

PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

APRESENTAÇÃO

Jadir Custódio Mendonça Junior

Conteúdo

- Introdução ao Android
- Configuração do Ambiente
- Conceitos Básicos do Android
- Classes Principais do Android (Activity e Intent)
- Interface Gráfica (Views)
- Interface Gráfica (Gerenciadores de Layout)
- HTTP, Sockets e WebServices
- Armazenamento de dados - Manipulação de Arquivos Textos e Arquivos de Preferência
- Armazenamento de dados - Banco de Dados

Conteúdo

3

❑ Introdução ao Android:

- ❑ Sistema operacional Linux;
- ❑ Máquina Virtual Dalvik;
- ❑ Código aberto livre;
- ❑ Desenvolvimento de aplicações com Android Studio e Java.

❑ Configuração do Ambiente:

- ❑ Android SDK;
- ❑ Requisitos de software e sistema;
- ❑ Instalação do Android Studio;
- ❑ Emulador do Android;
- ❑ Visualização das aplicações instaladas.

Conteúdo

4

- ❑ **Conceitos Básicos do Android:**
 - ▣ Activity, View e o método setContentView;
 - ▣ A classe R;
 - ▣ Criação de interface visual em XML;
 - ▣ Criação de interface visual com a API Java;
 - ▣ Acesso aos elementos da tela;
 - ▣ Tratamento de eventos.
- ❑ **Classes Principais do Android (Activity e Intent);**
- ❑ **Interface Gráfica (Views):**
 - ▣ Componentes TextView, EditText, Button, RadioButton, CheckBox, entre outros;
 - ▣ Definição de recursos de texto, cores, imagens e estilos.

Conteúdo

5

- ❑ **Interface Gráfica (Gerenciadores de Layout):**
 - ▣ ViewGroup – A classe pai dos gerenciadores de layout;
 - ▣ Os gerenciadores LinearLayout, FrameLayout, TableLayout, RelativeLayout, AbsoluteLayout, entre outros.
- ❑ **WebServices:**
 - ▣ Conceitos de WebServices;
 - ▣ Comunicação com servidor por meio de WebServices;
 - ▣ Métodos e formas de comunicação entre cliente e servidor com utilização de XML e/ou JSON.

Conteúdo

6

- ❑ **Armazenamento de dados - Manipulação de Arquivos Textos e Arquivos de Preferência:**
 - ▣ Manipulação de arquivos textos pelo Android;
 - ▣ Manipulação de arquivos de preferências pelo Android;
 - ▣ Aplicação para leitura e gravação de dados em arquivos.
- ❑ **Armazenamento de dados - Banco de Dados:**
 - ▣ Introdução ao SQLite e Ferramentas de manipulação do SQLite;
 - ▣ Criação de banco de dados com SQLite.
 - ▣ Abertura do banco de dados;
 - ▣ Criação de Aplicação para inserção, atualização, exclusão e pesquisa de registros no banco de dados.

Avaliações

- ❑ Quiz das Bandeiras – 0,5 pontos **(03/09/24)**
- ❑ Layouts de Tela – 0,5 pontos **(03/09/24)**
- ❑ BD com ListView- 0,5 pontos **(01/10/24)**
- ❑ Controle de Presença em Eventos
 - ❑ utilizar câmera e Qrcode - 1,5 pontos **(22/10/24)**
- ❑ Projeto Tema Livre – 2,0 pontos **(06/11/24)**
 - ❑ Atender a um dos 17 Objetivos de Desenvolvimento Sustentável.
- ❑ A1 – **03/12/24**
- ❑ AF – **17/12/24**

8

Bibliografía Básica

Bibliografia Básica

9



- **Autor(a):** Diversos
- **Título:** Android para Programadores
- **Subtítulo:** Uma Abordagem Baseada Em Aplicativos
- **Editora:** Bookman
- **ISBN:** 9788540702103
- **Páginas:** 512
- **Edição:** 1ª
- **Tipo de capa:** Brochura
- **Ano:** 2012
- **Idioma:** Português
- **Citação:**
 - DEITEL, Harvey M. et al. *Android para programadores: uma abordagem baseada em aplicativos*. Porto Alegre: Bookman Companhia Ed, 2013.
- **RECOMENDADO!**

Bibliografia Básica

10



- **Autor(a):** Ricardo R. Lecheta
- **Título:** Google Android
- **Subtítulo:** aprenda a criar aplicações para dispositivos móveis com o Android SDK
- **Editora:** Novatec
- **ISBN:** 9788575223444
- **Páginas:** 824
- **Edição:** 3ª
- **Tipo de capa:** Brochura
- **Ano:** 2013
- **Idioma:** Português
- **Citação:**
 - LECHETA, Ricardo R. *Google android: aprenda a criar aplicações para dispositivos móveis com o Android SDK*. 3.ed. São Paulo: Novatec, 2013.
- **RECOMENDADO!**

Processo de Compilação

11

O que é Compilação



Processo de Compilação

12

Definição

- Conjunto de etapas necessárias para a tradução e conversão de código-fonte em linguagem de máquina.
- Etapas:
 - ▣ **Preprocessamento:** etapa em que o pré-processador (programa às vezes acoplado ao compilador) lê o código-fonte e faz algumas substituições para que o programa possa ser compilado.
 - ▣ **Verificação sintática:** etapa que procura por eventuais erros nos códigos dos programas: parênteses não fechados, falta de ponto-e-vírgula no final da instrução, etc. Todos esses problemas são alertados e causam a interrupção da compilação.
 - ▣ **Compilação propriamente dita:** etapa que transforma o código preprocessado em um programa-objeto, que está em linguagem de máquina porém não pronto para ser executado.
 - ▣ **Linkedição (linking, em inglês):** etapa em que programas-objeto e bibliotecas necessárias são linkadas transformando em um único executável, feita pelo linkeditor (linker).

Como é feito
o processo de
compilação de
um aplicativo
Android



Processo de Compilação

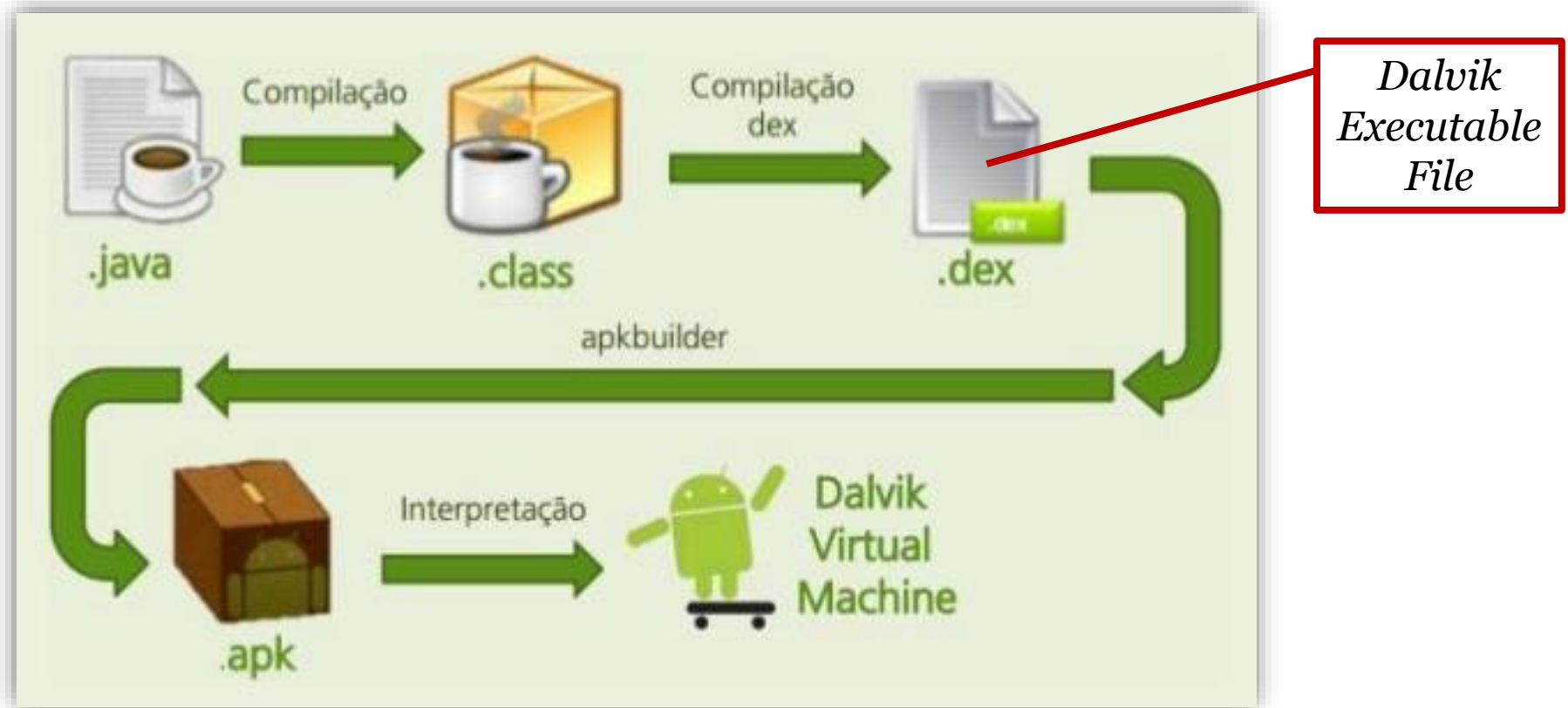
Processo de desenvolvimento e compilação Java tradicional:



- Todos os programas Java são **compilados** e **interpretados**
- O compilador transforma o programa em **bytecodes** independentes de plataforma
- O interpretador testa e executa os **bytecodes**

Processo de Compilação

Processo de desenvolvimento e compilação Android



Um aplicativo Android é um arquivo com extensão **.apk (Android Package)**, que basicamente é um pacote que contém o código compilado e os demais recursos, como XMLs e imagens.

Ferramentas de Desenvolvimento

O que eu preciso para desenvolver para o android?

- As ferramentas e programas necessários são todos gratuitos e disponíveis para todos os sistemas operacionais **(OS X, Windows e Linux)**;
- É necessário conhecer (ou aprender) a linguagem Java;
- Entre as ferramentas necessárias para o desenvolvimento estão:
 - ▣ **JDK** - *Java Development Kit*;
 - ▣ **Android SDK** - inclui as bibliotecas e várias ferramentas além do emulador;
 - ▣ **Android Studio** - a IDE que será utilizada para desenvolvimento para o **Android**.

O site do Android também disponibiliza o SDK de forma avulsa caso o desenvolvedor utilize um outro ambiente de desenvolvimento.

Acesse: <https://developer.android.com/sdk/index.html>

Sumário

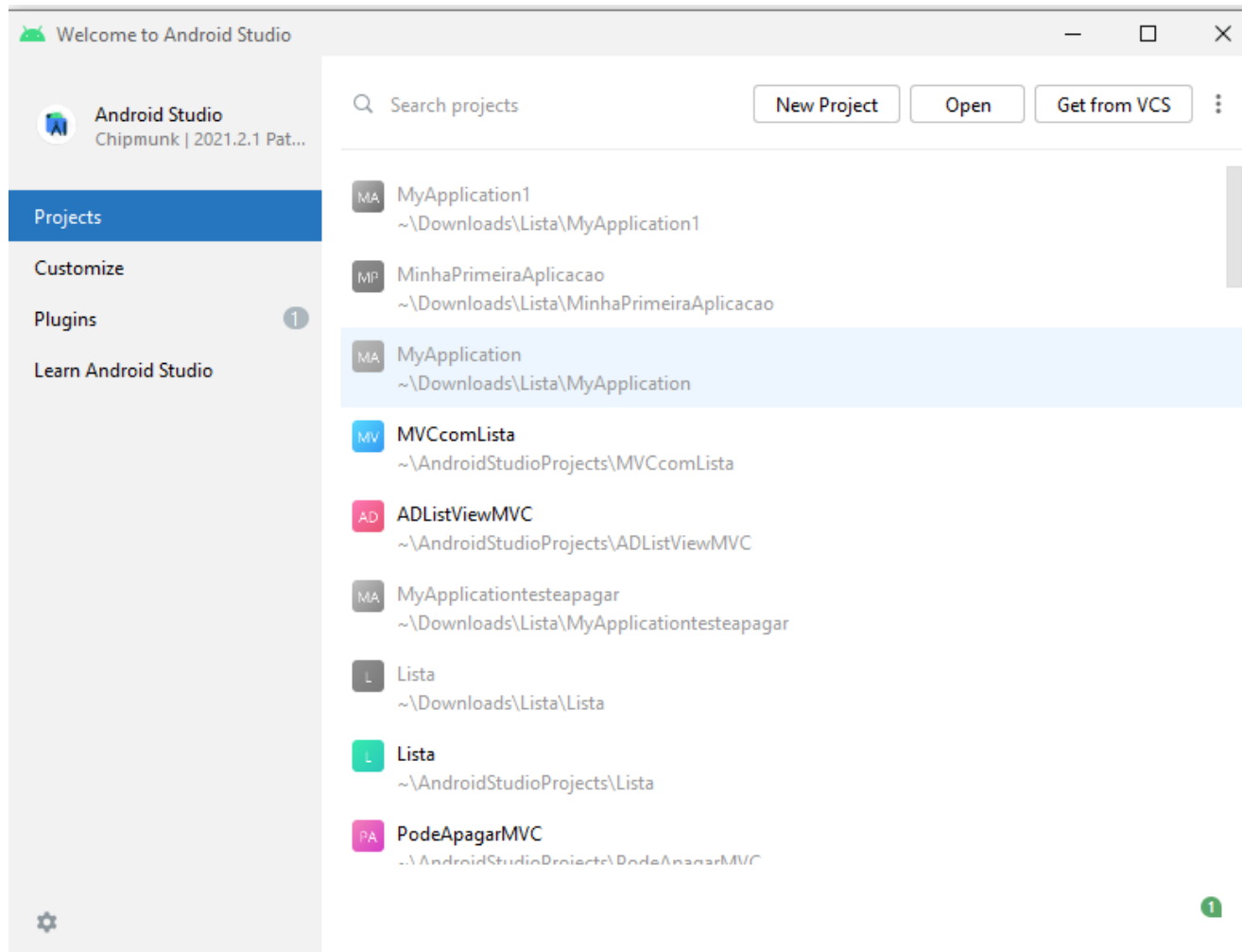
- Conceitos Básicos
 - ▣ Activity
 - ▣ View
 - ▣ Resources
 - ▣ A Classe R
- Interface Visual
 - ▣ Criação de interface Visual em XML
 - ▣ Criação de Interface Visual com API Java
 - ▣ Características Visuais dos Componentes

Android Studio

- É o ambiente de desenvolvimento de aplicativos para Android do Google
 - Utiliza linguagem Java ou Kotlin
 - Contém a IDE e o Android SDK
- Download
 - <https://developer.android.com/studio/index.html>

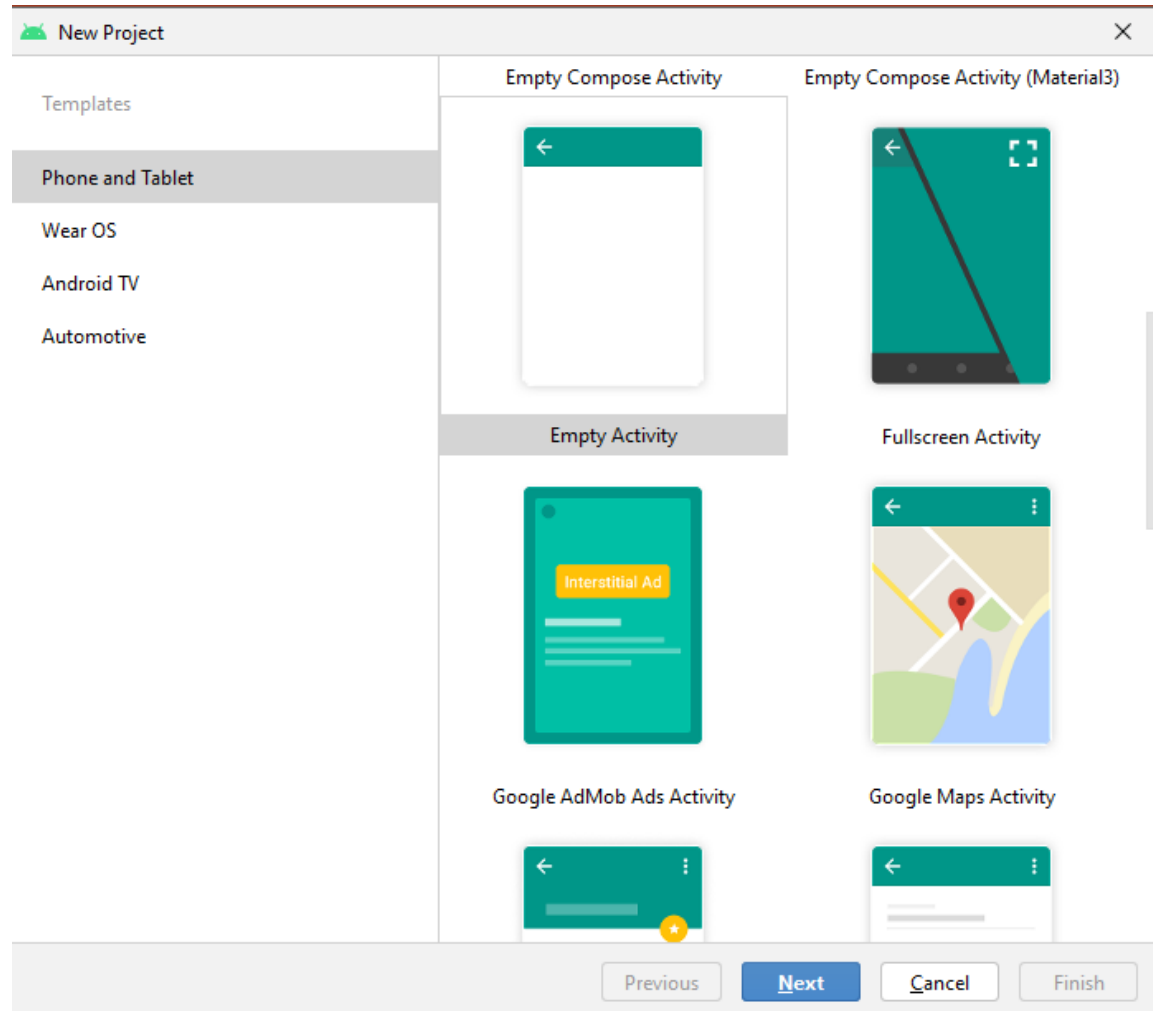
Criando um Aplicativo

- Abra Android Studio, Selecione a opção “New Project”



Criando um Aplicativo

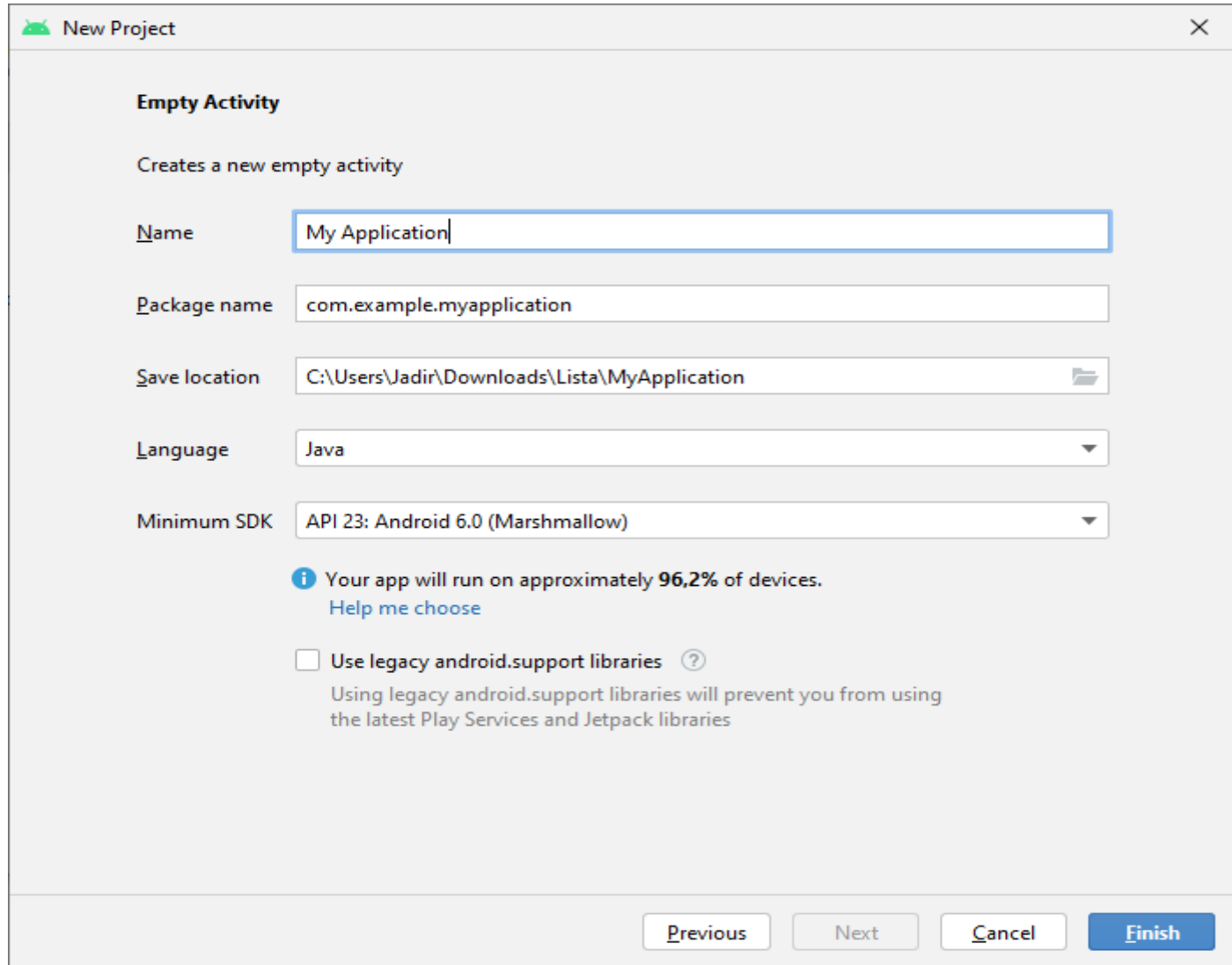
- Na tela **Choose Your Project**
 - Selecione a guia “Phone and Tablet”
 - Escolha “Empty Activity”
 - Clique no botão “Next”



Criando um Aplicativo

- Na tela **Configure Your New Project**
 - Informe o nome da aplicação (Name)
 - Informe o nome do pacote (Package name)
 - Isso será utilizado para montar os namespaces das classes e os imports!
 - Informe a localização do projeto (Save Location). Você pode selecionar a pasta no botão com o ícone de uma pasta!
 - Selecione a linguagem (Kotlin ou Java)
 - Selecione a API mínima (Minimum API level)
 - Clique no botão “Finish”

Criando um Aplicativo




New Project

Empty Activity

Creates a new empty activity


Name


Package name

Save location 

Language


Minimum SDK

 Your app will run on approximately **96,2%** of devices.
[Help me choose](#)

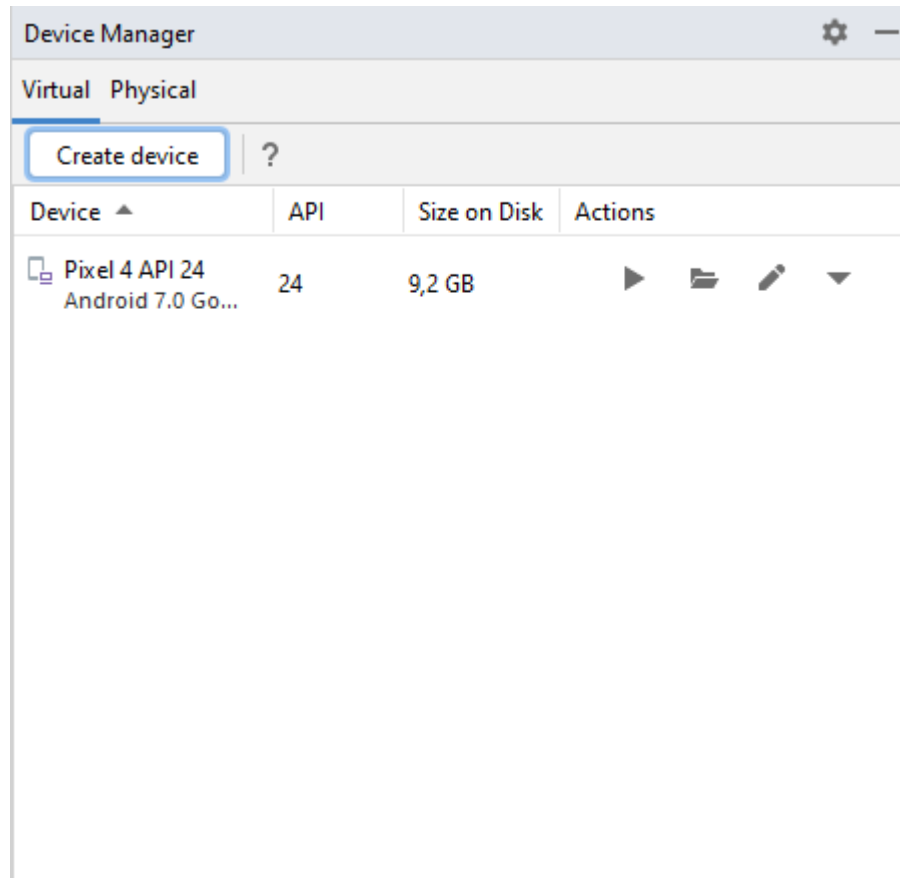
☐ **Use legacy android.support libraries** 

Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

Criando o Emulador

- O Emulador Android é um dispositivo virtual que simula a configuração real de um celular
 - É possível criar emuladores com diversas configurações diferentes, incluindo versão da API Android
- Clique no ícone do AVD Manager () para criar um novo emulador
- Na tela que abrir, clique em “Create virtual Device”

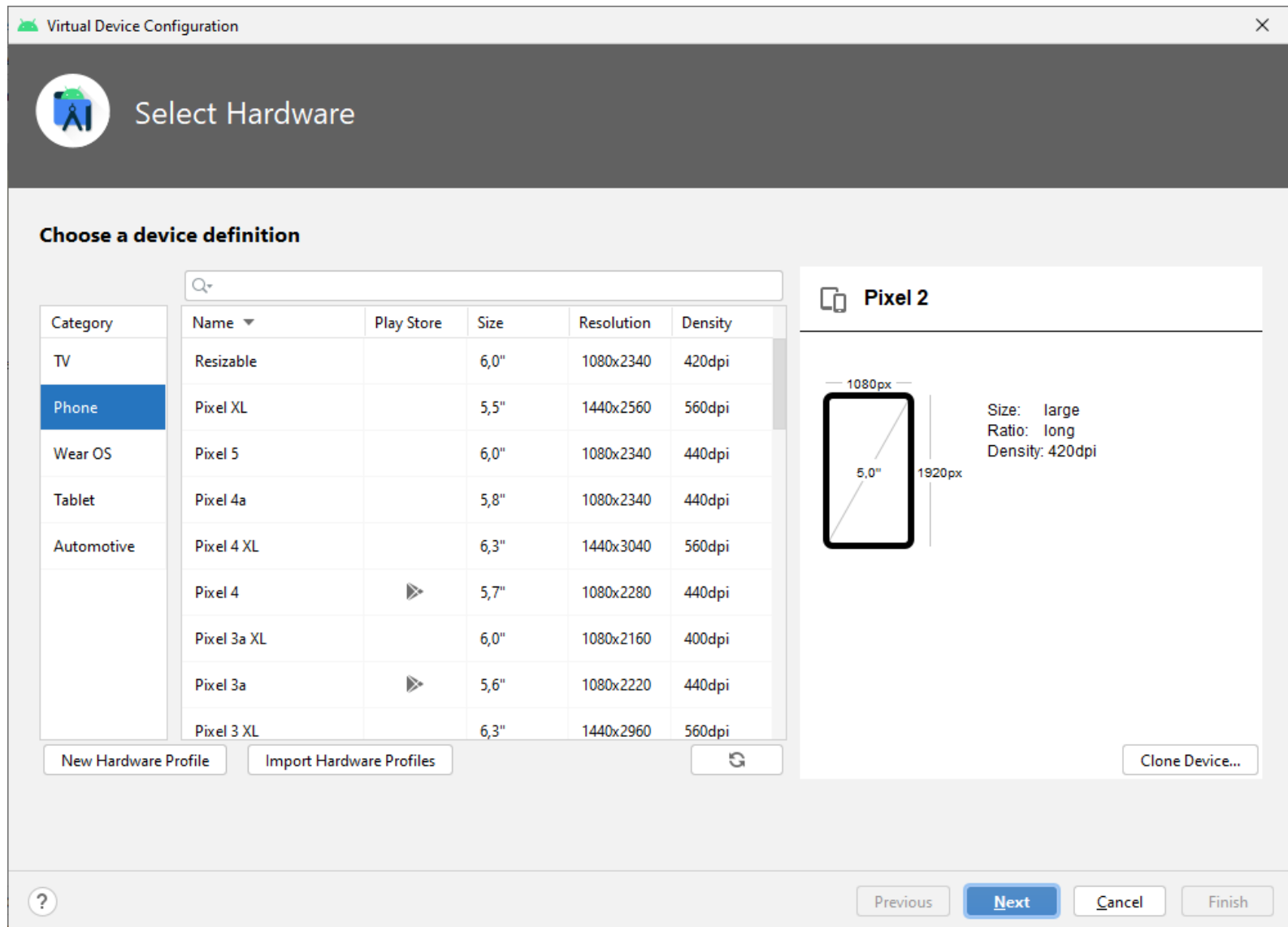
Criando o Emulador



Criando o Emulador

- A primeira tela mostra as opções de hardware
 - Existem opções pré-configuradas para diversos tipos de smartphones, tablets, wearables e TV
- Para a aula, vamos utilizar a opção 5.4'' FWVGA, dentro da categoria "Phone"
- Clique "Next"

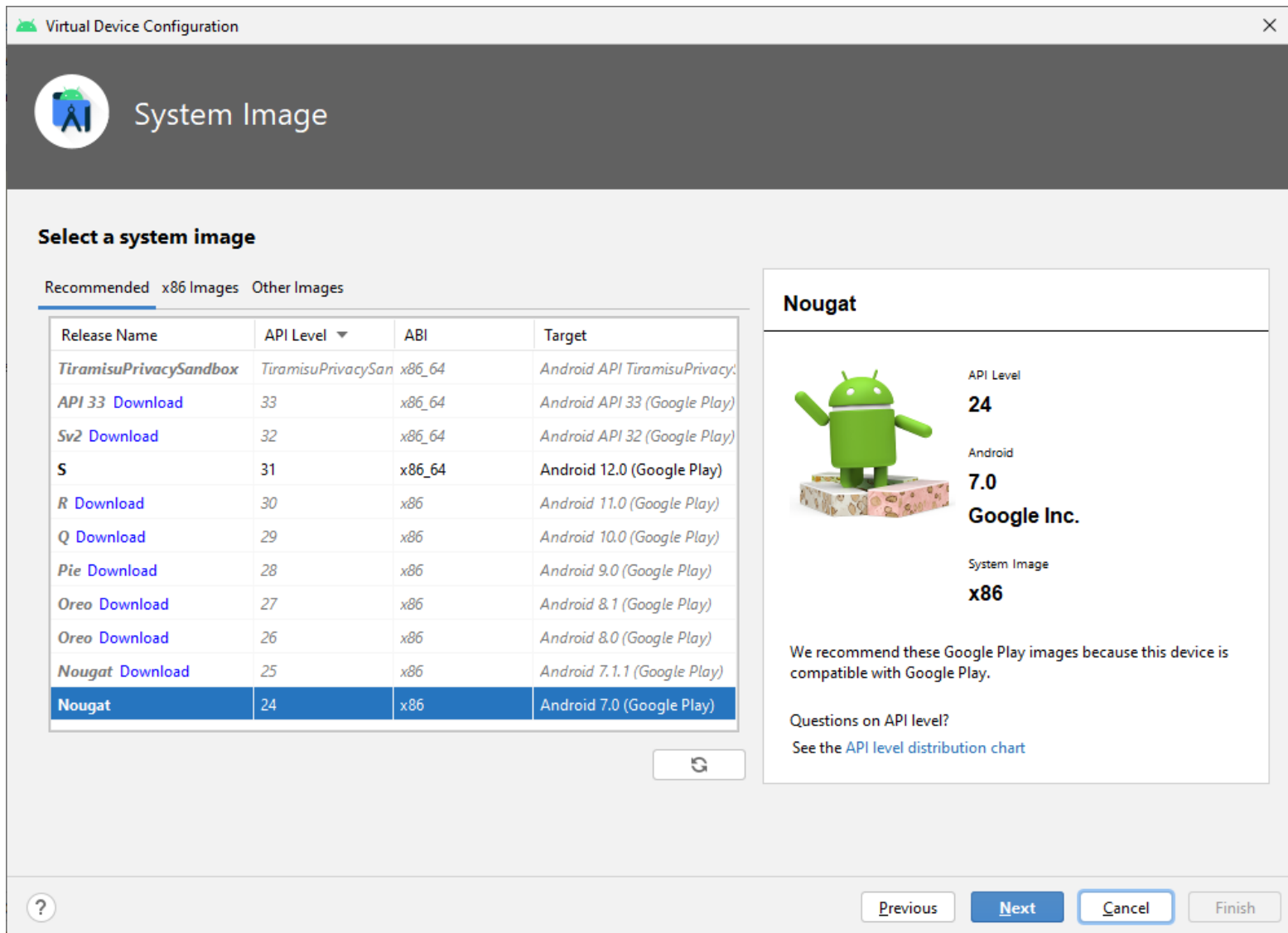
Criando o Emulador



Criando o Emulador

- Na próxima tela selecione a imagem do emulador, com a versão do Android
 - As opções mostradas serão aquelas já instaladas na máquina de desenvolvimento
 - Lembre-se sempre de escolher uma versão igual ou superior ao da API escolhida para o aplicativo
- Clique “Next”

Criando o Emulador




The screenshot shows the 'Virtual Device Configuration' window, specifically the 'System Image' tab. It features a table of recommended system images and a detailed view of the selected 'Nougat' image on the right.

Select a system image

Recommended x86 Images Other Images

Release Name	API Level	ABI	Target
<i>TiramisuPrivacySandbox</i>	<i>TiramisuPrivacySan</i>	<i>x86_64</i>	<i>Android API TiramisuPrivacy:</i>
API 33 Download	33	x86_64	Android API 33 (Google Play)
Sv2 Download	32	x86_64	Android API 32 (Google Play)
S	31	x86_64	Android 12.0 (Google Play)
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	24	x86	Android 7.0 (Google Play)

Nougat

 API Level **24**
Android **7.0**
Google Inc.
System Image **x86**

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

Questions on API level?
See the [API level distribution chart](#)

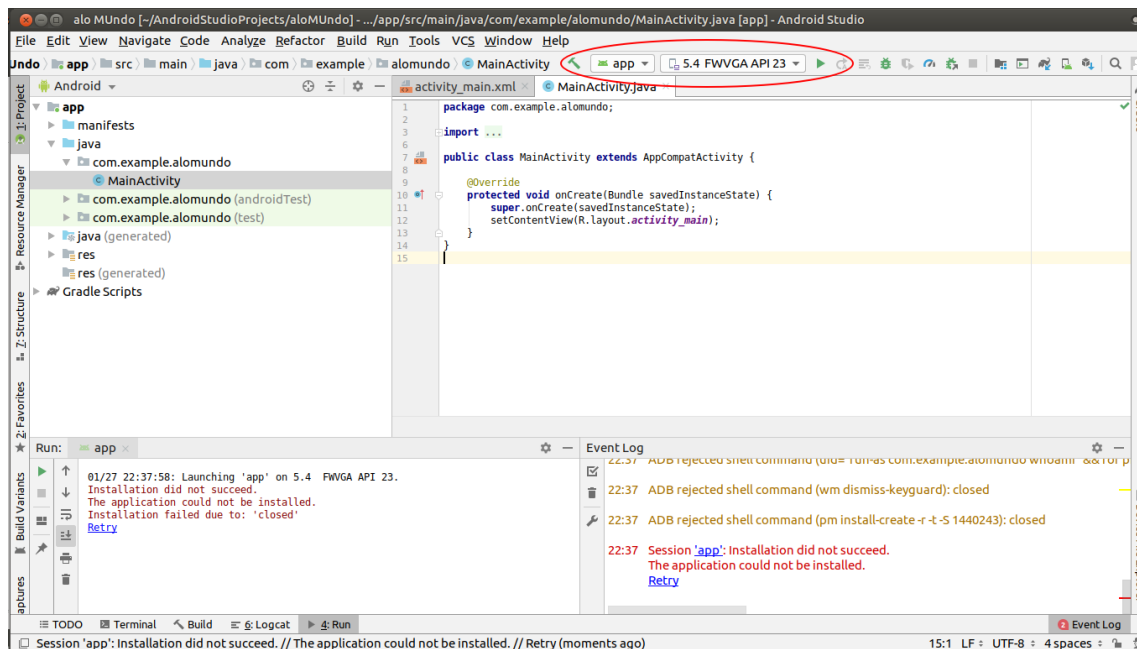
Previous Next Cancel Finish

Criando o Emulador

- Depois, basta verificar e alterar alguma configuração necessária
 - Para o hardware escolhido, temos que alterar o nome, que está inválido.
- Clique em Finish
- Você pode criar quantos emuladores quiser
 - Testar seu aplicativo em diferentes versões de API, configurações de hardware e tamanhos de tela

Executando o Aplicativo

- Volte a tela principal da IDE e execute seu aplicativo clicando no “Play” da barra superior



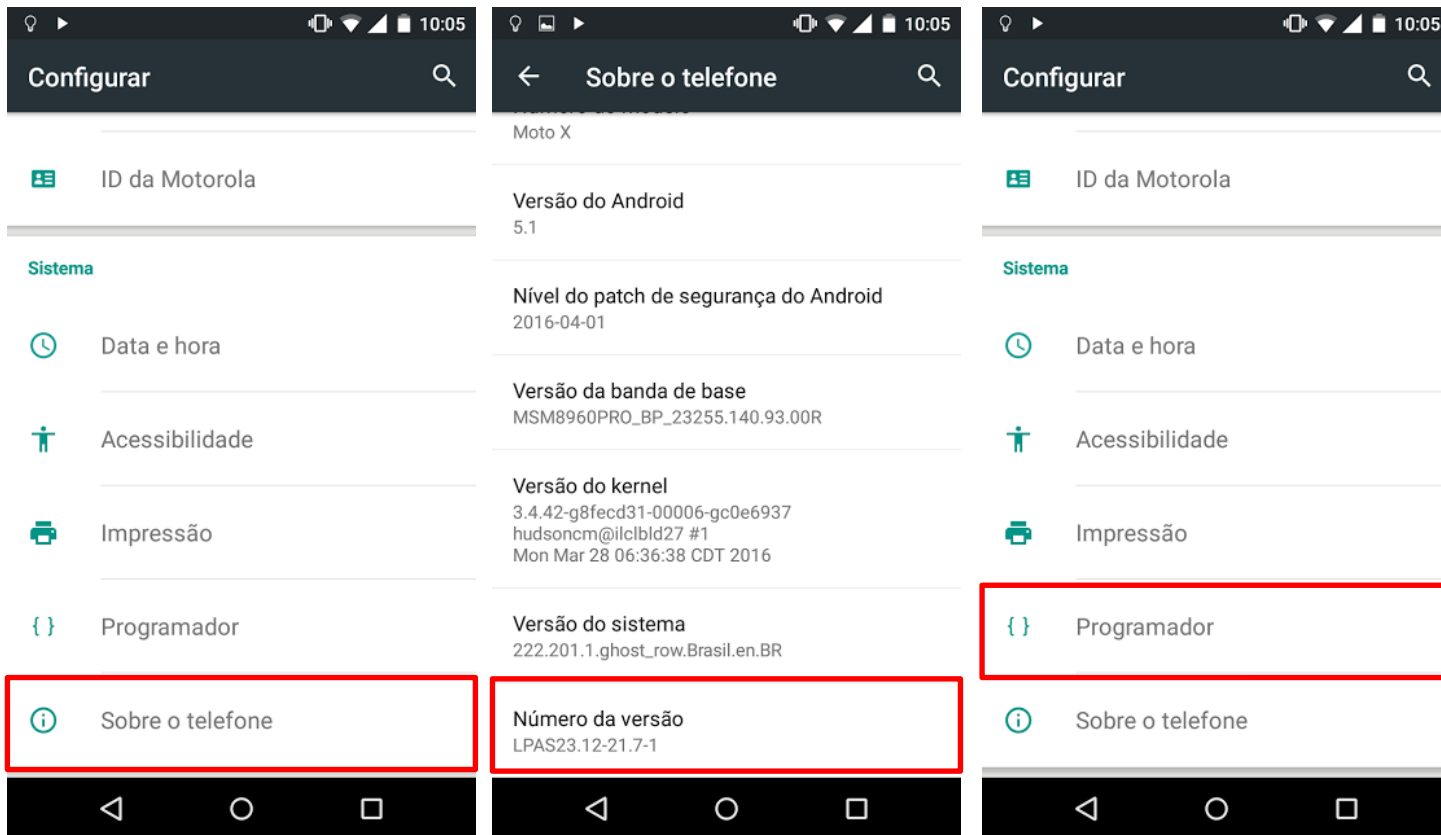
Executando o Aplicativo

- Você também pode executar o aplicativo diretamente do seu smartphone Android
- A configuração depende do fabricante do seu smartphone

Executando o Aplicativo

- Para conseguir executar diretamente no telefone, primeiro precisam habilitar seu telefone para desenvolvimento. Na versão Lollipop faça o seguinte:
 - Abra as configurações do telefone
 - Vá até a opção “Sobre o telefone” (último item)
 - Clique 7 vezes no item “Número da versão”
 - Volte às configurações, e haverá uma nova opção, “Programador” (penúltimo item)
 - Abra esta opção

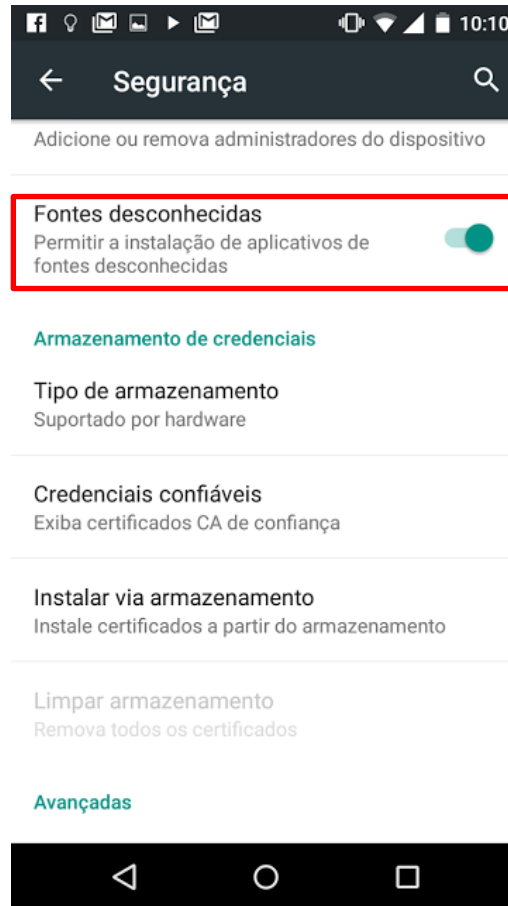
Executando o Aplicativo



Executando o Aplicativo

- Na opção programador:
 - Ative a opção
 - Ative a depuração USB
- Volte às configurações e selecione a opção segurança
 - Habilite a opção “Fontes Desconhecidas”

Executando o Aplicativo



Executando o Aplicativo

- Agora é preciso conectar o smartphone no computador via USB e ter o driver ADB interface, específico do fabricante, instalado
 - Para alguns fabricante, o driver é instalado automaticamente
 - Para outros é preciso instalar manualmente
 - Consulte <https://developer.android.com/studio/run/oem-usb.html>
- Depois disso, o seu smartphone vai aparecer na lista de dispositivos quando executar o aplicativo pelo Android Studio
 - Basta selecionar o dispositivo e clicar em OK

O que é
Activity



Conceitos Básicos

38

Activity

- **Activity** é uma classe que representa basicamente uma tela da aplicação e é responsável por tratar eventos gerados nessa tela, como, por exemplo, quando o usuário pressiona um botão ou quando um item de menu é escolhido:
 - Localizado no pacote `android.app`;
 - Podemos dizer que a classe `Activity` é similar a classe `JFrame` do swing Java.
- Toda classe que representa uma tela, **estende, direta ou indiretamente**, a classe **Activity**;
- Toda classe **Activity** implementa o método `onCreate()`, que é obrigatório e responsável por realizar a inicialização necessária para executar a aplicação.

Conceitos Básicos

39

Activity

- Em uma aplicação **Android**, cada tela criada e composta por;
 - ▣ *Uma classe que estende de **Activity** ou suas derivadas;*
 - ▣ *Um arquivo XML que representa o layout da tela.*
- Uma tela é composta por vários elementos visuais, os quais no **Android** são representados pela classe **View**;
 - ▣ *Localizada no pacote **android.view**;*
- Além de definir uma tela, uma **Activity** controla seu estado e a passagem de parâmetros de uma tela para outra e controla os eventos dos componentes visuais.

Activity

```
package br.unicid.app.teste;

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

public class Tela extends Activity {

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

Listagem de uma classe chamada **Tela** que estende da classe **Activity**.

Activity

```
package br.unicid.app.teste;
```

```
import android.os.Bundle;
```

```
import android.app.Activity;
```

Pacote da classe **Activity**

```
public class Tela extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

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

```
        setContentView(R.layout.tela);
```

```
    }
```

```
}
```

Listagem de uma classe chamada **Tela** que estende da classe **Activity**.

Activity

```
package br.unicid.app.teste;
```

```
import android.os.Bundle;
```

```
import android.app.Activity;
```

```
public class Tela extends Activity {
```

```
    @Override
```

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

```
}
```

Note também a definição do método obrigatório `onCreate(Bundle)`, responsável pela inicialização necessária para execução da aplicação.

Activity

```
package br.unicid.app.teste;

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

public class Tela extends Activity {

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

}
```

`setContentView()` é um método da classe `Activity` utilizado para vincular a interface gráfica (xml do layout) à classe `Tela` que por sua vez será a classe responsável por manipular o layout via programação.

O que é View



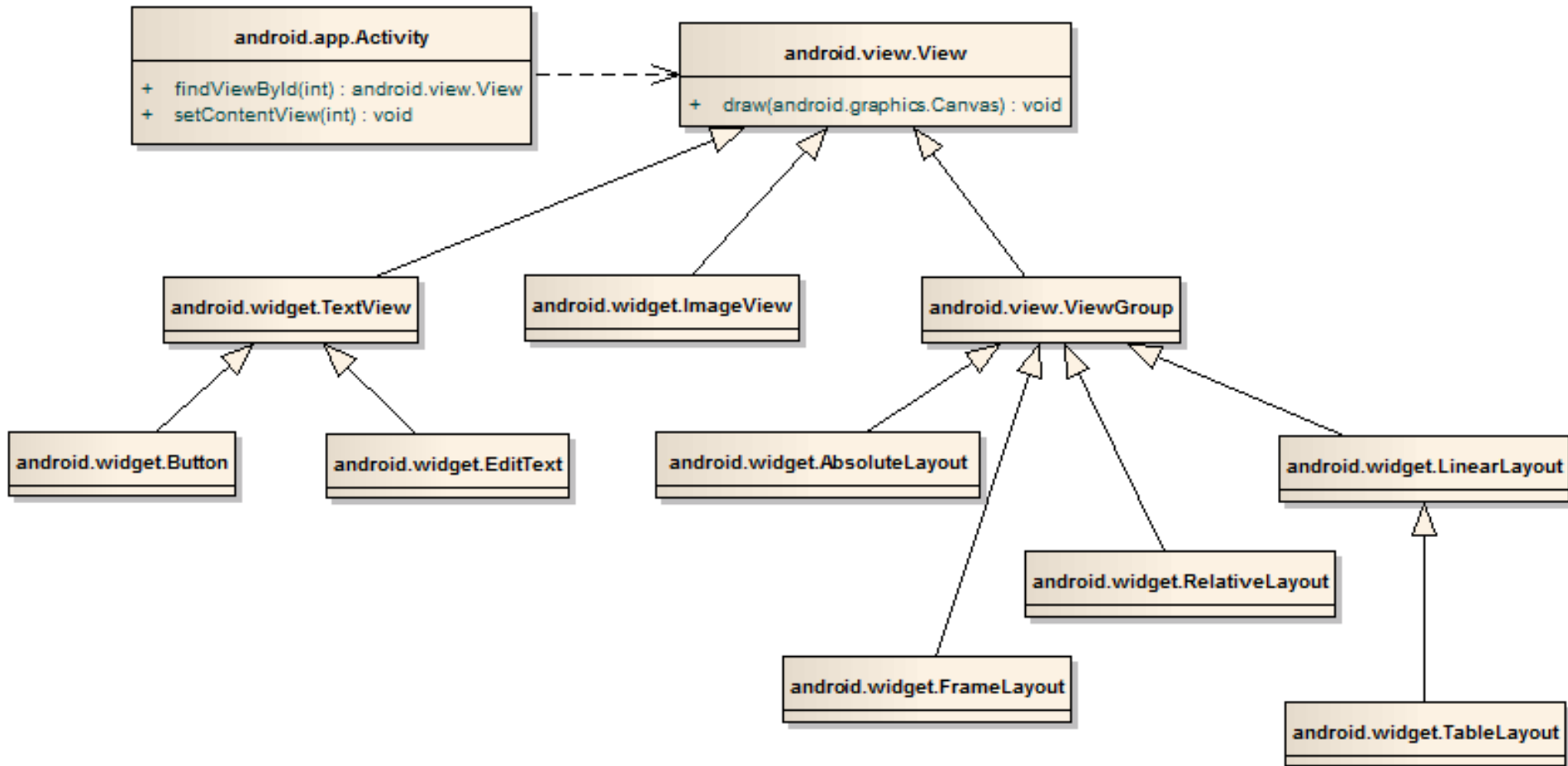
Conceitos Básicos

45

View

- A Classe **View** é responsável por desenhar algo na tela e tratar seus eventos se necessário, e é a classe base para todos os componentes visuais do Android:
 - ▣ Localizada no pacote `android.view`.
- Uma **View** pode ser:
 - ▣ Um simples componente gráfico (*botão, checkbox, caixa de texto, rótulos, entre outros*);
 - ▣ Ou algo mais complexo, como por exemplo, um gerenciador de layout (*RelativeLayout, LinearLayout, TableLayout, entre outros*);
- Os componentes gráficos, sejam eles simples ou complexos, **estendem, direta ou indiretamente, a classe View.**

View



O diagrama de classe acima mostra um esboço bem resumido da classe **View**, bem como as suas classes-filhas. Note que a classe **Activity** tem uma dependência da classe **View**.

O que é Resource



Conceitos Básicos

48

Resources

- ❑ **Resources** (*recursos*) são objetos externos que podem ser utilizados dentro do projeto de uma aplicação Android;
- ❑ Exemplos de Recursos:
 - ❑ *Textos;*
 - ❑ *Imagens;*
 - ❑ *Audio;*
 - ❑ *Vídeo;*
 - ❑ *Layout de Telas.*
- ❑ Podemos referenciar os recursos diretamente pelo código Java (***mas não é o ideal***):
 - ❑ *Repetição de código;*
 - ❑ *Difícil de manter.*
- ❑ O Android possui mecanismos para trabalhar com recursos de forma fácil e centralizada.

Conceitos Básicos

49

Resources

- Os **Resources** ficam localizados e armazenados dentro da pasta **/res**;
- Esta pasta contém os recursos da aplicação, como imagens, layouts de telas, arquivos de internacionalização, etc.
- Esta pasta contém sub-pastas de acordo com o tipo de recurso:
 - ▣ **/res/drawable**; armazena arquivos de imagens da aplicação;
 - ▣ **/res/layout**: armazena os arquivos XML de layouts para construção de telas;
 - ▣ **/res/menu**: armazena os arquivos XML para construção de menus de uma tela;
 - ▣ **/res/mipmap**; pasta exclusiva para armazenamento de imagens que representam ícones do aplicativo;
 - ▣ **/res/values**: contém os arquivos XML utilizados para internacionalização da aplicação, estilização de componentes e conteúdos e outras configurações;

O que é a classe R



Conceitos Básicos

51

A Classe R

- A **classe R** contém os identificadores de referência aos **resources** do projeto através de código de programação:
 - ▣ Cada identificador gerado nesta classe representa um único recurso dentro do projeto, como por exemplo:
 - *Um arquivo XML que representa o **layout**;*
 - *um arquivo XML que representa as **mensagens textuais da aplicação**;*
 - *Uma imagem localizada na pasta **drawable**, etc.*
 - ▣ Para cada recurso inserido na pasta **/res**, é gerado um identificador que representa aquele recurso.
- O Arquivo **R.java** faz a intermediação entre os recursos e o código-fonte da aplicação.
 - ▣ **Esta classe não pode ser alterada manualmente, pois o plugin do Android no Eclipse já faz isso automaticamente.**

A Classe R


A imagem abaixo representa graficamente um layout cujo cada componente foi definido com o atributo **android:id**, conforme mostra cada descrição:

The diagram illustrates the mapping between a visual Android layout and its XML representation. On the left, a screenshot of a 'Teste de Componentes Visuais' (Visual Component Test) interface is shown, featuring a title bar, two labels ('Primeiro Nome:' and 'Último Nome:'), two corresponding text input fields, and a 'Enviar' (Send) button at the bottom. Red arrows point from specific UI elements to their XML definitions on the right. A blue arrow points from the top of the interface to the root layout tag.

- <LinearLayout ...></LinearLayout>**: Points to the overall container of the interface.
- <TextView
android:id="@+id/lblNome" .../>**: Points to the 'Primeiro Nome:' label.
- <EditText
android:id="@+id/txtNome" .../>**: Points to the first text input field.
- <TextView
android:id="@+id/lblSobreNome" .../>**: Points to the 'Último Nome:' label.
- <EditText
android:id="@+id/txtSobreNome" .../>**: Points to the second text input field.
- <Button
android:id="@+id/btnEnviar" .../>**: Points to the 'Enviar' button.

A Classe R

```
public final class R {  
    public static final class attr {...}  
    public static final class dimen {...}  
    public static final class drawable {...}  
  
    public static final class id {  
        public static final int action_settings=0x7f080006;  
        public static final int btnEnviar=0x7f080005;  
        public static final int lblNome=0x7f080001;  
        public static final int lblSobreNome=0x7f080003;  
        public static final int txtNome=0x7f080002;  
        public static final int txtSobrenome=0x7f080004;  
    }  
  
    public static final class layout {...}  
    public static final class menu {...}  
    public static final class string {...}  
    public static final class style {...}  
}
```



Este trecho de código, por exemplo, representa os identificadores dos componentes visuais do layout mostrado no slide anterior.

Criação de interface visual em XML

- ❑ O Android é bastante flexível em relação à criação da interface gráfica e permite que a tela seja criada em XML ou diretamente pelo código-fonte utilizando a API Java, de forma similar ao Swing.
- ❑ Entretanto, a idéia do Android é separar a lógica de negócios da parte visual:
 - ▣ *Codificação mais limpa e enxuta de ambas as partes;*
 - ▣ *A manutenção do código fica mais simples*
- ❑ Para definir uma interface gráfica, basta inserir um arquivo XML de layout válido na pasta **/res/layout** do projeto.

Veja abaixo um esboço de código XML de um 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/LinearLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Este é um exemplo de layout no android" />

</LinearLayout>
```

<LinearLayout>...</LinearLayout>
Define o tipo de layout utilizado para organizar os elementos visuais da tela.

<TextView ... />
Define um componente utilizado para representar um conteúdo textual.

Interface Visual

56

Criação de interface visual em XML

- Para definir um componente em XML utilizamos a seguinte sintaxe:

```
<nome_componente atrib1="valor1" atrib2="valor2" ... />
```

- onde:

- ▣ **nome_componente**: nome do elemento que define um componente no layout;
- ▣ **atrib**: atributo do elemento que podem ser vários, cada um representando uma característica.
- ▣ **valor**: valor do atributo.

Após a definição do arquivo XML que representa o layout, devemos vinculá-lo à classe **Activity** que o controlará, através do método **setContentView()** passando como parâmetro o id que representa o layout em questão.

Interface Visual

57

Características Visuais dos Componentes

- No Android é possível definir algumas características dos componentes tais como:
 - ▣ formatações básicas:
 - Tamanho de fonte;
 - Cor de fontes
 - Estilo de fonte
 - ▣ Cor de fundo (Background);
 - ▣ Definição de hints;
 - ▣ Definição de links;
 - ▣ Dimensionamento de componentes;
 - ▣ entre outros.
- Veremos a seguir alguns atributos que modifica alguns estados dos componentes e conteúdos utilizados em um layout.

Interface Visual

58

Características Visuais dos Componentes

□ Definição de Tamanho da Fonte:

- ▣ O atributo `android:textSize` define o tamanho da fonte, exemplo:
`android:textSize="12pt"`

□ Definição de Cor da Fonte:

- ▣ O atributo `android:textColor` define a cor de fonte do conteúdo textual de um componente, exemplo:
`android:textColor="#00FF00"`

□ Definição de Alinhamento de Texto

- ▣ O atributo `android:gravity` define o alinhamento (*left*, *center*, *right*, etc.) do conteúdo textual de um componente, exemplo:
`android:gravity="left"`

□ Definição de Estilo da Fonte:

- ▣ O atributo `android:textStyle` define o estilo da fonte do conteúdo em normal, itálico (*italic*) e negrito (**bold**), exemplo:
`android:textStyle="bold"`

Exemplo de tamanhos de fonte:

The image shows an Android emulator window titled "5554:Android_2.3.3". The status bar at the top displays the time "7:18" and various icons. The main content area has a title bar "Teste de Componentes Visuais". Below the title bar, there are four lines of text, each with a red arrow pointing to a code snippet on the right:

- Line 1: "tamanho de fonte: 15pt" (red arrow points to the first code snippet)
- Line 2: "tamanho de fonte: 0.2in" (red arrow points to the second code snippet)
- Line 3: "tamanho de fonte: 30px" (red arrow points to the third code snippet)
- Line 4: "tamanho de fonte: 15sp" (red arrow points to the fourth code snippet)

The code snippets are:

- `<TextView
android:textSize="15pt" .../>`
- `<TextView
android:textSize="0.2in" .../>`
- `<TextView
android:textSize="30px" .../>`
- `<TextView
android:textSize="15sp" .../>`

The emulator also shows a "Basic Controls" panel on the right with buttons for volume, home, menu, and DPAD, and a "Hardware Keyboard" section with a note "Use your physical keyboard to provide input".

Exemplo de cores de fonte:

The image shows an Android emulator window titled "5554:Android_2.3.3". The main screen displays a "Teste de Componentes Visuais" (Visual Component Test) interface. It contains four lines of text with different colors: "conteúdo em vermelho" (red), "conteúdo em verde" (green), "conteúdo em azul" (blue), and "conteúdo cor de rosa" (pink). Below the text is a button labeled "Rótulo do Botão" (Button Label) and a text input field containing "Conteúdo na caixa de Texto" (Text in the text box). To the right of the emulator, four XML code snippets are shown, each with a red arrow pointing to a specific text element on the screen:

- `<TextView
android:textColor="#FF0000" .../>` (points to "conteúdo em vermelho")
- `<TextView
android:textColor="#00FF00" .../>` (points to "conteúdo em verde")
- `<TextView
android:textColor="#0000FF" .../>` (points to "conteúdo em azul")
- `<TextView
android:textColor="#FFC0CB" .../>` (points to "conteúdo cor de rosa")

The emulator's status bar at the top shows the time as 7:27 and various icons. The right side of the emulator displays a virtual hardware keyboard and other controls.

Exemplo de estilos de fonte:

The image shows an Android emulator window titled "5554:Android_2.3.3". The status bar at the top displays signal strength, battery level, and the time "7:31". The app's title bar reads "Teste de Componentes Visuais". The main content area displays four lines of text with different styles: "Conteúdo no estilo normal", "Conteúdo no estilo itálico", "Conteúdo no estilo negrito", and "Conteúdo no estilo negrito e itálico". To the right of the text is a virtual device control panel with buttons for volume, power, home, menu, and navigation. Four red arrows point from XML code snippets on the right to the corresponding text lines. The code snippets are:

```
<TextView  
  android:textStyle="normal" .../>
```

 for the first line,

```
<TextView  
  android:textStyle="italic" .../>
```

 for the second,

```
<TextView  
  android:textStyle="bold" .../>
```

 for the third, and

```
<TextView  
  android:textStyle="bold/italic" .../>
```

 for the fourth.

Conteúdo no estilo normal

Conteúdo no estilo itálico

Conteúdo no estilo negrito

Conteúdo no estilo negrito e itálico

`<TextView
 android:textStyle="normal" .../>`

`<TextView
 android:textStyle="italic" .../>`

`<TextView
 android:textStyle="bold" .../>`

`<TextView
 android:textStyle="bold/italic" .../>`

Interface Visual

62

Características Visuais dos Componentes

□ Definição de Cor de Fundo:

- ▣ O atributo **android:background** define a cor de fundo do componente, exemplo:

android:background="#0000FF"

□ Definição de Hints:

- ▣ O atributo **android:hint** é utilizado para definir um texto que deverá aparecer escrito no campo de texto que não foi preenchido.
- ▣ Na maioria das vezes utilizado como um indicador do que será preenchido no campo pelo usuário, exemplo:

android:hint="Digite aqui o seu nome"

□ Definição de Links:

- ▣ O atributo **android:autoLink** define a exibição do conteúdo no componente **TextView** na forma de link, exemplo:

android:autoLink="web"

- ▣ Utilizamos o valor "web" por se tratar de uma URL de um site.

Exemplo de cor de fundo:

The image shows an Android emulator window titled "5554:Android_2.3.3". The main display area is titled "Teste de Componentes Visuais" and contains four text views with different background colors: red, green, blue, and pink. Red arrows point from each text view to its corresponding XML code snippet on the right. The XML snippets are as follows:

- For the red text view:

```
<TextView  
  android:background="#FF0000" .../>
```
- For the green text view:

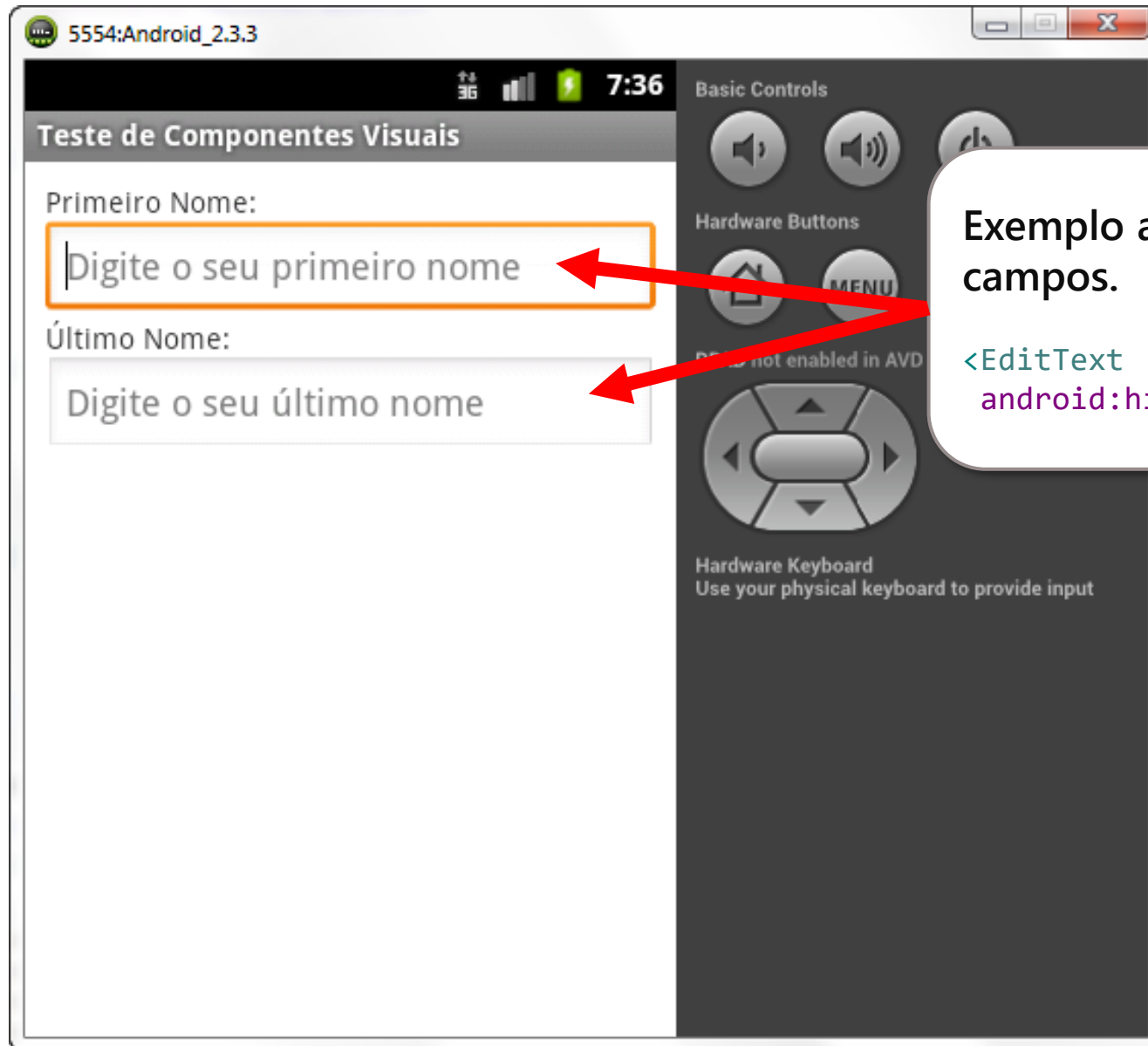
```
<TextView  
  android:background="#00FF00" .../>
```
- For the blue text view:

```
<TextView  
  android:background="#0000FF" .../>
```
- For the pink text view:

```
<TextView  
  android:background="#FFC0CB" .../>
```

The emulator interface also shows a status bar at the top with the time "7:29" and a "Basic Controls" panel on the right with various hardware buttons like volume, home, menu, and a DPAD.

Exemplo de definição de hints:



Exemplo antes de preencher os campos.

```
<EditText  
    android:hint="[conteúdo..]" .../>
```

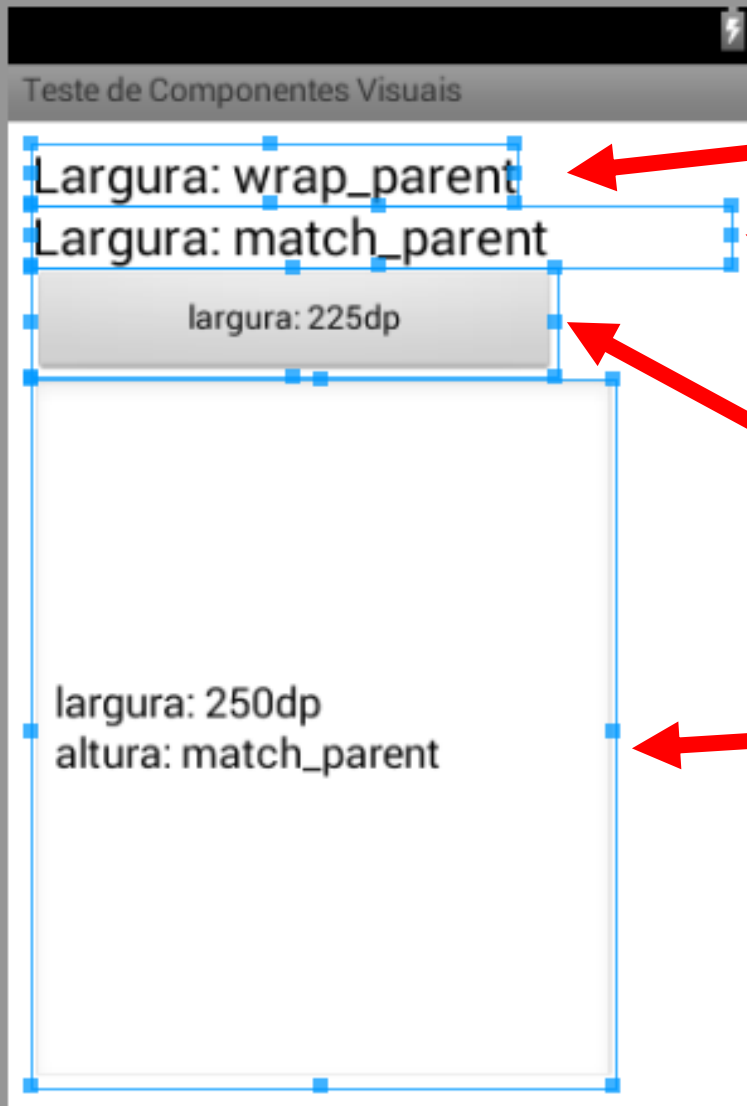

Características dos Componentes

65

□ Dimensionamento de Componentes

- ▣ Os atributos `android:layout_width` e `android:layout_height` definem a largura e altura básica de um componente/view. Estes atributos são obrigatórios para qualquer componente/view contido em um gerenciador de layout.
- ▣ Esses atributos podem receber os seguintes valores:
 - **Numérico:** Valores inteiros especificando o tamanho do componente. Esses valores podem ser em pixels (**px**), polegadas (**in**), milímetros (**mm**), pontos (**pt**), *Density-independent Pixels* (**dp** ou **dip**) e *Scale-independent Pixels* (**sp**).
 - **match_parent:** Constante que define que o componente deve ocupar todo o tamanho definido pelo **layout-pai**.
 - **fill_parent:** Constante que define a mesma característica da constante
 - **wrap_content:** Constante que define que o componente em questão deve ocupar apenas o tamanho necessário da tela, por exemplo, em se tratando de um componente **TextView**, o tamanho é caracterizado pela altura do caractere e o tamanho da cadeia de caracteres.

Exemplo de definição dimensionamento:



Dimensionamento largura necessária

Dimensionamento de largura total

Dimensionamento de largura em 225dp

Dimensionamento de largura em 250dp e altura total

Características Visuais dos Componentes

□ Alinhamento de Componentes

- ▣ O atributo **android:layout_gravity** permite alinhar o componente em relação ao seu pai (**layout**).
- ▣ Esse atributo pode receber os seguintes valores:
 - **left**: Define o alinhamento à esquerda do componente dentro do layout.
 - **center**: Define o alinhamento centralizado do componente dentro do layout.
 - **right**: Define o alinhamento à direita do componente dentro do layout.
 - Existem outros valores que podem ser usados no atributo **android:layout_gravity** tais como: **top**, **bottom**, **center_vertical**, **fill_vertical**, **center_horizontal**, **fill_horizontal** e **fill**.
- ▣ **IMPORTANTE:**
 - Cada um desses valores tem o seu funcionamento de acordo com o tipo de layout utilizado, ou seja, não funciona para todos os tipos de layouts.

Interface Visual

68

Características Visuais dos Componentes

❑ Definição de Tipo de Campos:

- ❑ O **EditText** possui uma propriedade chamada **android:inputType** que define o tipo de entrada de dados, como por exemplo, entrada de texto normal, apenas número, campo de senha, etc.
- ❑ Exemplo de uso:
android:inputType="text"
- ❑ Segue abaixo alguns valores de tipo de campo válidos:
 - text**: define um campo de texto simples;
 - number**: define um campo de dados numéricos;
 - textPassword**: define um campo de caracteres ocultos;
 - date**: define um campo de entrada de datas;
 - phone**: define um campo número de telefone;
 - entre outros.



Vamos as
Práticas

Sumário

- Introdução
- A Classe Intent
- Usando Intent com Activity
 - ▣ Transição entre telas
 - ▣ Transição entre telas com passagem de parâmetros

Introdução

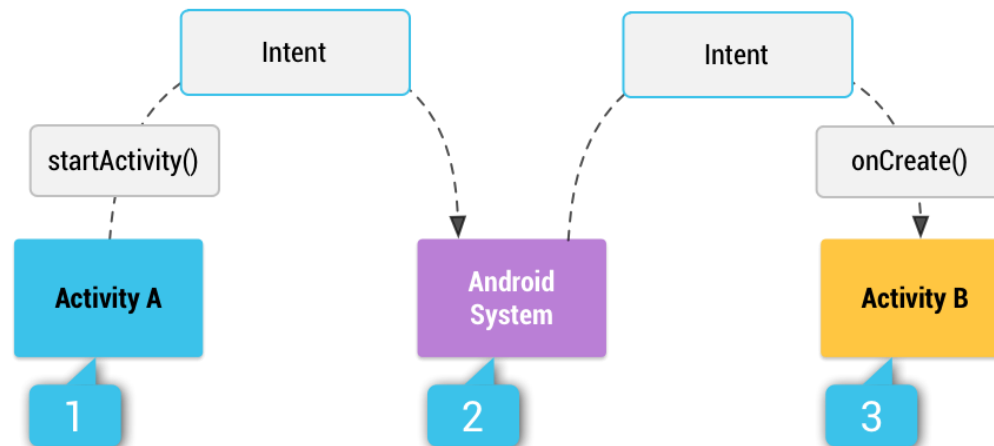
71

- Até o momento vimos apenas como utilizar uma **Activity** (tela) inicial na aplicação uma tela isolada e sempre iniciada pelo usuário.
- No entanto, em aplicações mais complexas será necessário criar mais de uma tela e um fluxo de navegação entre elas.
- Para isso, existem alguns recursos necessários para programar esta interação entre as telas:
 - ▣ *Objetos da classe **Intent**;*
 - ▣ *O método **startActivity(intent)**:*
 - usado para iniciar/chamar a próxima tela.

A Classe Intent

72

- A classe `Intent` representa uma “*ação*” que uma aplicação deseja executar, como por exemplo, abrir uma tela.
 - ▣ Localizado no pacote `android.content`.
- A tradução da palavra “*intent*” para o português é “*intenção*”, de forma que uma `Intent` representa a intenção de uma aplicação realizar uma determinada tarefa.
 - ▣ Na prática, essa intenção *é enviada ao sistema operacional Android como forma de uma mensagem.*
 - ▣ Ao receber esta mensagem, o S.O tomará as decisões necessárias, dependendo do conteúdo da mensagem, isto é, de sua intenção.



Usando Intent com Activities

73

Transição entre telas

- Para trabalhar com navegação entre telas é necessário o uso do método `startActivity(intent)`.
- O parametro `intent` a ser passado ao método contém as informações necessárias sobre a **Activity** que será ativada pelo sistema operacional.
- A listagem abaixo mostra um trecho de código em que uma tela qualquer solicitará ao S.O a iniciação de uma outra tela representada pela **Activity** `Tela2`:

```
//Declarando uma variável do tipo Intent
Intent it = new Intent(getApplicationContext(), Tela2.class);

//Iniciando a Tela desejada (tela 2)
startActivity(it);
```

Usando Intent com Activities

74

Transição entre telas com passagem de parâmetros

- Além utilizarmos a **Intent** para programar do processo de navegação entre as telas, também podemos passar parâmetros (dados) de uma tela para outra.
- Para isso utilizamos o método **putExtra(chave, valor)**, onde:
 - ▣ *Chave*: representa a identificação do valor a ser passado.
 - ▣ *Valor*: representa o dado a ser enviado que pode ser de qualquer tipo primitivo.

```
//Declarando uma variável do tipo Intent
Intent it = new Intent(this, Tela2.class);

//Utilizando o método p/ enviar parâmetros
it.putExtra("msg", "Olá Mundo");

//Iniciando a Tela desejada (tela 2)
startActivity(it);
```

Usando Intent com Activities

75

Transição entre telas com passagem de parâmetros

- Na segunda tela, utilizamos o método `getStringExtra()` para recuperar os dados enviados da primeira tela. Veja a listagem abaixo:

```
public class SegundaTela extends Activity {  
    private TextView lblMensagem;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.segunda_tela);  
  
        lblMensagem = (TextView) findViewById(R.id.lbl_mensagem);  
  
        //Obtendo o objeto Intent enviado da tela Principal.  
        Intent it = getIntent();  
  
        //Recuperando a informação através da chave "msg".  
        String mensagem = it.getStringExtra("msg");  
  
        //Exibindo a informação em um TextView na segunda tela.  
        lblMensagem.setText(mensagem);  
    }  
}
```

Sumário

- APP Restaurante Balaio de Lenha
- Recursos de Strings
- Componentes
 - ▣ ListView
- Gravando/Lendo um Arquivo Texto
- Envio de Email

Restaurante Balaio de Lenha

77

Balaio de Lenha Restaurante

Consumo Total

Couvert Artístico


Dividir Conta por

CALCULAR CONTA FINAL

Taxa de Serviço 10%

Conta Total R\$

Valor por pessoa

**Balaio de Lenha**
RESTAURANTE

```
<LinearLayout
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"" />

    <LinearLayout
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView2"" />

        <EditText
            android:id="@+id/edtConsumo"/>
    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView3"/>

        <EditText
            android:id="@+id/edtCouvert"/>
    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView4"/>

        <EditText
            android:id="@+id/edtConta"/>
    </LinearLayout>
```

```
<Button
    android:id="@+id/btnCalcular"" />

<LinearLayout
    android:orientation="horizontal">

    <TextView
        android:id="@+id/lblTaxa"/>

    <EditText
        android:id="@+id/edtTaxa"/>
</LinearLayout>

<LinearLayout
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView8"/>

    <EditText
        android:id="@+id/edtTotal"/>
</LinearLayout>

<LinearLayout
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView9"/>

    <EditText
        android:id="@+id/edtPessoa"/>
</LinearLayout>

<ImageView
    android:id="@+id/imageView" />
</LinearLayout>
```

Restaurante Balaio de Lenha

78

@Override

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

```
    btnCalcular = findViewById(R.id.btnCalcular);  
    edtConsumo = findViewById(R.id.edtConsumo);  
    edtCouvert = findViewById(R.id.edtCouvert);  
    edtConta = findViewById(R.id.edtConta);  
    edtTaxa = findViewById(R.id.edtTaxa);  
    edtTotal = findViewById(R.id.edtTotal);  
    edtPessoa = findViewById(R.id.edtPessoa);
```

```
    edtTaxa.setEnabled(false);  
    edtTotal.setEnabled(false);  
    edtPessoa.setEnabled(false);
```

```
    btnCalcular.setOnClickListener(new View.OnClickListener() {
```

@Override

```
    public void onClick(View v) {  
        double consumo = Double.parseDouble(edtConsumo.getText().toString());  
        double couvert = Double.parseDouble(edtCouvert.getText().toString());  
        int qtdPessoas = Integer.parseInt(edtConta.getText().toString());  
        double taxa = (consumo + couvert) * 0.1;  
        double total = consumo + taxa + couvert;
```

```
        edtTaxa.setText(String.format("%.2f", taxa));  
        edtTotal.setText(String.format("%.2f", total));  
        edtPessoa.setText(String.format("%.2f", total/qtdPessoas));
```

```
    }
```

```
});
```

```
}
```

Recursos de String

79

Uma string é um recurso simples que é referenciado usando o valor fornecido no atributo **name** (e não no nome do arquivo XML). Dessa forma, é possível combinar recursos de string com outros recursos simples em um arquivo XML, em um elemento **<resources>**.

localização do arquivo:

```
res/values/filename.xml
```

O nome do arquivo é arbitrário. O `name` do elemento `<string>` é usado como o ID do recurso.

tipo de dados do recurso compilado:

Ponteiro do recurso para um `String`.

referência de recurso:

Em Java: `R.string.string_name`

Em XML: `@string/string_name`

Recursos de String

80

exemplo:

Arquivo XML salvo em `res/values/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello!</string>
</resources>
```



Esse XML de layout aplica uma string em uma visualização:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```



Recursos de String/Arrays

81

exemplo:

Arquivo XML salvo em `res/values/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>
```

Esse código do aplicativo extrai uma matriz de strings:

Kotlin Java

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

Componentes – Caixa de Diálogo

82

// MainActivity

```
public void abrirDialogo(View view) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Exclusão");
    builder.setMessage("Deseja excluir?");
    builder.setPositiveButton("SIM", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(MainActivity.this, "Você excluiu", Toast.LENGTH_SHORT).show();
            return;
        }
    });

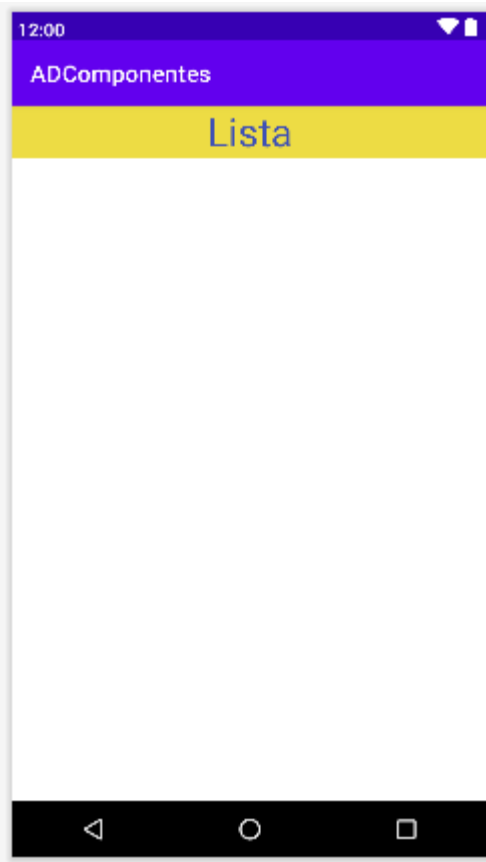
    builder.setNegativeButton("Não", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(MainActivity.this, "Que Pena!!!!", Toast.LENGTH_SHORT).show();
            return;
        }
    });

    AlertDialog dialogo = builder.create();
    dialogo.show();
}
```



Componentes - ListView

83



```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/lblTitulo"/>

    <ListView
        android:id="@+id/lvDados"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Componentes - ListView

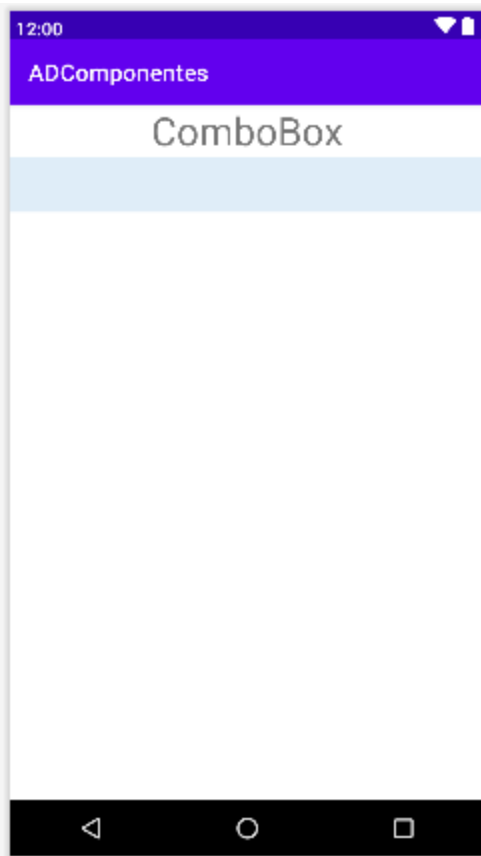
84

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_tela1);  
  
    //Componente Visual de Lista  
    lvDados = findViewById(R.id.lvDados);  
  
    //Fonte de Dados  
    String[] nomes = {"Fulano", "Ciclano", "Beltrano", "Astrolabio", "Astrogildo", "Xincrano",  
        "Josefina", "Roque"};  
    //Adaptador do ListView  
    ArrayAdapter<String> adaptador = new ArrayAdapter<String>(  
        this, android.R.layout.simple_list_item_1, nomes);  
    //Relacionando o adaptador ao componente ListView  
    lvDados.setAdapter(adaptador);  
  
    //implementacao do evento de click do item  
    lvDados.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
        @Override  
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
            // recebe o nome da lista  
            String nome = String.valueOf(parent.getAdapter().getItem(position));  
            Toast.makeText(Tela1.this, nome, Toast.LENGTH_SHORT).show();  
        }  
    });  
}
```

Componentes - Spinner

85



```
<LinearLayout  
    android:orientation="vertical">  
  
    <TextView  
        android:id="@+id/textView"/>  
  
    <Spinner  
        android:id="@+id/spiDados"/>  
</LinearLayout>
```

Componentes - Spinner

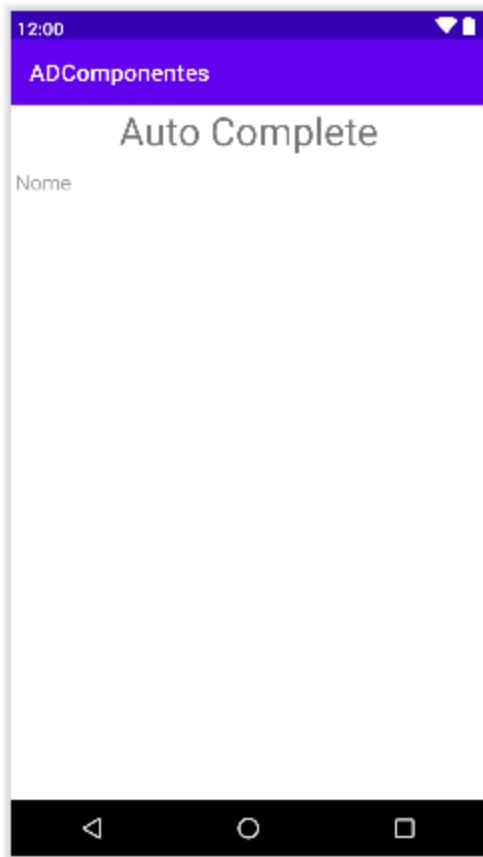
86

`@Override`

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_tela2);  
    //Componente Visual (Combobox - Spinner)  
    spiDados = (Spinner) findViewById(R.id.spiDados);  
    //Fonte de Dados  
    String[] nomes = {"Fatec São Paulo", "Fatec Garulhos", "Fatec Barueri"};  
    //Adaptador do Spinner  
    ArrayAdapter<String> adaptador = new ArrayAdapter<String>(  
        this, android.R.layout.simple_list_item_1, nomes  
    );  
    //Relaciona o Adaptador ao componente Spinner  
    spiDados.setAdapter(adaptador);  
    //Implementando o Evento de Seleção do Item  
    spiDados.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
        @Override  
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
            String nome = String.valueOf(parent.getAdapter().getItem(position));  
  
            Toast.makeText(Tela2.this, nome, Toast.LENGTH_SHORT).show();  
        }  
        @Override  
        public void onNothingSelected(AdapterView<?> parent) {  
        }  
    });  
}
```

Componentes - AutoComplete

87



```
<LinearLayout  
    android:orientation="vertical">  
  
    <TextView  
        android:id="@+id/textView3"/>  
  
    <AutoCompleteTextView  
        android:id="@+id/actNome"/>  
</LinearLayout>
```

Componentes - AutoComplete

88

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_tela3);  
  
    actNome = findViewById(R.id.actNome);
```

//Fonte de Dados

```
Resources res = getResources();  
String[] nomes = res.getStringArray(R.array.nomes_array);
```

//Adaptador do Auto Complete

```
ArrayAdapter<String> adaptador = new ArrayAdapter<String>(  
    this, android.R.layout.simple_list_item_1, nomes  
);
```

//Relaciona o adaptador ao componente AutoComplete

```
actNome.setAdapter(adaptador);  
actNome.setThreshold(1);  
}
```

```
<resources>  
    <string name="app_name">ADComponentes</string>  
    <string name = "txtTitulo">Componentes</string>  
  
    <string-array name="nomes_array">  
        <item>Fulano</item>  
        <item>Ciclano</item>  
        <item>Beltrano</item>  
    </string-array>  
</resources>
```


Gravando/Lendo um Arquivo Texto

89



```
<LinearLayout
    android:orientation="vertical">
    <TextView
        android:text="Leitor de Arquivo Texto"/>

    <EditText
        android:id="@+id/edtNome" />

    <EditText
        android:id="@+id/edtEndereco"/>

    <EditText
        android:id="@+id/edtEmail"/>

    <Button
        android:id="@+id/btnSalvar" />

    <Button
        android:id="@+id/btnRecuperar"/>

    <Button
        android:id="@+id/btnLimpar"/>
</LinearLayout>
```

Gravando um Arquivo Texto

90

```
public void salvar(View view){
    // botao salvar
    try {
        //criar o arquivo - somente a sua aplicacao pode acessar esta informacao
        FileOutputStream arquivo = openFileOutput(edtNome.getText().toString()+".txt",
Context.MODE_PRIVATE);
        //criando o fluxo
        OutputStreamWriter fluxo = new OutputStreamWriter(arquivo);
        // criando a classe para gravar os dados
        PrintWriter out = new PrintWriter(fluxo);
        out.println(edtNome.getText().toString());
        out.println(edtEndereco.getText().toString());
        out.println(edtEmail.getText().toString());
        Toast.makeText(getApplicationContext(), "Gravado com sucesso",Toast.LENGTH_LONG).show();
        // fechando os objetos
        out.close();
        arquivo.close();
        fluxo.close();
    }catch(Exception e){
        Toast.makeText(getApplicationContext(),"Erro ao gravar o arquivo",Toast.LENGTH_LONG).show();
    }
}
```

Lendo um Arquivo Texto

91

```
public void recuperar(View view){  
  
    // recuperar dados  
    try {  
        // nome do arquivo que  
        FileInputStream arquivo = openFileInput(edtNome.getText().toString() + ".txt");  
        // criar o fluxo de entrada de dados  
        InputStreamReader fluxo = new InputStreamReader(arquivo);  
        // criando um leitor de dados  
        BufferedReader in = new BufferedReader(fluxo);  
        // leitura dos dados  
        edtNome.setText(in.readLine());  
        edtEndereco.setText(in.readLine());  
        edtEmail.setText(in.readLine());  
        Toast.makeText(getApplicationContext(), "Lido com sucesso!", Toast.LENGTH_LONG).show();  
    } catch (Exception e) {  
        Toast.makeText(getApplicationContext(), "Erro ao ler o arquivo", Toast.LENGTH_LONG).show();  
    }  
}
```

Envio de Email

92



```
<LinearLayout
```

```
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">
```

```
<TextView
```

```
    android:id="@+id/textView" />
```

```
<EditText
```

```
    android:id="@+id/edtDestinatario" />
```

```
<EditText
```

```
    android:id="@+id/edtAssunto"/>
```

```
<EditText
```

```
    android:id="@+id/edtMensagem" />
```

```
<Button
```

```
    android:id="@+id/btnEnviar"/>
```

```
<Button
```

```
    android:id="@+id/btnSair"/>
```

```
</LinearLayout>
```

Envio de Email

93

```
public void enviar(View view){
    destinatario = txtDestinatario.getText().toString();
    assunto = txtAssunto.getText().toString();
    mensagem = txtMensagem.getText().toString();
    // abre tela do email
    intent = new Intent(Intent.ACTION_SEND);

    intent.putExtra(intent.EXTRA_EMAIL, new String[]{destinatario});
    intent.putExtra(intent.EXTRA_SUBJECT, assunto);
    intent.putExtra(intent.EXTRA_TEXT, mensagem);
    // rfc822 padrao mundial de mensagem
    intent.setType("message/rfc822");
    startActivity(intent.createChooser(intent, "Selecione um aplicativo"));
}
```

Envio de Email

94

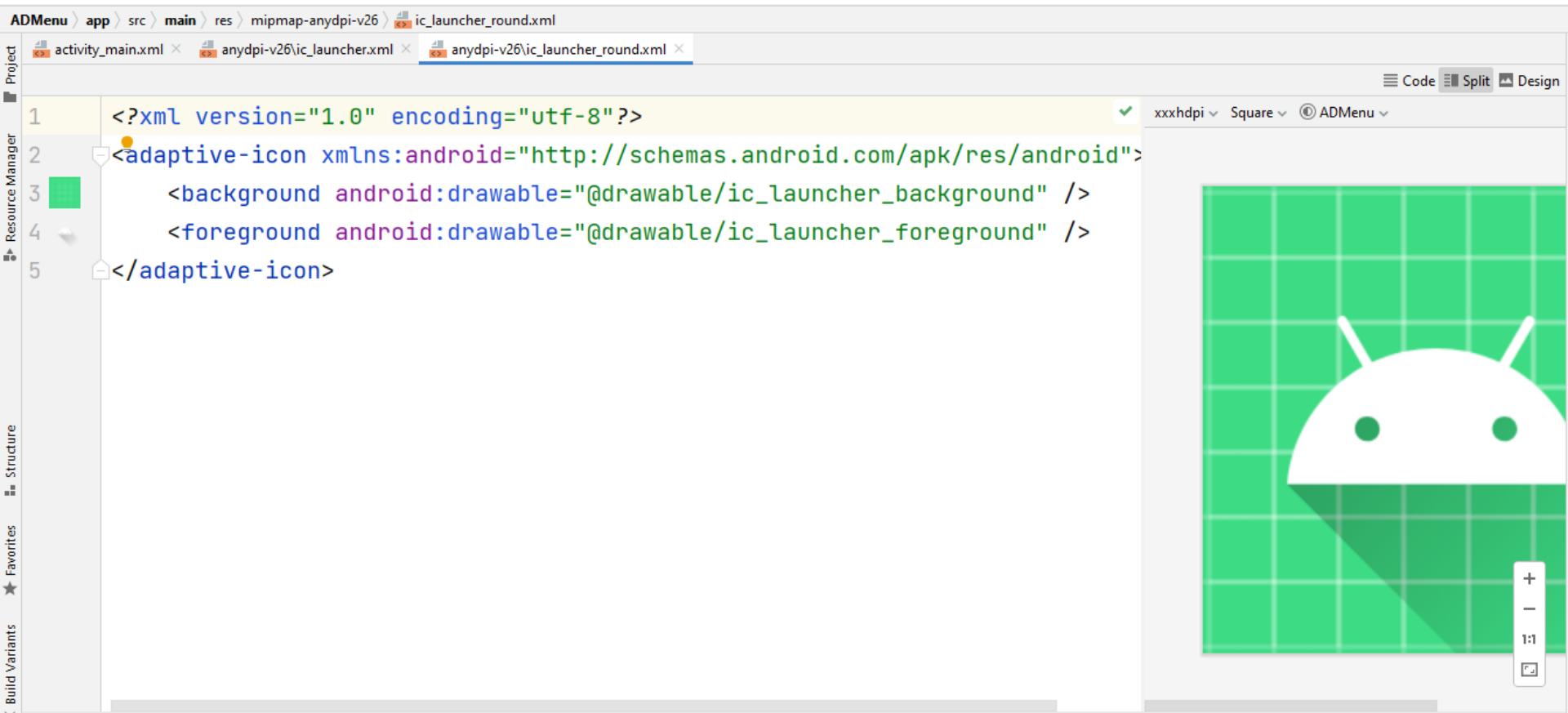
@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    txtAssunto = findViewById(R.id.edtAssunto);  
    txtMensagem = findViewById(R.id.edtMensagem);  
    txtDestinatario = findViewById(R.id.edtDestinatario);  
    btnEnviar = findViewById(R.id.btnEnviar);  
    btnSair = findViewById(R.id.btnSair);
```

```
    btnSair.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View view) {  
            finish();  
        }  
    });  
}
```

Alterando o ícone do Android

95

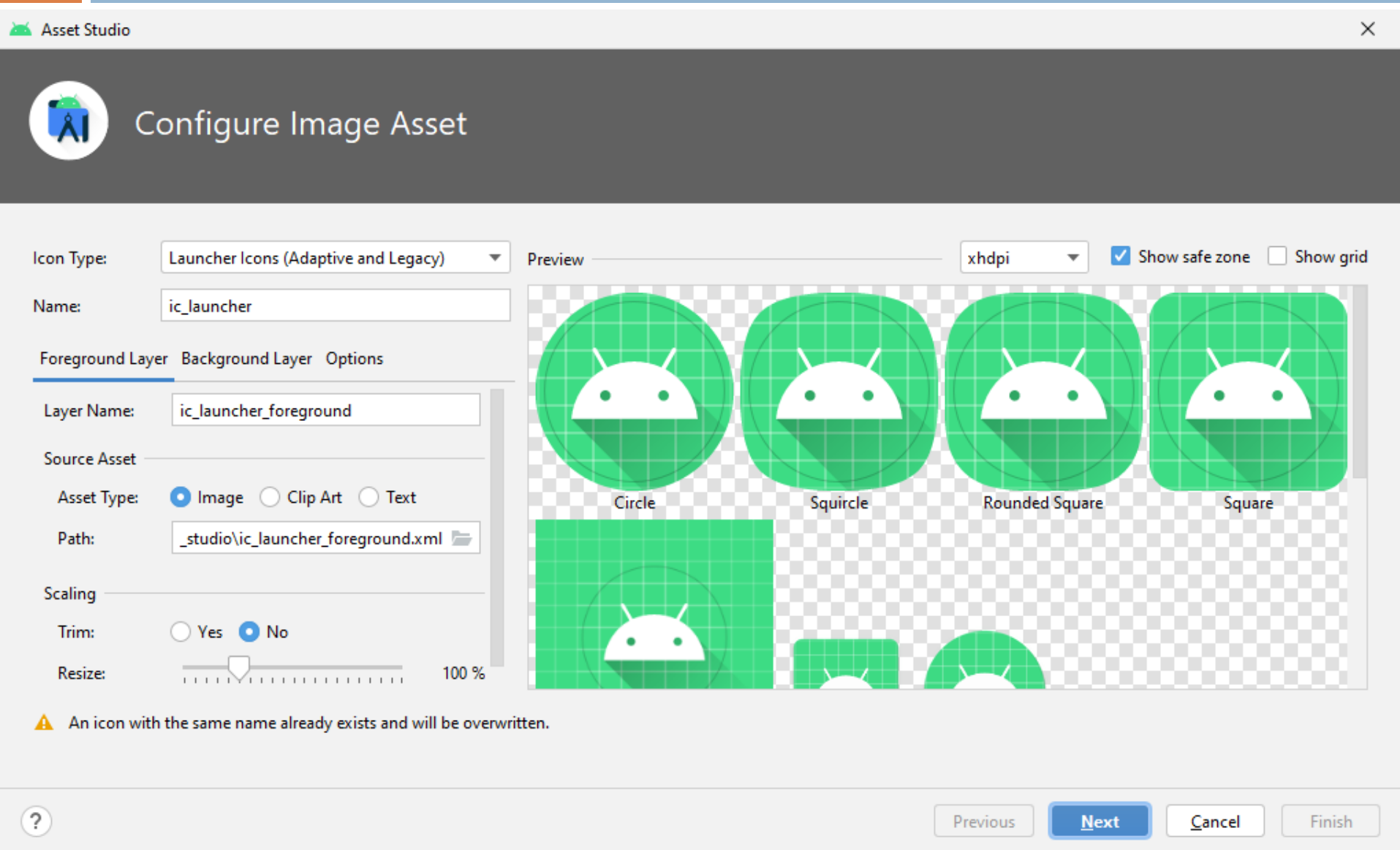


Alterando o ícone do Android

96

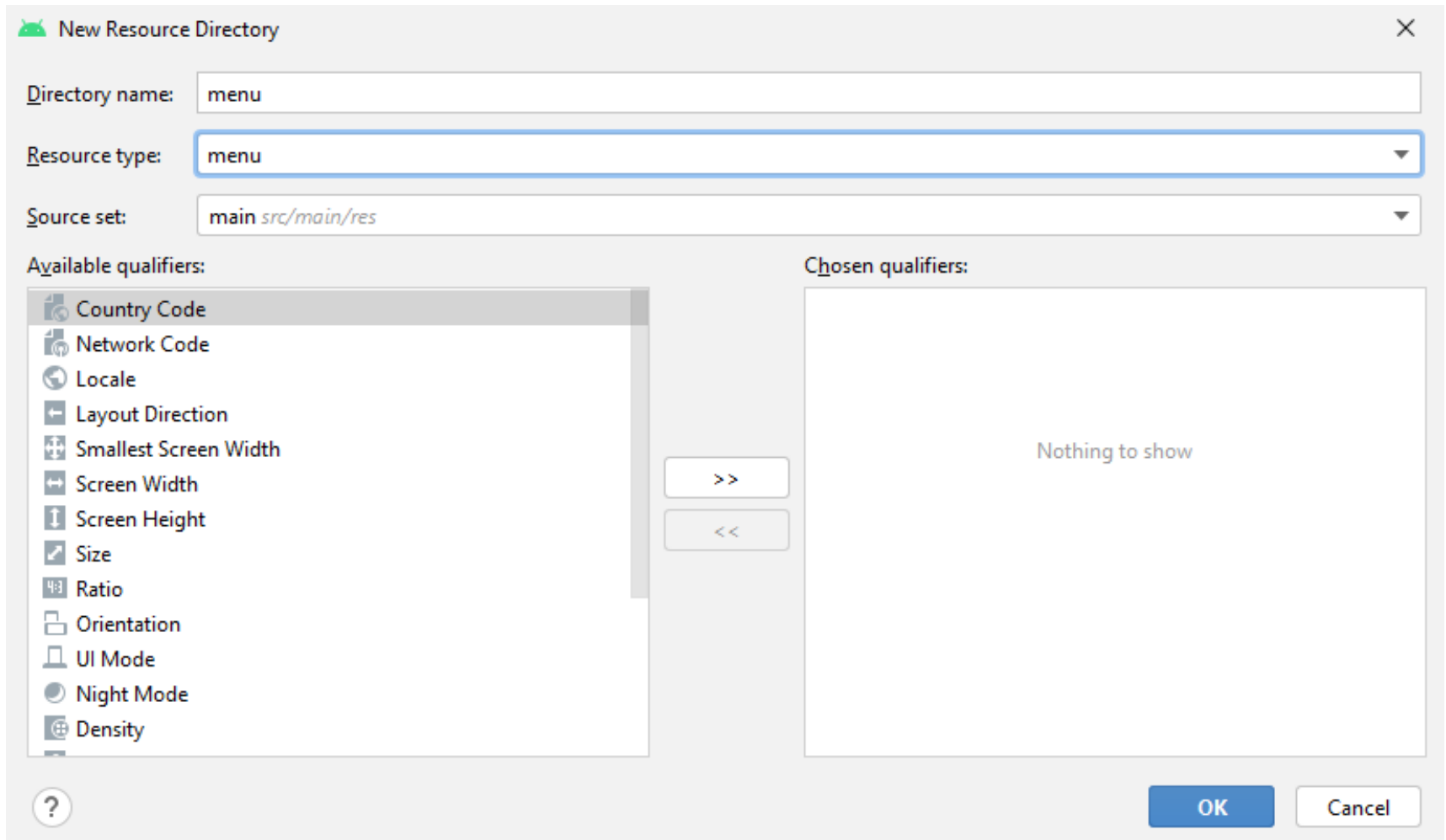
Alterando o ícone do Android

97




Menu

98



Menu

99

 New Resource File ✕

File name:

Resource type:

Root element:

Source set:

Directory name:

Available qualifiers:

Country Code

Network Code

Locale

Layout Direction

Smallest Screen Width

Screen Width

Screen Height

Size

Ratio

Orientation

>>

<<

Chosen qualifiers:

Nothing to show

?

OK

Cancel

Menu

100

ADMenu > app > src > main > res > menu > menu_principal.xml

activity_main.xml x menu_principal.xml x

Code Split Design

Palette

- Cast Button
- Menu Item
- Search Item
- Switch Item
- Menu
- Group

Component Tree

- menu
 - Salvar
 - Editar
 - Alterar
 - itemSair

Attributes

item <unnamed>

Declared Attributes

id	itemSair
title	Sair

Transforms

Common Attributes

id	itemSair
title	Sair
icon	
showAsAction	
visible	
enabled	
checkable	

All Attributes

actionLayout	
actionProviderCl...	
actionProviderCl...	
actionViewClass	
actionViewClass	
alphabeticShortc...	

Menu

101

```
public boolean onCreateOptionsMenu(Menu menu){
    MenuInflater i = getMenuInflater();
    i.inflate(R.menu.menu_principal,menu);
    SearchView sv = (SearchView) menu.findItem(R.id.menuBarConsultar).getActionView();
    sv.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String s) {
            System.out.println("Digitou "+s);
            return false;
        }
        @Override
        public boolean onQueryTextChange(String s) {
            System.out.println("Digitou "+s);
            return false;
        }
    });

    return true;
}

public void mostrarSalvar(MenuItem item){
    Toast.makeText(getApplicationContext(),"Menu Salvar",Toast.LENGTH_LONG).show();
}

public void mostrarEditar(MenuItem item){
    Toast.makeText(getApplicationContext(),"Menu Editar",Toast.LENGTH_LONG).show();
}

public void mostrarSair(MenuItem item){
    finish();
}
```

Classe ConnectionFactory

102

```
public class ConnectFactory extends SQLiteOpenHelper {

    private static final String NAME = "banco.db";
    private static final int VERSION = 1;

    public Conexao(@Nullable Context context) {
        super(context, NAME, null, VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table aluno(id integer primary key autoincrement, "+
            "nome varchar(50), cpf varchar(50), telefone varchar(50))");
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int i1) {
        String sql = "DROP TABLE IF EXISTS aluno";
        db.execSQL(sql);
        onCreate(db);
    }
}
```

Classe Javabeau

103

```
public class Aluno {  
    private Integer id;  
    private String nome;  
    private String cpf;  
    private String telefone;  
    // construtor  
    public Aluno(Integer id, String nome, String cpf, String telefone) {  
        this.id = id;  
        this.nome = nome;  
        this.cpf = cpf;  
        this.telefone = telefone;  
    }  
    public Aluno(){  
    } // getters e setters  
    public Integer getId() {  
        return id;  
    }  
  
    public void setId(Integer id) {  
        this.id = id;  
    }  
}
```

Classe AlunoDAO (construtor)

104

```
public class AlunoDAO {  
    private ConnectionFactory conexao;  
    private SQLiteDatabase banco;  
    public AlunoDAO(Context context){  
        //ConnectionFactory com o banco de dados  
        conexao = new ConnectionFactory(context);  
        banco = conexao.getWritableDatabase();  
    }  
}
```


Classe AlunoDAO(insert)

105

// método inserir

```
public long insert(Aluno aluno){  
    ContentValues values = new ContentValues();  
    values.put("nome", aluno.getNome());  
    values.put("cpf", aluno.getCpf());  
    values.put("telefone", aluno.getTelefone());  
    return(banco.insert("aluno", null, values));  
}
```

Classe AlunoDAO(update)

106

// método alterar

```
public void update(Aluno aluno){  
    ContentValues values = new ContentValues();  
    values.put("nome", aluno.getNome());  
    values.put("cpf", aluno.getCpf());  
    values.put("telefone", aluno.getTelefone());  
    String args[] = {aluno.getId().toString()};  
    banco.update("aluno", values, "id=?",args);  
}
```

Classe AlunoDAO(delete)

107

// Méthode Exclure

```
public void delete(Aluno aluno){  
    String args[] = {aluno.getId().toString()};  
    banco.delete("aluno","id=?",args);  
}
```

Classe AlunoDAO(list)

108

```
public List<Aluno> obterTodos() {  
    List<Aluno> alunos = new ArrayList<>();  
    Cursor cursor = banco.query("aluno", new String[]{"id", "nome", "cpf",  
"telefone"},  
        null, null, null, null, null);  
    while (cursor.moveToNext()) {  
        Aluno a = new Aluno();  
        a.setId(cursor.getInt(0));  
        a.setNome((cursor.getString(1)));  
        a.setCpf((cursor.getString(2)));  
        a.setTelefone((cursor.getString(3)));  
        alunos.add(a);  
    }  
    return alunos;  
}
```

Classe AlunoDAO(read)

109

```
public Aluno read(Integer id) {  
    String args[] = {String.valueOf(id)};  
    Cursor cursor = banco.query("aluno", new String[]{"id", "nome", "cpf", "telefone"},  
        "id=?", args, null, null, null);  
    cursor.moveToFirst();  
    Aluno aluno = new Aluno();  
    if(cursor.getCount() > 0){  
        aluno.setId(cursor.getInt(0));  
        aluno.setNome((cursor.getString(1)));  
        aluno.setCpf((cursor.getString(2)));  
        aluno.setTelefone((cursor.getString(3)));  
    }  
    return aluno;  
}
```

Classe MainActivity

110

```
public class MainActivity extends AppCompatActivity {
```

```
    private EditText edtNome;  
    private EditText edtCpf;  
    private EditText edtTelefone;  
    private EditText edtListar;  
    private AlunoDAO dao;  
    private List<Aluno> alunos;
```

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    edtNome = findViewById(R.id.edtNome);  
    edtCpf = findViewById(R.id.edtCpf);  
    edtTelefone = findViewById(R.id.edtTelefone);  
    edtListar = findViewById(R.id.edtListar);  
}
```

Classe MainActivity

111

```
public void limpar(View view){  
    edtNome.setText(null);  
    edtCpf.setText(null);  
    edtTelefone.setText(null);  
    edtListar.setText(null);  
}
```

```
public void salvar(View view){  
    Aluno a = new Aluno();  
    a.setNome(edtNome.getText().toString());  
    a.setCpf(edtCpf.getText().toString());  
    a.setTelefone(edtTelefone.getText().toString());  
    dao = new AlunoDAO(this);  
    long id = dao.insert(a);  
    Toast.makeText(getApplicationContext(), "Aluno inserido com o ID "+id,  
    Toast.LENGTH_LONG).show();  
}
```

Classe MainActivity

112

```
public void listar(View view){
    dao = new AlunoDAO(this);
    alunos = dao.obterTodos();
    for (Aluno aluno : alunos) {
        edtListar.append("ID      : " + aluno.getId() + "\n");
        edtListar.append("Nome    : " + aluno.getNome() + "\n");
        edtListar.append("CPF     : " + aluno.getCpf() + "\n");
        edtListar.append("Telefone: " + aluno.getTelefone() + "\n");
    }
}

public void proxima(View view){
    Intent it = new Intent(this, Manutencao.class);
    startActivity(it);
}
}
```


Classe Manutencao

113

```
public class Manutencao extends AppCompatActivity {
```

```
    private EditText edtNome;  
    private EditText edtCpf;  
    private EditText edtTelefone;  
    private EditText edtId;  
    private AlunoDAO dao;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_manutencao);  
  
        edtNome = findViewById(R.id.edtNome);  
        edtCpf = findViewById(R.id.edtCpf);  
        edtTelefone = findViewById(R.id.edtTelefone);  
        edtId = findViewById(R.id.edtId);  
    }
```

Classe Manutencao (update)

114

```
public void alterar(View view){
    Aluno a = new Aluno();
    a.setId(Integer.parseInt(edtId.getText().toString()));
    a.setNome(edtNome.getText().toString());
    a.setCpf(edtCpf.getText().toString());
    a.setTelefone(edtTelefone.getText().toString());
    dao = new AlunoDAO(this);
    dao.update(a);
    Toast.makeText(getApplicationContext(), "Aluno alterado! ",
    Toast.LENGTH_LONG).show();
}
```

Classe Manutencao (read)

115

```
public void consultar(View view){  
    dao = new AlunoDAO(this);  
    Aluno a = dao.read(Integer.parseInt(edtId.getText().toString()));  
    edtNome.setText(a.getNome());  
    edtCpf.setText(a.getCpf());  
    edtTelefone.setText(a.getTelefone());  
}
```

Classe Manutencao

116

```
public void excluir(View view){
    Aluno a = new Aluno();
    a.setId(Integer.parseInt(edtId.getText().toString()));
    dao = new AlunoDAO(this);
    dao.delete(a);
    Toast.makeText(getApplicationContext(), "Aluno Excluido!",
    Toast.LENGTH_LONG).show();
}
```

Classe Manutencao (voltar)

117

```
public void voltar(View view){  
    finish();  
}
```