



---

# android

Prof. Diogo Soares

# Android

- Sistema Operacional e Plataforma de Desenvolvimento para Dispositivos Móveis com maior taxa de crescimento
  
- FOSS (*Free Open Source Software*)
  - Usa o kernel do Linux
  - Usa diversas bibliotecas livres
    - *SQLite, LibC, WebKit*
    - *OpenSSL, OpenGL*
    - *zLib, FreeType*



# Desenvolvimento

## ■ Aplicativos Android podem ser desenvolvidos nas linguagens:

### □ Java

- *Recomendada para quem está aprendendo a programar*
- *Além de ser usado pelo Android, é usado em milhares de outros projetos*
- *Comunidade imensa de programadores. Maior documentação e oportunidades*

### □ Kotlin

- *Linguagem oficial do Android, desde 2017*
- *Possui uma sintaxe mais limpa que o Java, dentre outras vantagens*
- *Fácil de aprender, principalmente se você já conhece Java*
- *Compilado para o mesmo bytecode do Java, executado pela máquina virtual java (JVM)*
- *Usado basicamente apenas pelo Android*

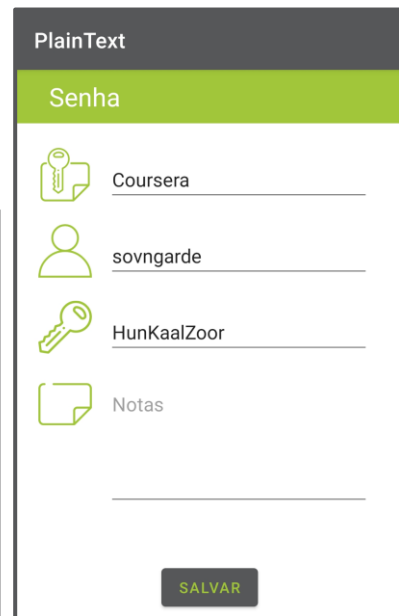
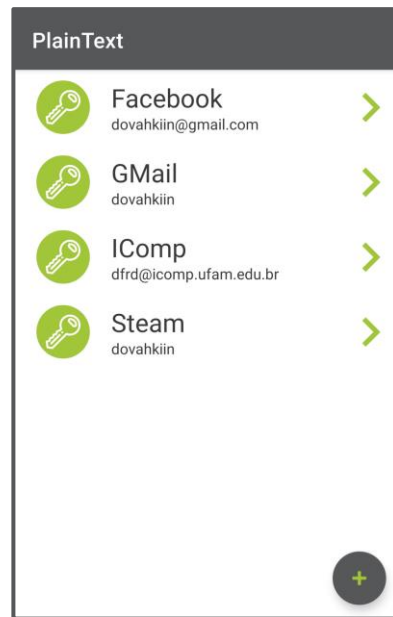
### □ C++

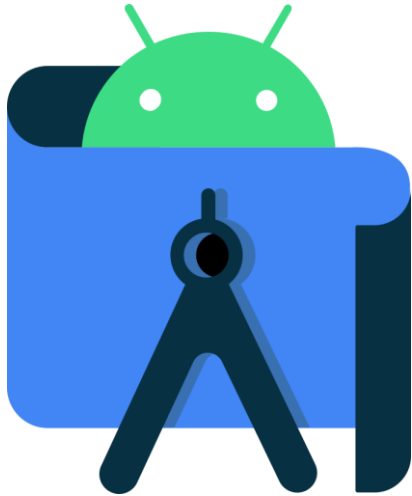
- *Código é executado diretamente pelo processador do dispositivo*
- *Permite utilizar bibliotecas existentes desenvolvidas em C/C++*

# Sumário

- Android Studio
  - Instalação
  - Hello World
- Interface Gráfica
- Activities e Intents
  - Preferências
  - Listas
- Banco de Dados

## ■ Aplicativo de Exemplo





# android studio

Prof. Diogo Soares

# Android Studio



- Modo recomendado de desenvolvimento desde 12/2014
  - Substitui Eclipse ADT
  
- Contêm:
  - IntelliJ IDE
  - Android SDK Tools
  - Android Platform-tools
  - Imagem do Sistema Android para o Emulator

# Download

---

▣ <https://developer.android.com/studio/>



Android Studio provides the fastest tools for building apps on every type of Android device.

**Download Android Studio**

Android Studio Chipmunk | 2021.2.1 Patch 2 for Linux 64-bit (964 MiB)

# Instalação (Linux)

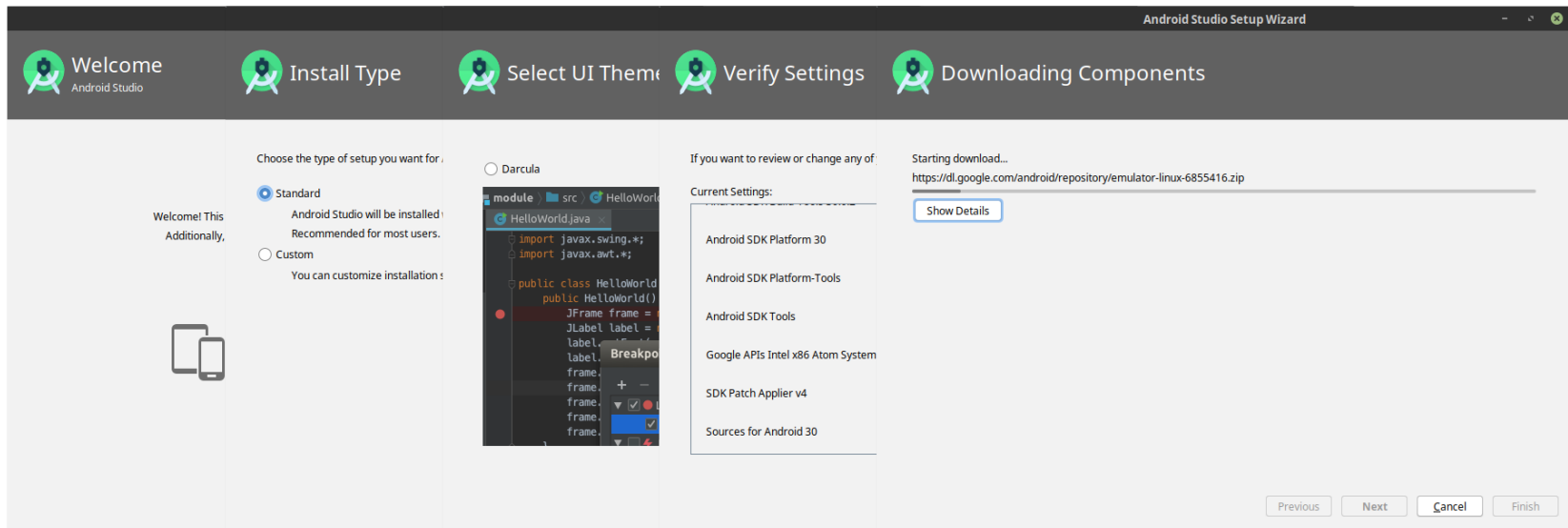
```
$ tar -xvf android-studio-xxxx.x.x.xx-linux.tar.gz
```

```
$ cd android-studio/bin/
```

```
$ ./studio.sh
```

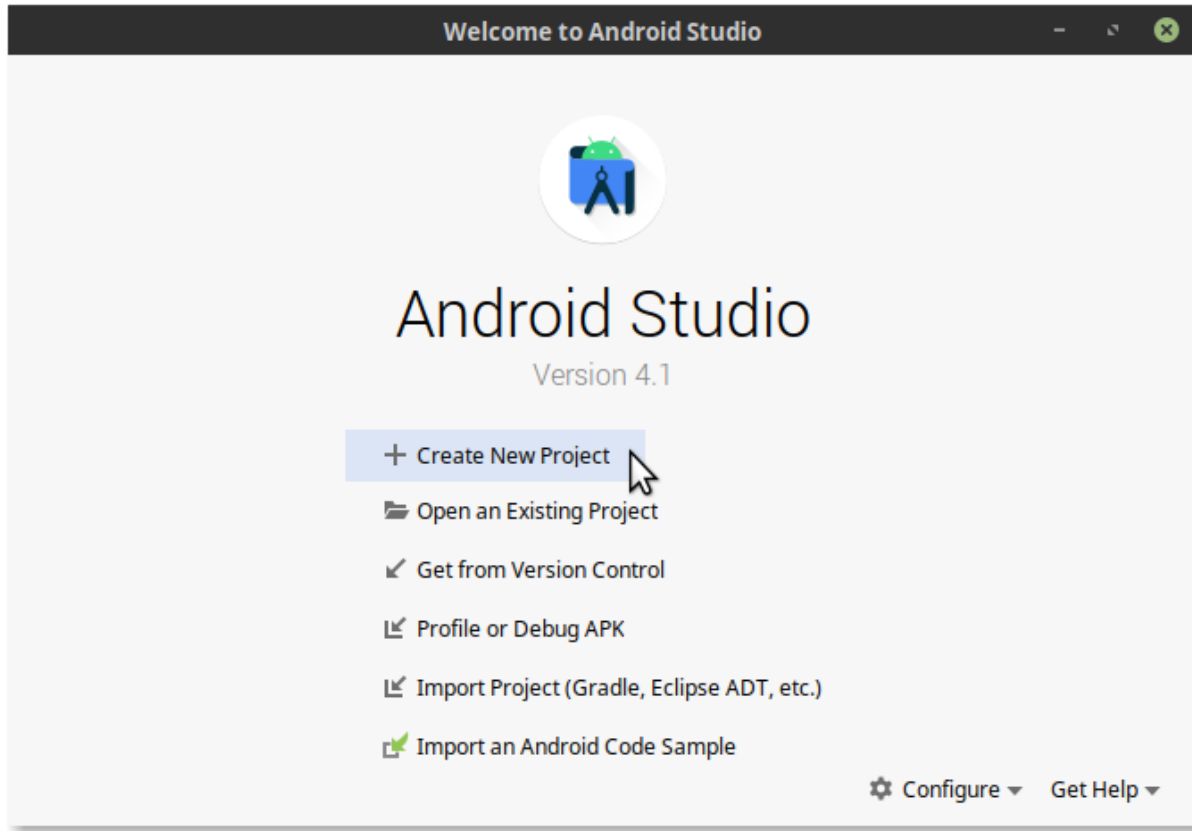
→ Na primeira execução, serão baixados e instalados componentes

→ No total, serão baixados aproximadamente 1.6GB

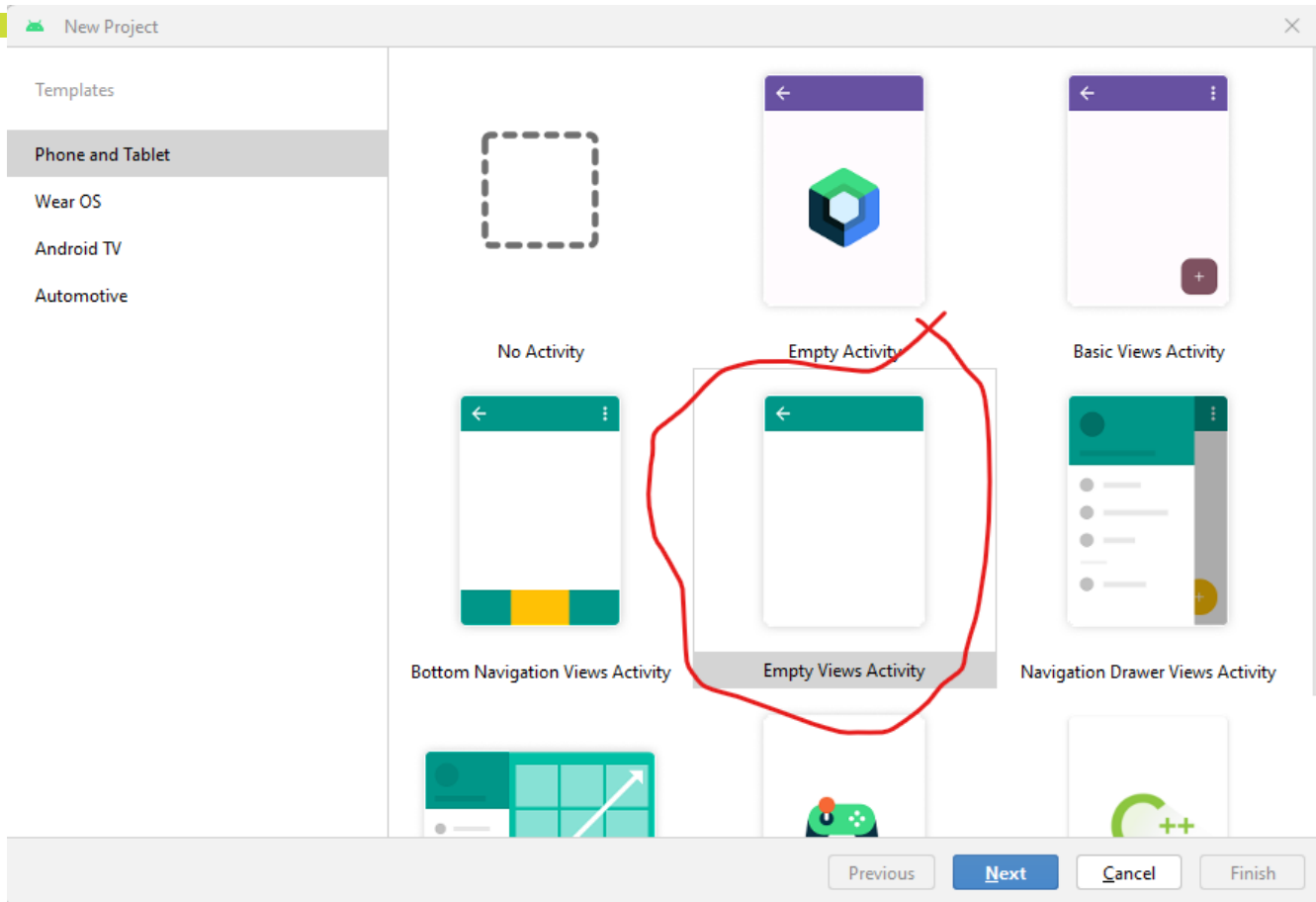




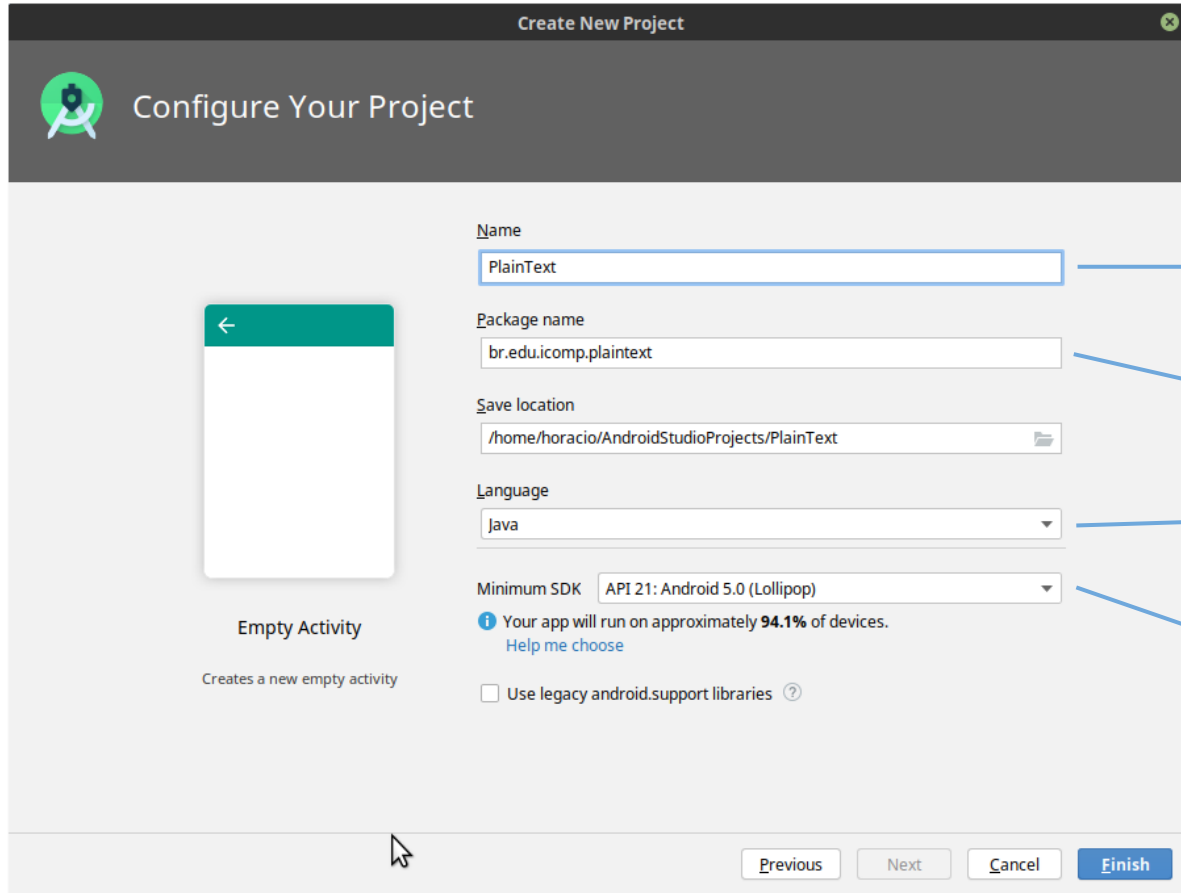
# Tela Inicial, Novo App



# Novo App



# Novo App



**Create New Project**

**Configure Your Project**

**Name**  
PlainText

**Package name**  
br.edu.icomp.plaintext

**Save location**  
/home/horacio/AndroidStudioProjects/PlainText

**Language**  
Java

**Minimum SDK** API 21: Android 5.0 (Lollipop)

**Empty Activity**  
Creates a new empty activity

**Use legacy android.support libraries** ☐

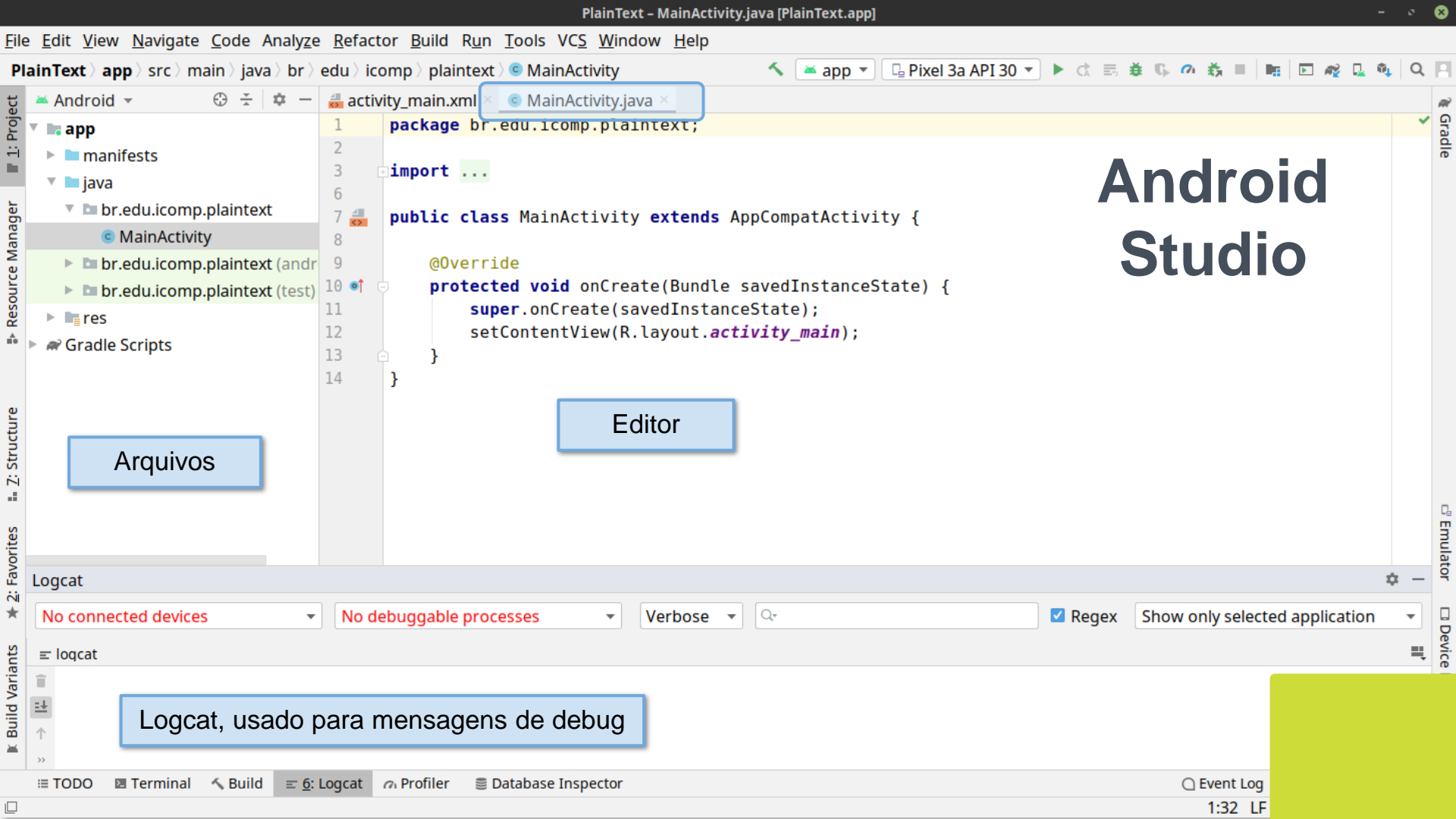
**Finish**

Nome do App

Pacote

Selecione Java

Versão mínima

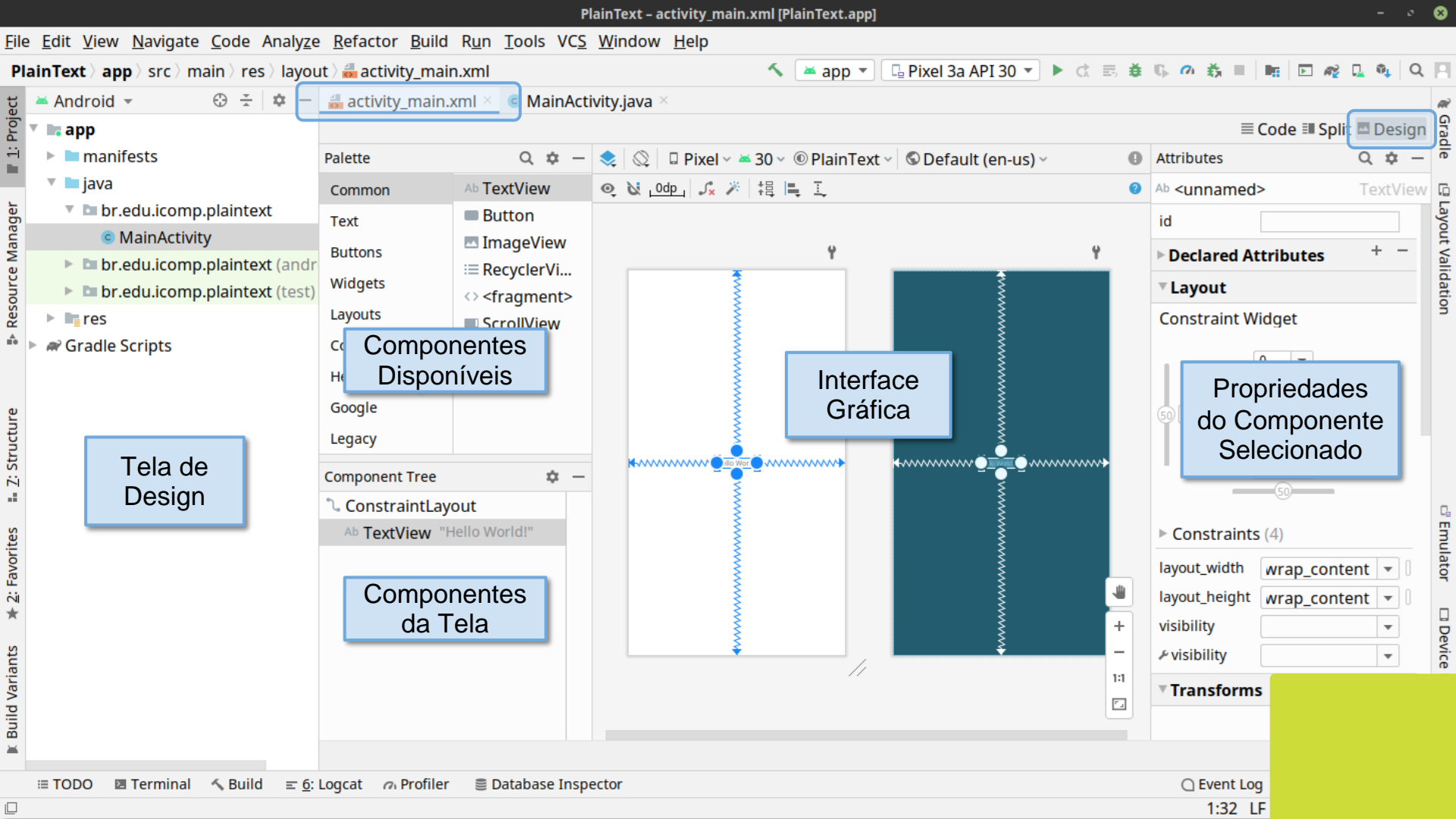


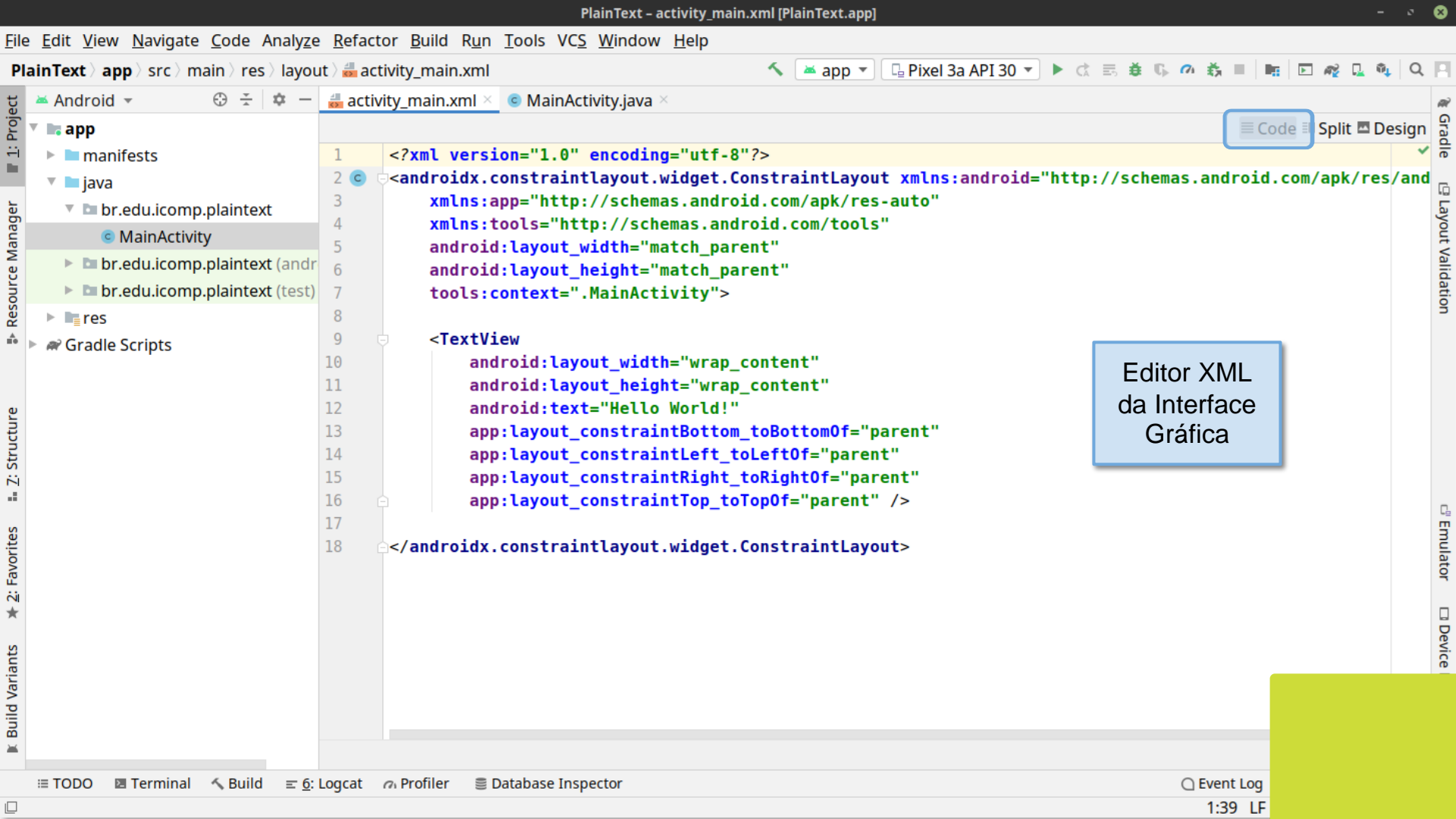
Android  
Studio

Editor

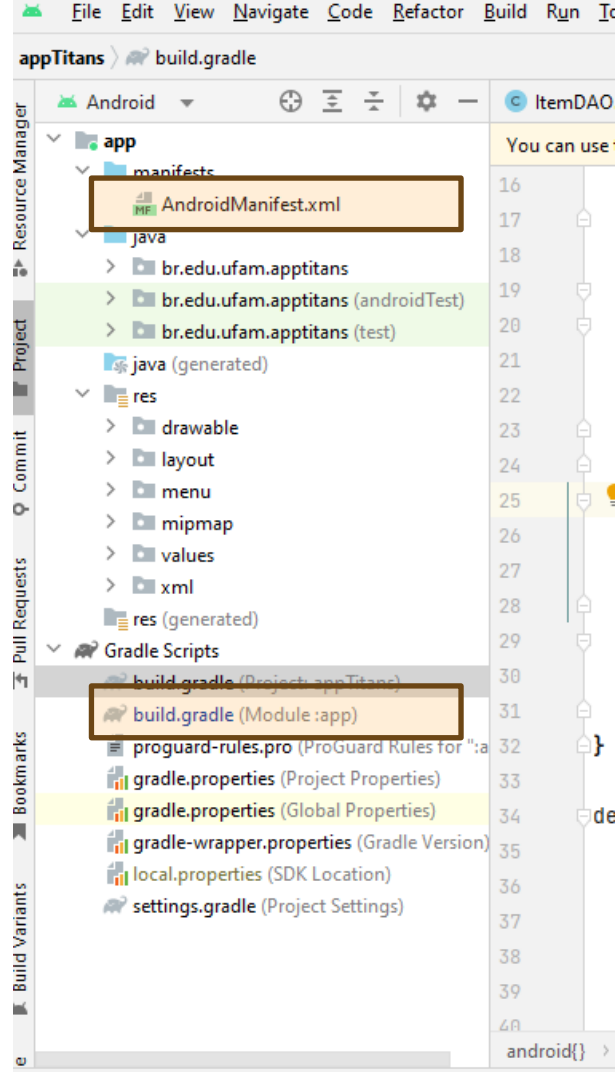
Arquivos

Logcat, usado para mensagens de debug





## ■ Outros itens essenciais

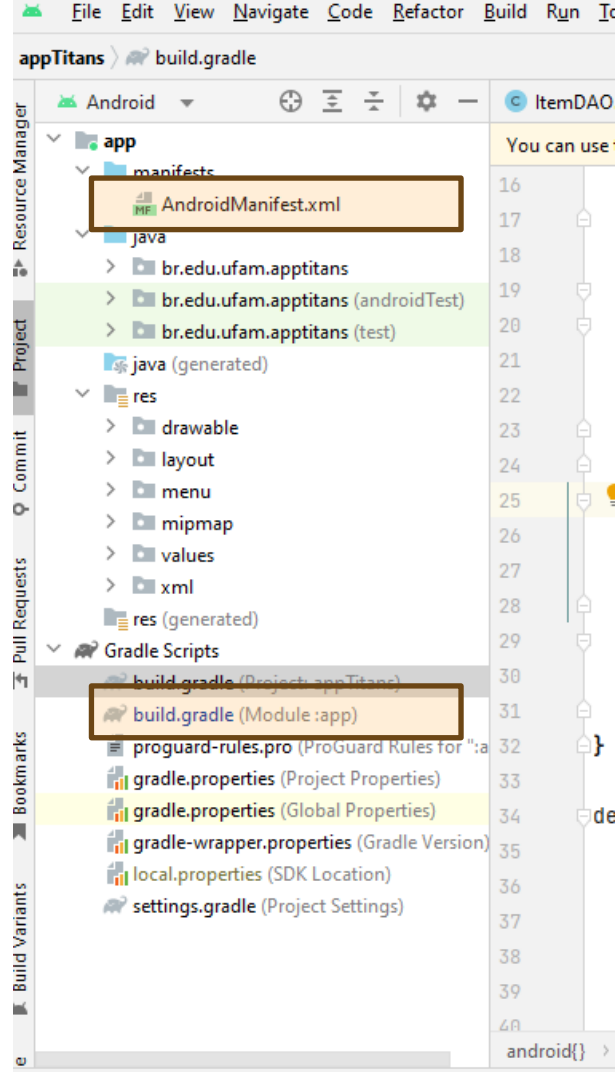


### AndroidManifest

Componente crucial para declarar as activities, services, permissões do app (como wifi, bluetooth, etc) e até requisitos de hardware/software.

Se um conteúdo não é listado no manifesto, não será visto pelo sistema e não executará

## ■ Outros itens essenciais



### Build.gradle

Arquivo de configuração usado no desenvolvimento. É usado para definir o build, dependências e bibliotecas adicionais do app.

Isso garante que seu app executará sob dadas condições de compilação/dependências

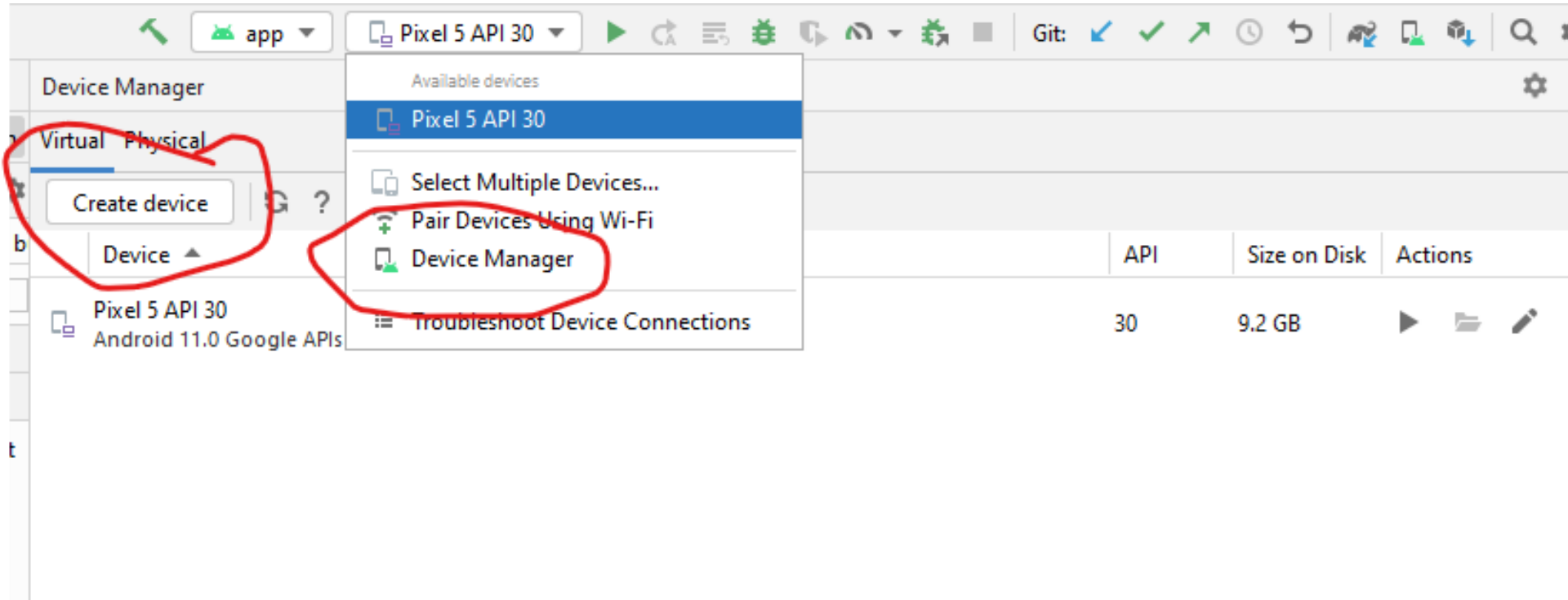


# AVD - Android Virtual Device

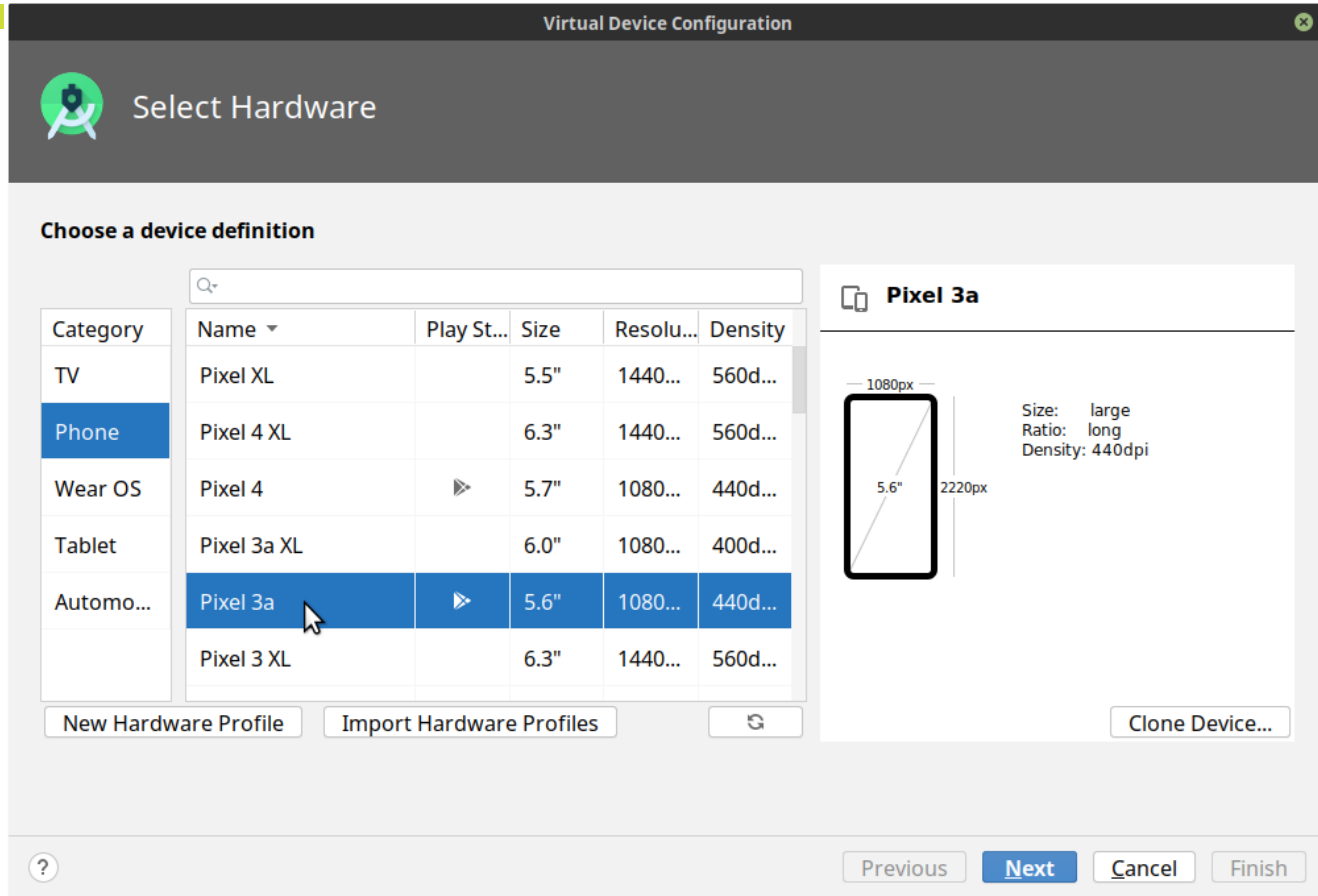


- Emulador que permite executar os aplicativos em sua máquina
  - Ele emula um celular e executa uma versão completa do Android
  - O mesmo do seu celular, mas sem os aplicativos da operadora e do fabricante
  - Permite testar o aplicativo em celulares de configurações e Android diferentes
  
- Para abrir o gerenciador de AVDs:
  - Tools → Device Manager
    - *Talvez seja necessário criar um AVD novo (próximos slides)*

# AVD - Android Virtual Device



# AVD - Android Virtual Device



# AVD - Android Virtual Device

Clica no Download

Depois do download, Next

Virtual Device Configuration

System Image

Select a system image

Recommended x86 Images Other Images

Release Name	API Level	ABI	Target
R Download	30	x86	Android 11.0 (Google Play)
Q Download	29	x86	Android 10.0 (Google Play)
Pie Download	28	x86	Android 9.0 (Google Play)
Oreo Download	27	x86	Android 8.1 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat Download	25	x86	Android 7.1.1 (Google Play)
Nougat Download	24	x86	Android 7.0 (Google Play)

A system image must be selected to continue.

Android 11.0 (Google Inc.)

API Level 30


System Image x86

We recommend these Google Play images because this device is compatible with Google Play.

Previous Next Cancel Finish

# AVD - Android Virtual Device


Virtual Device Configuration

 Android Virtual Device (AVD)

Verify Configuration


AVD Name

Pixel 3a API 30

 Pixel 3a

5.6 1080x2220 xxhdpi


Change...

 R


Android 11.0 x86

Change...

Startup orientation



Portrait



Landscape

Emulated Performance

Graphics: Automatic

Show Advanced Settings

AVD Name

The name of this AVD.

Clica no Finish

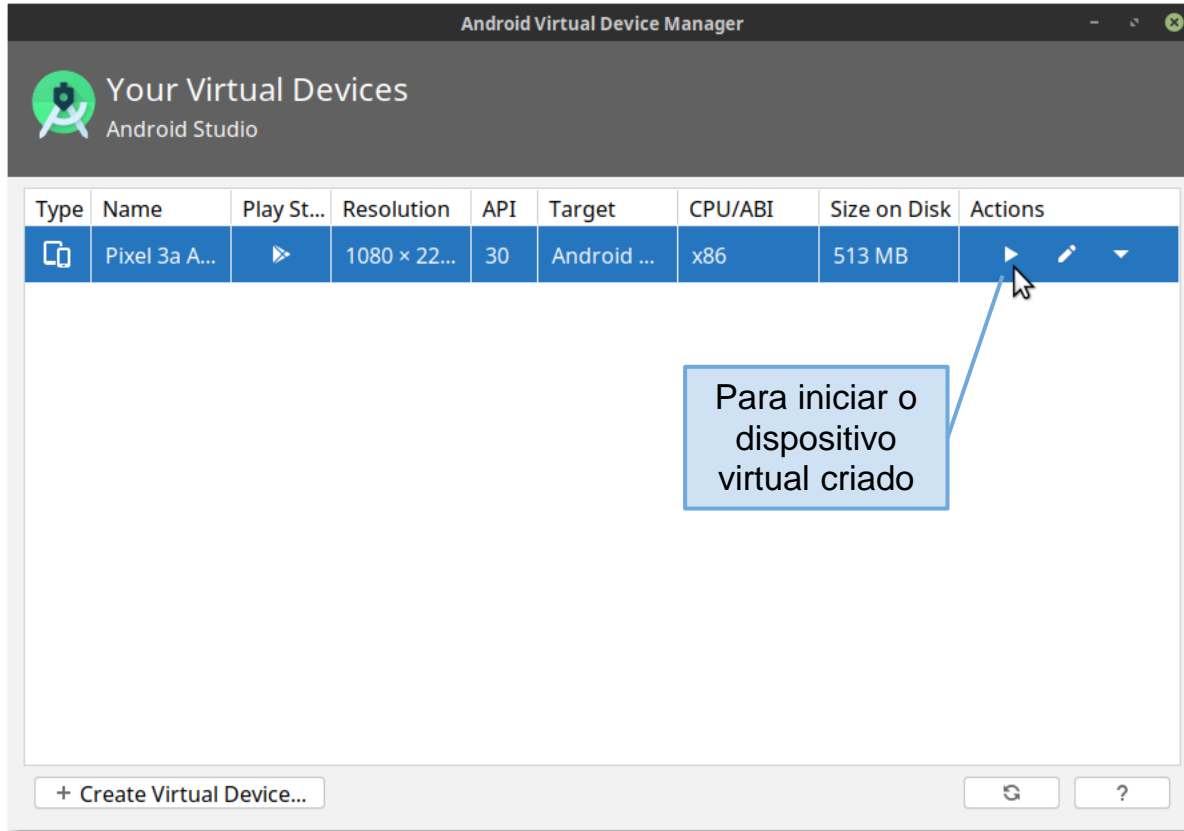
Previous

Next

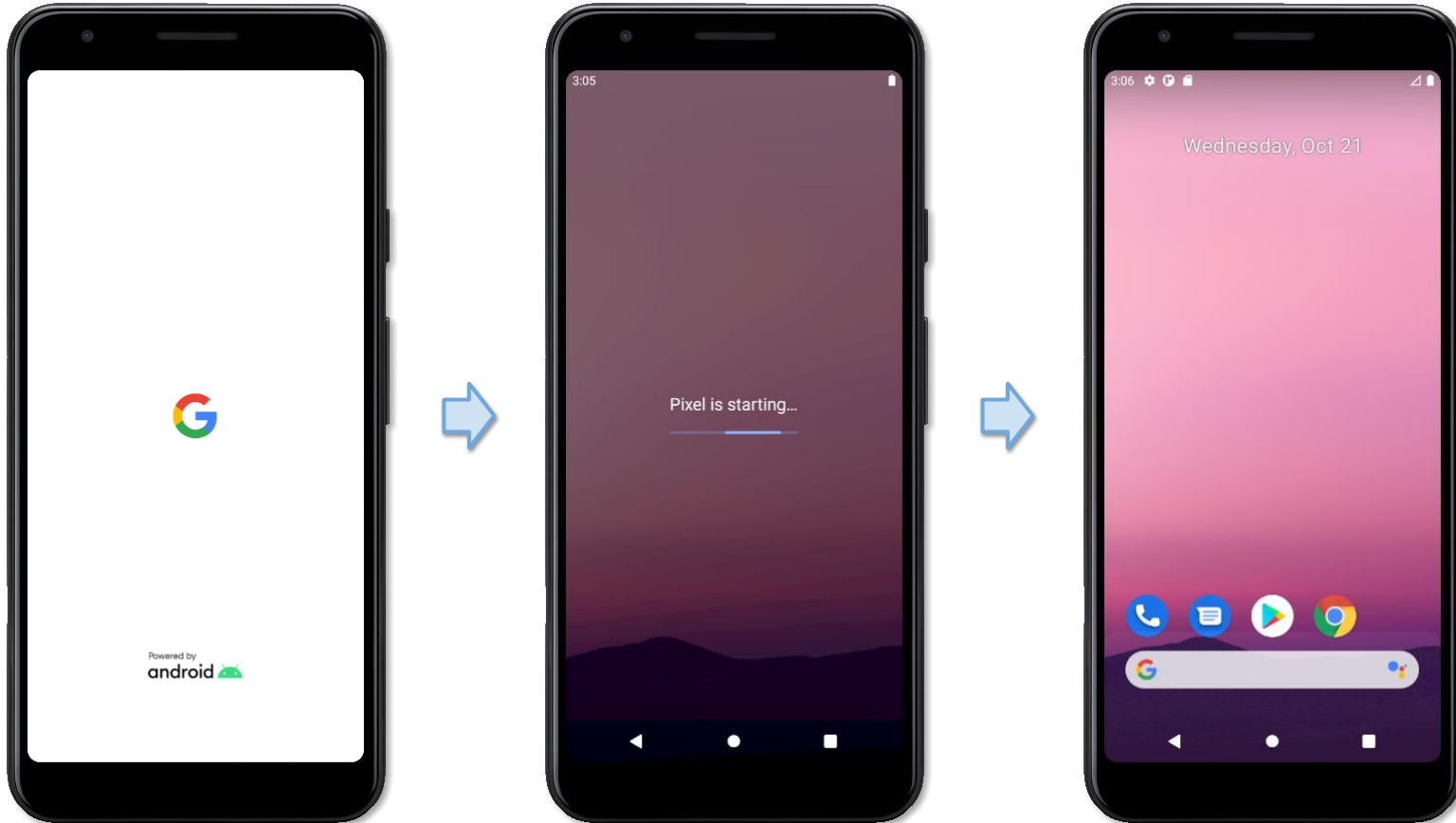
Cancel

Finish

# AVD - Android Virtual Device

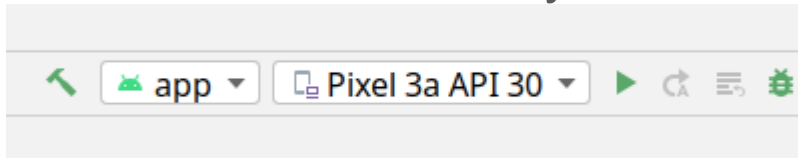


# AVD - Android Virtual Device



# Executando o App no AVD

- Clicando no botão “Play”

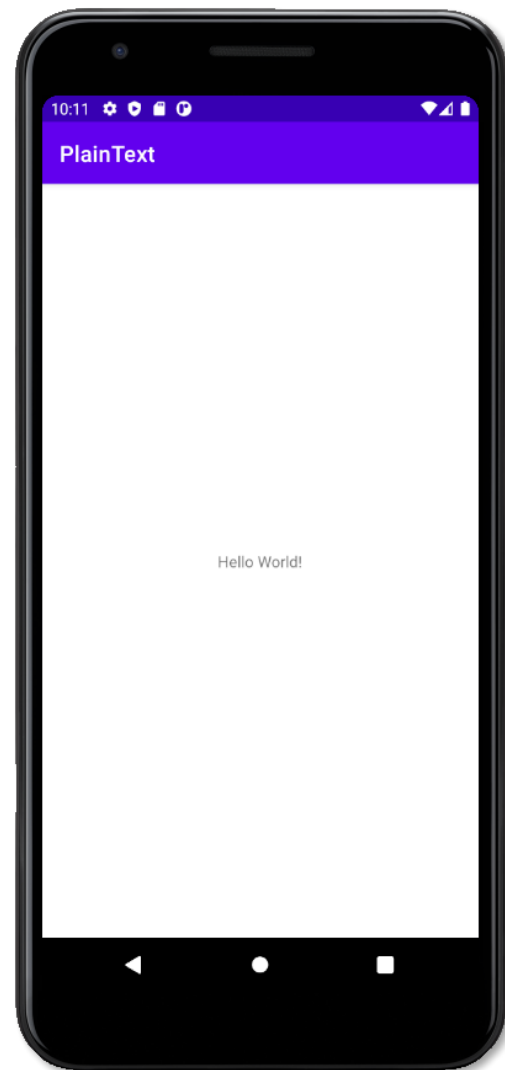


- No Menu

- ☐ Run → Run ‘App’
- ☐ Shift + F10

- Teclado

- ☐ Botão ESC → Voltar
- ☐ Botão HOME → Home
- ☐ F2 → Menu





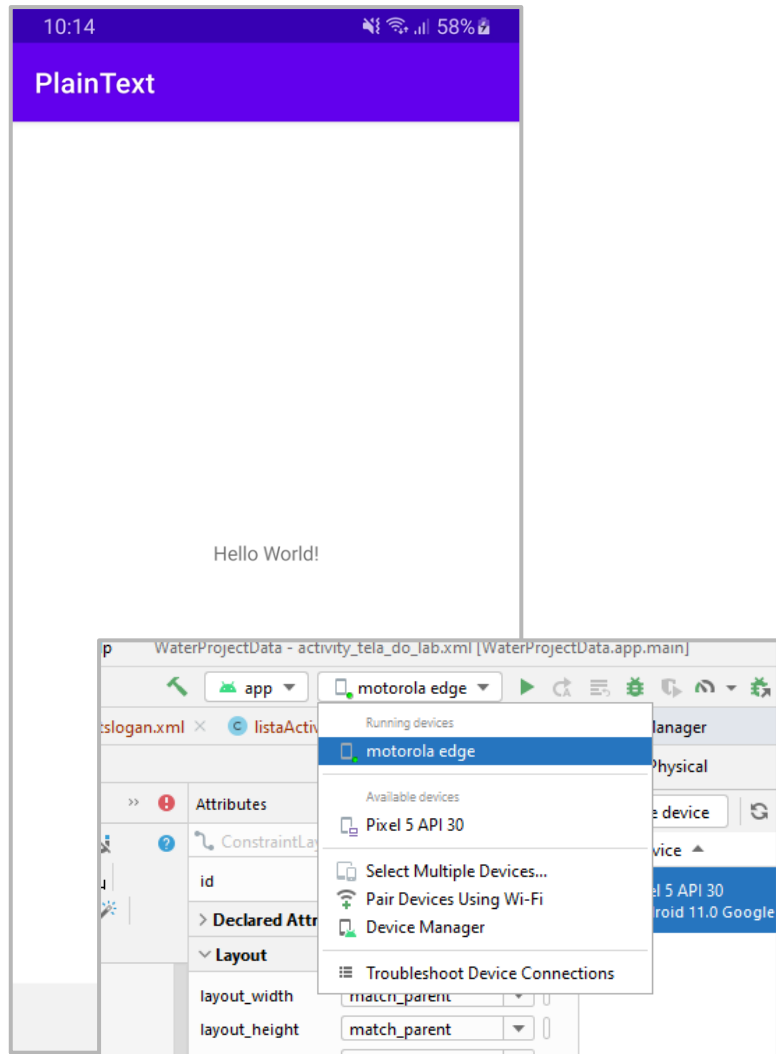
# Executando no Celular

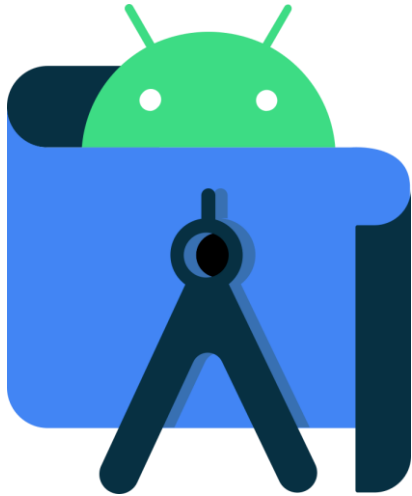


- Dependendo da configuração da sua máquina de desenvolvimento, o AVD pode ficar muito lento ou mesmo nem executar
- Além disso, o AVD não tem suporte a todos os recursos
  - Bluetooth
  - Movimentos realistas (para testar sensores de movimento)
  - Câmera do celular (que, normalmente, é melhor que as webcams)
- Outras vezes, você quer ver seu App executando em um celular real

# Executando no Celular

- Passos para executar no celular
  - Conecta celular via USB
  - Habilita “Depuração de USB” no dispositivo
    - *Android 7 em diante: Opções → Sobre o dispositivo → Info. Software → Clique no N. de Compilação diversas vezes → Voltar → Voltar → Opções do Desenv. → Depuração de USB*
    - *Em outros celulares, este passo pode variar*
  - No Android Studio, Run → Run
    - *O Android Studio detecta seu celular conectado, instala e inicia o App no celular*





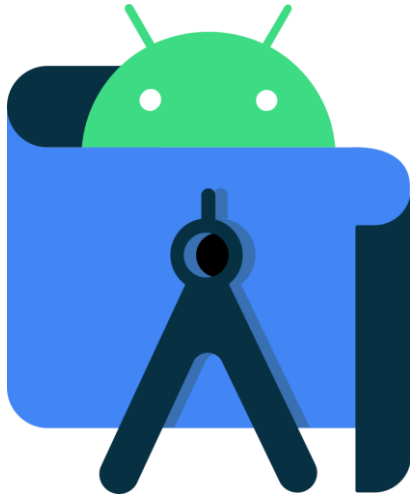
# Hello World 2.0

Prof. Diogo Soares

# Mude o Hello World

- Vamos agora adicionar um Hello personalizado
  - Mude a frase para Hello user!
  - No controller (MainActivity.java):
    - Crie uma String com seu nome como variável global da classe
    - Na função onCreate, vamos trocar o valor de user no XML
    - Crie uma variável helloText para receber o id da activityMain.xml
    - Modifique usando as funções da TextView
  - Recompile

```
private TextView helloText;  
1 usage  
private String meuNome = "Diogo";  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    helloText = (TextView) findViewById(R.id.helloUser);  
    String aux = helloText.getText().toString();  
    aux.replace(target: "user", meuNome);  
    helloText.setText(aux);  
  
    Log.i(tag: "Debug", msg: "username alterado");  
}
```

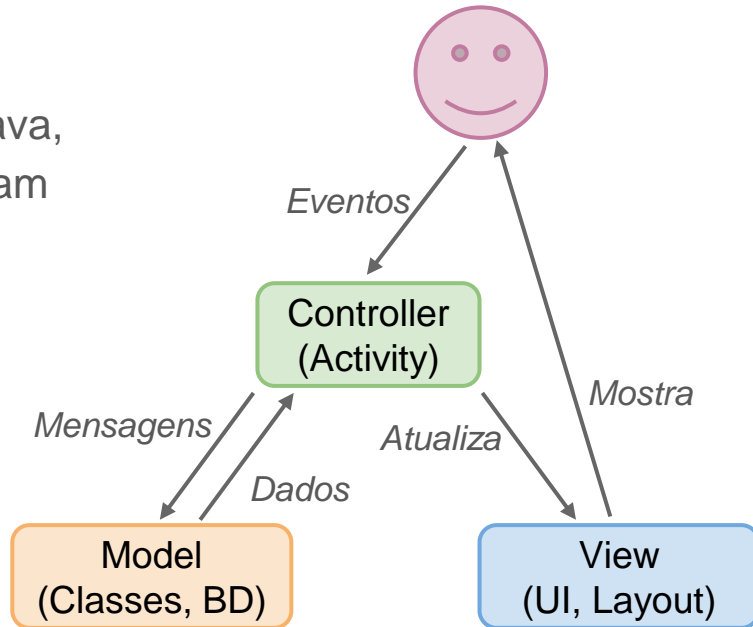


# Interface Gráfica

Prof. Diogo Soares

# Modelo MVC

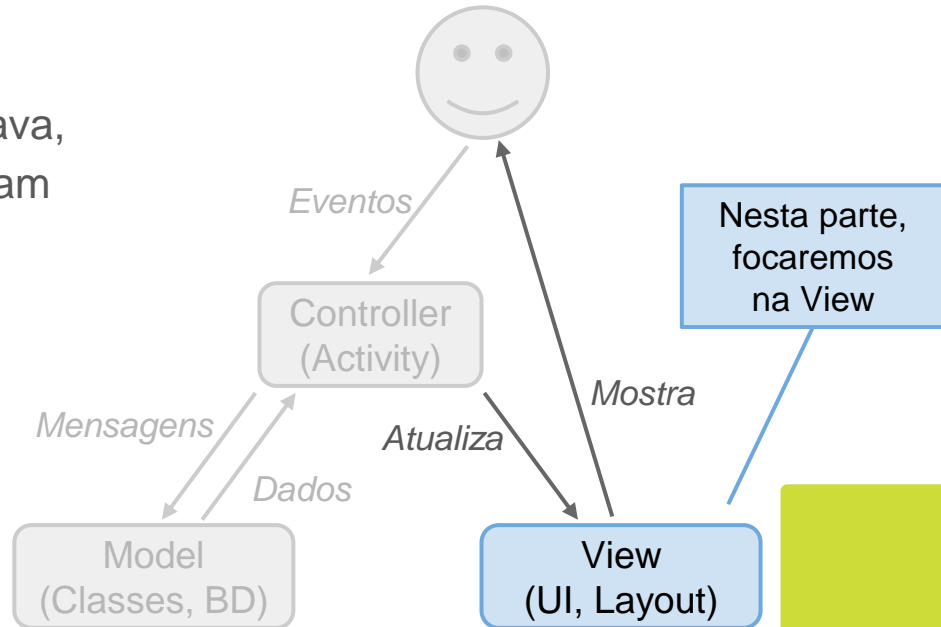
- O Android usa o modelo MVC – *Model, View, Controller*
  - *Model*: classes implementadas em Java
  - *View*: componentes da Interface normalmente feitos em XML
  - *Controller*: implementados em Java, instanciam os modelos a controlam as views. São conhecidas como “activities” no Android



# Modelo MVC

## ■ O Android usa o modelo MVC – *Model, View, Controller*

- *Model*: classes implementadas em Java
- *View*: componentes da Interface normalmente feitos em XML
- *Controller*: implementados em Java, instanciam os modelos a controlam as views. São conhecidas como “activities” no Android



# Interface Gráfica

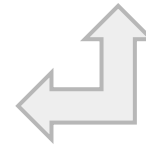
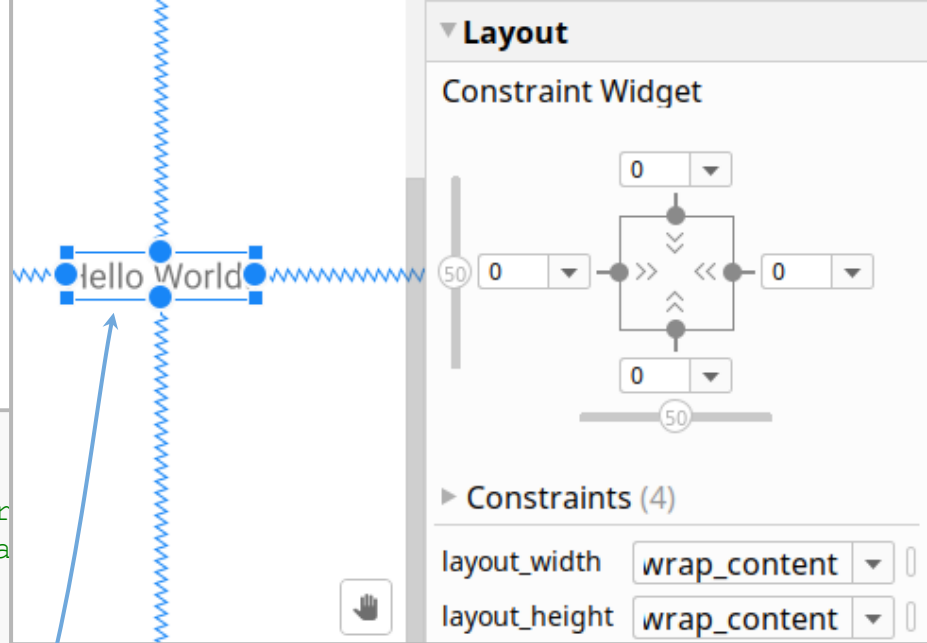
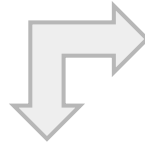


- A interface gráfica pode ser feita de três formas:
  - Usando o Editor do Android Studio (Design)
    - *O editor irá gerar um XML do Layout automaticamente*
  - Editando o XML (Text) manualmente
    - *O Android faz um parse do XML do Layout e gera internamente os objetos dos componentes (Views). Este processo é conhecido como “inflate” do XML.*
    - *Toda tag XML tem uma classe Java correspondente (subclasse da classe View)*
  - Gerando Códigos em Java manualmente
    - *Instanciando objetos de componentes gráficos (Views) manualmente e adicionando-os à interface (similar ao feito no Swing)*
    - *Não recomendado por quebrar o modelo MVC.*



# Design e Text

- Do design ao text
  - Na prática, alternamos entre os dois



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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: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>
```

# Layouts



## ▣ Layouts

- ▣ Agrupam componentes UI (Views) relacionados
  - Podem ter outros Layouts internamente (que agruparão outras Views)
- ▣ Definem como os componentes serão posicionados na tela
  - Similar aos layouts do Swing
- ▣ Layouts são classes (e tags) que herdam a classe ViewGroup

## ▣ Todo componente (botões, textos, etc) deve estar dentro de um Layout

## ▣ Principais Layouts:

- ▣ ConstraintLayout (recomendado)
- ▣ RelativeLayout
- ▣ LinearLayout

# Constraint Layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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:id="@+id/textHello"
        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" />
    <TextView
        android:id="@+id/textIntroducao"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="42dp"
        android:layout_marginTop="32dp"
        android:text="Introdução ao Android"
        app:layout_constraintLeft_toLeftOf="@+id/textHello"
        app:layout_constraintTop_toBottomOf="@+id/textHello" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



Attributes

Ab textIntroducao TextView

id textIntroducao

Declared Attributes + -

Layout

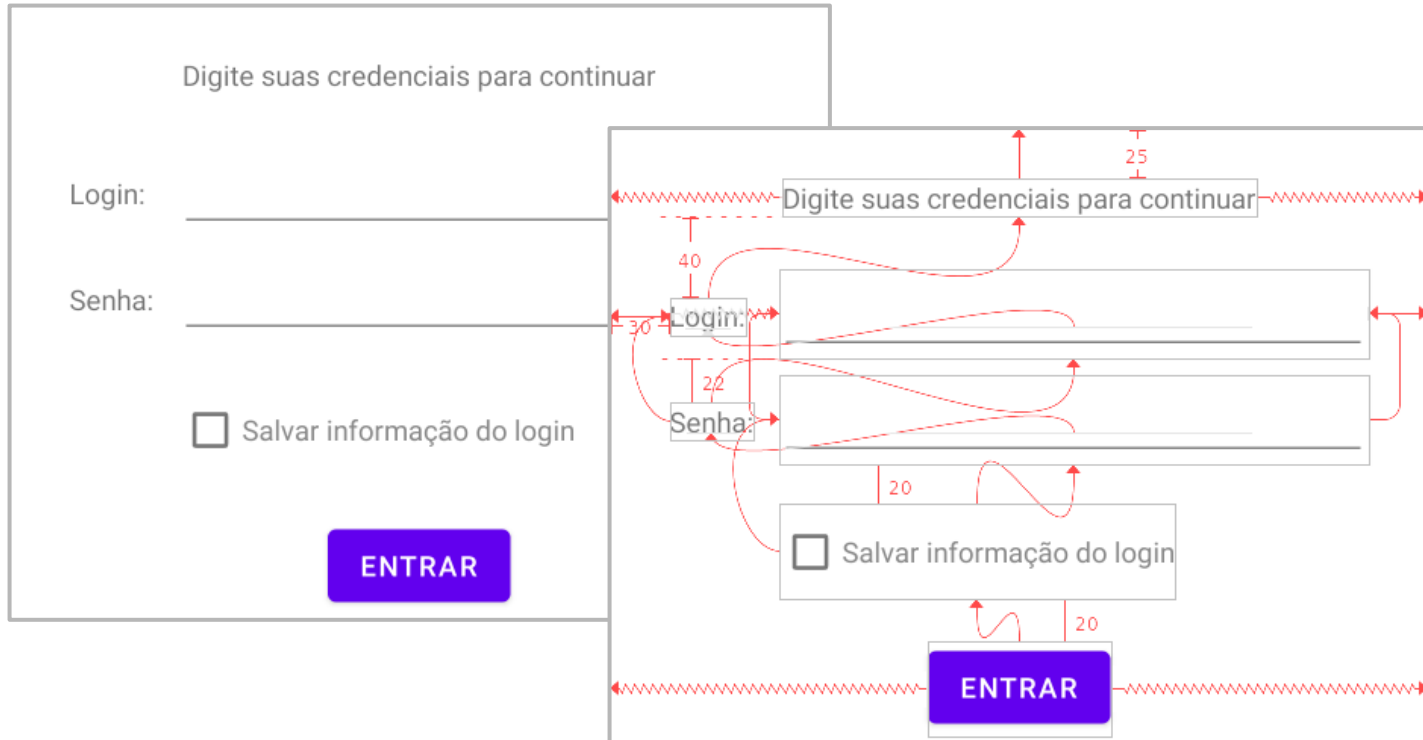
Constraint Widget

Constraints

- Left → LeftOf textHello (0dp)
- Top → BottomOf textHello (3...

# ConstraintLayout

- Posiciona elementos com base em outros, de forma relativa



# Recursos (*Resources*)




- Recursos são arquivos externos ao seu código-fonte (java)
- São “externalizados” do seu código para serem mantidos independentemente e mais facilmente
- Ficam na pasta “res”
- Exemplos:
  - Imagens
  - Layouts
  - Menus
  - Outros valores
    - *Strings, dimensões, estilos, etc*

# Recursos (*Resources*)

## ■ Exemplo de recursos de Strings (res/values/strings.xml)

```
<resources>
  <string name="app_name">PlainText</string>
  <string name="introducao">Introdução ao Android</string>
</resources>
```



## ■ São acessados pelo nome usando o “@”:

```
<TextView
  android:id="@+id/textIntroducao"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_marginStart="42dp"
  android:layout_marginTop="32dp"
  android:text="@string/introducao"
  app:layout_constraintStart_toStartOf="@+id/textHello"
  app:layout_constraintTop_toBottomOf="@+id/textHello" />
```

# Componentes

---

- São as partes da tela que o usuário vê e interage
  - Assim como no Swing, são objetos de uma determinada classe
  - Possuem atributos e métodos
- Os componentes são também chamados de *views*, pois todas as classes que as implementam herdam a classe `View`
- Exemplos de componentes (*views*):
  - `TextView`
  - `EditText`
  - `CheckBox`
  - `Button`
  - `ImageView`

# TextView

■ Representa um texto (label)

The image shows the Android Studio interface with several annotations and dialog boxes:

- Palette:** The 'Common' tab is selected, and 'TextView' is highlighted in the list.
- Design View:** A 'TextView' widget is being dragged onto the design surface.
- Properties Panel:** The 'Declared Attributes' section shows 'text' with the value 'TextView'.
- New String Value Dialog:** The 'Resource name' is 'message', the 'Resource value' is 'Digite suas credenciais para', and the 'File name' is 'strings.xml'.
- Pick a Resource Dialog:** The 'String' resource is selected, and the 'Preview' shows the new string value.

Annotations and callouts:

- Clique no botão para selecionar um recurso (String):** Points to the 'TextView' value in the 'Declared Attributes' section.
- Selecione o TextView:** Points to the 'TextView' widget in the Palette.
- Arraste para a tela de design:** Points to the 'TextView' widget being dragged onto the design surface.
- Novo valor Escreva uma String value:** Points to the 'Resource value' field in the 'New String Value' dialog.



# TextView

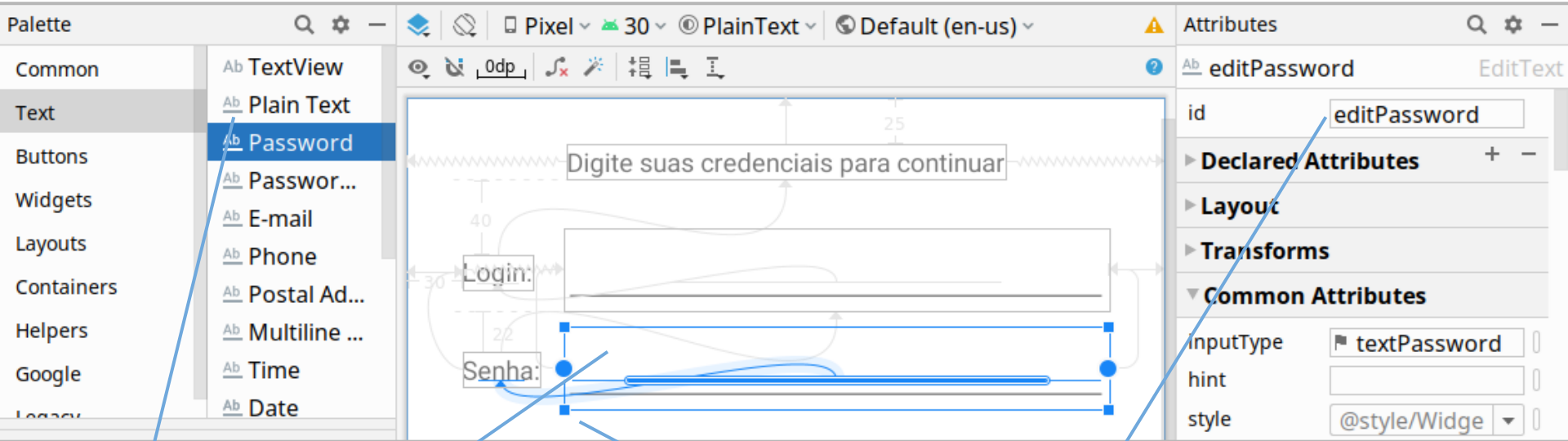
■ Representa um texto (label)

■ XML resultante do slide anterior (campo de senha apenas)

```
<TextView
    android:id="@+id/textMessage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dp"
    android:text="@string/message"
    android:textSize="14dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

# EditText

- Texto de entrada do usuário
- Vários tipos: `text`, `textPassword`, `number`, etc



Tipos de entrada

Arraste para a tela de design

Ajuste os alinhamentos

Diga o ID do componente. Ele será usado para acessar o valor digitado a partir do Java

# EditText

- Texto de entrada do usuário
- Vários tipos: `text`, `textPassword`, `number`, etc

- XML resultante do slide anterior (campo de senha apenas)

```
<EditText
    android:id="@+id/editPassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    app:layout_constraintStart_toStartOf="@id/editLogin"
    app:layout_constraintEnd_toEndOf="@id/editLogin"
    app:layout_constraintBaseline_toBaselineOf="@id/textPassword" />
```

# CheckBox

## Caixa de seleção

The screenshot shows the Android Studio IDE with a login form design. The **Palette** on the left shows the **CheckBox** widget selected under the **Buttons** category. The **Component Tree** lists the form's components: `ConstraintLayout`, `textMessage "@string/..."`, `textLogin "Login:"`, `editLogin (E-mail)"`, and `textPassword "Senha:"`. The **Design View** shows a login form with a title "Digite suas credenciais para continuar", two text input fields labeled "Login:" and "Senha:", and a "Salvar informação de login" checkbox. The **Attributes** panel on the right shows the `checkSaveLogin` attribute for the checkbox, with its ID set to `checkSaveLogin`. The **Declared Attributes** section shows the **Layout** constraints: `Start → StartOf editPassword` and `Top → BottomOf editPassword`. Blue arrows point from text boxes at the bottom to the following elements:

- Component e**: Points to the **CheckBox** widget in the **Palette**.
- Arraste para a tela de design**: Points to the **CheckBox** widget being dragged onto the design.
- Ajuste os alinhamentos**: Points to the alignment handles of the checkbox in the design.
- Mude o texto**: Points to the text "Salvar informação de login" of the checkbox.
- Ajuste o ID**: Points to the `id` attribute in the **Attributes** panel.

# CheckBox

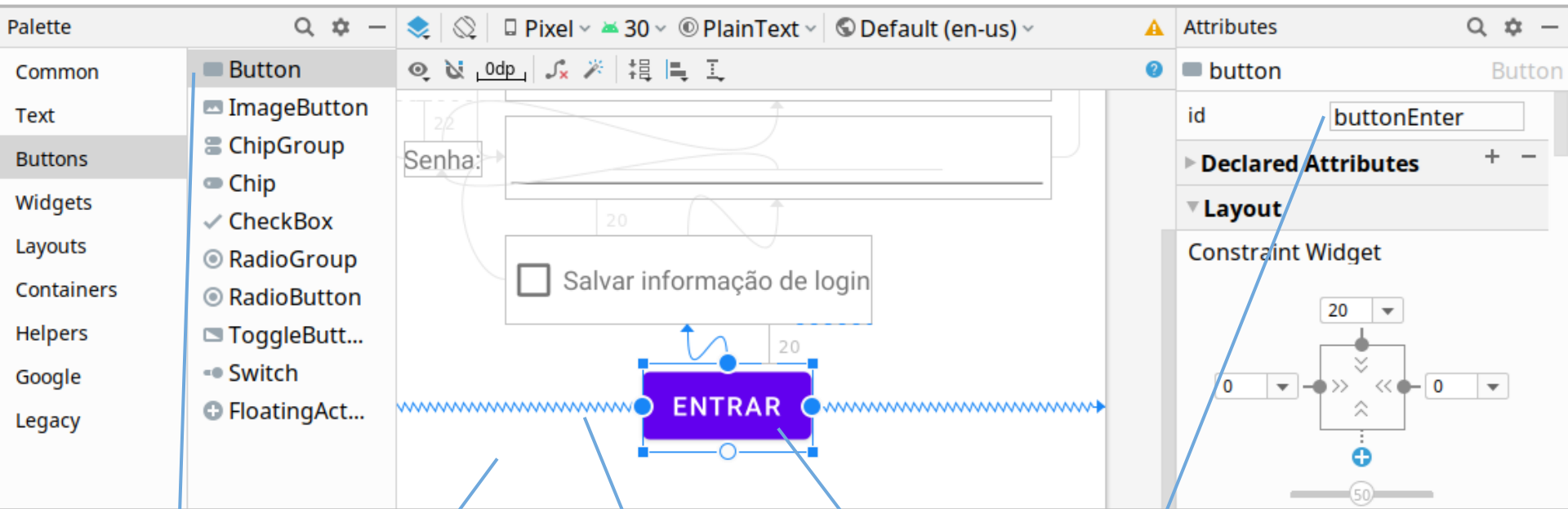
☐ Caixa de seleção

## ☐ XML resultante do slide anterior

```
<CheckBox
    android:id="@+id/checkSaveLogin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="@string/saveLogin"
    android:textColor="#777777"
    app:layout_constraintStart_toStartOf="@+id/editPassword"
    app:layout_constraintTop_toBottomOf="@+id/editPassword" />
```

# Button

■ Botão clicável



Component  
e

Arraste para a  
tela de design

Ajuste os  
alinhamentos

Mude o texto

Ajuste o ID

# Button

■ Botão clicável

## ■ XML resultante do slide anterior

```
<Button
    android:id="@+id/buttonEnter"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="@string/enter"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/checkSaveLogin" />
```

# Mudando as Cores



- No arquivo “res/values/themes/themes.xml” é possível mudar as cores básicas do tema do seu aplicativo
- Na verdade, é possível personalizar basicamente todos os componentes do aplicativo
  - Entretanto, os detalhes desse arquivo vão além do escopo do curso



# Mudando as Cores

```
<resources xmlns:tools="http://schemas.android.com/tools">

  <style name="Theme.PlainText" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
    <!-- Primary brand color. -->
    <item name="colorPrimary">#565758</item>
    <item name="colorPrimaryVariant">#565758</item>
    <item name="colorOnPrimary">#a1c639</item>
    <!-- Secondary brand color. -->
    <item name="colorSecondary">#565758</item>
    <item name="colorSecondaryVariant">#a1c639</item>
    <item name="colorOnSecondary">#a1c639</item>
    <!-- Status bar color. -->
    <item name="android:statusBarColor" tools:targetApi="1">#565758</item>
    <!-- Customize your theme here. -->
    <item name="android:textColor">#39393a</item>
  </style>

</resources>
```

Fundo da barra do app

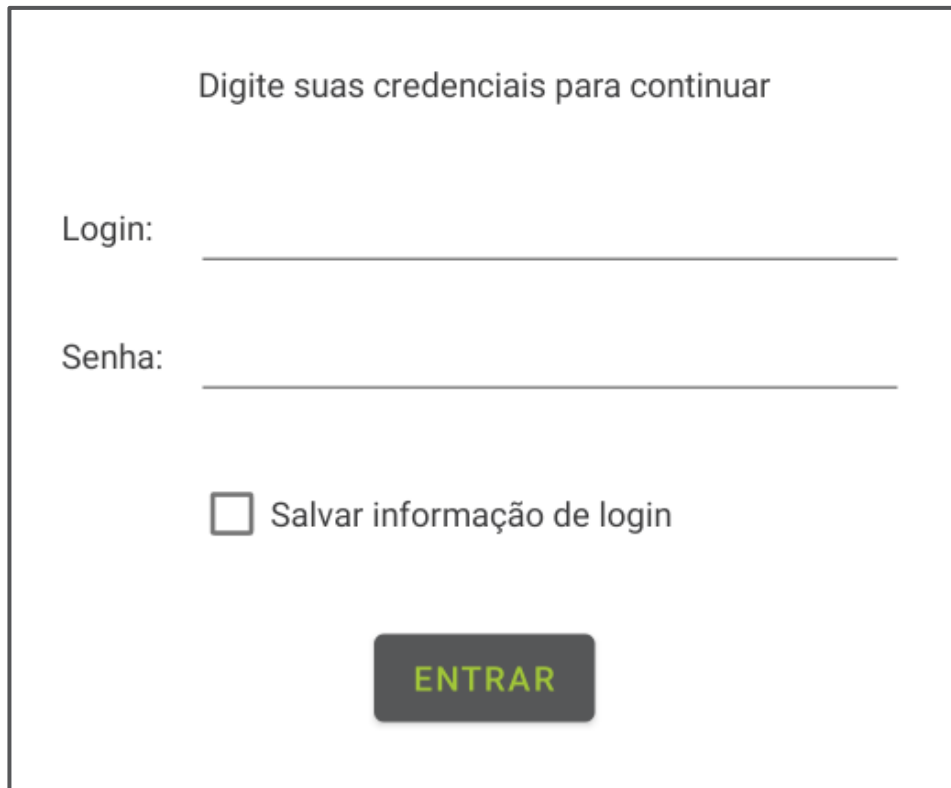
Texto dos botões

Fundo da barra do relógio

Textos das telas

# Mudando as Cores

- Ao modificar o XML, basta voltar para a tela de design

A login form mockup with a light gray background and a dark gray border. At the top, the text "Digite suas credenciais para continuar" is centered. Below it are two input fields: "Login:" and "Senha:". Under the password field is a checkbox labeled "Salvar informação de login". At the bottom center is a dark gray button with the text "ENTRAR" in yellow.

Digite suas credenciais para continuar

Login: \_\_\_\_\_

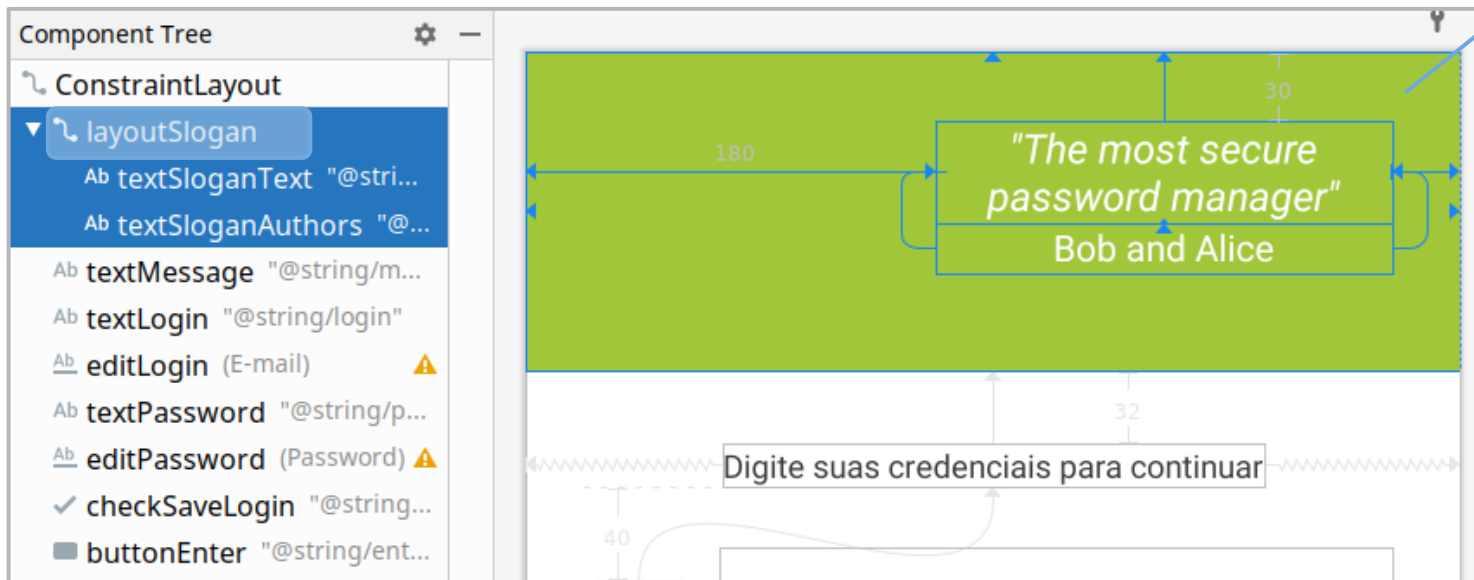
Senha: \_\_\_\_\_

☐ Salvar informação de login

**ENTRAR**

# Layout dentro de Layout

- Um layout pode ser um sub-componente de um layout
- Combine layouts para obter os efeitos desejados



layoutSlogan  
é um layout com  
componentes  
dentro

# ImageView

---

- O componente ImageView permite inserir imagens na interface

- Tipos de Imagens

  - Não-Vetoriais

    - *Imagem é composta por um conjunto de “pixels”*
    - *O tamanho/qualidade da imagem depende da densidade de pixels da tela*
    - *PNG, GIF, JPG*

  - Vetoriais

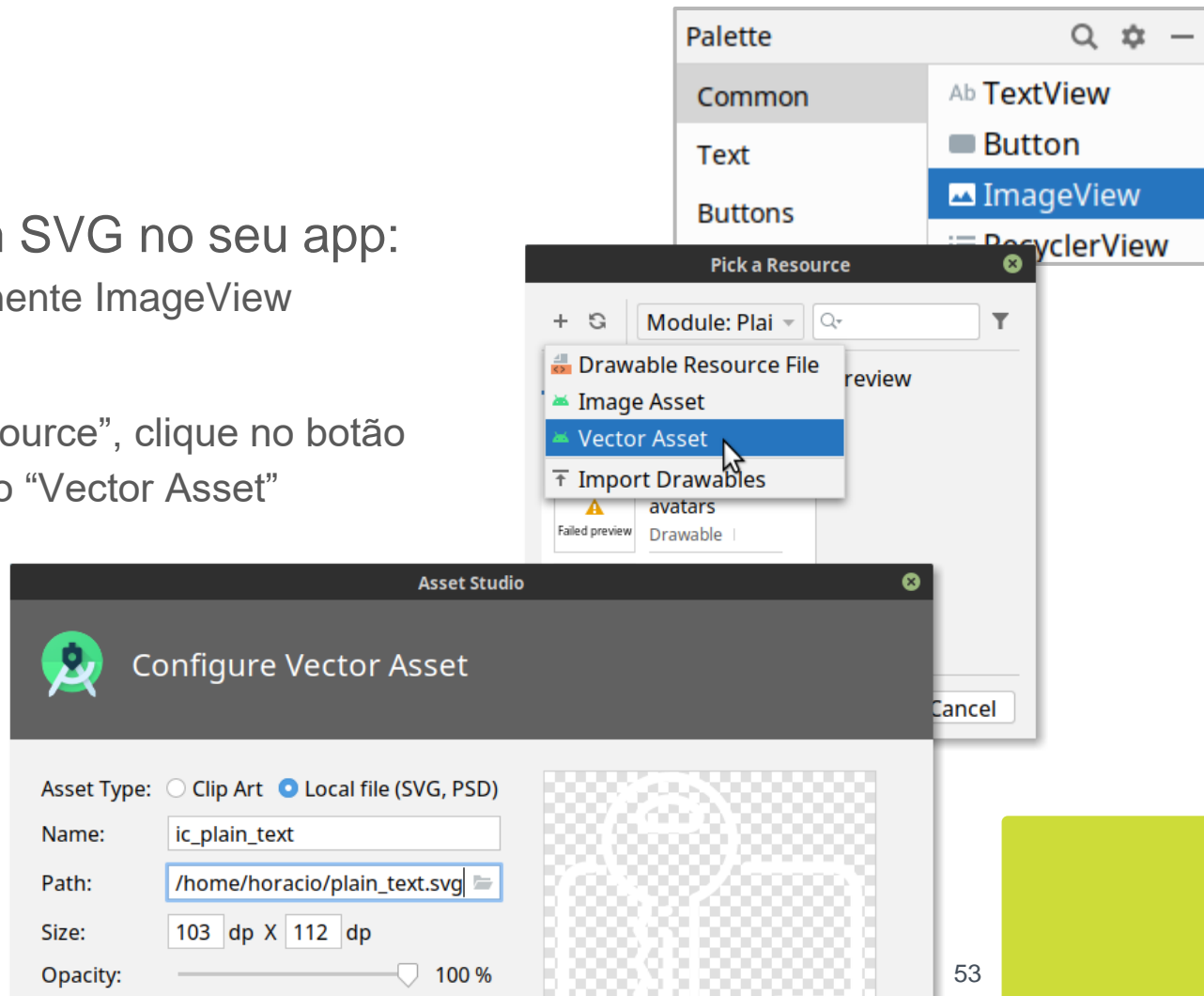
    - *Imagem é composta por um conjunto de linhas, quadrados, paths, etc*
    - *Qualidade é mantida independente do tamanho, zoom, densidade de pixes, etc*
    - *SVG, EPS*

- Como os apps executam em diferentes telas e dispositivos, recomenda-se o uso de imagens vetoriais

  - O Android tem suporte a imagens SVG

# ImageView

- Para adicionar um SVG no seu app:
  - Arraste o componente ImageView para a interface
  - Na janela “Pick a Resource”, clique no botão “+” e, depois na opção “Vector Asset”
- Selecione “Local File”
- Indique o Path
- Por fim, clique em “Next” → “Finish”, e selecione a nova imagem na janela “Pick a Resource”



# ImageView

Ajuste os  
alinhamentos

Ajuste o ID

Palette

- Common
- Text
- Buttons
- Widgets
- Layouts
- Containers

Ab TextView

- Button
- ImageVie...
- Recycler...
- <> <fragme...
- ScrollView
- Switch

Component Tree

- ConstraintLayout
- layoutSlogan
- imageLogo
- Ab textSloganText "@stri...
- Ab textSloganAuthors "@...
- Ab textMessage "@string/m...
- Ab textLogin "@string/login"
- Ab editLogin (E-mail)
- Ab textPassword "@string/p...
- Ab editPassword (Password)
- checkSaveLogin "@string...
- buttonEnter "@string/ent...

Attributes

imageLogo

id imageLogo

Declared Attributes

Layout

Constraint Widget

0 30 0

Start → StartOf parent (30dp)

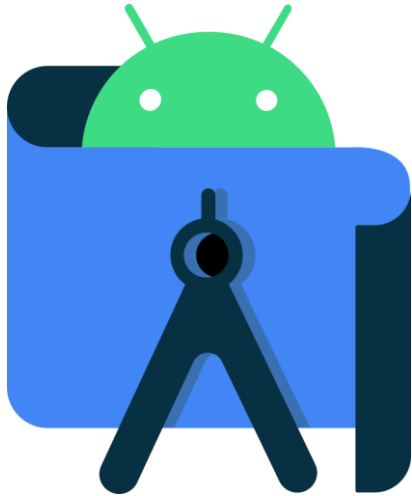
Top → TopOf parent (0dp)

Bottom → BottomOf parent (...)

layout\_width 151dp

layout\_height 110dp

visibility

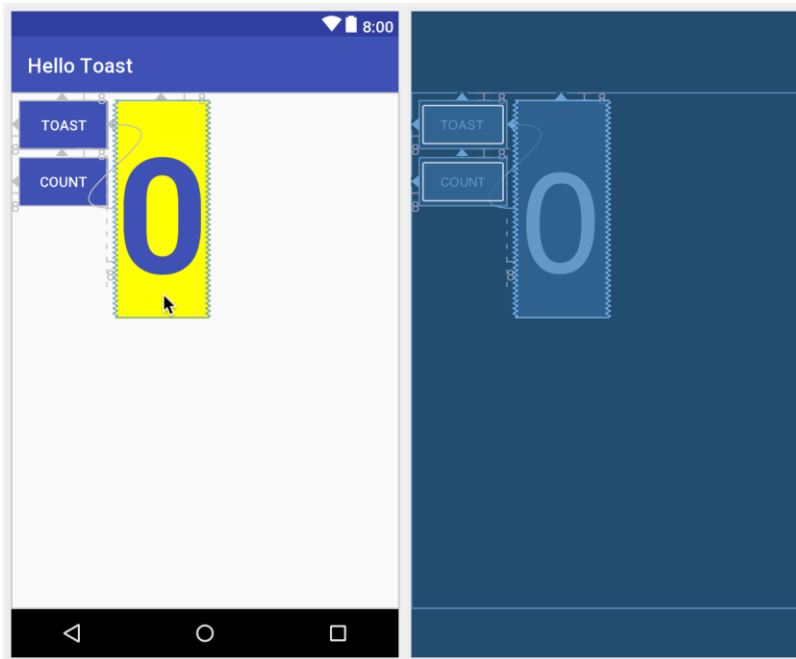


# Challenging

Prof. Diogo Soares

# HelloToast

- Faça um app HelloToast com 3 componentes
  - 2 botões (um de mensagem e um de contagem) e um textview entre os 2 botões com o número de vezes que o botão contagem foi clicado
- Desafio 2: deixe seu app responsivo para visão horizontal ou vertical
- Exemplo:





# Referências



- ▣ Training for Android Developers
  - ▣ <http://developer.android.com/training/>
- ▣ Android API Guide
  - ▣ <http://developer.android.com/guide/components/>
- ▣ Android API Reference
  - ▣ <http://developer.android.com/reference/>