



PROGRAMAÇÃO PARA  
DISPOSITIVOS MÓVEIS

# PROJETOS ANDROID

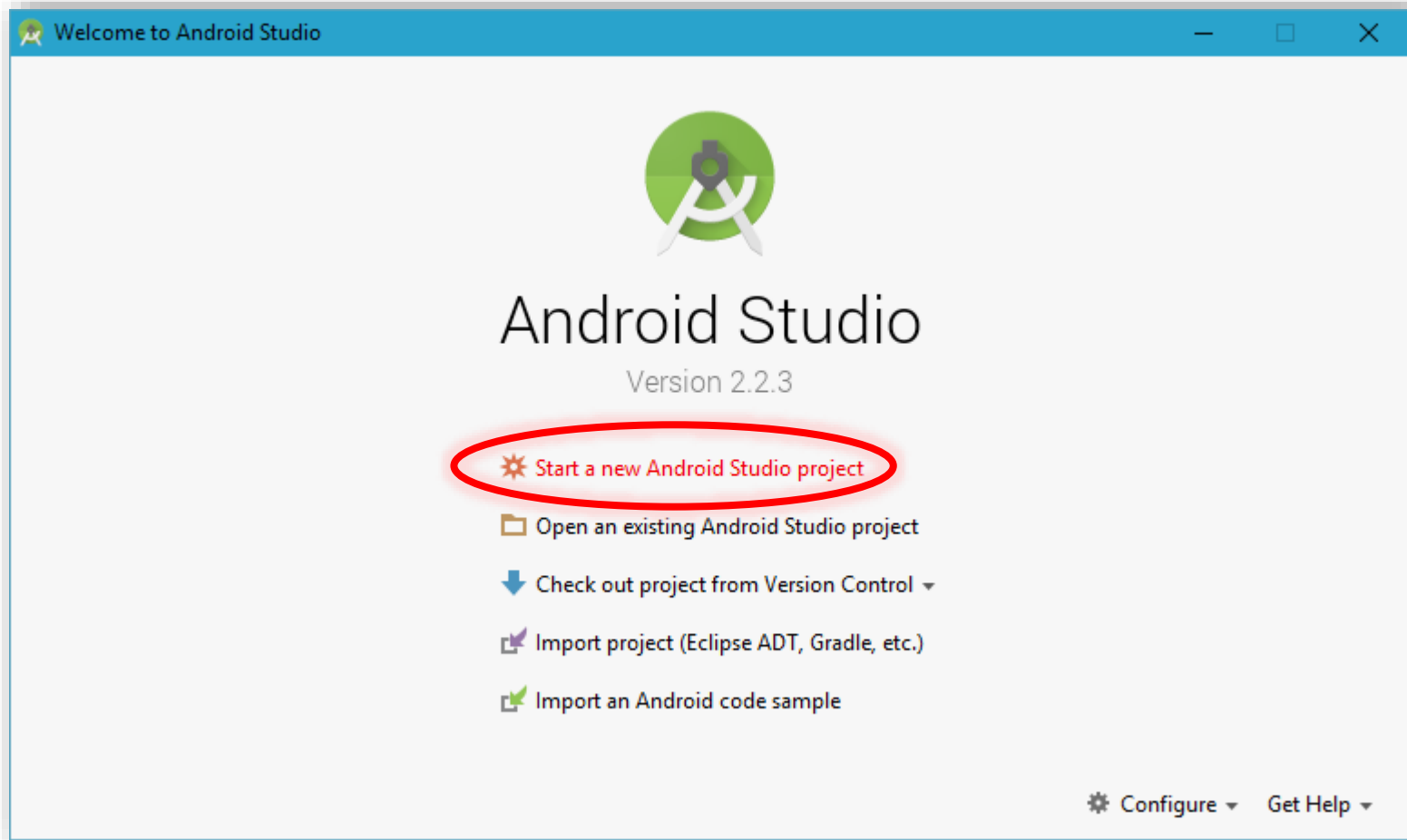
# Aplicações Android

- Aplicações Android normalmente são compostas de múltiplas telas com diversos comportamentos
- As telas, junto a demais códigos de acionamento de serviços, modelos e configurações, compõem a aplicação
- Uma aplicação é representada por um projeto Android contendo as classes e recursos da aplicação

# Novo Projeto

- Vamos criar uma aplicação Android para primeiro contato com os conceitos a serem estudados;
- Para isso, inicie o Android Studio;
- No Wizard, selecione a opção “Start a New Android Studio Project”;

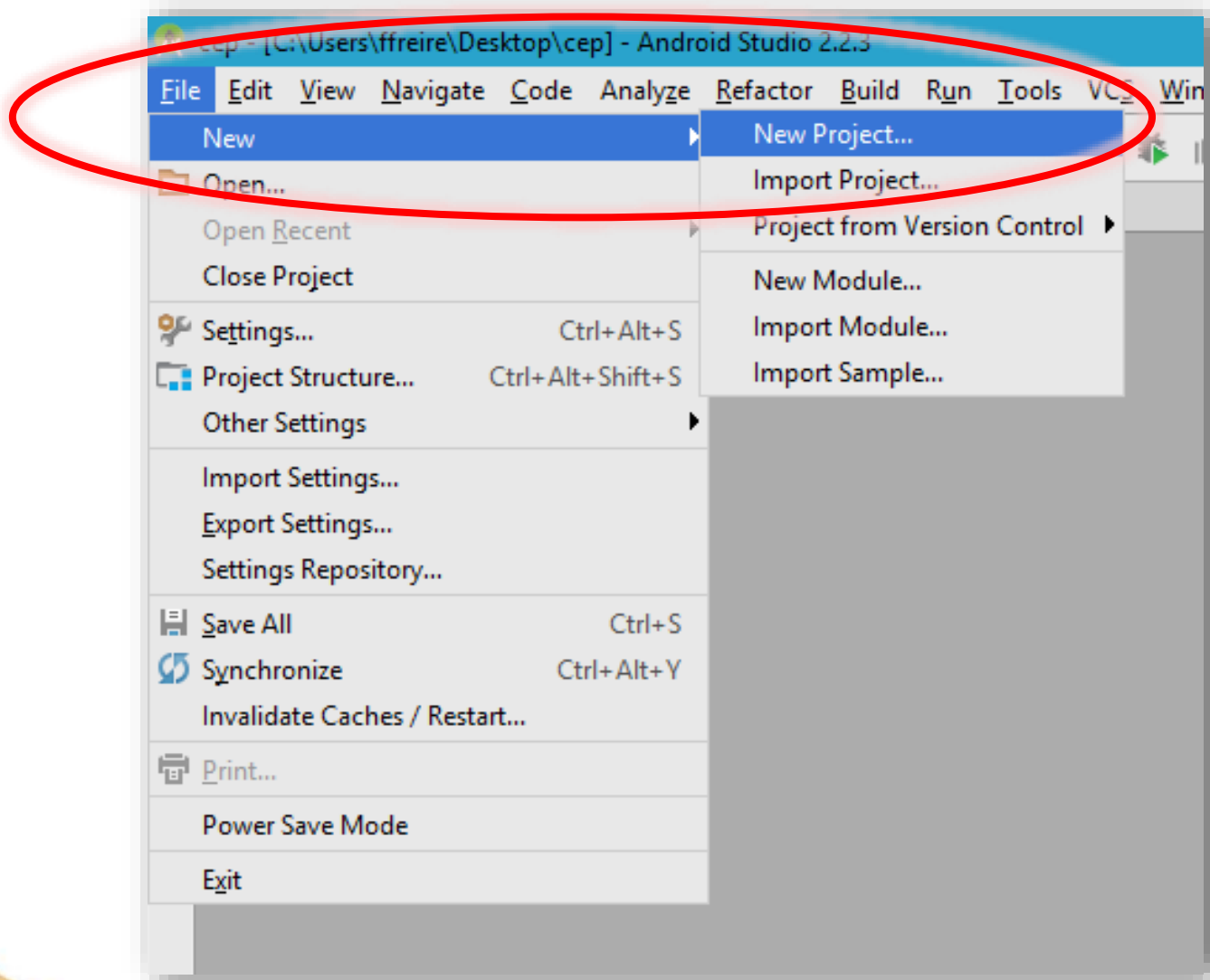
# Novo Projeto



# Novo Projeto

- Se o Android Studio abrir um projeto aberto anteriormente, é possível criar um novo projeto selecionando o Item “New” > “New Project...” no menu “File”.

# Novo Projeto



# Assistente de Criação de Projeto

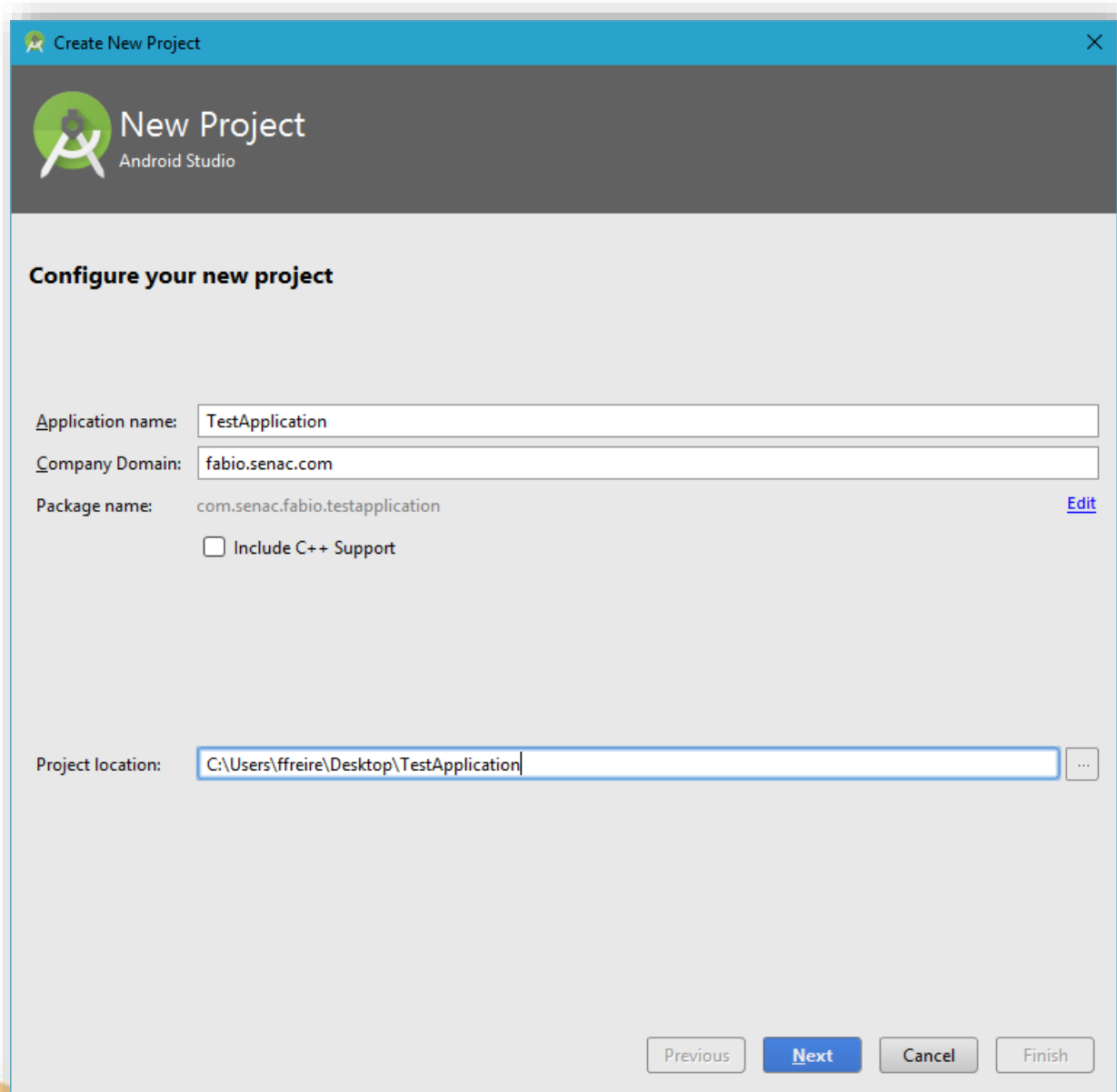
- No assistente que surgir, é possível configurar o projeto a ser criado;
- Na primeira tela, será possível informar o nome do projeto, o domínio da empresa criadora do aplicativo (utilizado para identificação no Google Play e em contextos) e o local do projeto;




# Assistente de Criação de Projeto

- Em “Application Name”, informe “TestApp”;
- Em “Company Domain”, informe “seunome.senac.com”;
- Em “Project Location”, informe o local onde o projeto será salvo no disco rígido;
- Após inserir as informações, clique em “Next”.

# Assistente de Criação de Projeto



Create New Project

 New Project  
Android Studio

**Configure your new project**

Application name:

Company Domain:

Package name:  [Edit](#)

☐ Include C++ Support

Project location:  ...

Previous Next Cancel Finish

# Tipo de Dispositivo e Versão

- Na próxima “página” do assistente, será possível configurar a plataforma destino da aplicação a ser criada;
- É possível escolher entre tipo de dispositivo (celular e tablets, relógios, TVs e etc) e a API mínima do Android que o dispositivo precisa possuir para executar a aplicação;

# Tipo de Dispositivo e Versão

- Quanto maior a versão da API, mais recursos estarão disponíveis para utilização no dispositivo;
- No entanto, maiores versões estão em números reduzidos de dispositivos, de forma que é necessário efetuar uma avaliação criteriosa e escolher uma versão com marketshare significativo, mas que não seja muito antiga;

# Tipo de Dispositivo e Versão

- Marque a opção “Phone and Tablet” e escolha a versão “API 23: Android 6.0 (Marshmallow)”;
- Se seu aparelho possui uma versão menor do Android, escolha esta versão na opção “Phone and Tablet”;
- Evitar utilizar versões muito antigas do Android (< 5.0), ou muitos recursos poderão não estar disponíveis;
- Ao concluir a seleção, clique em “Next”.

# Assistente de Criação de Projeto

Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK: API 23: Android 6.0 (Marshmallow)

Lower API levels target more devices, but have fewer features available.

By targeting API 23 and later, your app will run on approximately 15,3% of the devices that are active on the Google Play Store.

[Help me choose](#)

Stats load failed. Value may be out of date.

☐ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Glass (Not Available)

Minimum SDK:

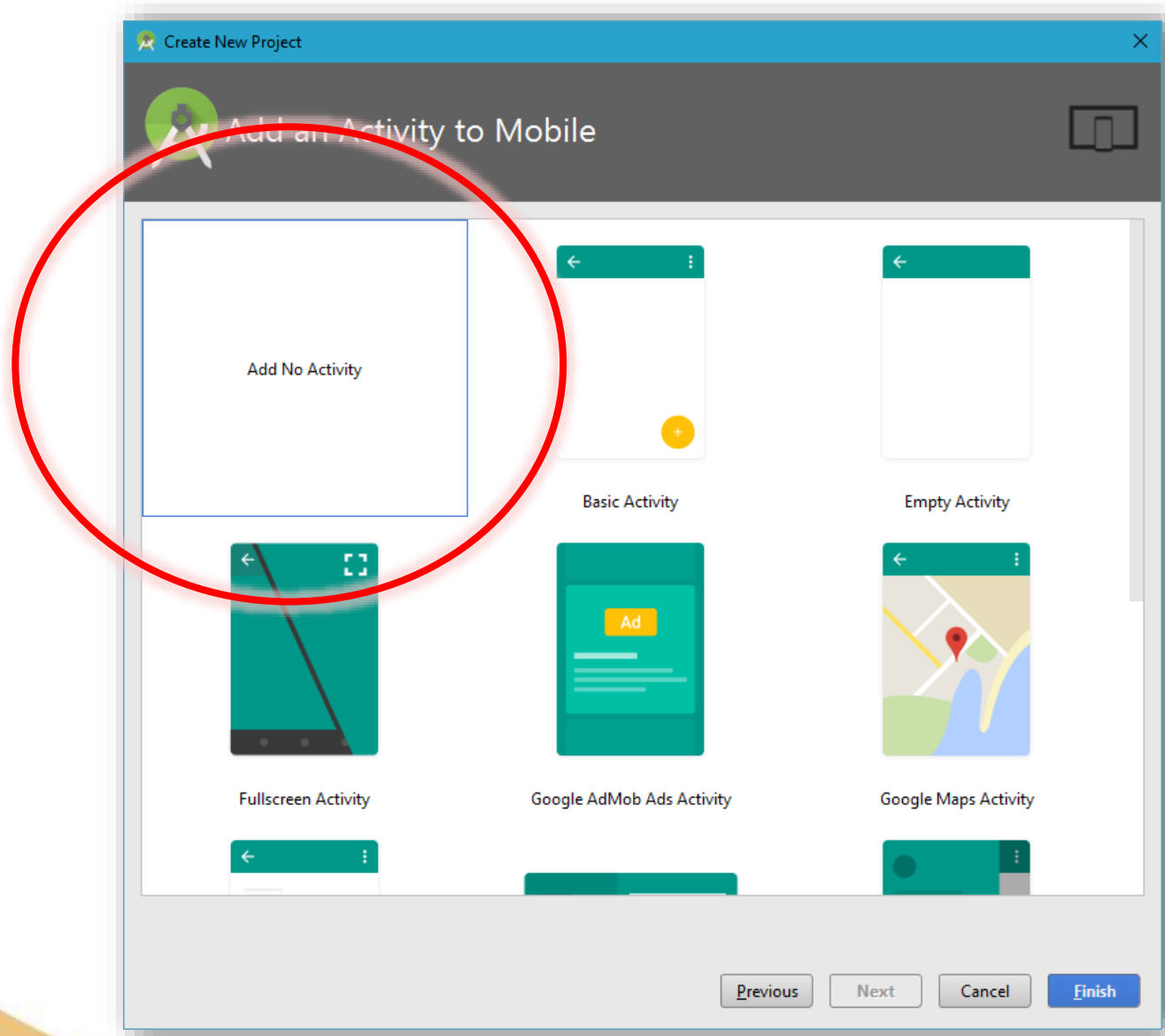
Previous Next Cancel Finish

Indica o % de dispositivos Android que estão atualizados com a versão selecionada

# Adição de Atividade de Template

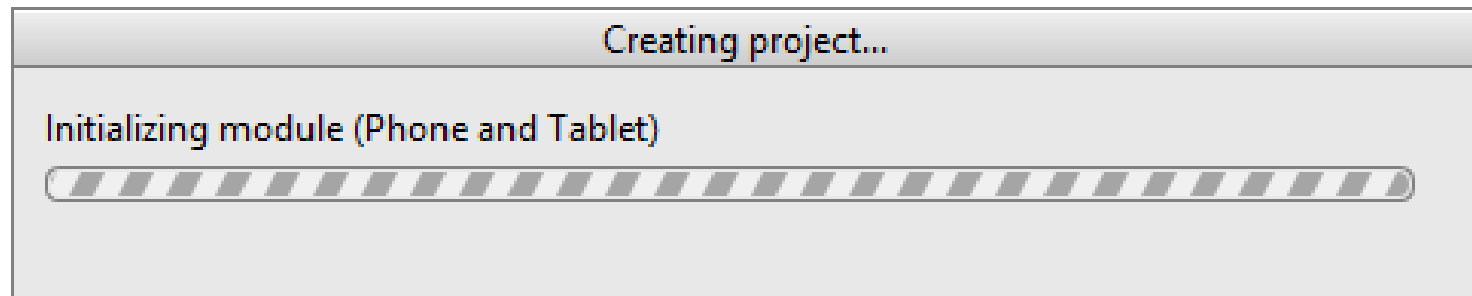
- Na próxima página, é possível selecionar uma atividade baseada em um template para que o Android Studio crie-a e adicione-a ao projeto automaticamente;
- Não vamos utilizar uma atividade de template, então escolha a opção “Add No Activity” e clique em “Finish” para concluir o assistente e iniciar a criação do projeto;

# Adição de Atividade de Template





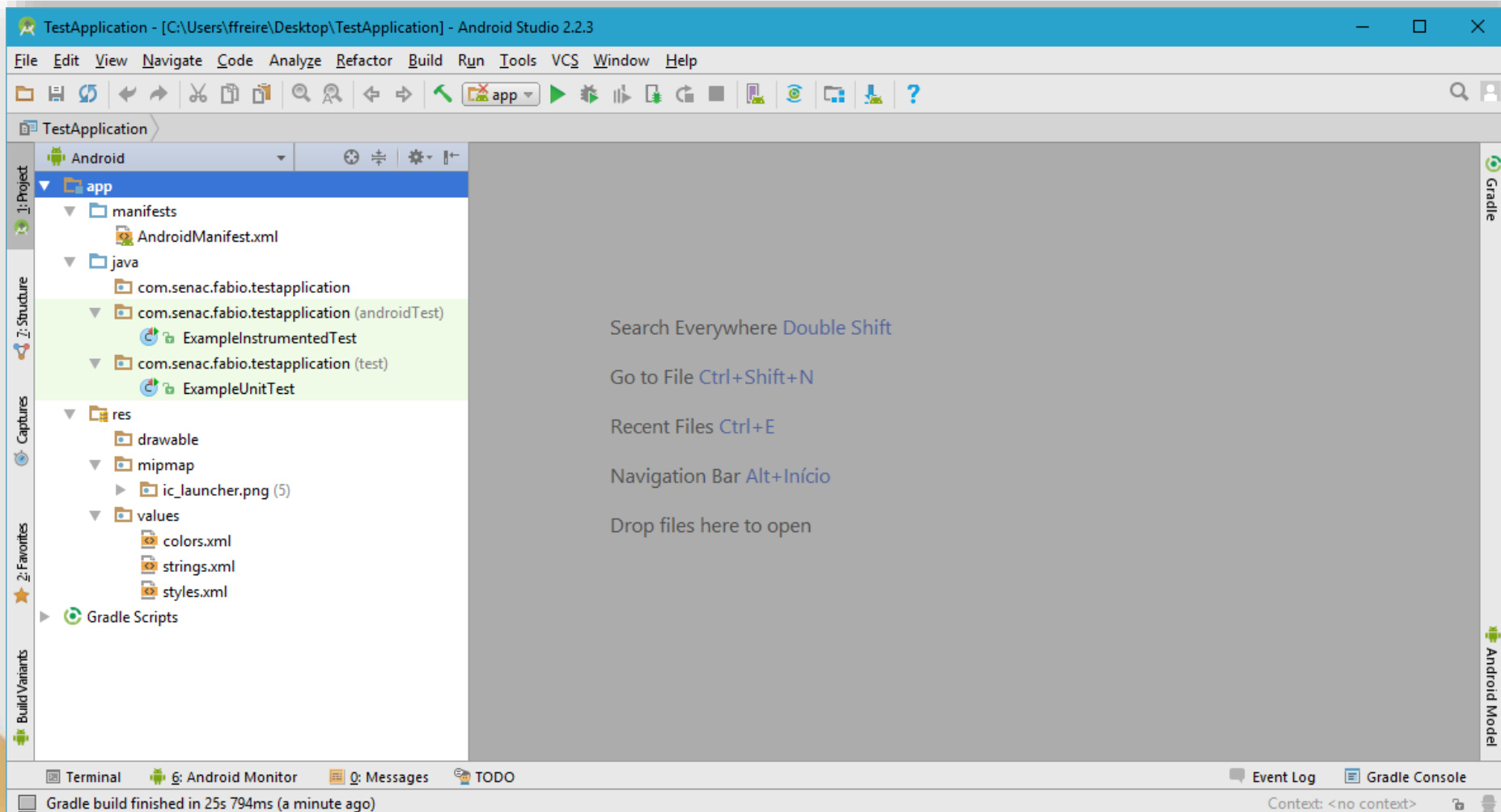
# Criação do Projeto



# Conclusão da Criação do Projeto

- Quando o assistente concluir (todas as barras de progresso desaparecerem) o projeto terá sido criado e uma tela semelhante a seguinte será exibida;
- A tela mostra a estrutura do projeto sem nenhum arquivo aberto no editor.

# Conclusão da Criação do Projeto



# ATIVIDADES

# O que é uma atividade

- Cada “tela” de aplicação corresponde a uma atividade
- Cada atividade possui elementos visuais (por exemplo, um layout, componentes visuais e etc) e comportamento (por exemplo, mostrar uma caixa de diálogo quando um botão for tocado)

# Formas de Construção de Activities

- É possível utilizar duas abordagens para criação de interface de usuário:
  - Layout em XML;
  - Programática em Java;

# Diferenças entre XML e Java

- Qual a diferença?
- Por que não usar só uma abordagem?
  - O XML é utilizado para construir o layout
  - O Java trata as ações e eventos da interface
- O Java sempre é necessário, o XML, não (embora seja a abordagem mais comum)

# XML e Java

- Desta forma, utilizaremos a abordagem de XML, onde todas as telas (atividades) do Android possuem um XML que representa o aspecto visual da atividade e a classe Java vai utilizar este XML para carregar a interface e ficará responsável por todo o comportamento da aplicação



# XML e Java

## Java

Cria a interface, baseada na leitura de um XML (a classe Java “infla o XML”) ou na especificação de cada item que representa a interface; Trata os eventos dos botões e controla a interface



## XML

Especifica como a interface deverá ser construída

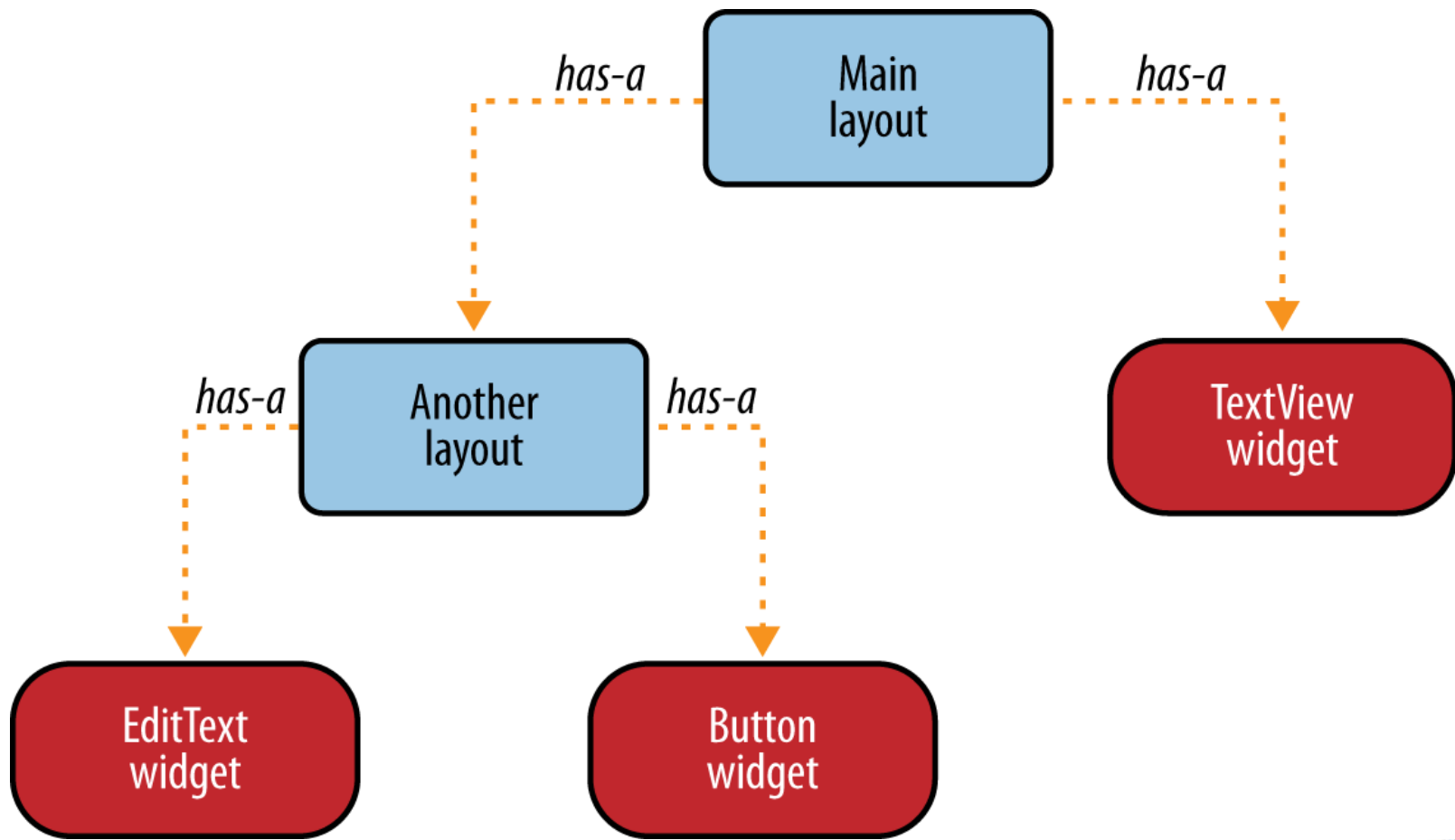
# Elementos da interface

- Android organiza a interface em dois elementos:
  - Layouts
  - Views

# Diferenças entre view e layout

- O que são layouts?
  - Layouts organizam views
  - Agrupam os elementos em uma determinada disposição
  - Podem conter outros layouts
- O que são views?
  - Tudo que você vê na interface de usuário é uma view
  - Exemplos?

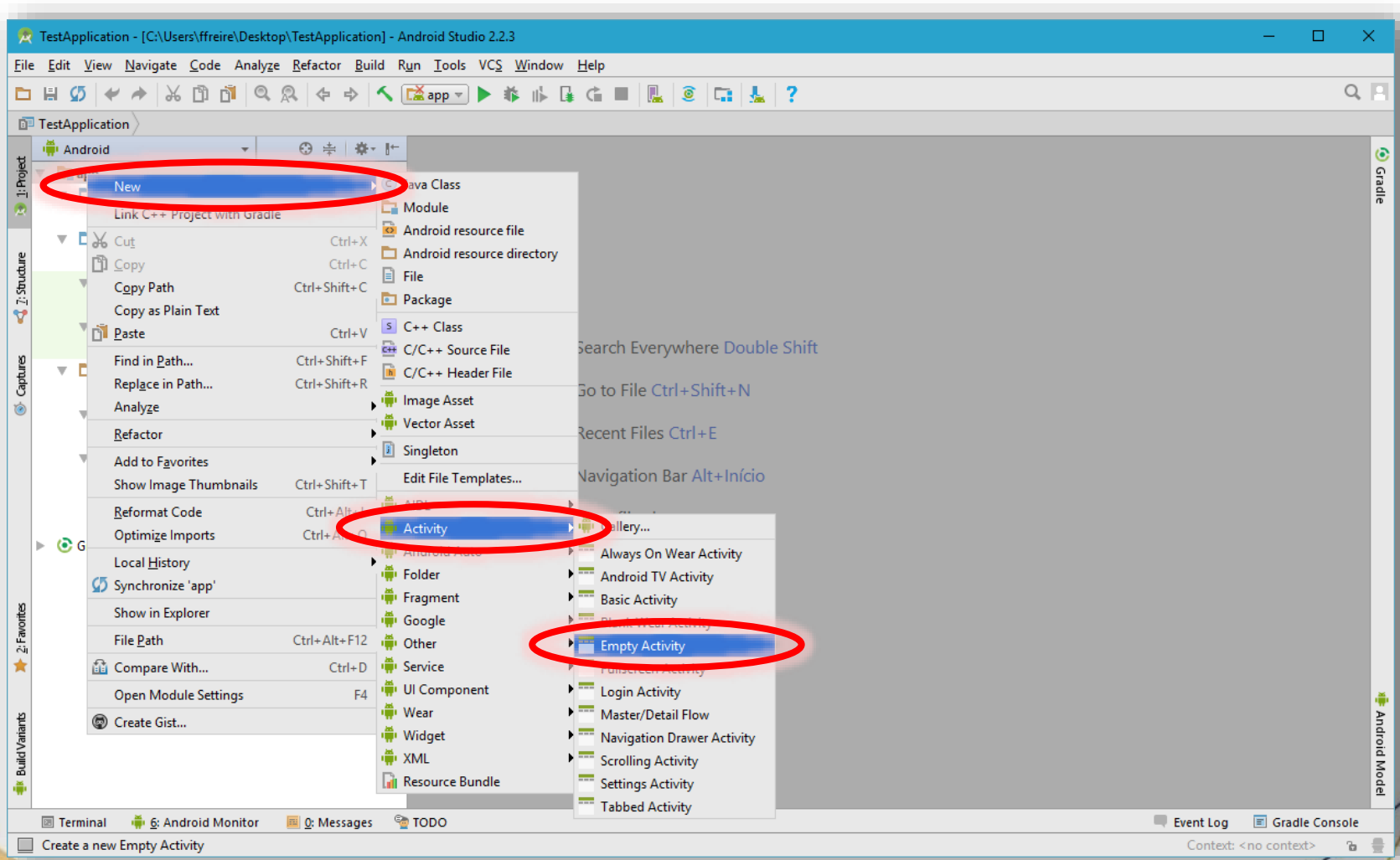
# Views x Layouts



# Criando uma Atividade

- Para a criar uma atividade, clique com o botão direito no projeto e escolha as opções “New” > “Activity” > “Empty Activity”;

# Conclusão da Criação do Projeto



# Assistente de Criação de Atividades

- No assistente que surgir, é possível configurar as opções da atividade a ser criada, como o seu nome, se será criado um XML de layout, o nome do XML de layout e o pacote onde a classe da atividade será criada;
- Também é possível especificar se esta atividade aparece na tela inicial do usuário (“launcher”). Se não marcado, só será possível iniciar a atividade de outra atividade.


# Assistente de Criação de Atividades

- Em “Activity Name”, digite “TestActivity”;
- Deixe marcada a opção “Generate Layout File” para que um arquivo de layout XML seja gerado;
- Em “Layout Name”, deixe o nome sugerido pelo Android Studio (“activity\_test”);
- Marque a opção “Launcher Activity”, para que esta atividade aparece na tela inicial e possa ser aberta;
- Em pacote, deixe como sugerido pelo Android Studio;
- Deixe as demais opções como estão;
- Clique em “Finish” para que a atividade seja criada.




# Conclusão da Criação do Projeto

New Android Activity

 **Configure Activity**  
Android Studio

Creates a new empty activity



Activity Name:

☒ Generate Layout File

Layout Name:

☒ Launcher Activity

☒ Backwards Compatibility (AppCompat)

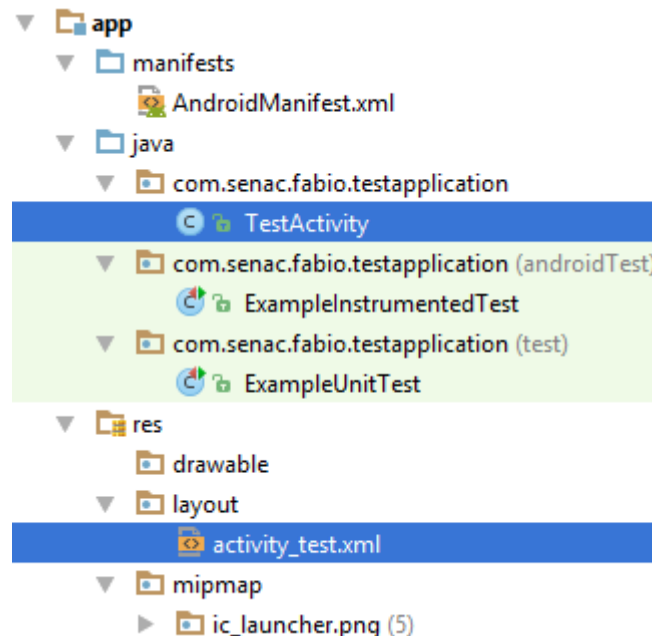
Package name:

Target Source Set:

If true, this activity will have a CATEGORY\_LAUNCHER intent filter, making it visible in the launch

# Elementos da Atividade

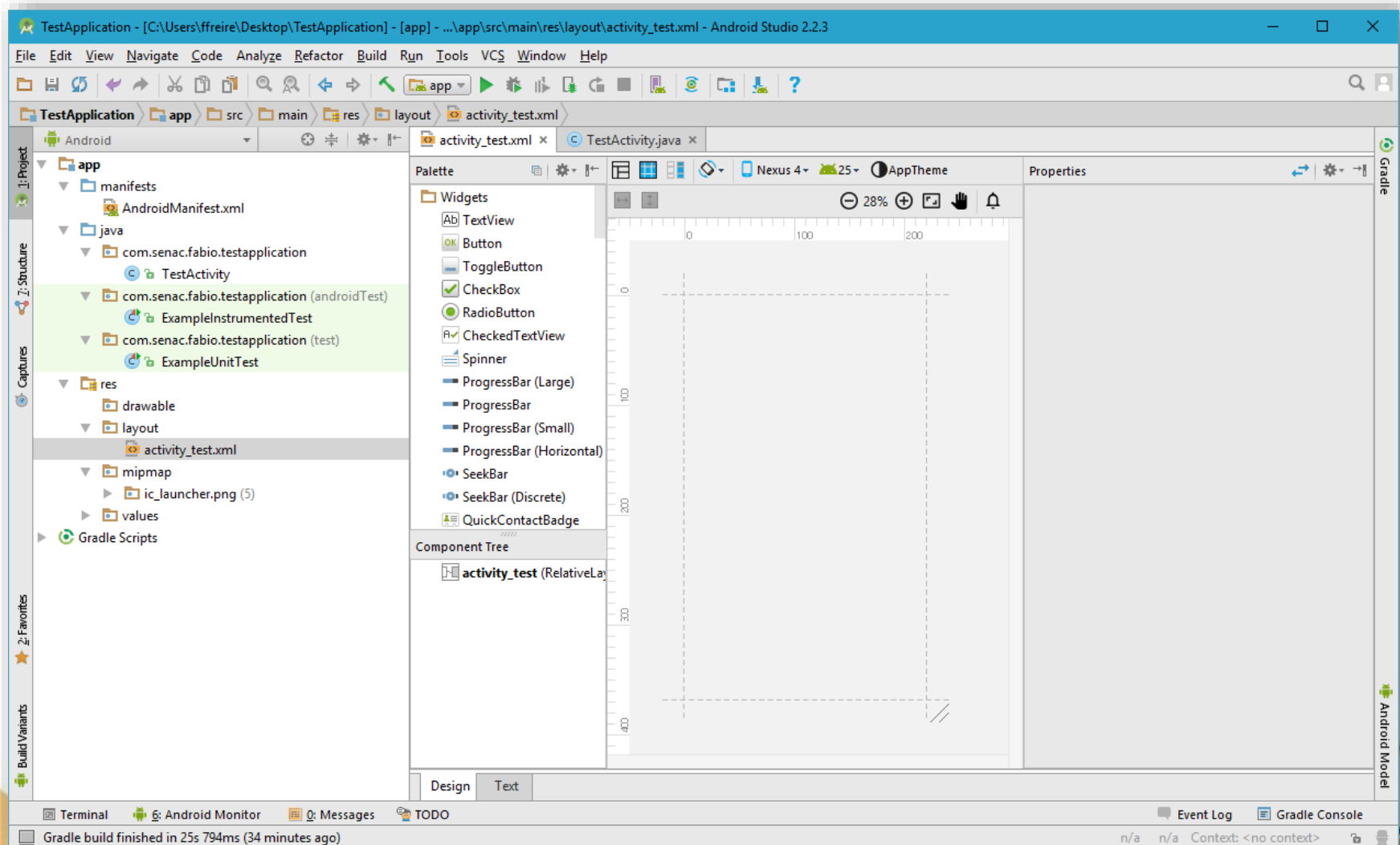
- Após a conclusão do assistente, serão gerados dois novos elementos, a classe da atividade (“TestActivity.java”), em “app” > “java” e o XML do layout, em “res” > “layout”



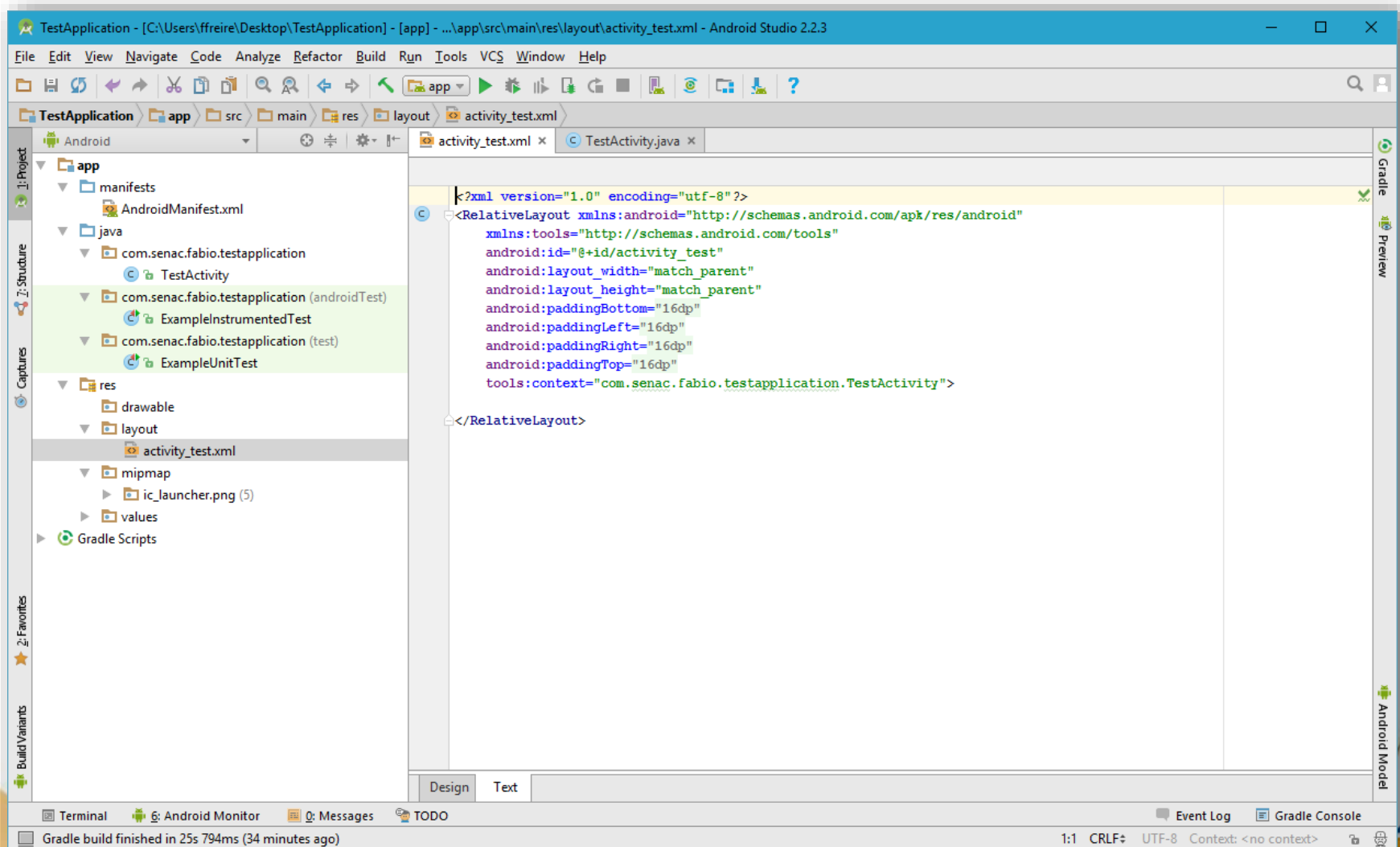
# XML de Layout

- Clicando duas vezes no XML da atividade, o editor visual de layouts do Android Studio será aberto;
- É possível visualizar e editar o visual e o código XML gerado pelo editor, alternando entre as abas no canto inferior esquerdo (“Design” e “Text”).

# XML de Layout - Design



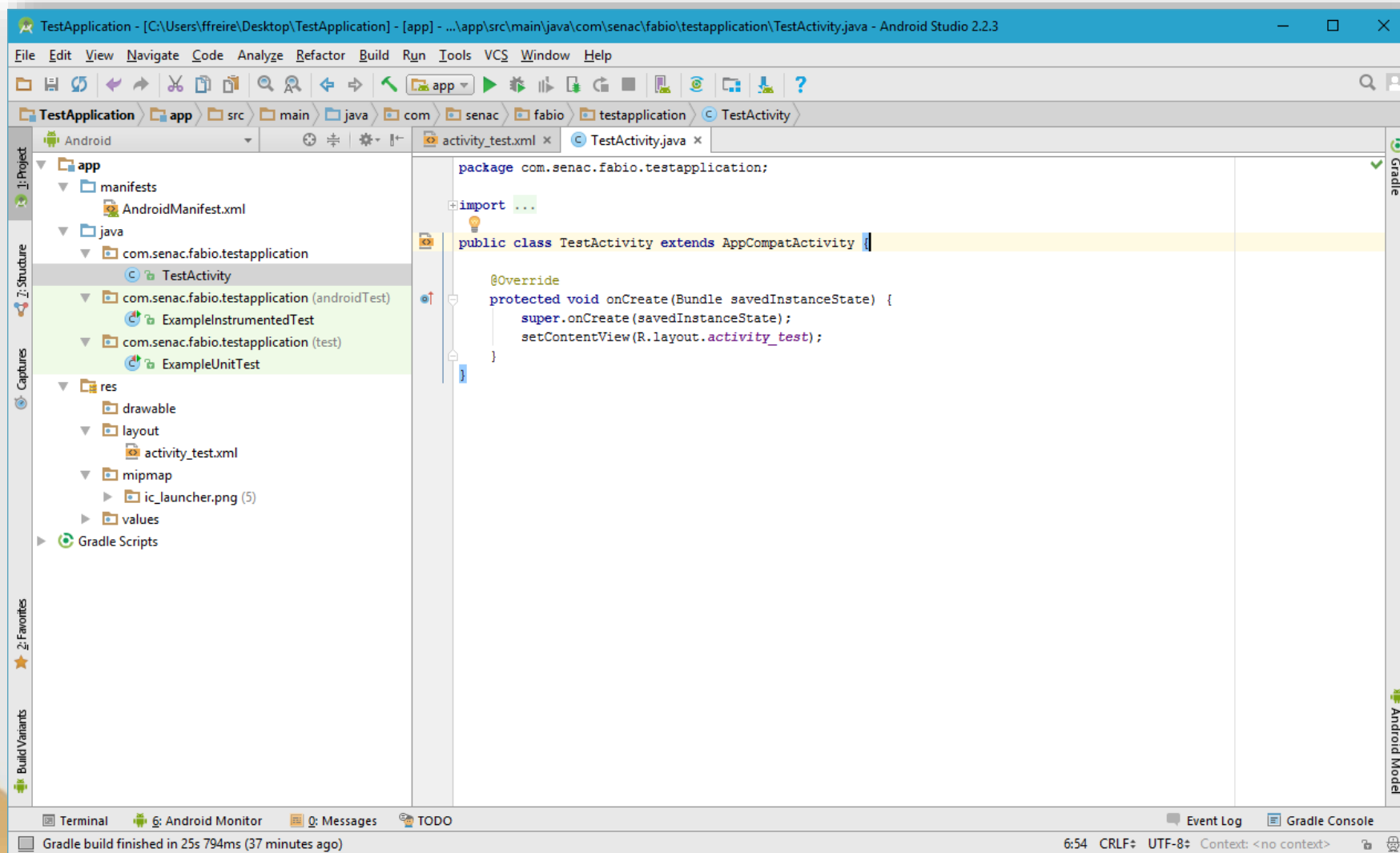
# XML de Layout - Texto



# Classe da Atividade

- Clicando duas vezes na classe da atividade, o editor de código Java do Android Studio será aberto e exibirá o código-fonte da atividade;
- A atividade vem com o método de callback “onCreate” implementado por padrão;

# Classe da Atividade



# Classe da Atividade

```
public class TestActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_test);  
    }  
}
```

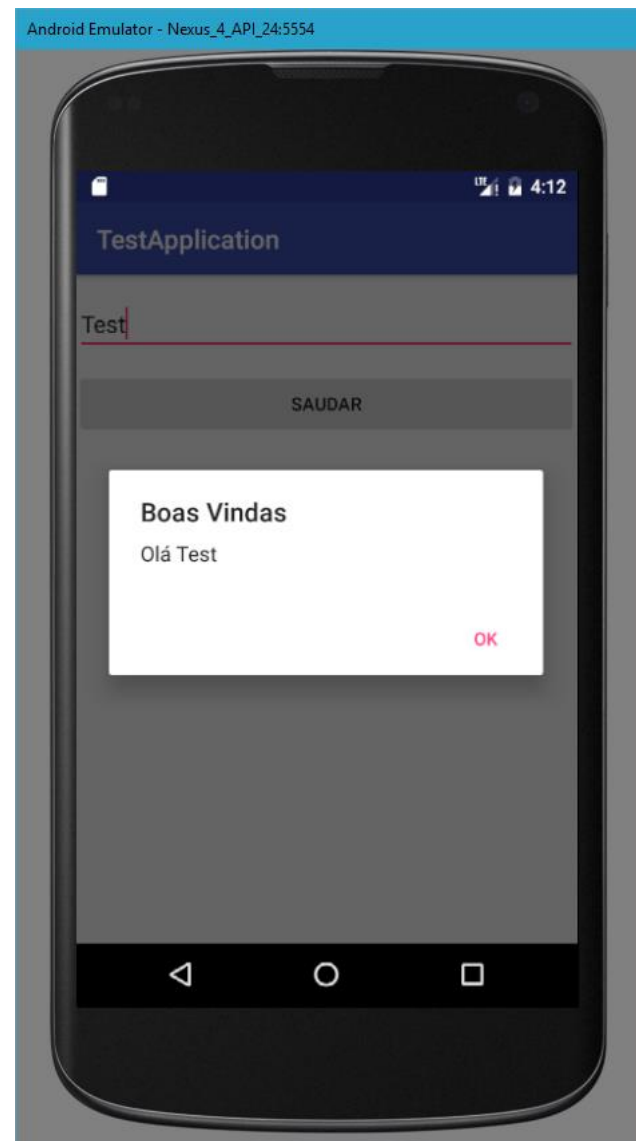


# INSERINDO VIEWS E TRATANDO EVENTOS

# Tela de Boas-Vindas

- Vamos criar uma composição de tela e uma aplicação simples, capaz de exibir uma mensagem de boas-vindas ao usuário quando este toca no botão (view) correspondente;
- O resultado final deve ser como a seguir.

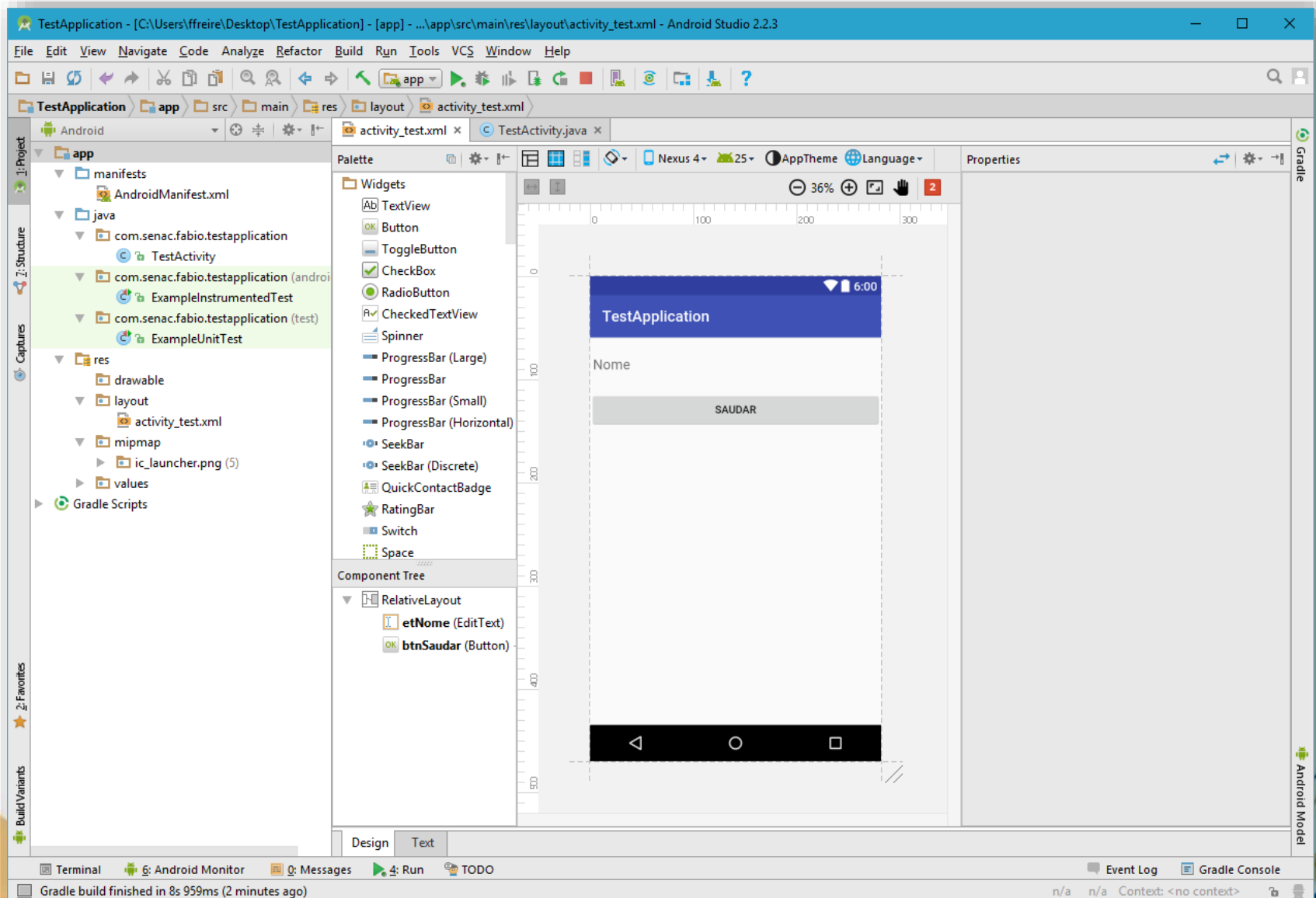
# Tela de Boas-Vindas



# Criando o Layout

- Arraste as views da paleta de views e posicione-as no layout de forma a compor o visual da atividade conforme o exemplo a seguir;
- Utilize os componentes “**Plain Text**” de “Text Fields” e “**Button**” de “Widgets”.

# Layout da Tela de Boas-Vindas



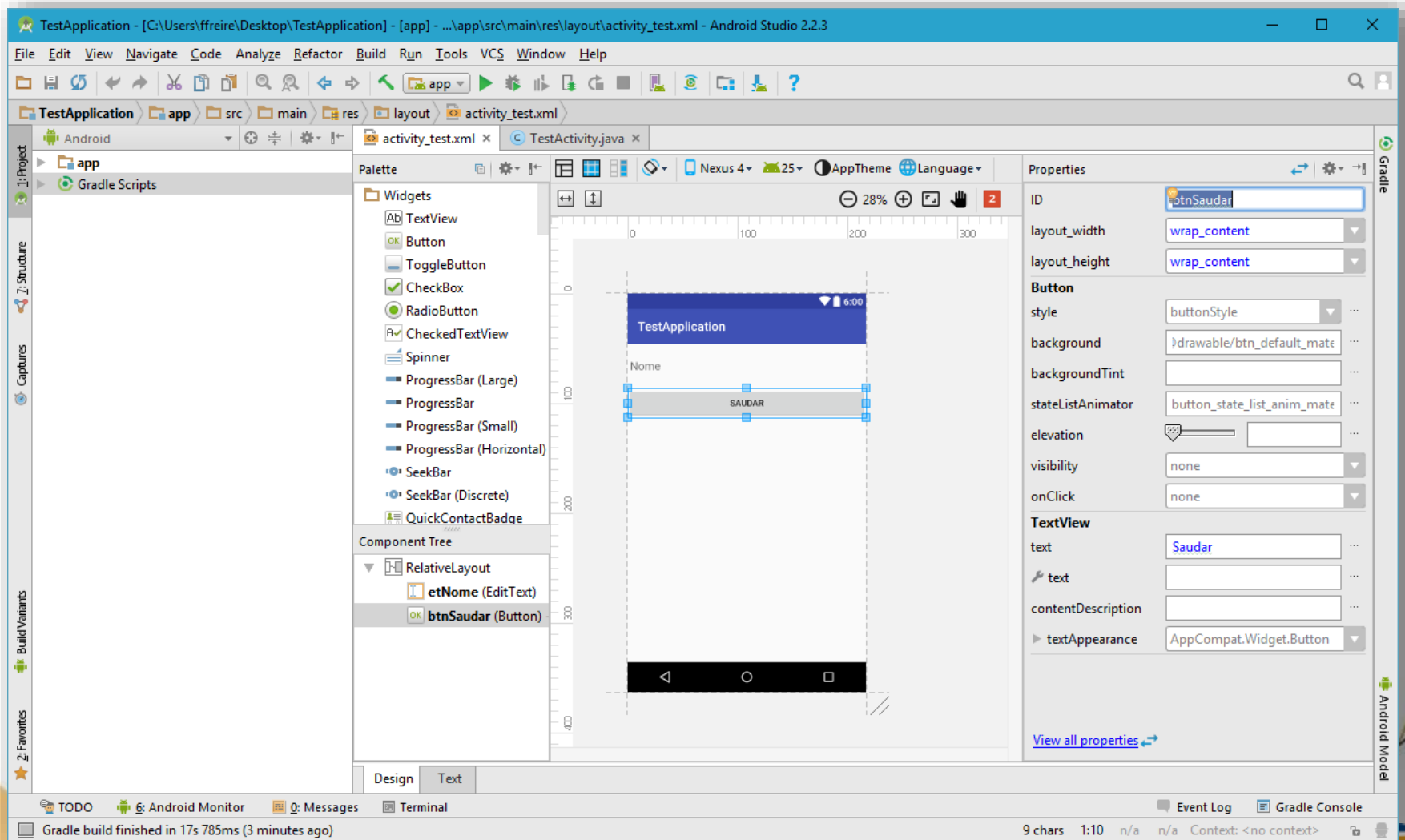
# ID dos Componentes

- Atribua um ID para os componentes criados;
- Um ID é fundamental para acessar os componentes na classe Java;
- Para atribuir um ID, selecione os componentes no editor visual e, na view “Properties”, altere o valor do campo “ID”.

# ID dos Componentes

- Altere o ID da caixa de texto para “etNome” (ID composto pela abreviação das palavras “Edit Text Nome”);
- Altere o ID do botão para “btnSaudar” (ID composto pela abreviação das palavras “Button Saudar”).

# ID dos Componentes





# Referência aos Componentes

- É necessário adicionar variáveis para representar os componentes na classe Java. Além disso, será preciso associar cada variável ao elemento do XML (processo conhecido como “binding”);

# Referência aos Componentes

- Para isso, primeiramente, adicione dois atributos a classe, um para representar o campo de texto e outro para representar o botão;
- O atributo do campo de texto deve ser do tipo “EditText” e se chamar “etNome”;
- O do botão deve ser do tipo “Button” e se chamar “btnSaudar”.

# Referência aos Componentes

```
public class TestActivity extends AppCompatActivity {  
    //Representa o componente de campo de texto do nome  
    EditText nome;  
    //Representa o botão de saudação  
    Button btn;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_test);  
    }  
}
```

# Referência aos Componentes

- Em seguida, associe as variáveis dos componentes aos componentes no layout;
- Para isso, utilize o método da atividade “findViewById” e a classe “R” para encontrar os elementos pelo ID;
- Será necessário fazer um “casting” para os tipos específicos.

# Referência aos Componentes

```
public class TestActivity extends AppCompatActivity {  
    //Representa o componente de campo de texto do nome  
    EditText nome;  
    //Representa o botão de saudação  
    Button btn;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_test);  
  
        //Binding dos componentes com os elementos do layout  
        nome = (EditText) findViewById(R.id.etNome);  
        btn = (Button) findViewById(R.id.btnSaudar);  
    }  
}
```

# Ação do Botão (Listener)

- Agora, vamos atribuir uma ação ao botão;
- Isto pode ser feito através de um listener;
- Um listener é associado a uma view (componente visual) e é chamado quando ocorre um evento (por exemplo, toque no botão);
- É possível criar um listener de clique (toque) no android através da classe (View.OnClickListener).

# Listener do Botão

...

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

    //Binding dos componentes com os elementos do layout
    nome = (EditText) findViewById(R.id.etNome);
    btn = (Button) findViewById(R.id.btnSaudar);

    //Define um listener de ação
    View.OnClickListener listener = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //Ação aqui!
        }
    };

    //Associa o listener de ação com o botão
    btn.setOnClickListener(listener);
}
```

...

# Exibir um Diálogo

- Vamos criar um método para exibição de um diálogo;
- O método visa parametrizar a criação, facilitando chamadas em outros locais para criação de diálogos;
- Pode ser externalizado para uma classe utilitária e usado em mais locais (em toda a aplicação);
- Siga os comandos a seguir para construir um método que mostra um diálogo (crie o método “showMessage” logo abaixo do método “onCreate”)



# Método para Exibição de Diálogo

```
...  
private void showDialog(String message, String title) {  
    //Declara e instancia uma fábrica de construção de diálogos  
    AlertDialog.Builder builder = new AlertDialog.Builder(TestActivity.this);  
    //Configura o corpo da mensagem  
    builder.setMessage(message);  
    //Configura o título da mensagem  
    builder.setTitle(title);  
    //Impede que o botão seja cancelável (possa clicar  
    //em voltar ou fora para fechar)  
    builder.setCancelable(false);  
    //Configura um botão de OK para fechamento (um  
    //outro listener pode ser configurado no lugar do "null")  
    builder.setPositiveButton("OK", null);  
    //Cria efetivamente o diálogo  
    AlertDialog dialog = builder.create();  
    //Faz com que o diálogo apareça na tela  
    dialog.show();  
}  
...
```

# Exibindo o Diálogo no Listener

- Por fim, faça a chamada do método para exibição do diálogo no Listener;
- Para montar a mensagem a ser exibida no diálogo, obtenha o valor da caixa de texto, através do método “getText”.

# Método para Exibição de Diálogo

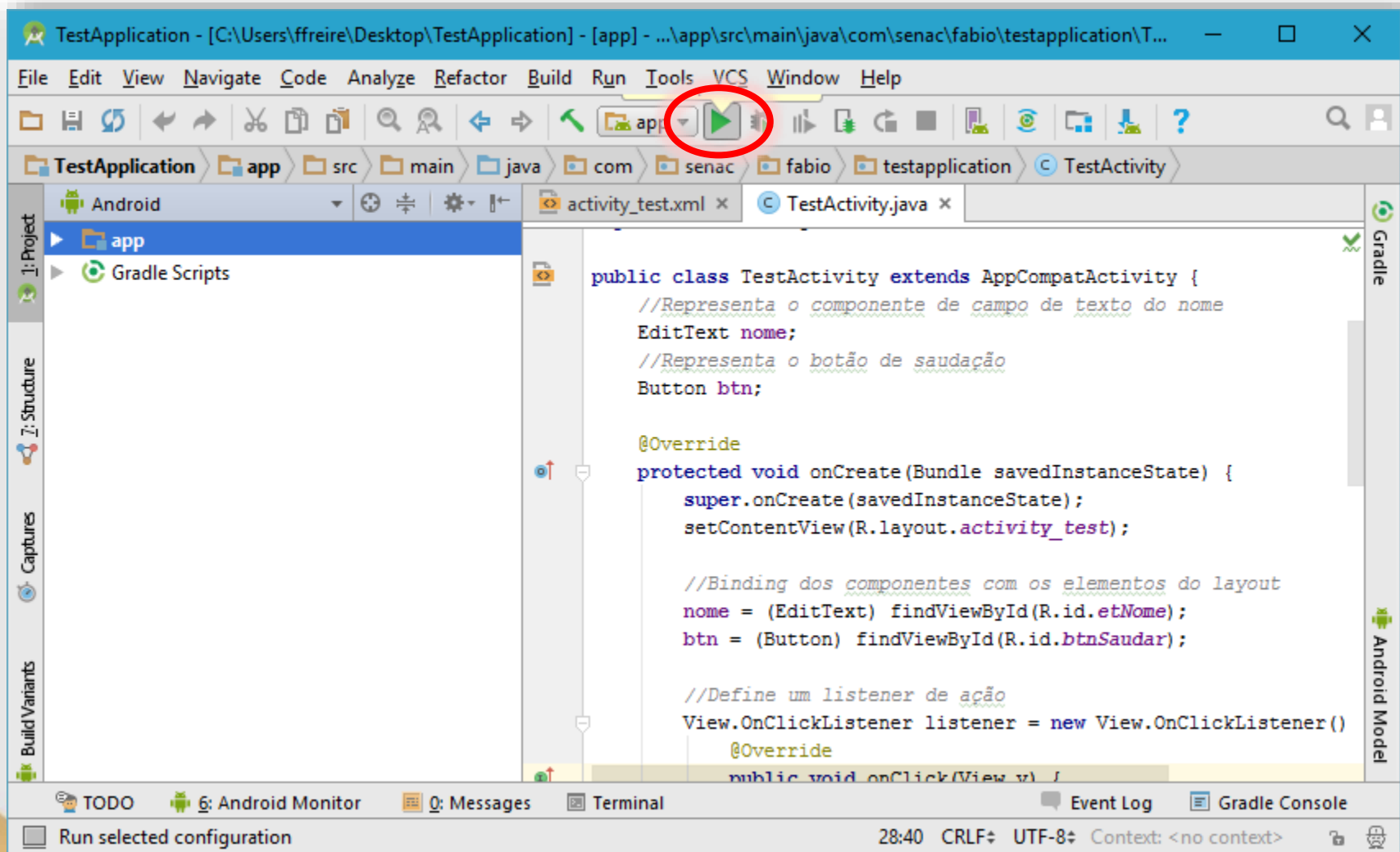
...

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_test);  
  
    //Binding dos componentes com os elementos do layout  
    nome = (EditText) findViewById(R.id.etNome);  
    btn = (Button) findViewById(R.id.btnSaudar);  
  
    //Define um listener de ação  
    View.OnClickListener listener = new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            //Ação aqui!  
            showDialog("Olá " + nome.getText().toString(), "Boas Vindas");  
        }  
    };  
  
    //Associa o listener de ação com o botão  
    btn.setOnClickListener(listener);  
}  
...
```

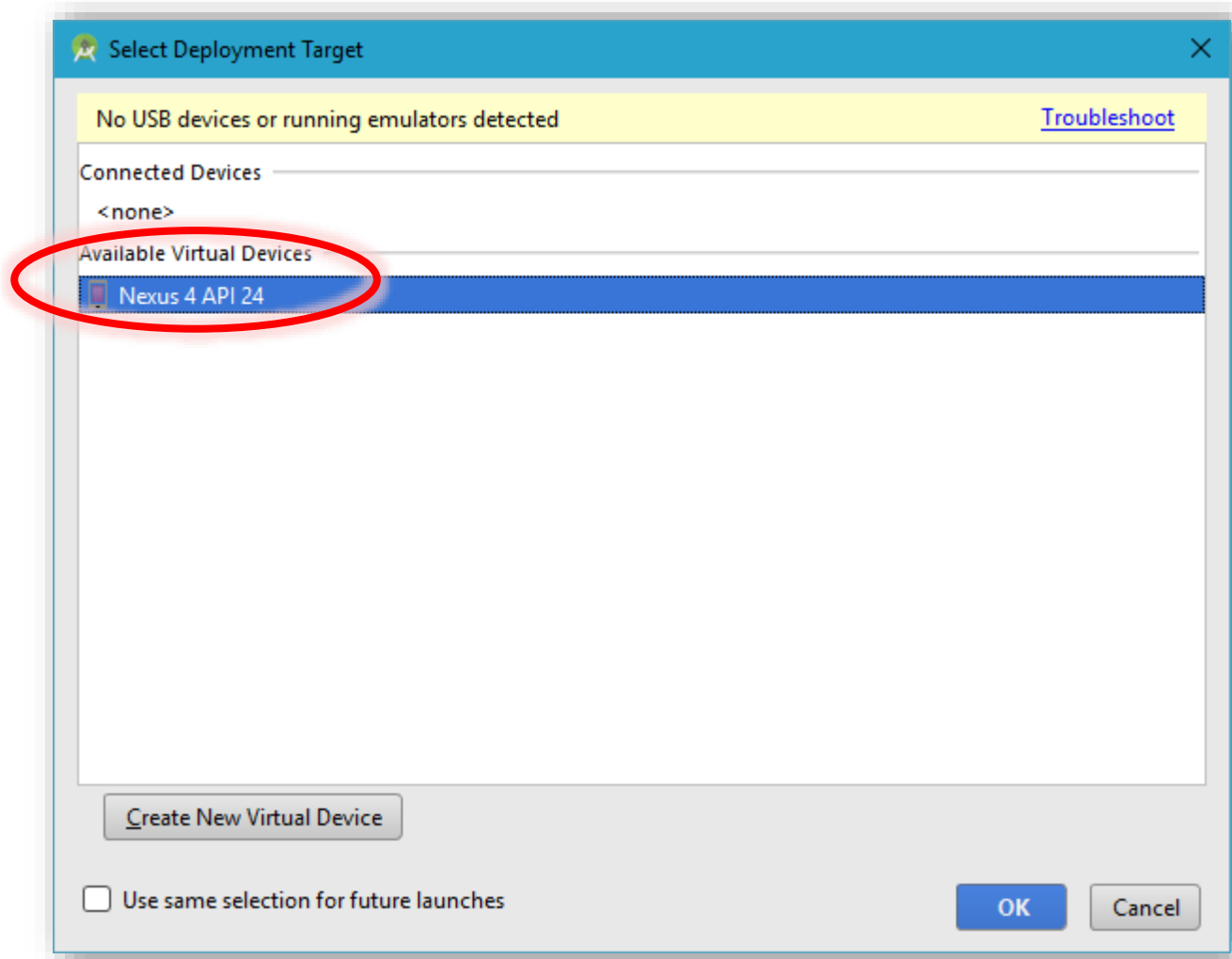
# Executando a Aplicação

- Execute a aplicação para testá-la, clicando no botão “Play”;
- Selecione o dispositivo conectado ou o emulador na lista que surgir;

# Executando a Aplicação



# Executando a Aplicação



# SOBRE ATIVIDADES

# Classes de atividades

- O framework do Android provê superclasses para os principais blocos de construção
- Para criar uma atividade em Java, estendemos normalmente de:
- **android.app.Activity** ou
- **android.support.v7.app.AppCompatActivity** (recomendado)



# Tarefas das classes de atividades

- Principais tarefas da atividade:
  - “Inflar” o layout XML
  - Inicializar as views como objetos Java
  - Lidar com eventos da interface

# Passos para criação de uma atividade

- Sobrescrever o método “onCreate()”
  - Sempre que um método de uma superclasse do framework Android for sobrescrito, chamar o mesmo método da classe pai através do “super”, nesse caso, “super.onCreate()”

# Passos para criação de uma atividade

- Sobrescrever o método “onCreate()”
  - O método “onCreate()” recebe como parâmetro um Bundle, que armazena uma pequena quantidade de dados
  - Esses dados são passados quando a atividade está sendo encerrada, de modo que, quando a atividade for reiniciada, seja possível recriar seu estado anterior

# Passos para criação de uma atividade

- Inflar o layout XML
  - No “onCreate()”, chamar o método “setContentView()”, informando qual layout XML deve ser “inflado”
  - Todos os layouts criados são referenciados na classe de recursos da aplicação, a “R”
  - R.layout.<nome\_do\_arquivo\_xml>

# Exemplo de Atividade

```
import android.os.Bundle;
import android.app.Activity;

public class MyActivity extends Activity {

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

## Para inicialização de views como objetos Java

- É preciso encontrar as views através do id e “liga-las” a objetos Java correspondentes, processo conhecido como “binding”
- Todos os ids de todos os elementos criados são referenciados através da classe especial de recursos do Android (“R”)
  - `R.id.<id_do_elemento>`

# Exemplo de Referência a Elementos

```
import android.os.Bundle;
import android.app.Activity;
import android.widget.Button;
import android.widget.EditText;

public class MyActivity extends Activity {

    private EditText editField;
    private Button myButton;

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

        editField = (EditText) findViewById(R.id.editField);
        myButton = (Button) findViewById(R.id.myButton);
    }
}
```

# Tratando eventos

- Para lidar com eventos da interface gráfica:
- Exemplos de eventos?
- É necessário implementar uma interface Java do framework Android para cada evento (listeners)



# Formas de Tratamento de eventos

- A sua própria atividade implementa a interface Java e o método correspondente ao evento é sobrescrito
- Implementação da interface Java através de uma definição de classe anônima, isto é, “em linha”
  - É a mais utilizada (e o que utilizamos)

# Exemplo de Tratamento de Eventos

```
public class MyActivity extends Activity {  
    private Button myButton;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_my);  
  
        myButton = (Button) findViewById(R.id.myButton);  
        myButton.setOnClickListener(  
            new View.OnClickListener() {  
                @Override  
                public void onClick(View view) {  
                    //Faz alguma coisa  
                }  
            }  
        );  
    }  
}
```

# Acionando caixas de diálogo

- Para exibir uma caixa de diálogo de resposta ou confirmação, é necessário construí-la, através da “fábrica” `AlertDialog.Builder`;
- É possível customizar o título, a mensagem, os botões e seus eventos;
- Quando pronto, basta chamar a função `create` da fábrica e o método `show` do diálogo.

# Exemplo de Tratamento de Eventos

```
buttonShowMessage.setOnClickListener(  
    new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            AlertDialog.Builder builder = new  
AlertDialog.Builder(MainActivity.this);  
  
            builder.setMessage("Hello, " + editTextName.getText());  
            builder.setTitle("Hello");  
            builder.setCancelable(false);  
            builder.setPositiveButton("OK", null);  
  
            AlertDialog dialog = builder.create();  
            dialog.show();  
        }  
    }  
);
```

# SOBRE LAYOUTS E RECURSOS

# Principais tipos de layouts

- **LinearLayout**
  - Agrupa as views uma após a outra, vertical ou horizontalmente
- **TableLayout**
  - Divide a tela em linhas (TableRow), cada uma agrupando as views uma ao lado da outra
- **FrameLayout**
  - Adiciona uma view em cima da outra
- **RelativeLayout**
  - Cada view é posicionada de forma relativa às outras (à esquerda, à direita, acima, abaixo, etc.)

# Linear Layout

## Linear Layout

Orientation: vertical


Orientation: horizontal

--	--	--	--	--	--

# Linear Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:orientation="vertical">

    <Button android:id="@+id/button2" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="Button 01" />
    <Button android:id="@+id/button3" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="Button 02" />
    <Button android:id="@+id/button4" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="Button 03" />
    <Button android:id="@+id/button5" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="Button 04" />
    <Button android:id="@+id/button6" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="Button 05" />
    <Button android:id="@+id/button7" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="Button 06" />
    <Button android:id="@+id/button8" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:text="Button 07" />

</LinearLayout>
```



# Linear Layout



# Table Layout

<TableLayout>

Row 1		
Row 2 column 1	Row 2 column 2	Row 2 column 3
Row 3 column 1	Row 3 column 2	

</TableLayout>

# Table Layout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:shrinkColumns="*" android:stretchColumns="*" android:background="#ffffff">

    <TableRow android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:gravity="center_horizontal">
        <Button android:id="@+id/button1" android:layout_width="match_parent"
            android:layout_height="wrap_content" android:text="Button 01" />
        <Button android:id="@+id/button2" android:layout_width="match_parent"
            android:layout_height="wrap_content" android:text="Button 02" />
        <Button android:id="@+id/button3" android:layout_width="match_parent"
            android:layout_height="wrap_content" android:text="Button 03" />
    </TableRow>

    <TableRow android:id="@+id/tableRow1" android:layout_height="wrap_content"
        android:layout_width="match_parent">
        <Button android:id="@+id/button4" android:layout_width="match_parent"
            android:layout_height="wrap_content" android:text="Button 04" />
    </TableRow>

    <TableRow android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:gravity="center_horizontal">
        <Button android:id="@+id/button5" android:layout_width="match_parent"
            android:layout_height="wrap_content" android:text="Button 05" />
    </TableRow>
</TableLayout>
```

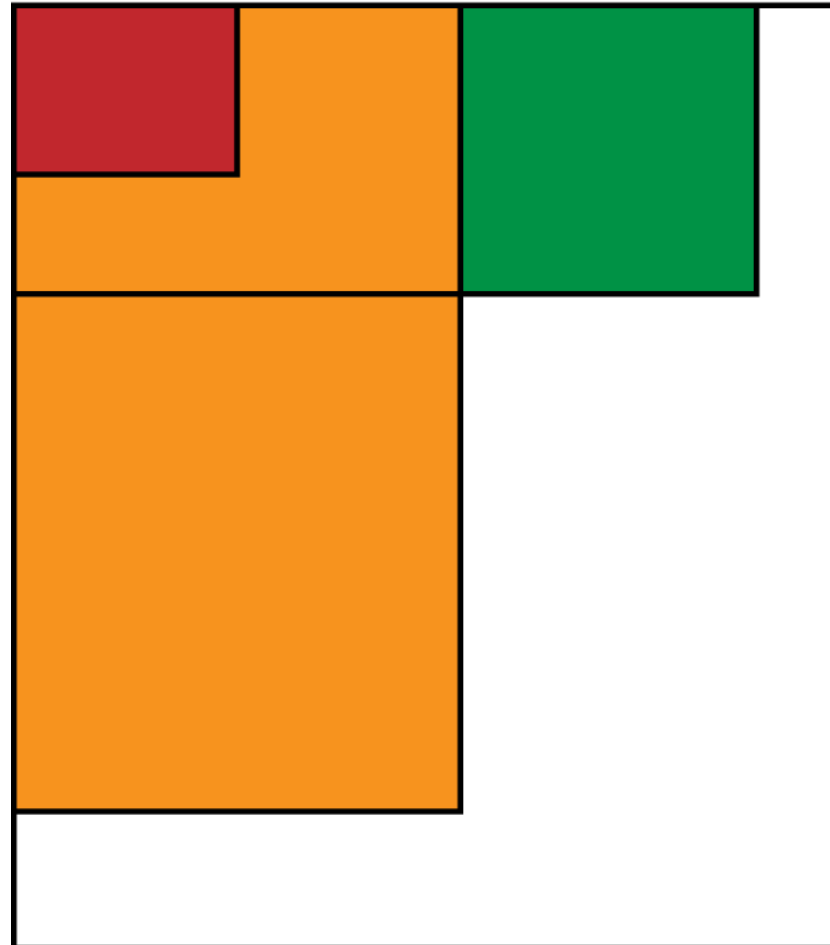


# Table Layout



# Frame Layout

Frame Layout



# Frame Layout

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#D6FFD6">
    <ImageView
        android:src="@drawable/android"
        android:scaleType="fitCenter"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"/>
    <TextView
        android:text="TEXT0 TESTE AULA 03"
        android:textSize="30sp"
        android:textStyle="bold"
        android:textColor="#003399"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        android:gravity="center"/>
</FrameLayout>
```

# Frame Layout



# Relative Layout

Relative Layout

id=F toLeftOf E above D	id=E center_horizontal ParentTop	id=G toRightOf E above B
id=D center_vertical ParentLeft	id=A Center	id=B center_vertical ParentRight
id=I toLeftOf C below D	id=C center_horizontal ParentBottom	id=H toRightOf C below B



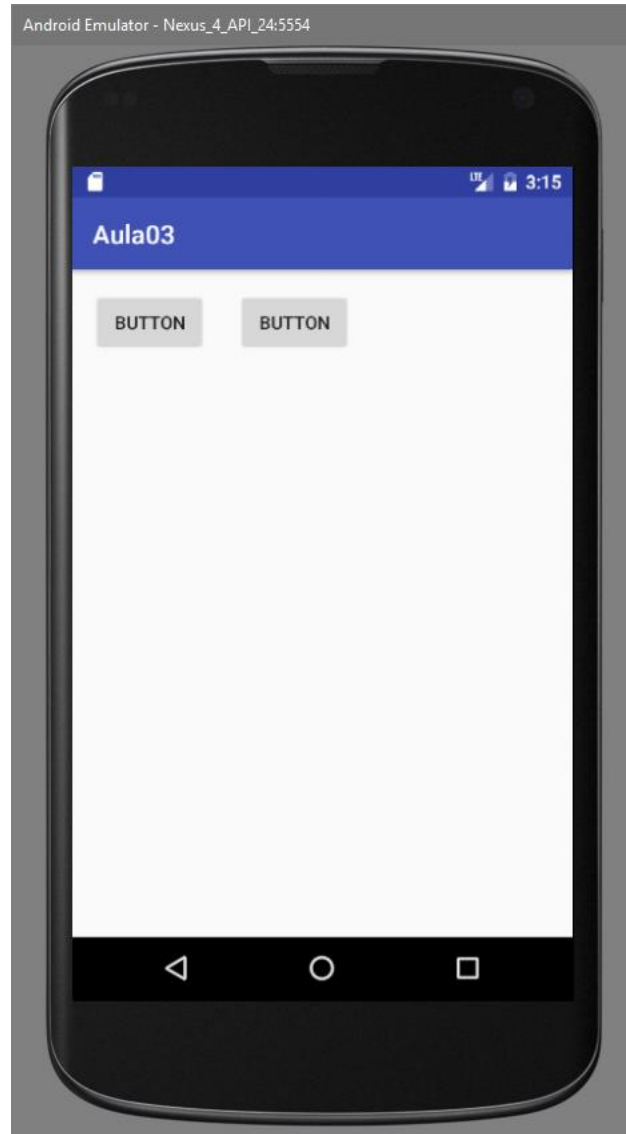
# Relative Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.fabio.aula03.MainActivity">

    <Button
        android:text="Button" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" android:id="@+id/button" />

    <Button
        android:text="Button" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/button2"
        android:layout_marginStart="23dp" android:layout_alignParentTop="true"
        android:layout_toEndOf="@+id/button" />
</RelativeLayout>
```

# Relative Layout



# Recursos de Layouts

- Os layouts criados em XML ficam no diretório “res/layout”
- Layouts em formatos específicos ou alternativos ficam em diretórios com nomes específicos

# Layouts para tamanhos de telas

- Para tamanhos de tela diferentes:
  - “res/layout-small” (426 dp x 320 dp)
  - “res/layout-normal” (470 dp x 320 dp)
  - “res/layout-large” (640 dp x 480 dp)
  - “res/layout-xlarge” (960 dp x 720 dp)

# Layouts de orientação de tela

- Para orientação específica:
  - “res/layout-land”
  - “res/layout-port”

# Layouts combinados

- É possível combinar os dois:
  - “res/layout-xlarge-land”
- Os diretórios são gerenciados pelo Android Studio

# Propriedades de views e layouts

- id: é o identificador único do elemento, sendo utilizado para recuperá-lo no Java
- text: os elementos que possuem texto podem ser definidos através dessa propriedade

# Propriedades de views e layouts

- `layout_gravity`: indica como o elemento vai ser posicionado vertical e horizontalmente, possuem alguns valores pré-definidos, como `top`, `center`, `left`, etc.
  - Não funciona em todos os tipos de layout
- `gravity`: indica como o conteúdo dentro do elemento será posicionado



# Propriedades de views e layouts

- `layout_height` e `layout_width`: definem a altura e largura, possuem dois valores pré-definidos
  - `match_parent`: preenche todo o espaço disponível no pai
  - `wrap_content`: ocupa somente o espaço suficiente para exibi-lo

# Outras propriedades

- `paddingBottom`, `paddingLeft`, `paddingRight` e `paddingTop`
- `layout_marginBottom`, `layout_marginLeft`, `layout_marginRight` e `layout_marginTop`
- `hint`
- `inputType`

# Recursos de texto

- Os textos fixos do layout podem ser colocados no arquivo “strings.xml”
  - As strings possuem um identificador e um valor
    - O identificador é utilizado no layout da seguinte forma:  
`android:text="@string/identificador”`
    - O identificador será substituído pelo valor na hora de exibir a tela

# Recursos de texto

- Localiza-se em “res/values/strings.xml”
- Para outros idiomas, nomear o diretório “values” da seguinte forma:
  - “res/values-es/strings.xml”: para espanhol
  - “res/values-fr/strings.xml”: para francês
  - E assim por diante
  - O Android Studio gerencia os diretórios

# Formato do arquivo de strings

- O formato do arquivo é:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="title">My Application</string>
  <string name="hello_world">Hello world!</string>
</resources>
```

- No diretório “values-es”, o arquivo ficaria:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="title">Mi Aplicación</string>
  <string name="hello_world">Hola Mundo!</string>
</resources>
```

- E no “values-fr”, ficaria:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="title">Mon Application</string>
  <string name="hello_world">Bonjour le monde !</string>
</resources>
```

# Recursos de estilos

- Também é possível separar os estilos aplicados aos elementos no arquivo “styles.xml”
- É possível replicar os estilos em diversos elementos

# BOAS PRÁTICAS

# Boas práticas

- Dê um nome significativo para sua atividade, sempre seguido de “Activity”
- Por exemplo, “ProdutoActivity” se for uma atividade para exibir informações de produto
- Deixe que o Android Studio renomeie os demais arquivos (XMLs)



# Boas práticas

- Ao adicionar elementos (views) no XML do layout, sempre dê id's significativos
- Muito útil para recuperá-los no Java depois

# Boas práticas

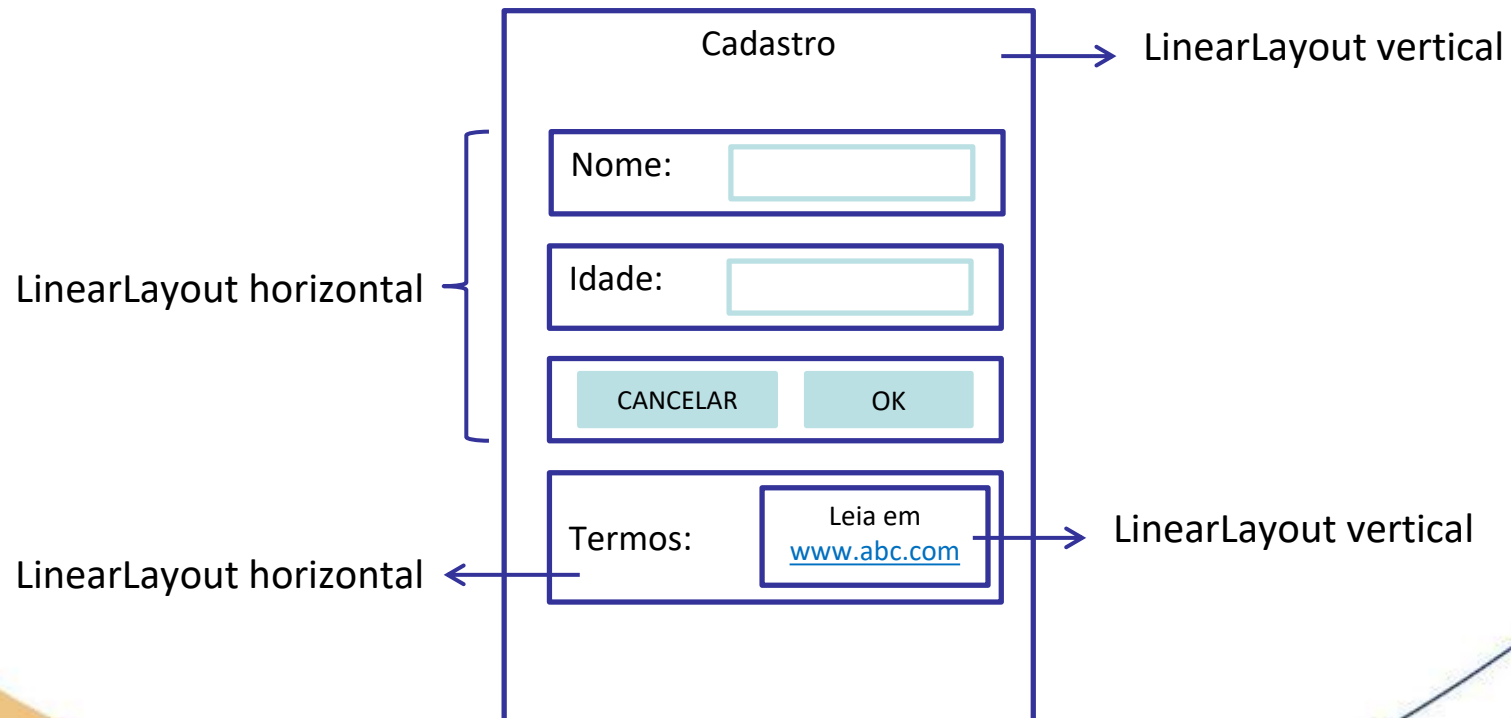
- Por padrão, o Android Studio coloca o nome da view seguido de um número
- Por exemplo, “editText”, “editText2”, “textView”, etc.
- Isso é péssimo!

# Boas práticas

- Para recuperar o texto digitado pelo usuário em um editText, é muito mais fácil se você souber o que ele representa
- Por exemplo, dar o id “sobrenome” no editText usado pelo usuário para digitar seu sobrenome

# Boas práticas

- Construa um layout utilizando o LinearLayout
- Componha vários LinearLayouts para montar a interface do jeito que você quiser



*"That's all Folks!"*