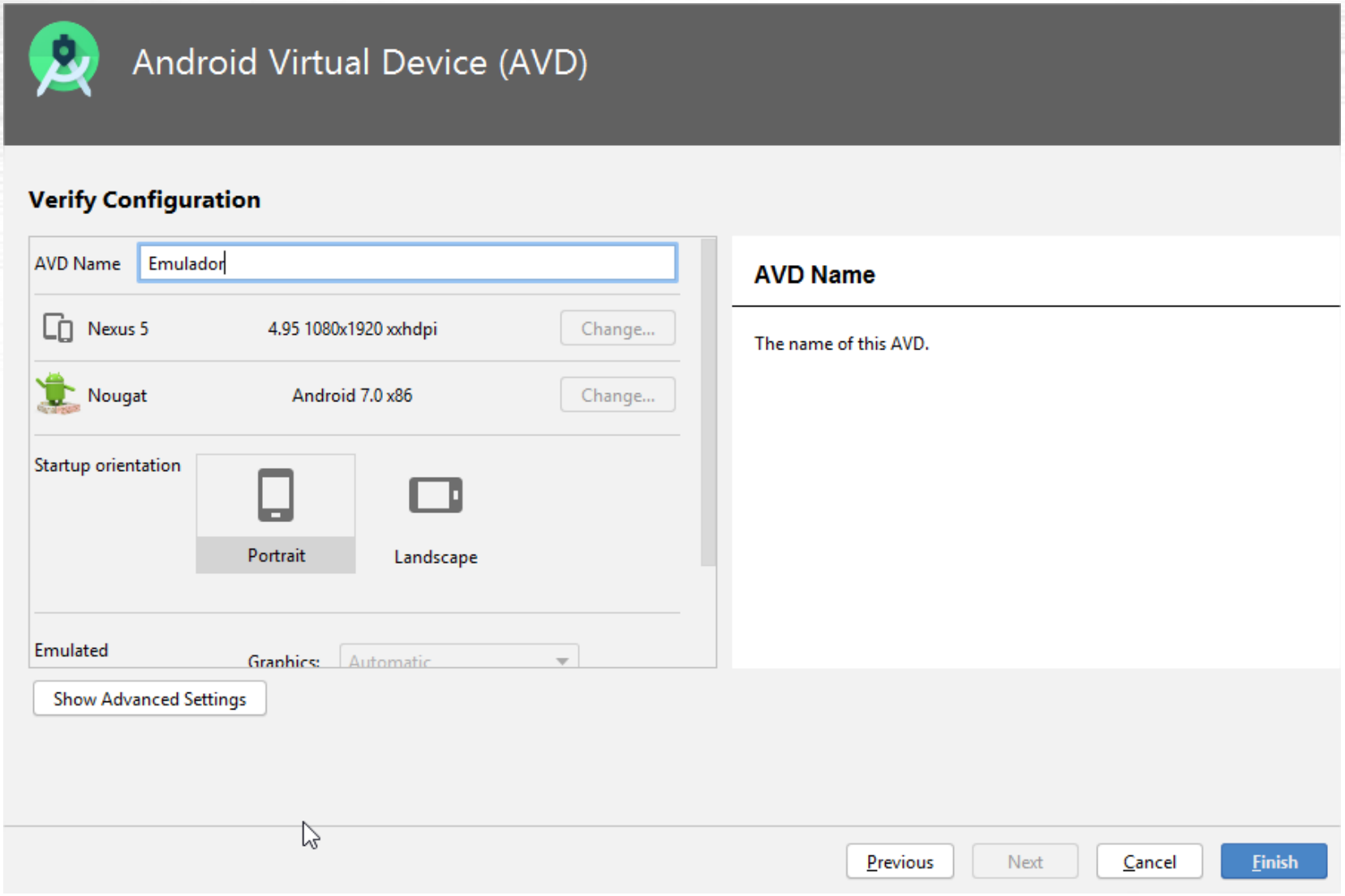
Executando a nossa aplicação

- Como modelo de **smatphone** utilizado para executar as nossas aplicações selecione a opção “**Nougat, API Level 24, Target Android 7.0 (Google Play)**”,conforme indicado na figura abaixo:



Executando a nossa aplicação

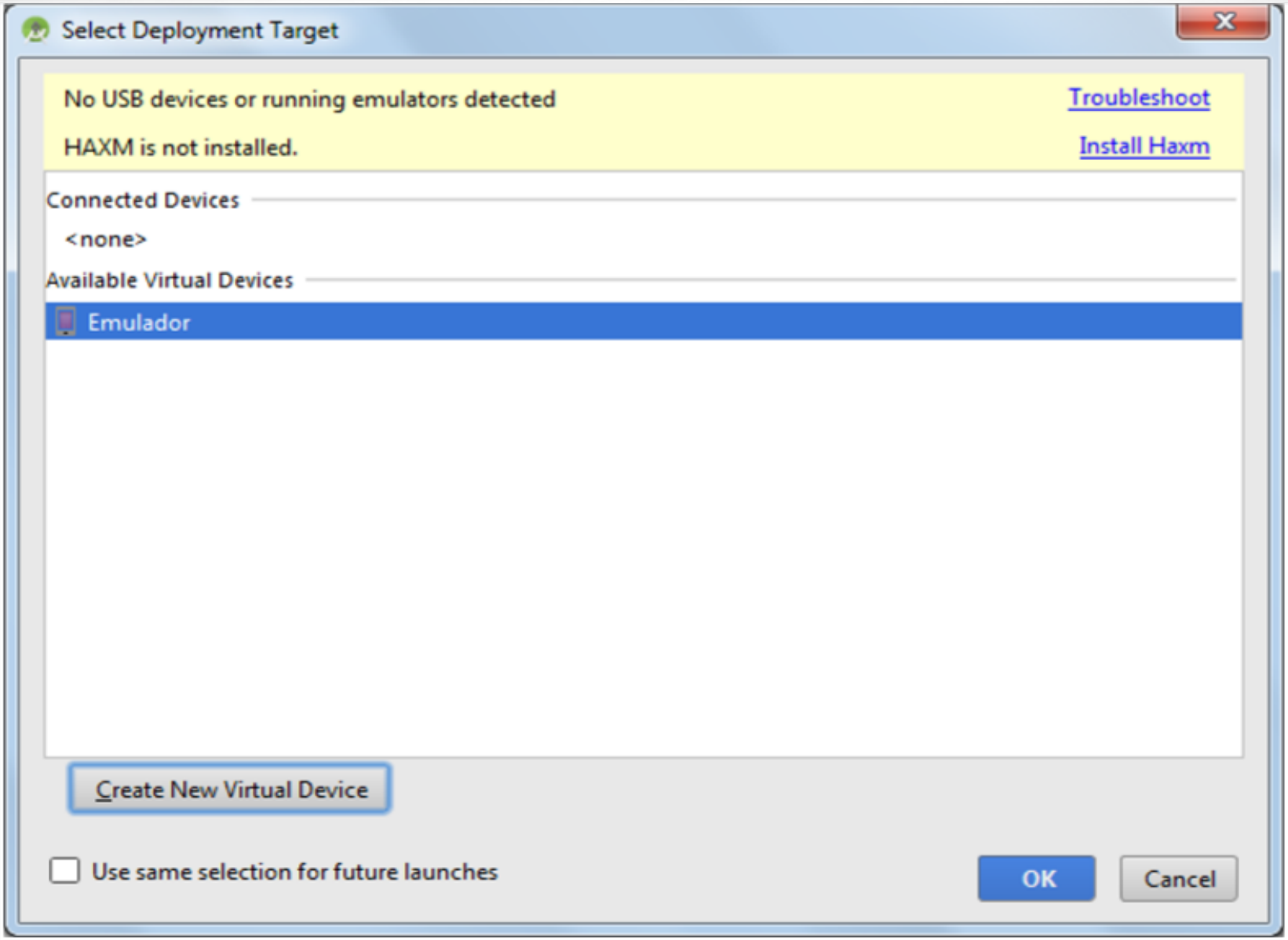
- AVD Name: ***Emulador*** e ***Finish***:



Executando a nossa aplicação

- No campo “**ADV Name**” vamos especificar o nome do nosso dispositivo virtual, que se chamará “**Emulador**”.
- Logo, vamos digitar **Emulador**. As demais configurações vamos deixá-las como está (**default**). Feito isso, vamos clicar no botão “**Finish**” para que o nosso dispositivo virtual possa ser gerado.
- Vejamos a figura no próximo slide de como ficou o dispositivo criado para a execução da nossa aplicação:

Executando a nossa aplicação



Dispositivo virtual criado

Executando a nossa aplicação

- Vamos selecionar agora o dispositivo criado (chamado de "**Emulador**") e em seguida clique em "**OK**". Feito isso nosso emulador irá executar, conforme mostra a figura seguinte :

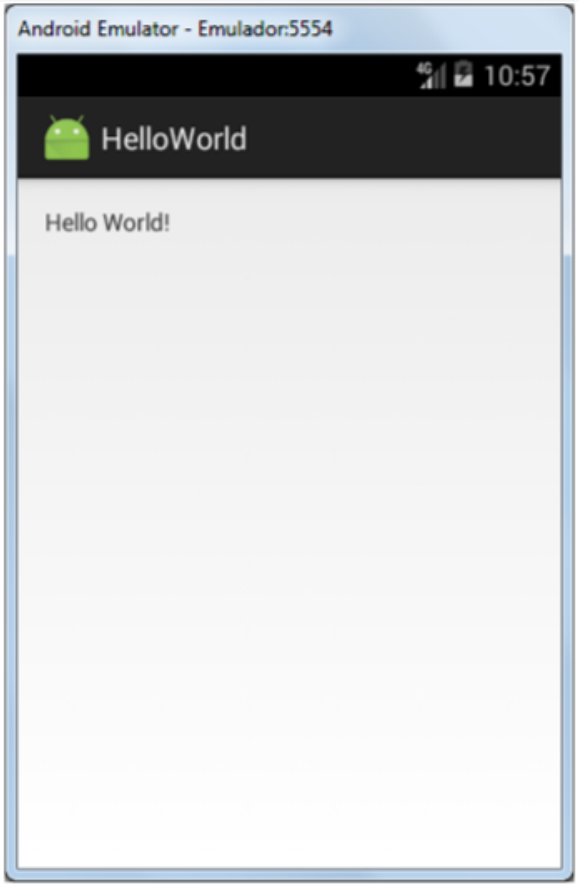
Emulador do Android em Execução

Executando a nossa aplicação

- No início da execução do emulador mostra o título Android, conforme você vê na figura no slide anterior. Depois vem um outro título escrito “Android”, um pouco maior e cinza , com uma animação. Esse processo normalmente demora em torno de menos de **1 minuto a 10 minutos** (dependendo da sua máquina).
- É recomendável que você tenha no mínimo 2 GB de memória e um processador bem rápido para um bom desempenho da execução para a aplicação ser exibida, mesmo sendo essa aplicação algo muito simples.

Executando a nossa aplicação

- Passado o tempo citado no slide anterior, será mostrada a nossa aplicação em execução, conforme podemos ver na figura seguinte:



Emulador do Android em Execução

Executando a nossa aplicação

- Esse emulador já vem com uma série de recursos como Navegador, Aplicações de demonstração, Mapas, Lista de contatos e etc.

- Se você neste exato momento **fechou** o **emulador** após a execução da aplicação, vou te dizer uma coisa: "Não era para você ter feito isso". Se você esperou muito tempo para ver essa aplicação em execução, ao executar novamente a aplicação, possivelmente você vai esperar o mesmo tempo.

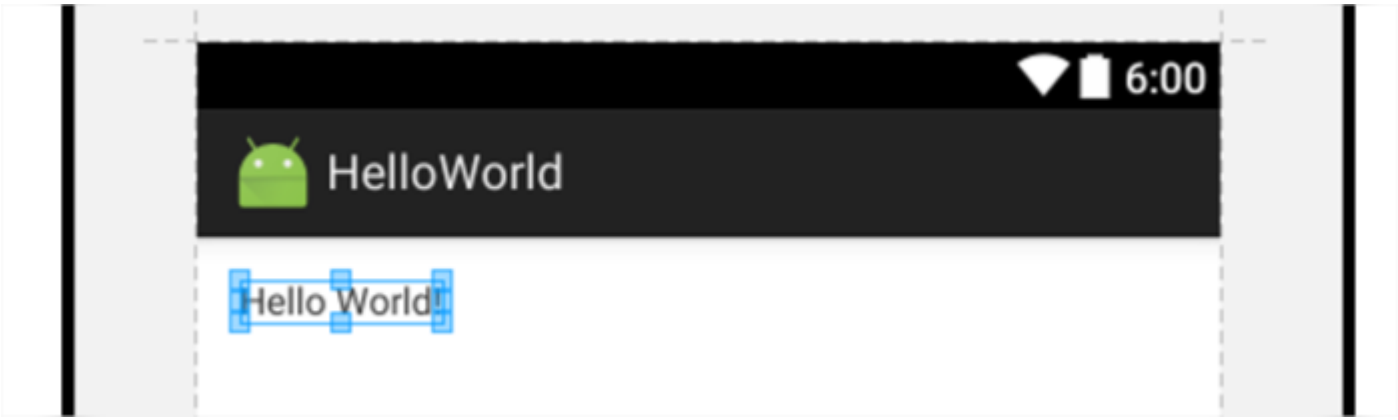
Executando a nossa aplicação

- Ao executar pela primeira vez o emulador, e caso vá executar outros programas, minimize o emulador ao invés de fechar, pois se você esperou muito tempo para executar esse programa, com ele minimizado, ao executar outro programa, o Android Studio vai fazer uso do emulador já aberto em vez de abrir outro, com isso a aplicação levará em torno de 9 a 12 segundos em média para ser executada.

- Nunca esqueçam disso!!!!!!

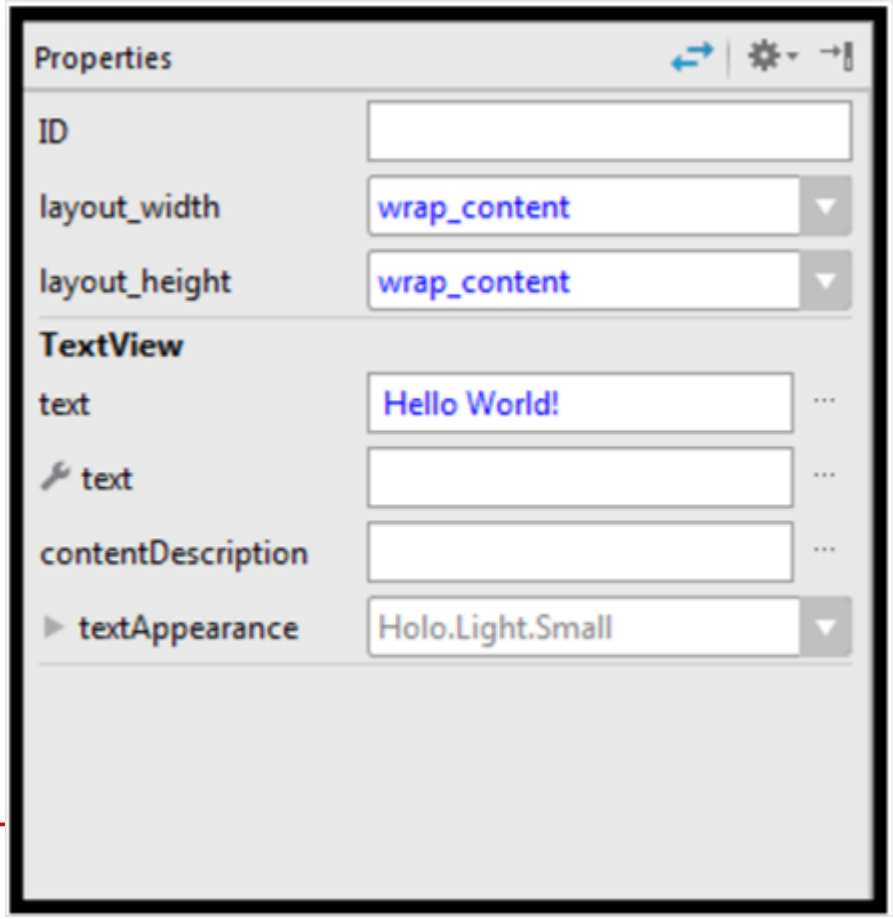
Executando a nossa aplicação

- Agora vamos fazer algumas modificações em nossa aplicação.
  - Minimize o emulador e vamos no projeto.
  - Para começar vamos mudar o texto que está sendo exibido na tela.
  - Selecione o texto (componente **TextView**), conforme demonstra a figura seguinte:



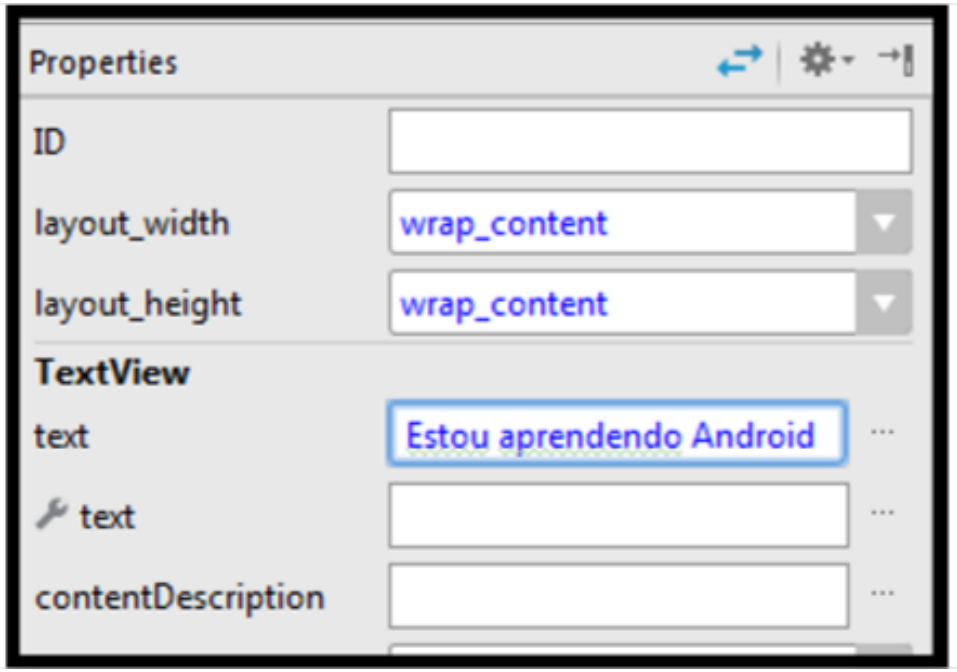
Executando a nossa aplicação

- Se observarmos no lado direito, temos as propriedades do componente selecionado, conforme demonstra a figura seguinte:



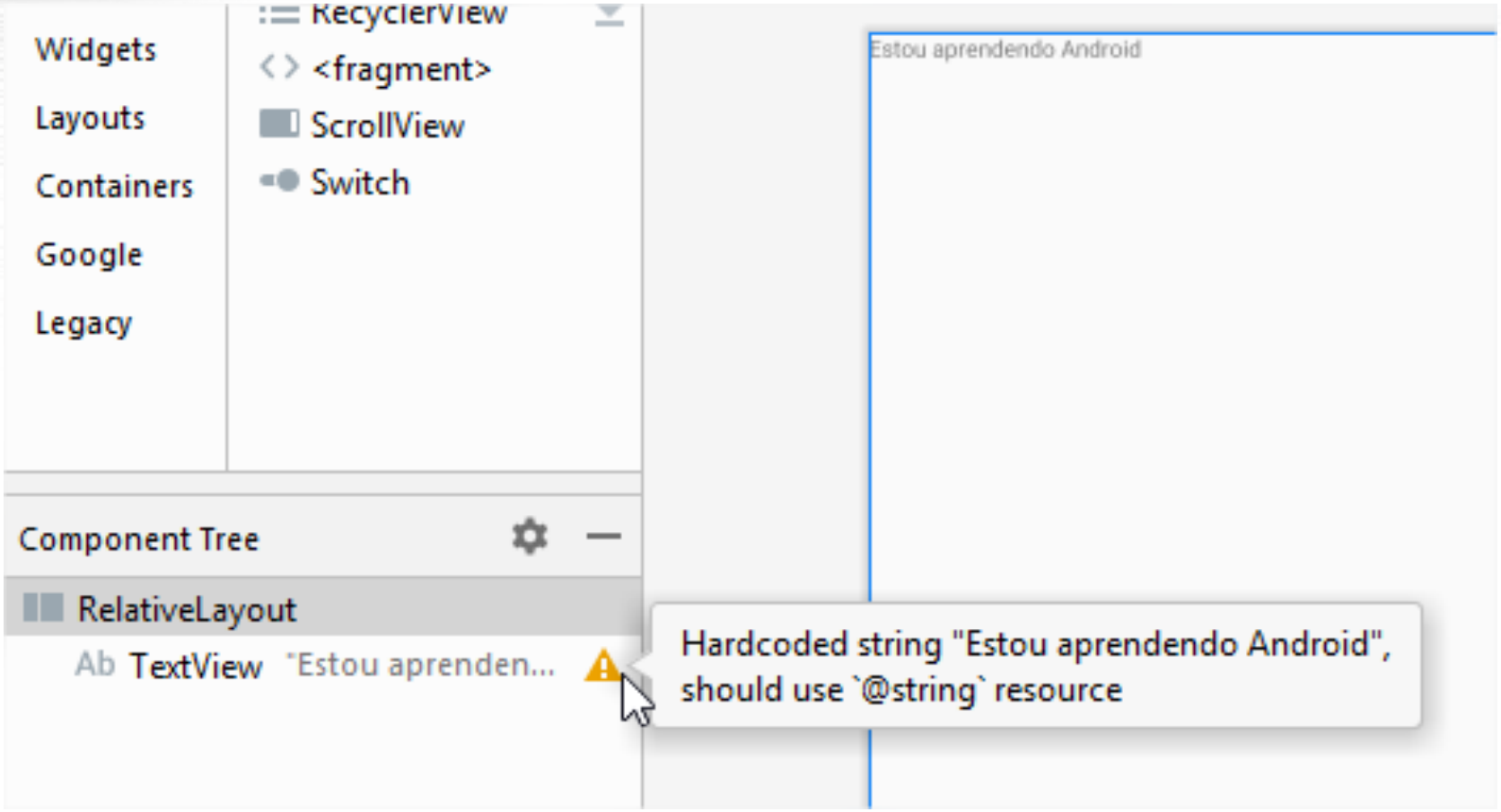
Executando a nossa aplicação

- Para alterarmos o conteúdo do texto, devemos alterar a propriedade "**text**" (conforme você pode ver na figura no slide anterior).
- Vamos digitar agora a seguinte frase: "**Estou aprendendo Android**". Vejamos como ficou na figura abaixo:



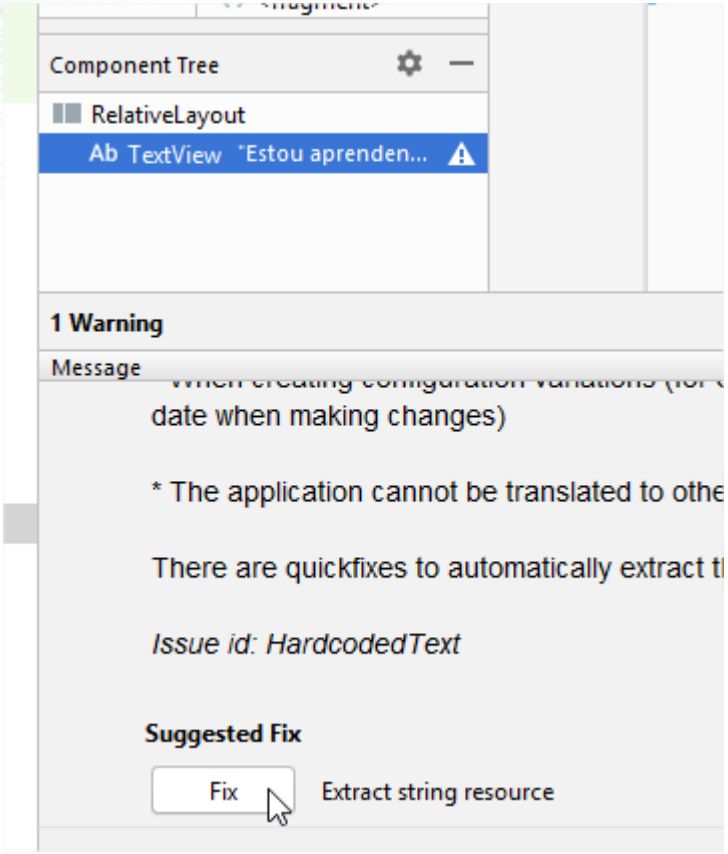
Executando a nossa aplicação

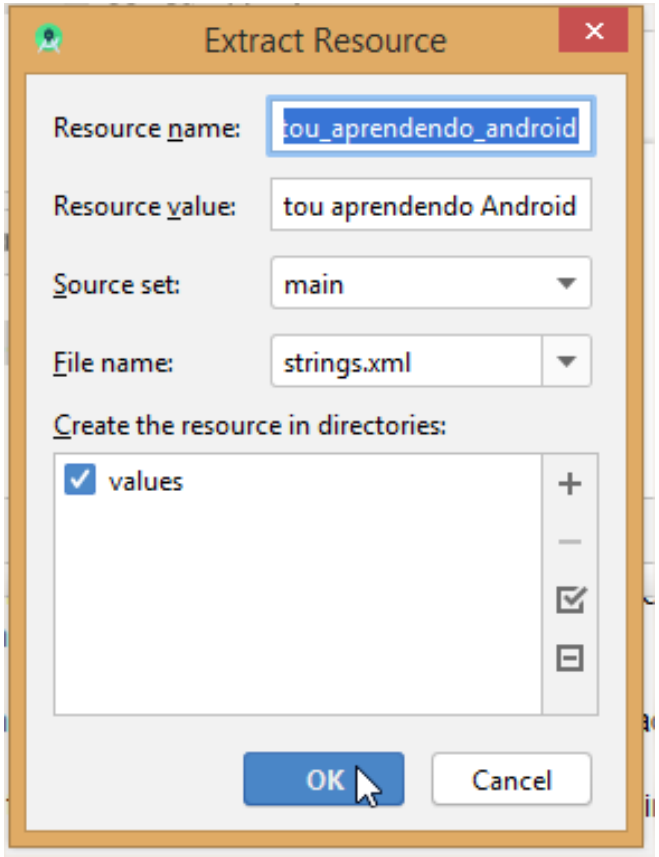
- Quando um texto é alterado, é solicitado a inclusão dele no arquivo @string (strings.xml)



Executando a nossa aplicação

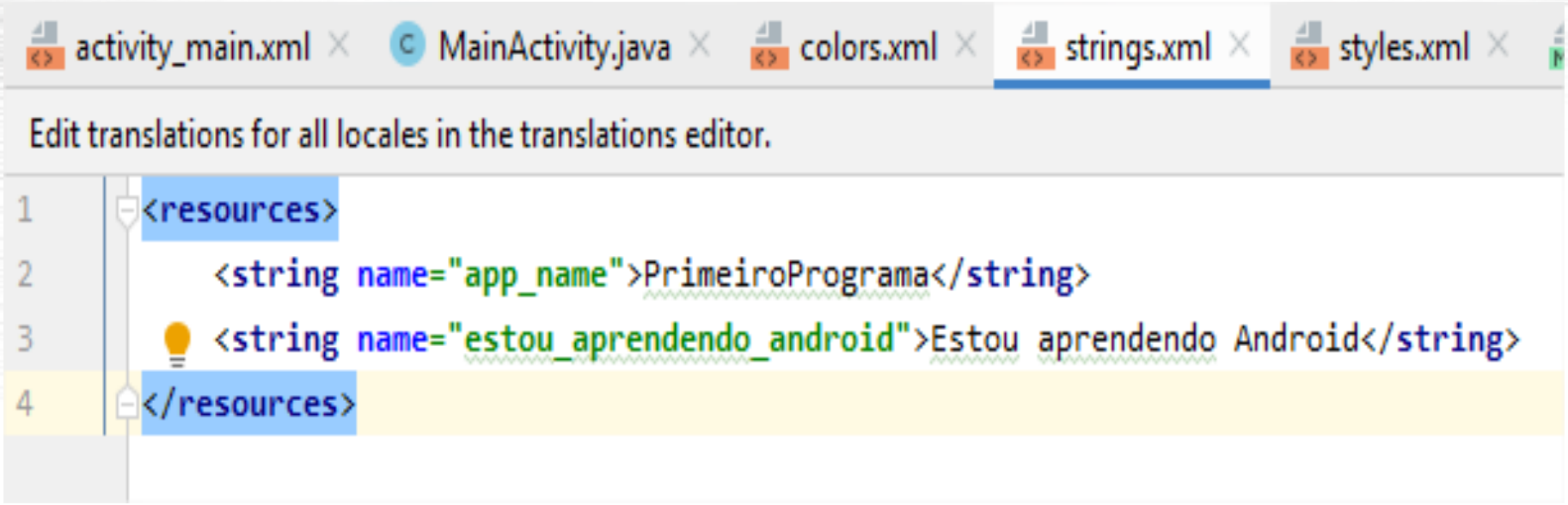
- Quando um texto é alterado, é solicitado a inclusão dele no arquivo @string (strings.xml)





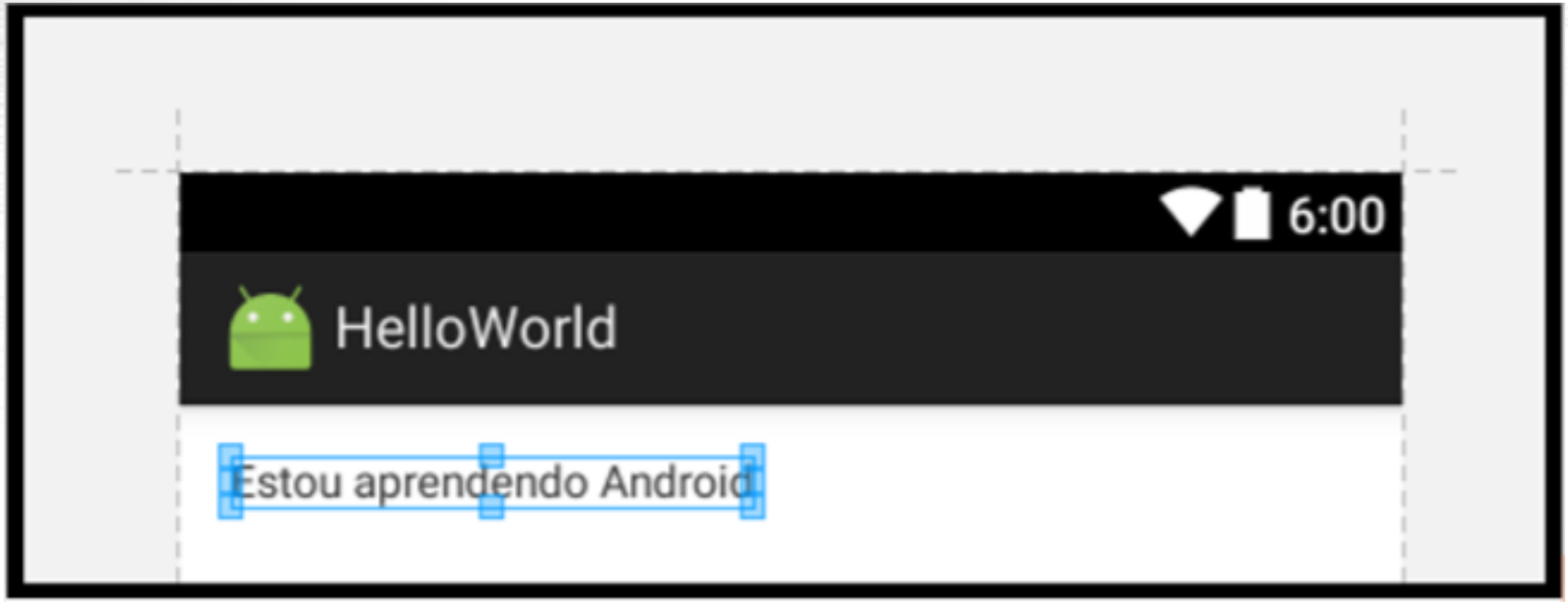
Executando a nossa aplicação

- Arquivo strings.xml atualizado:



Executando a nossa aplicação

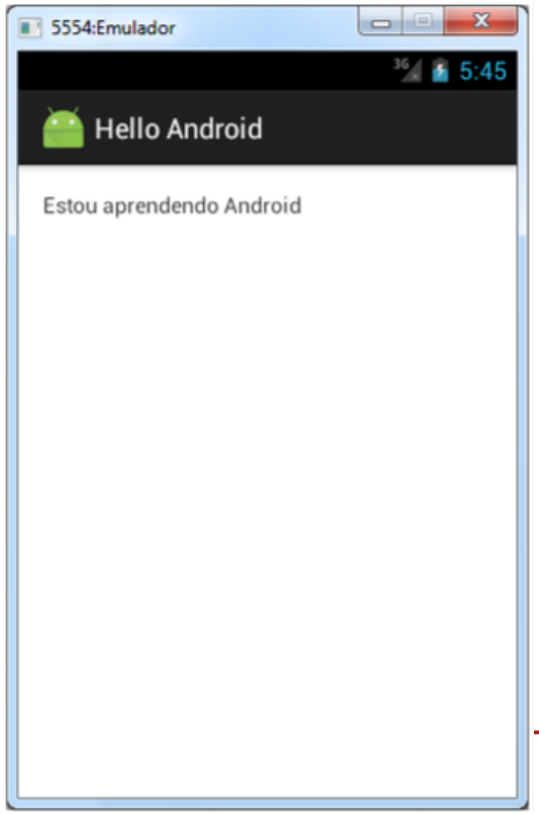
- O resultado você confere em seguida :



Conteúdo modificado

Executando a nossa aplicação

- Depois de alterar o conteúdo da **TextView** vamos executar novamente a aplicação (conforme já foi mostrado). O resultado podemos conferir na figura seguinte:



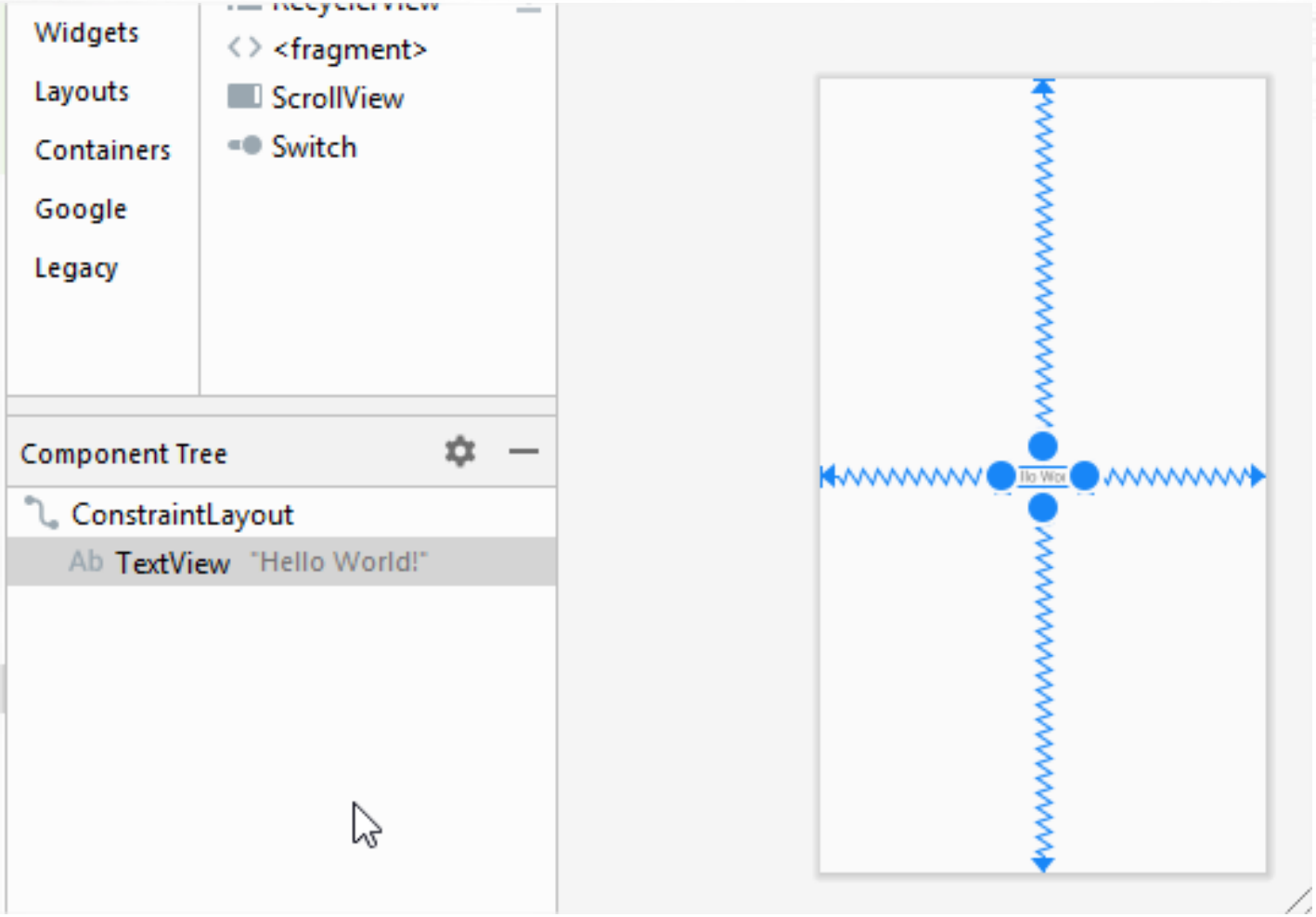
Aplicação em execução

Executando a nossa aplicação

- Toda **tela de uma aplicação** é uma estrutura de **layout** que permite organizar e distribuir os componentes nela inseridos conforme a nossa necessidade.
- A estrutura de **layout** dessa versão do **Android SDK**, que e inserida por padrão toda vez que criamos este um projeto em Android, é a "**ConstraintLayout**", mas pode ser convertida para "**RelativeLayout**". Essa estrutura permite que os componentes nela inseridos possam ser organizados e distribuídos de forma "**relativa**" a outros componentes presentes também na tela.

Executando a nossa aplicação

- Veja abaixo a Estrutura do **ConstraintLayout**:



Executando a nossa aplicação

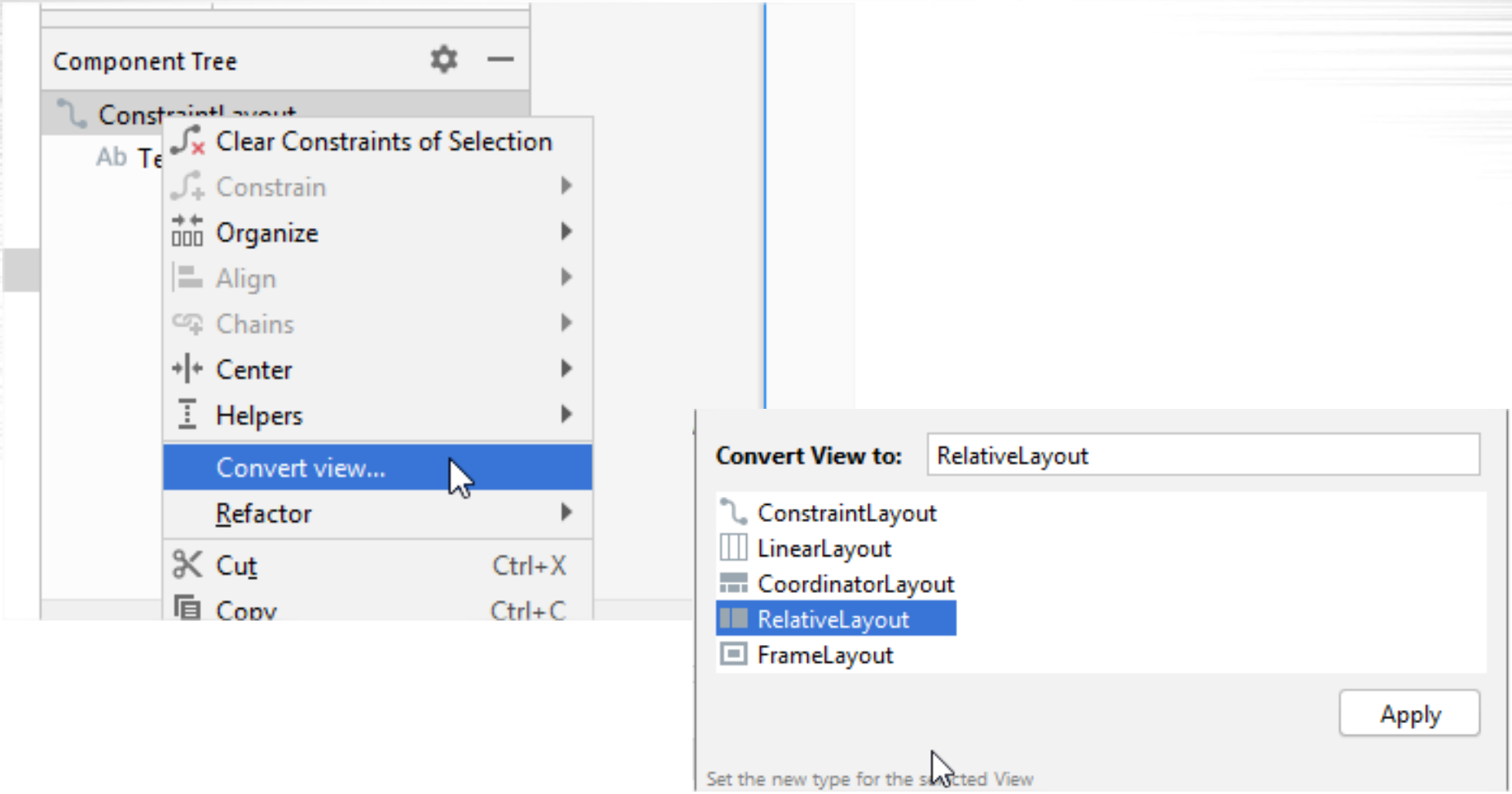
- XML da Estrutura do **ConstraintLayout**:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    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="Hello World!"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent" />

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

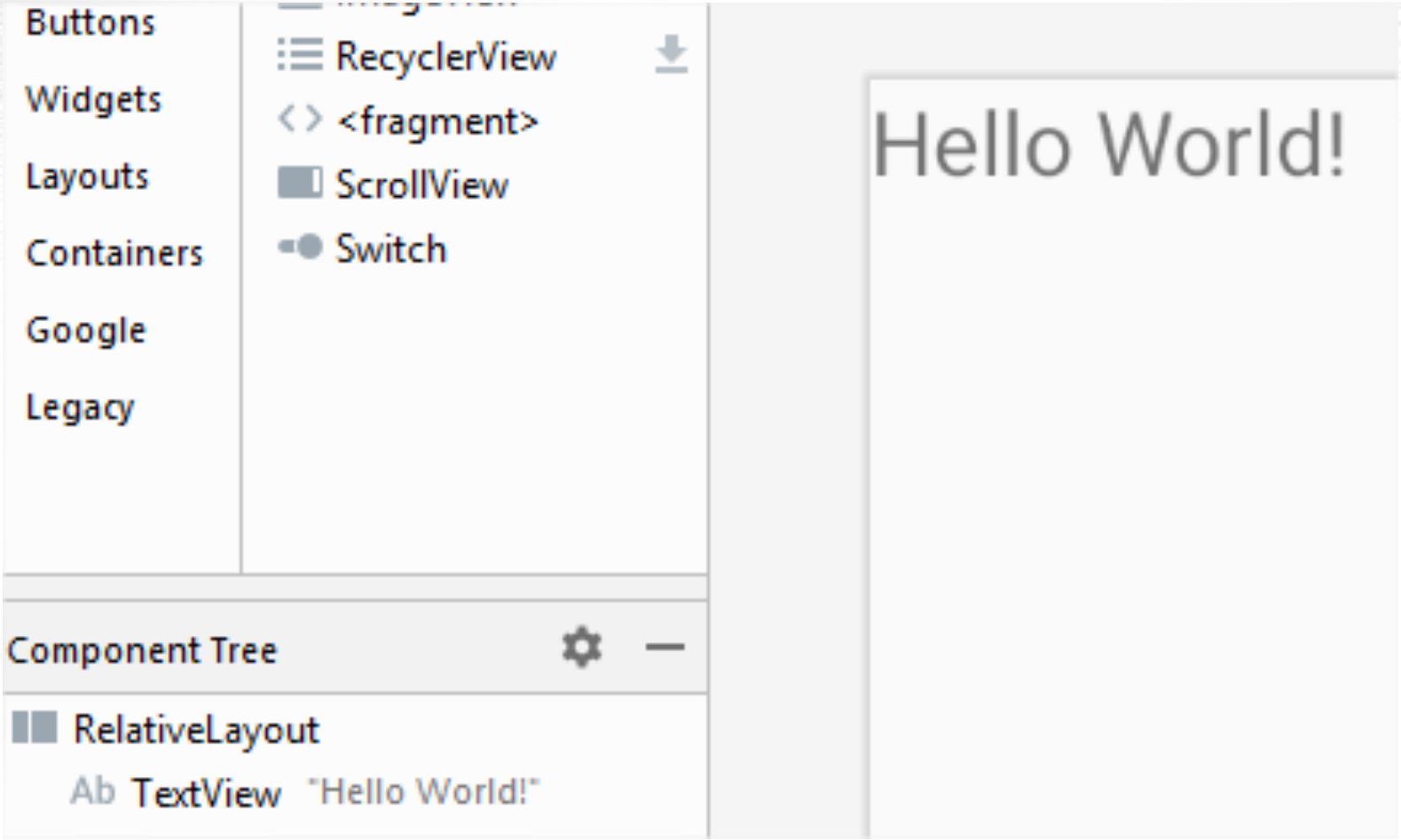
Executando a nossa aplicação

- Convertendo para o **RelativeLayout**:



Executando a nossa aplicação

- Veja abaixo a Estrutura do **RelativeLayout**:



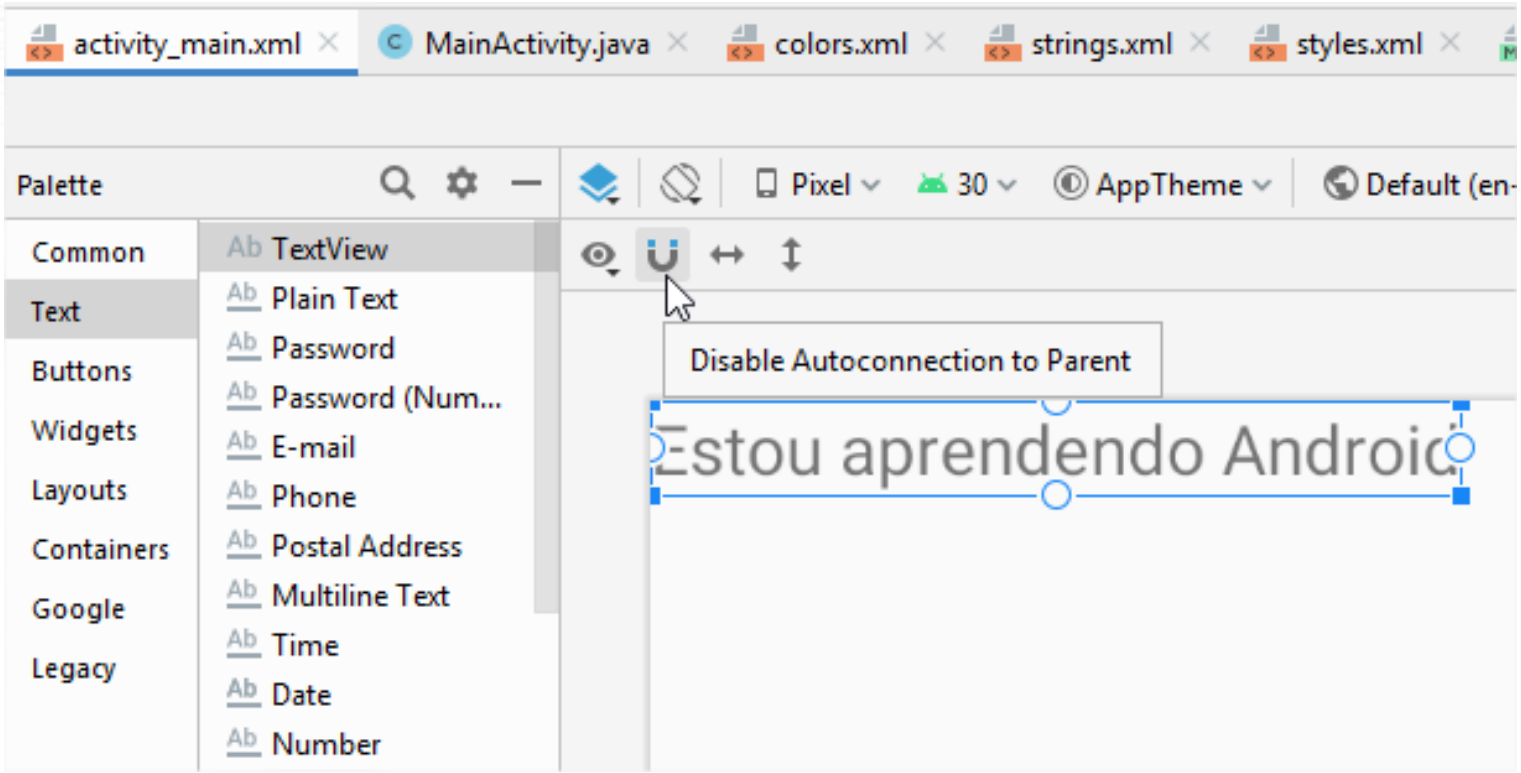
Executando a nossa aplicação

- XML da Estrutura do **RelativeLayout**:

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       tools:context=".MainActivity">
8
9       <TextView
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content"
12          android:text="Hello World!" />
13
14  </RelativeLayout>
```

Executando a nossa aplicação

- Vamos agora antes de **arrastar e soltar** componentes na tela, clicar no "**U**" para ***Disable Autoconnection to Parent***, conforme abaixo:



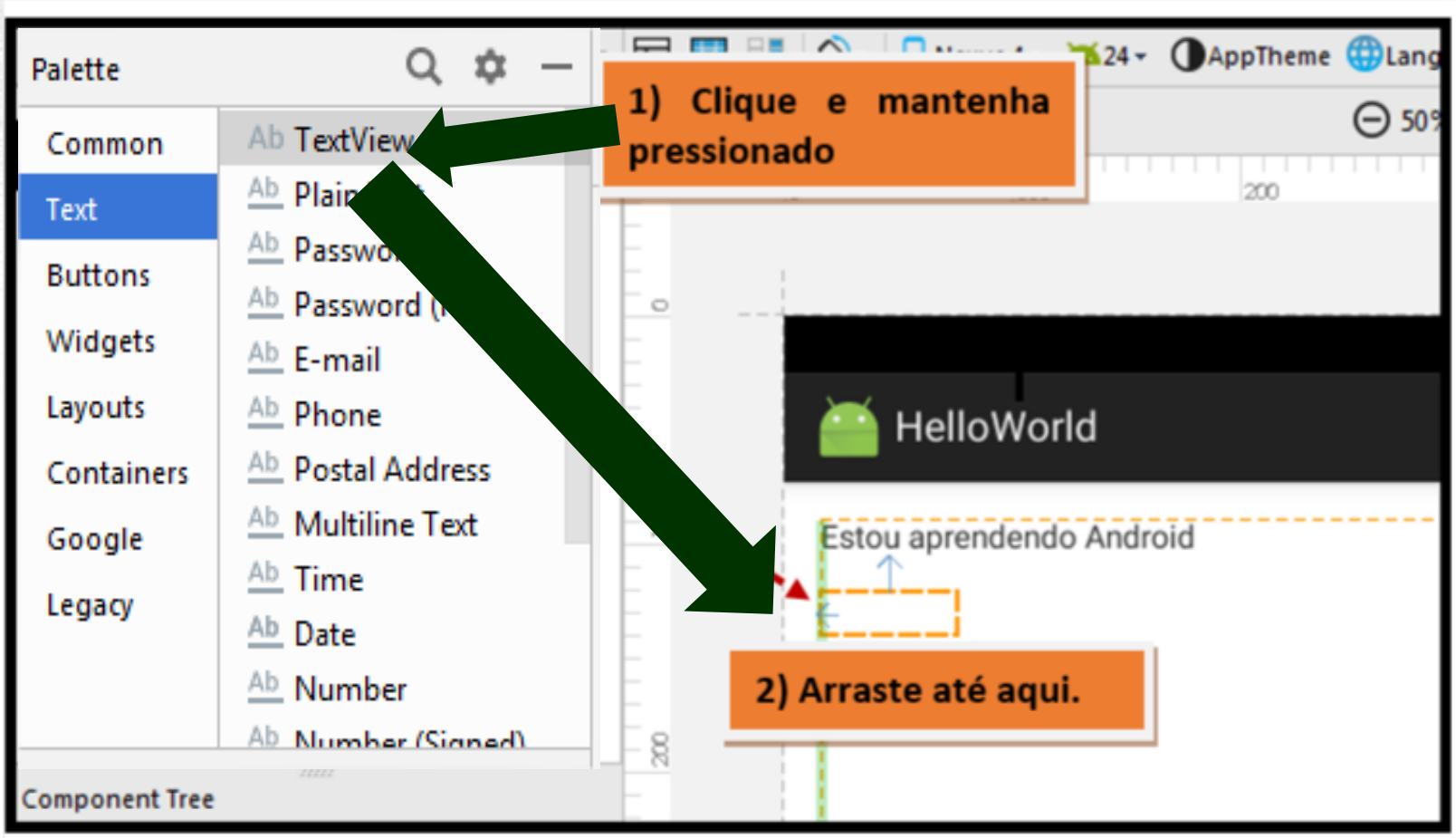
Executando a nossa aplicação

- Vamos agora usando a paleta de componentes **arrastar e soltar** na tela do dispositivo o componente **TextView** (que se encontra na seção "**Common**" ou "**Text**", com o título "**TextView**").

- Acompanhe o processo no próximo slide.

Executando a nossa aplicação

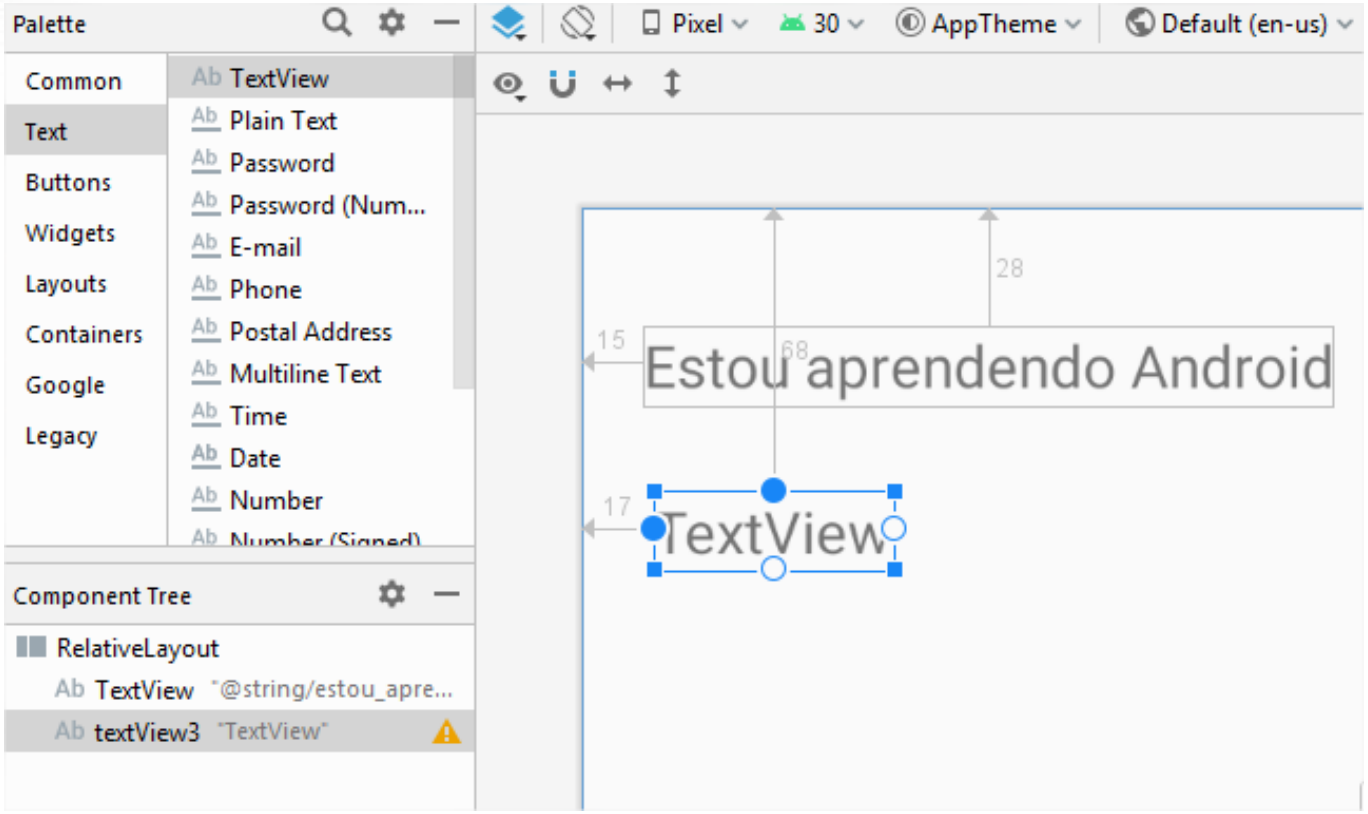
- Adicionando o componente “**TextView**”:



Adicionando o componente

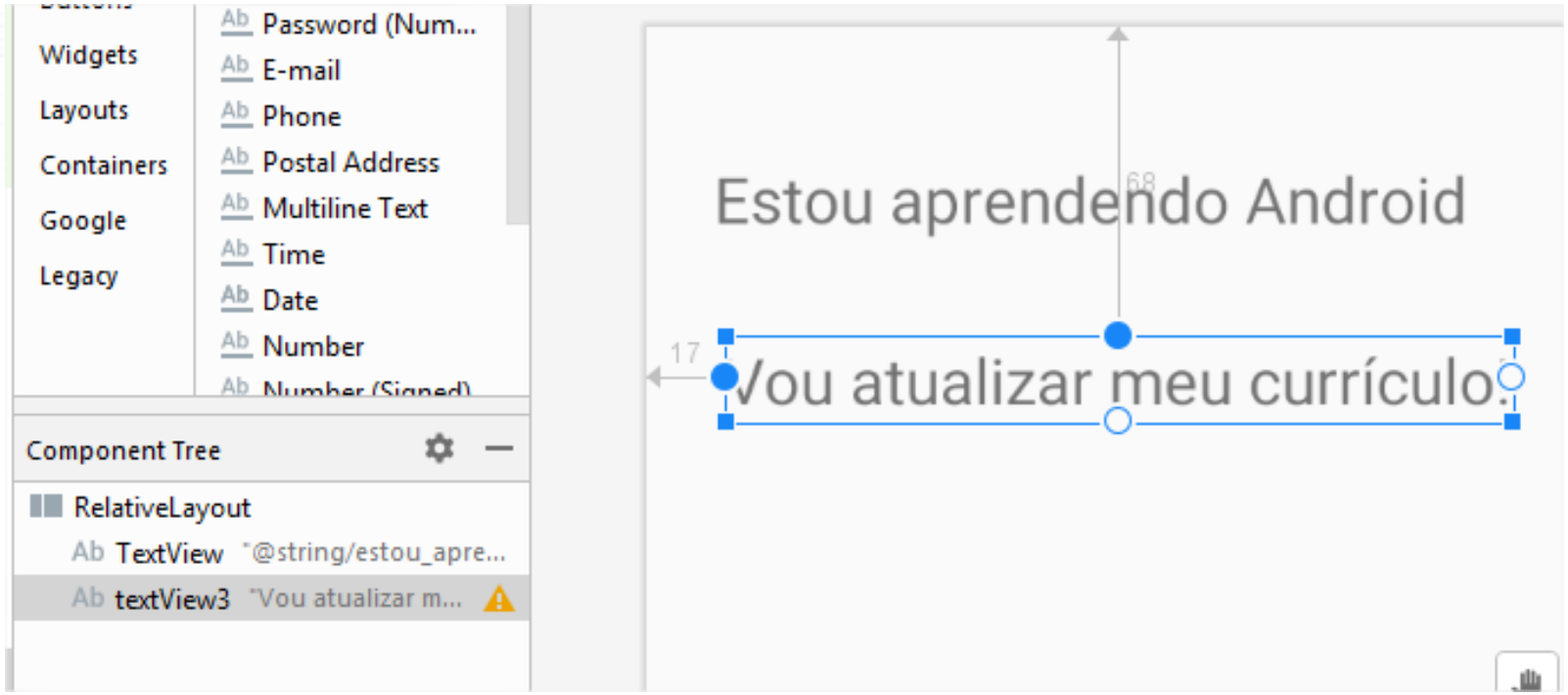
Executando a nossa aplicação

- Seguindo os passos solicitados teremos o seguinte resultado mostrado na figura em seguida:



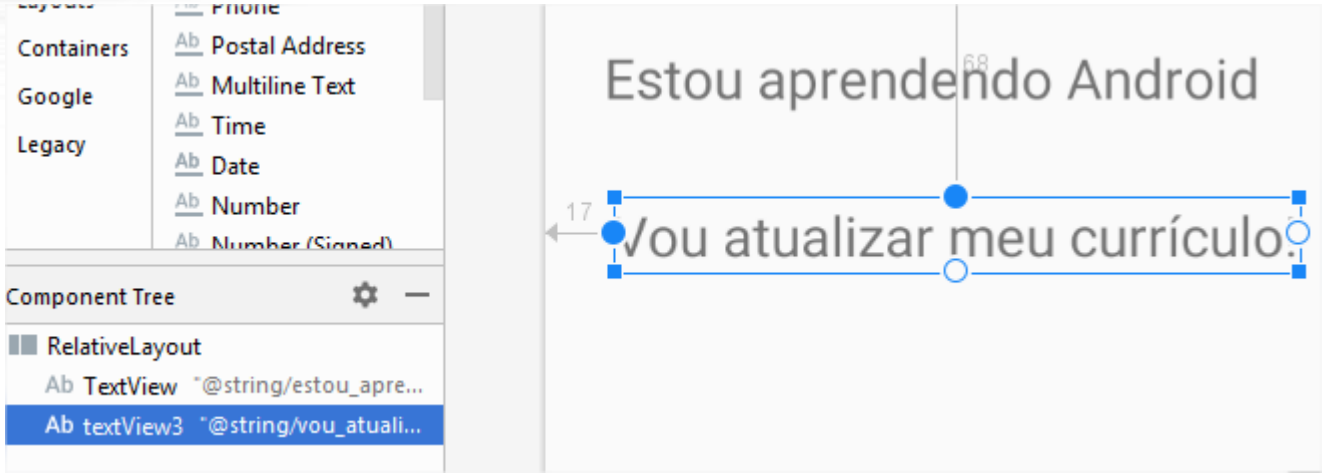
Executando a nossa aplicação

- Vamos agora digitar na propriedade “**text**” do segundo componente **TextView** (que inserimos agora) a seguinte frase “**Vou atualizar meu currículo!**”. Veja o resultado na figura seguinte:



Executando a nossa aplicação

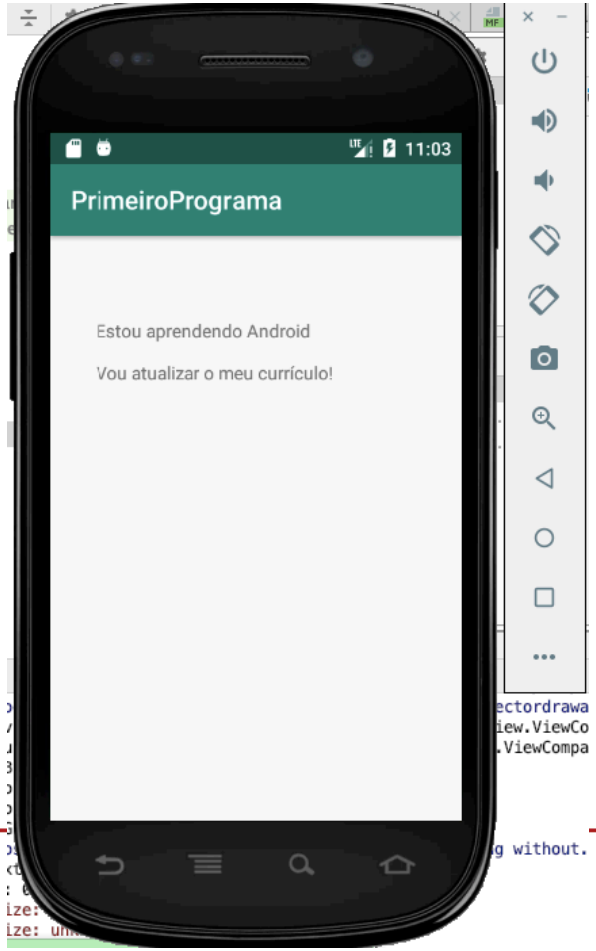
- Vamos também incluir este novo texto no arquivo **strings.xml**, conforme exemplo anterior.




```
1  <resources>
2      <string name="app_name">PrimeiroPrograma</string>
3      <string name="estou_aprendendo_android">Estou aprendendo Android</string>
4      <string name="vou_atualizar_meu_curr_culo">Vou atualizar meu currículo!</string>
5  </resources>
```

Executando a nossa aplicação

- Vamos salvar as alterações feitas no arquivo e em seguida executar a aplicação para conferir o resultado, como demonstra a figura seguinte:



Executando a nossa aplicação

- Conforme já havia mencionado, toda estrutura que constitui a tela de uma aplicação em Android nada mais é do que um ***código XML***.
- Para visualizarmos o código XML basta clicar na guia “**Text**”, situado ao lado da guia “**Design**”.

- Veja o seu código XML no próximo slide:

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginStart="15dp"
        android:layout_marginTop="28dp"
        android:text="@string/estou_aprendendo_android" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginStart="17dp"
        android:layout_marginTop="68dp"
        android:text="@string/vou_atualizar_meu_curr_culo" />
</RelativeLayout>
```

Executando a nossa aplicação

- Tudo certo? Estão entendo aos poucos como se faz aplicações **Android** ? Com certeza que **sim**!
- Como podemos ver no **Android Studio** ele já oferece um utilitário que permite a criação de aplicações de forma rápida, simplesmente arrastando e soltando os componentes. Isso acelera o processo de desenvolvimento de aplicações, mas nada impede que vocês codifiquem suas aplicações via "**xml**", pois os códigos ficam mais simples e claros para as devidas manutenções.

Exercício

- Vamos alterar nossa aplicação incluindo mais textos variados e alterando as propriedades dos textos e verificando os resultados.

- Bom trabalho!!!!

Sobre o Projeto Final Mobile

- Enviar por e-mail (joao.pimentel@projecao.br) a ideia do seu APP.

- Assunto: Projeto LTP Mobile - Matutino / Noturno

- Além de enviar os nomes das pessoas do grupo (máximo 2) ou poderá ser individual, é para enviar também uma breve descrição sobre o referido APP a ser desenvolvido durante o semestre.

Por hoje é só!!!!

Até a próxima aula...