



Android QuickStart



Android

Introdução

- Sistema Operacional e Plataforma de Desenvolvimento para Dispositivos Móveis com maior taxa de crescimento
- FOSS (*Free Open Source Software*)
 - Baseado no Linux
- Programado na Linguagem Java
 - Permite aplicações nativas em C e C++



Android

Introdução

- Última versão: Android 5.0 (Lollipop)

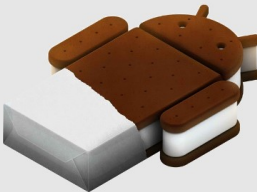
- Interface melhorada e padronizada
- Performance
- Notificações melhoradas
- Suporte a TV



KitKat
(4.4)



Jelly Bean
(4.1-4.3)



Ice Cream
Sandwich
(4.0)

- Últimas notícias

- 06/12/2014: *Android Studio 1.0*
 - *Novo IDE para o Android*
- 06/01/2014: *Open Automotive Alliance*
 - *Google, Audi, General Motors, Hyundai e Honda*
 - *New roads ahead for Android and the Open Automotive Alliance*

Android

Passos para Desenvolvimento

Configuração

Instalação do
Android Studio

Hello World
App

Configuração
do AVD

Execução
do App

Desenvolvimento

Interface
Gráfica

*Activities e
Intents*

Banco de
Dados

Tópicos
Avançados

Android

Passos para Desenvolvimento

Configuração

**Instalação do
Android Studio**

Hello World
App

Configuração
do AVD

Execução
do App

Desenvolvimento

Interface
Gráfica

*Activities e
Intents*

Banco de
Dados

Tópicos
Avançados

Android

Instalação do Android Studio (Linux)

- Android Studio:
 - Modo recomendado de desenvolvimento desde 12/2014
 - *Substitui o antigo Eclipse ADT*
 - Contêm:
 - *Android SDK Tools*
 - *Android Platform-tools*
 - *Imagem do Sistema Android para o Emulator*
- Download do Android Studio:
 - <http://developer.android.com/sdk/>
 - *Download Android Studio*
 - *Linux*
 - *android-studio-ide-xxx.xxxxxxx-linux.zip (233MB)*
- Instalando e executando (terminal):

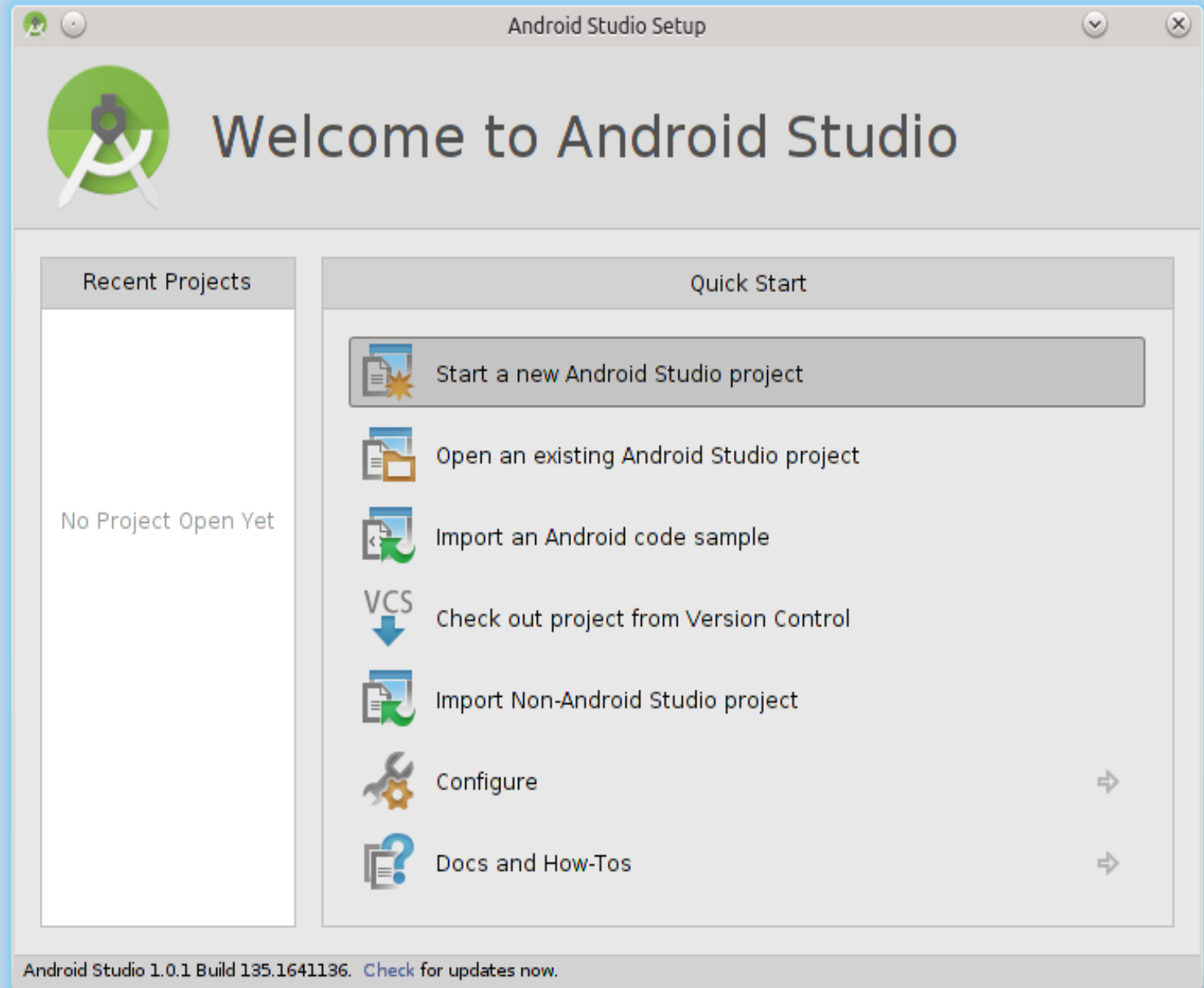
```
# unzip android-studio-ide-xxx.xxxxxxx-linux.zip
# cd android-studio/bin/
# ./studio.sh
```

 - *Na primeira execução, serão baixados, instalados e configurados alguns componentes*

Android

Iniciando o Android Studio

- Tela Inicial do Android Studio



Android

Passos para Desenvolvimento

Configuração

Instalação do
Android Studio

Hello World
App

Configuração
do AVD

Execução
do App

Desenvolvimento

Interface
Gráfica

*Activities e
Intents*

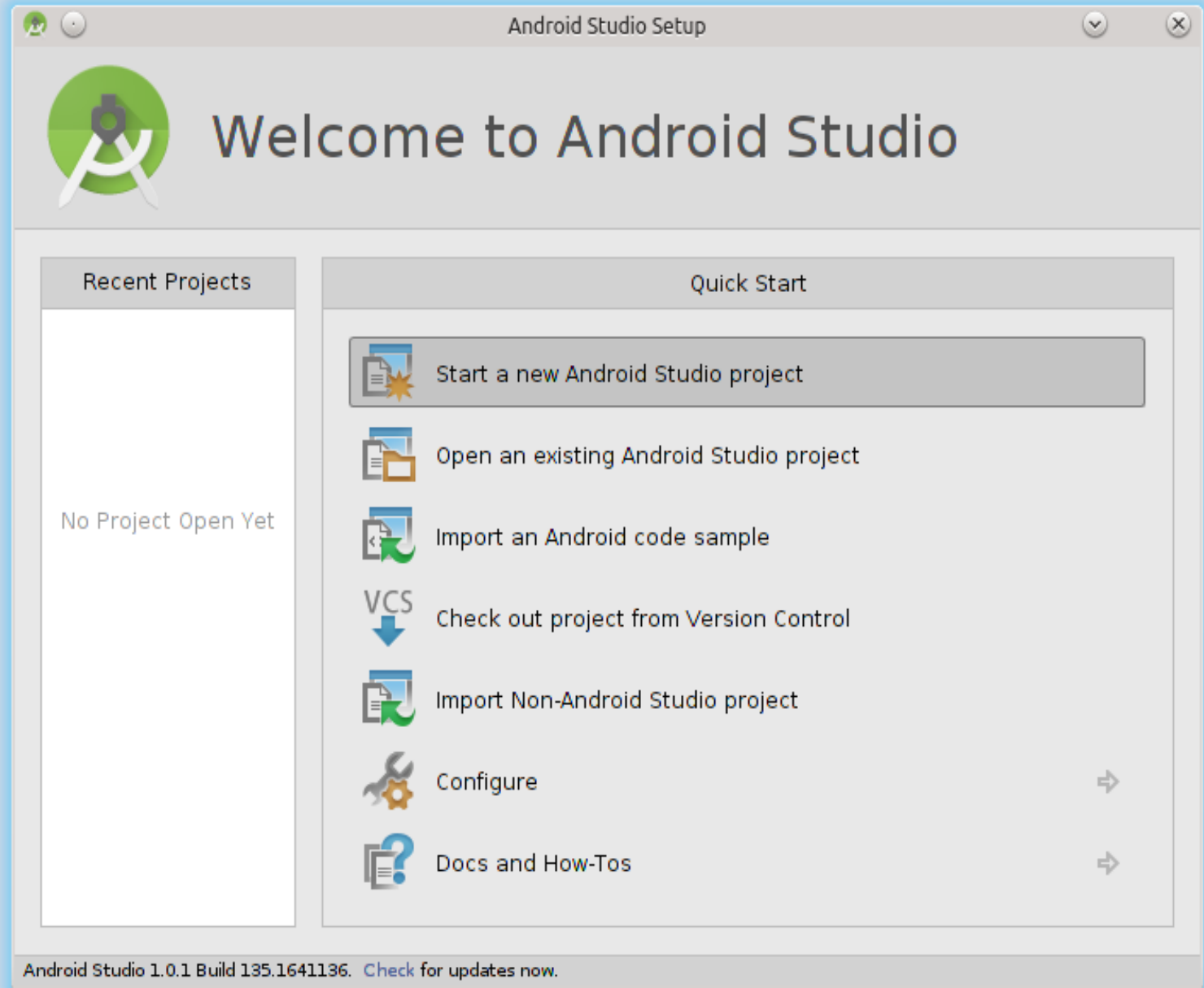
Banco de
Dados

Tópicos
Avançados

Android

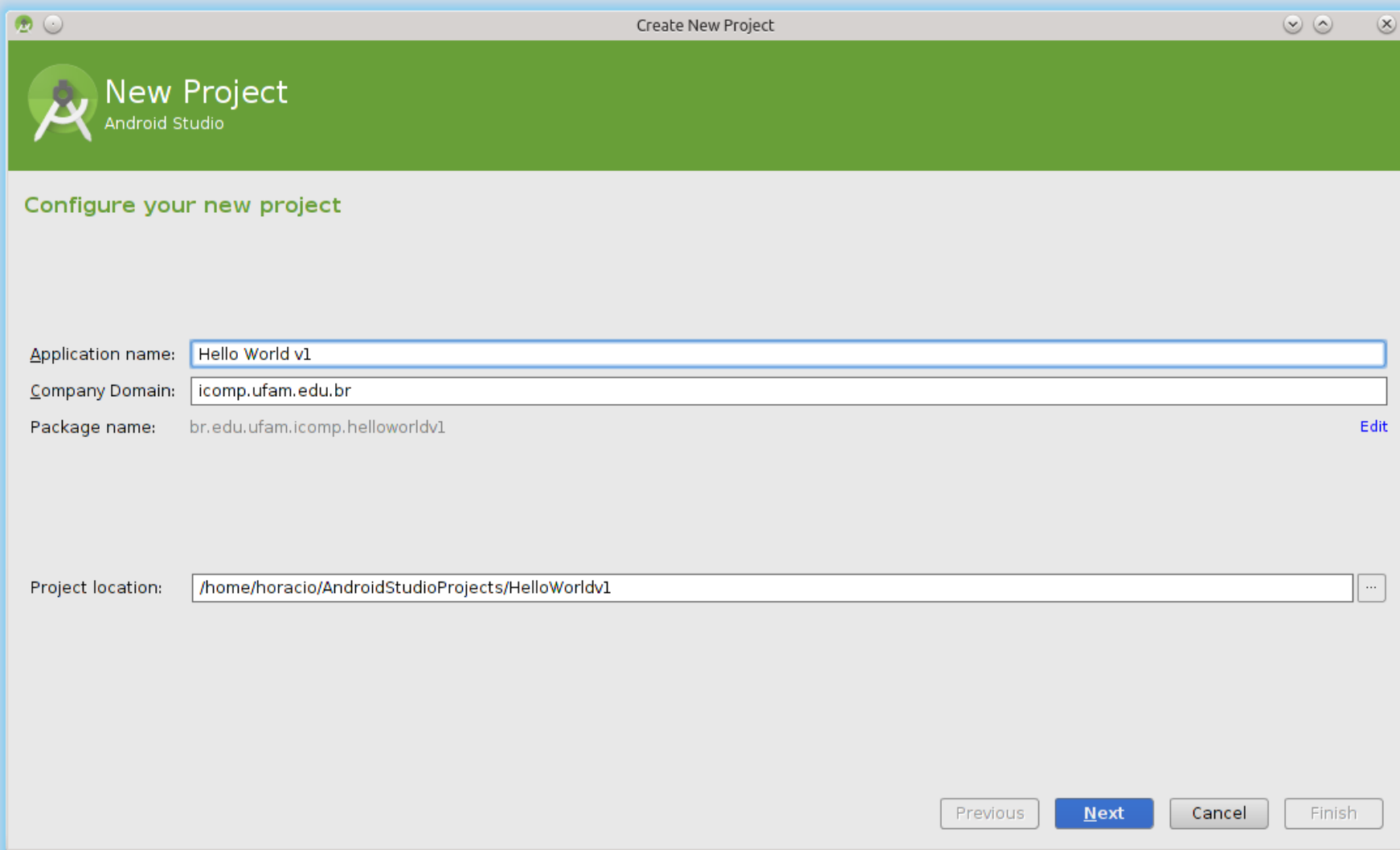
Criando uma Nova Aplicação

- Start a new Android Studio project



Android

Criando uma Nova Aplicação



The screenshot shows the 'Create New Project' dialog in Android Studio. The window has a title bar 'Create New Project' and standard OS window controls. The main area has a green header with the Android Studio logo and the text 'New Project' and 'Android Studio'. Below this is the heading 'Configure your new project'. There are four input fields: 'Application name' with the value 'Hello World v1', 'Company Domain' with 'icomp.ufam.edu.br', 'Package name' with 'br.edu.ufam.icomp.helloworldv1' and an 'Edit' link, and 'Project location' with '/home/horacio/AndroidStudioProjects/HelloWorldv1' and a file explorer icon. At the bottom are four buttons: 'Previous', 'Next' (highlighted in blue), 'Cancel', and 'Finish'.

Create New Project

New Project
Android Studio

Configure your new project

Application name: Hello World v1

Company Domain: icomp.ufam.edu.br

Package name: br.edu.ufam.icomp.helloworldv1 [Edit](#)

Project location: /home/horacio/AndroidStudioProjects/HelloWorldv1 ...

Previous Next Cancel Finish

Android

Criando uma Nova Aplicação

Create New Project

Create New Project

New Project
Android Studio

Select the form factors your app will run on

Different platforms require separate SDKs

☒ Phone and Tablet

Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available. By targeting API 15 and later, your app will run on approximately **87.9%** of the devices that are active on the Google Play Store. [Help me choose.](#)

☐ TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

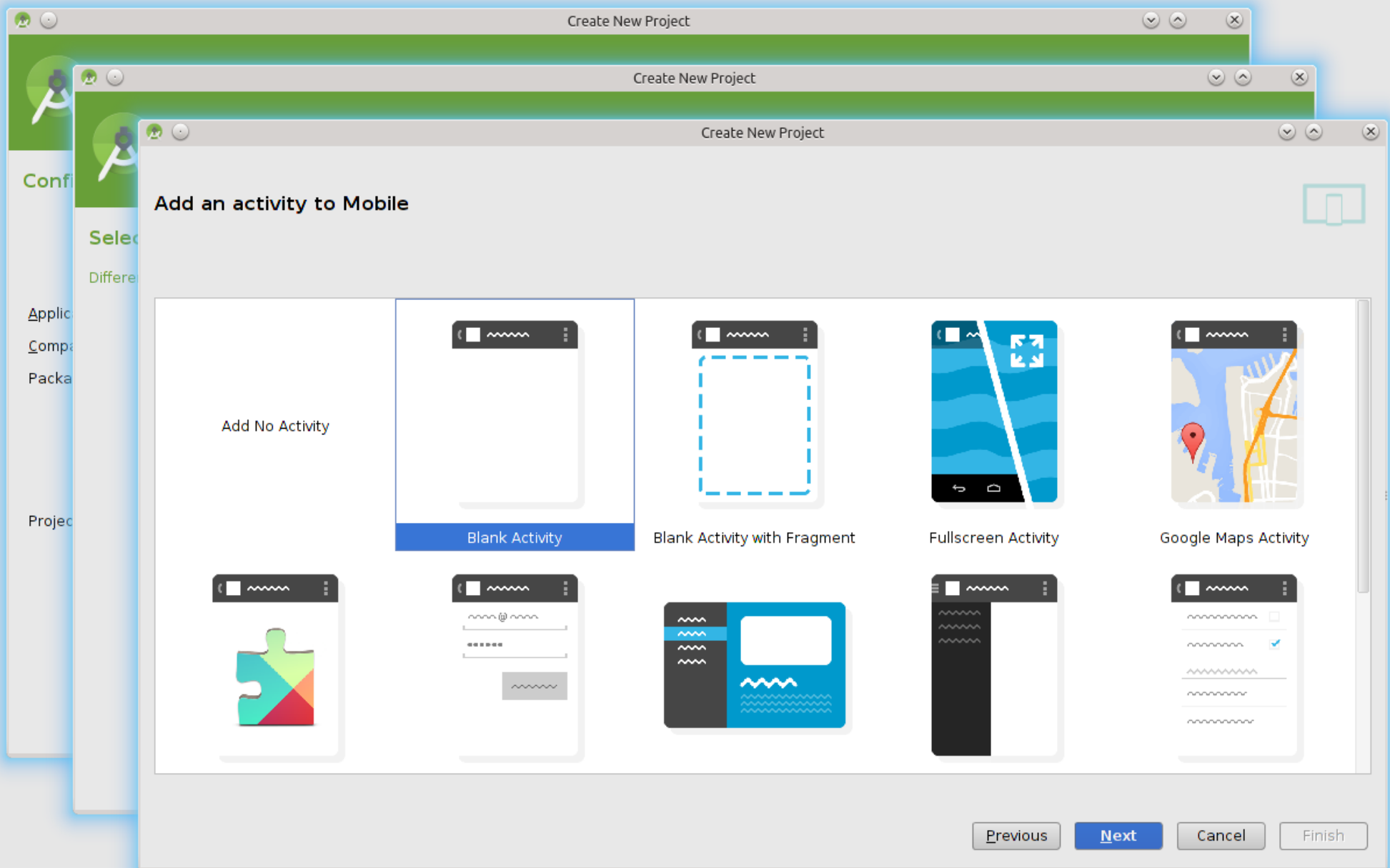
☐ Glass (Not Installed)

Minimum SDK:

Previous Next Cancel Finish

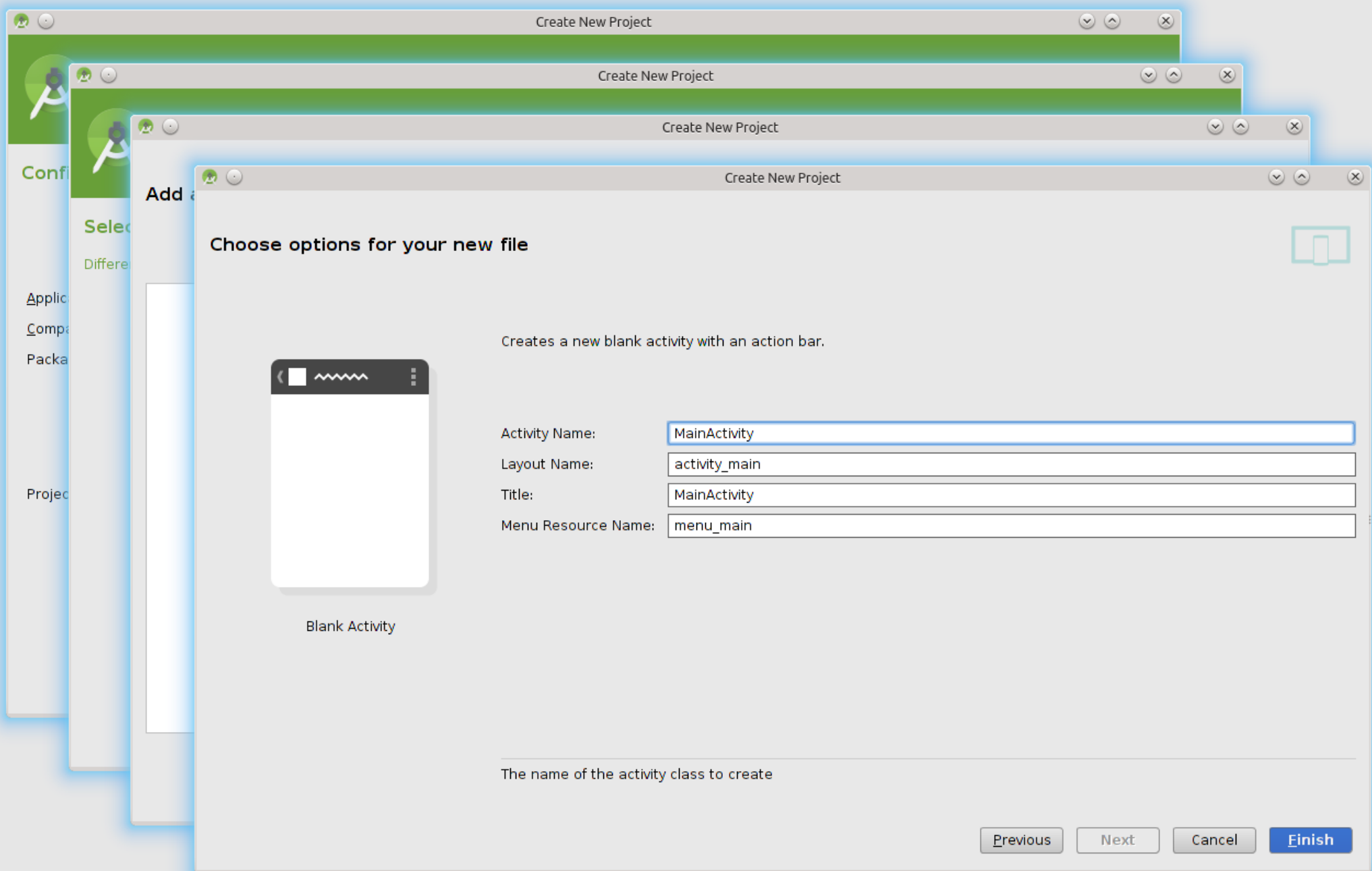
Android

Criando uma Nova Aplicação



Android

Criando uma Nova Aplicação



Android

Tela de Desenvolvimento

The image shows the Android Studio IDE interface with several annotations in green boxes:

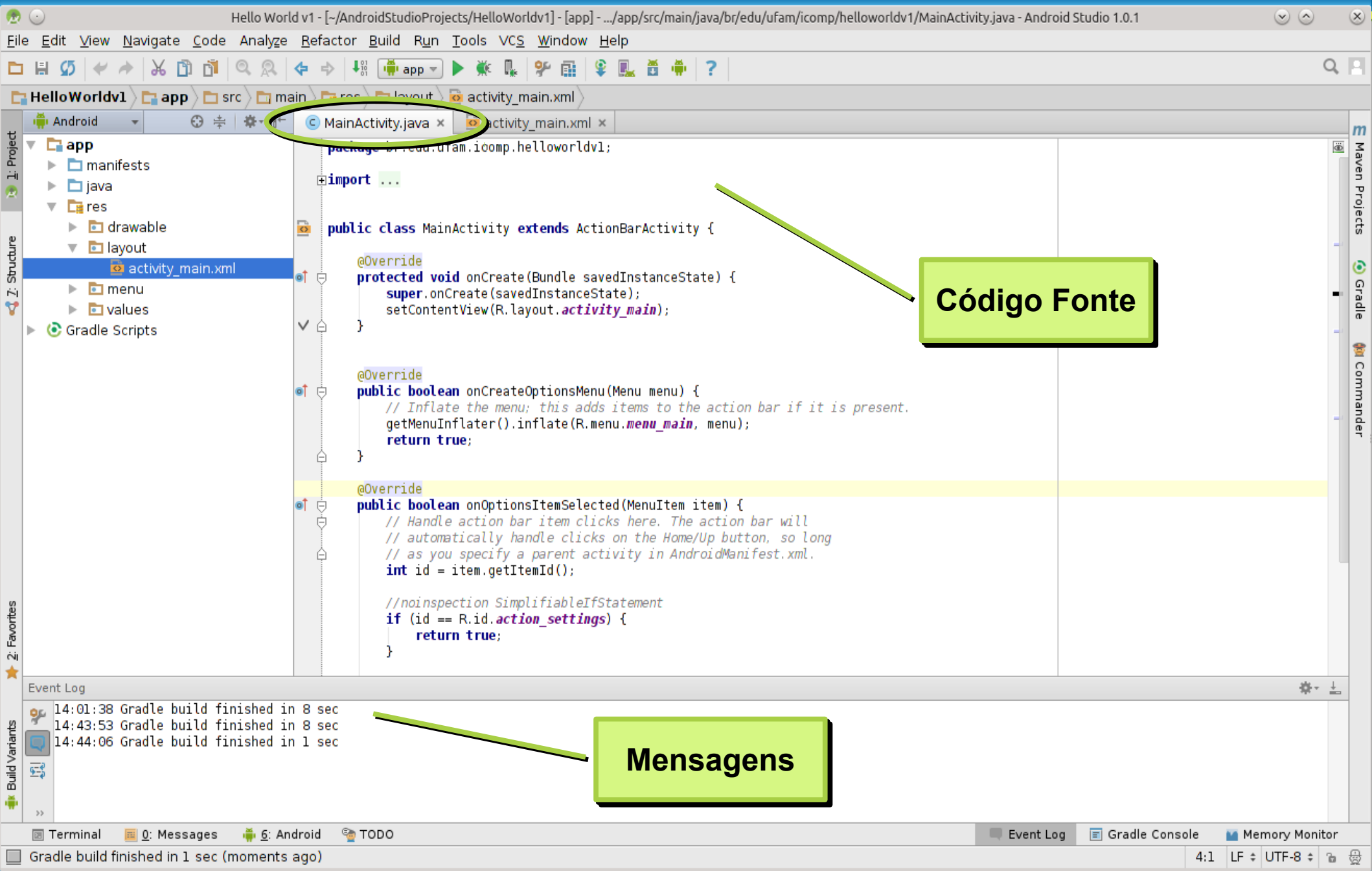
- Arquivos Fonte (classes, pacotes)**: Points to the **Project** tab on the left, showing the file structure of the app.
- Recursos (imagens, layouts)**: Points to the **res** folder in the Project tab.
- Componentes da UI**: Points to the **Widget** section in the **Palette** on the left.
- Interface Gráfica (UI)**: Points to the central **Design** view showing a mobile device screen with a "Hello world" text view.
- XML da UI**: Points to the **Text** button in the **Design** toolbar at the bottom.
- Organiz. do Layout**: Points to the **Component Tree** on the right, showing the hierarchy of the UI components.
- Props. do Componente**: Points to the **Properties** panel on the right, showing the attributes of the selected text view.

The **Properties** panel shows the following attributes for the **TextView**:

Property	Value
layout:width	wrap_content
layout:height	wrap_content
layout:margin	[]
layout:alignEnd	
layout:alignParent	<input type="checkbox"/>
layout:alignParent	<input type="checkbox"/>
layout:alignStart	
layout:toEndOf	
layout:toStartOf	
layout:align	
layout:alignPa	
layout:center	
style	

Android

Tela de Desenvolvimento



Android

Passos para Desenvolvimento

Configuração

Instalação do
Android Studio

Hello World
App

**Configuração
do AVD**

Execução
do App

Desenvolvimento

Interface
Gráfica

*Activities e
Intents*

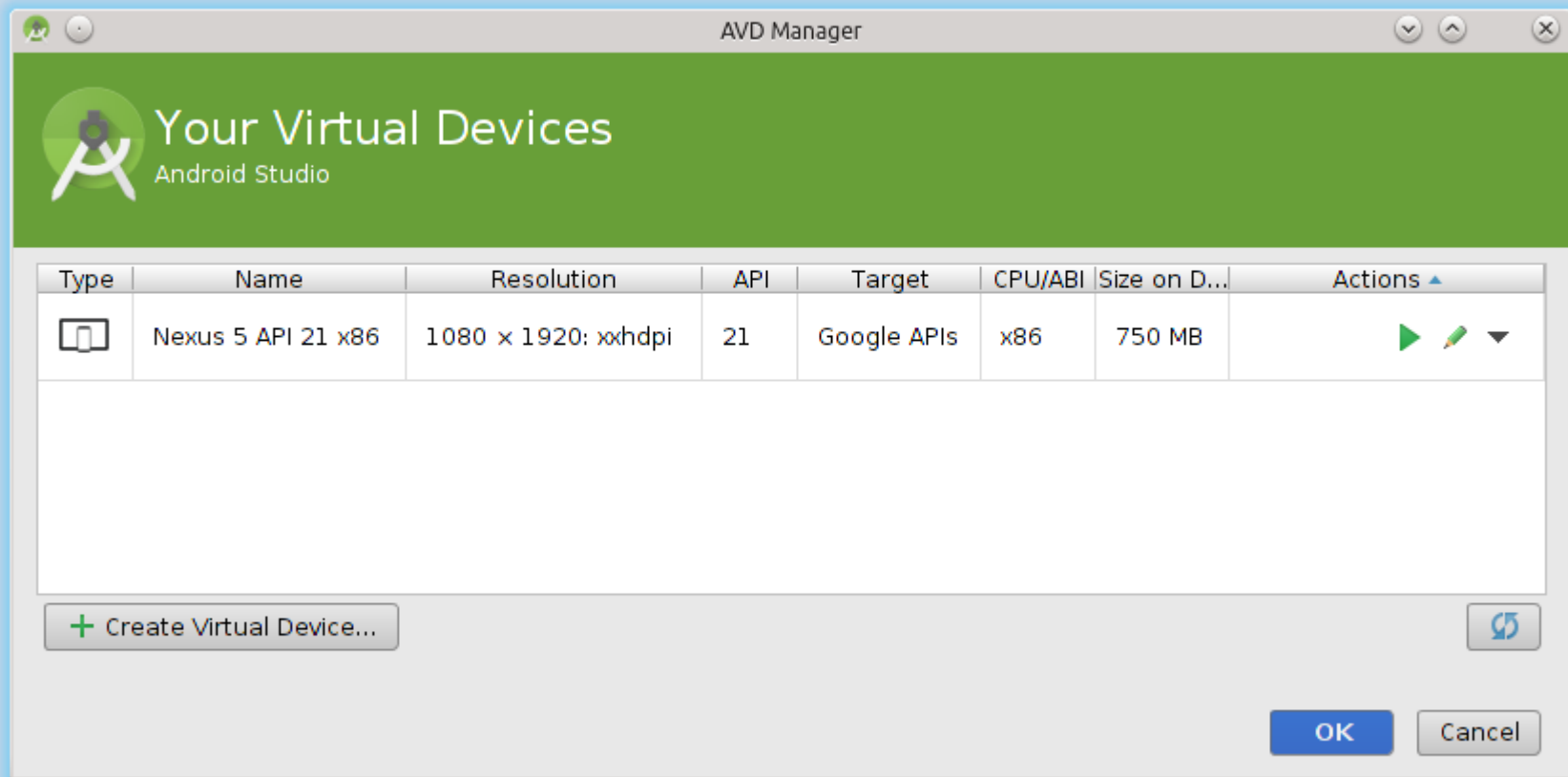
Banco de
Dados

Tópicos
Avançados

Android

Criando um AVD

- Android Virtual Device (AVD)
 - Emulador que permite executar os aplicativos em sua máquina como se fosse em um dispositivo Android real.
- Gerenciando AVDs:
 - *Tools* → *Android* → *AVD Manager*



Android

Iniciando o **AVD**

- Clique no botão “Play”
do AVD Manager
 - Botão ESC → Voltar
 - Botão HOME → Home
 - F2 → Menu



Android

Passos para Desenvolvimento

Configuração

Instalação
do ADT Bundle

Hello World
App

Configuração
do AVD

**Execução
do App**

Desenvolvimento

Interface
Gráfica

*Activities e
Intents*

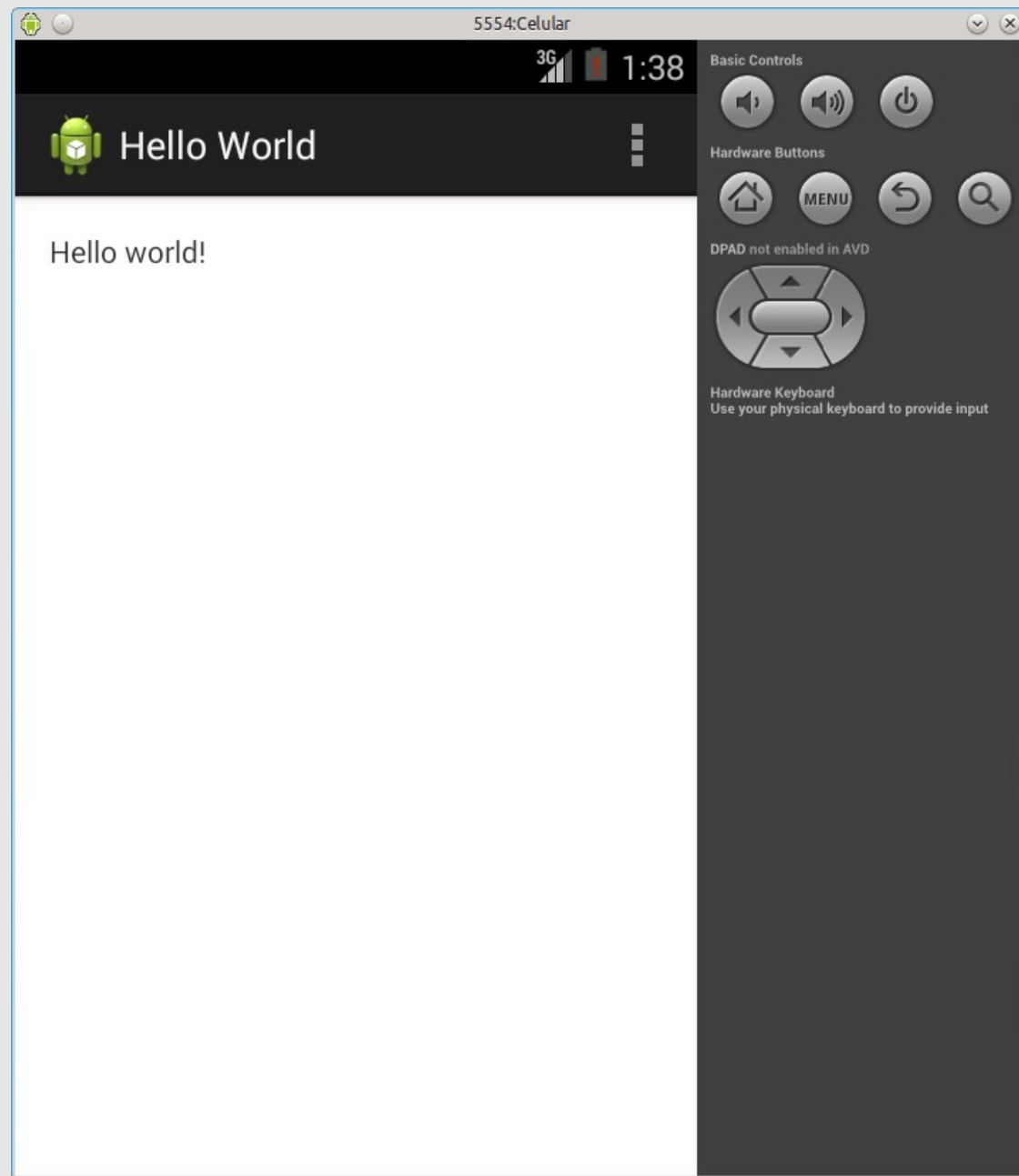
Banco de
Dados

Tópicos
Avançados

Android

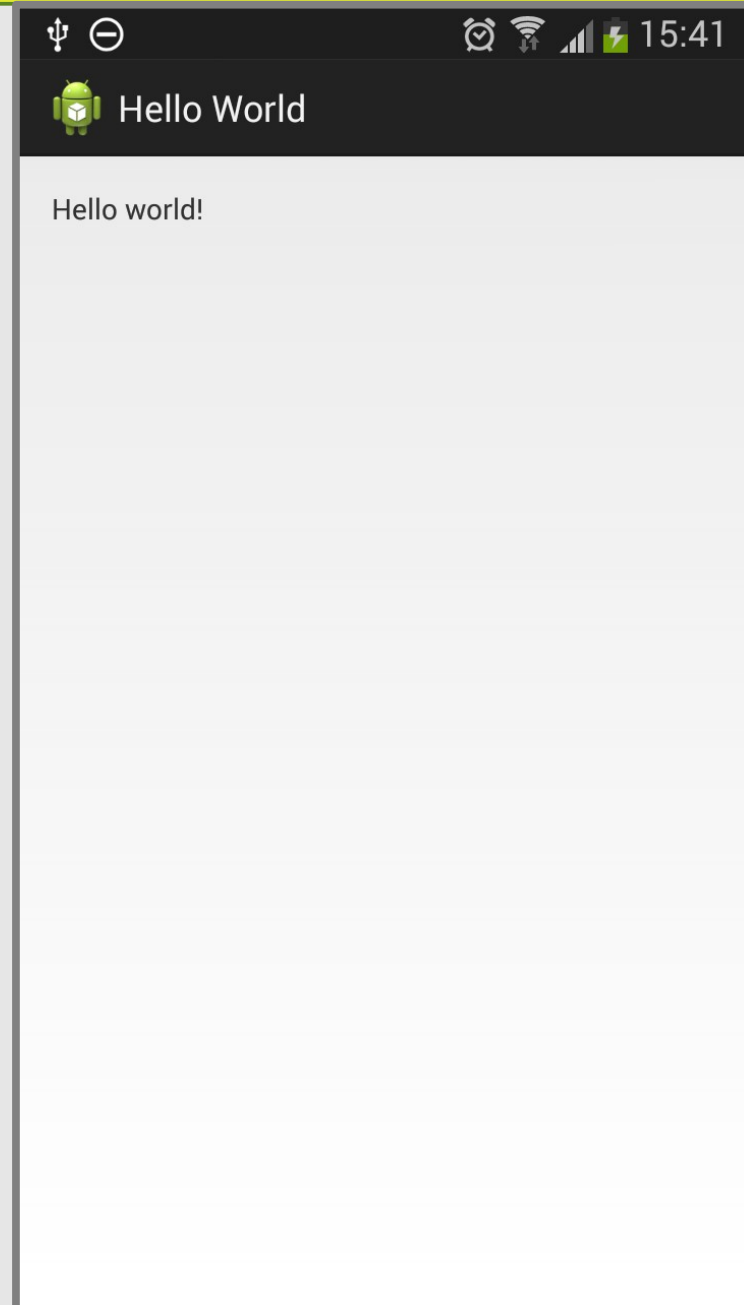
Executando o Aplicativo no AVD

- Executando o App:
 - Run → Run (Shift+F10)



Executando o Aplicativo em um Celular Real

- Conecta o celular via USB
- Habilita “Depuração de USB” no dispositivo
 - 2.2: *Opções → Opções de Desenvolvedor → Opções de desenv. → Depuração de USB*
 - 2.3: *Mais → Opções → Sobre o dispositivo → Clique no N. de Compilação diversas vezes → Voltar → Opções do Desenv. → Dep. de USB*
 - *OBS: Em outros celulares, este passo pode variar.*
- No Eclipse, *Run → Run As → Android Application*
 - O Eclipse detecta seu celular conectado; instala e inicia o App no celular



Android

Passos para Desenvolvimento

Configuração

Instalação
do ADT Bundle

Hello World
App

Configuração
do AVD

Execução
do App

Desenvolvimento

**Interface
Gráfica**

*Activities e
Intents*

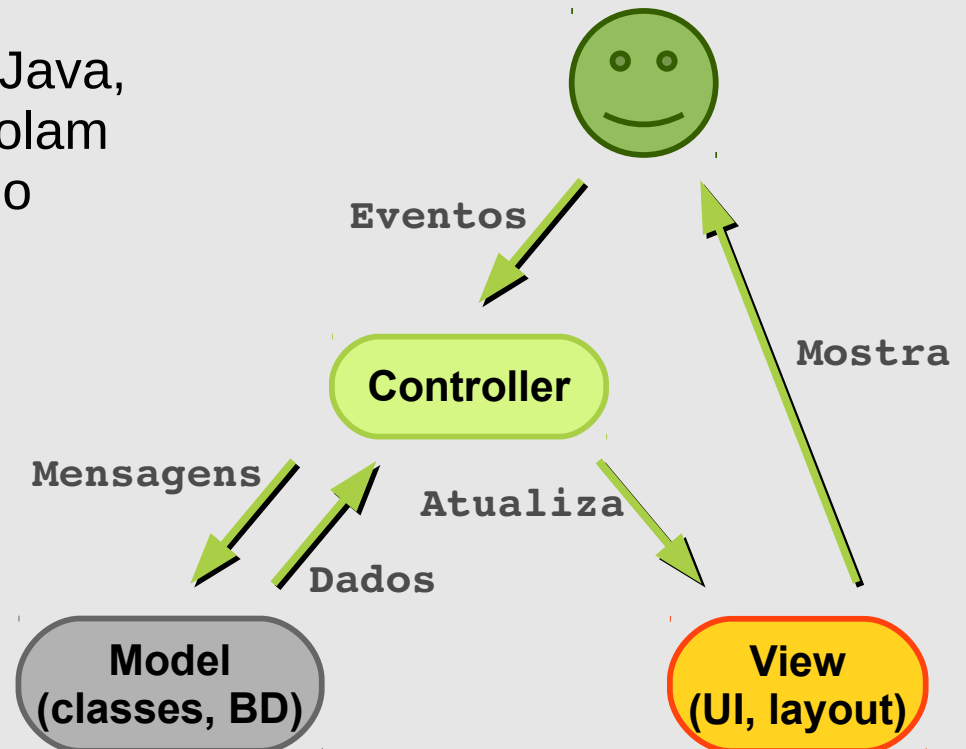
Banco de
Dados

Tópicos
Avançados

Android

Interface Gráfica

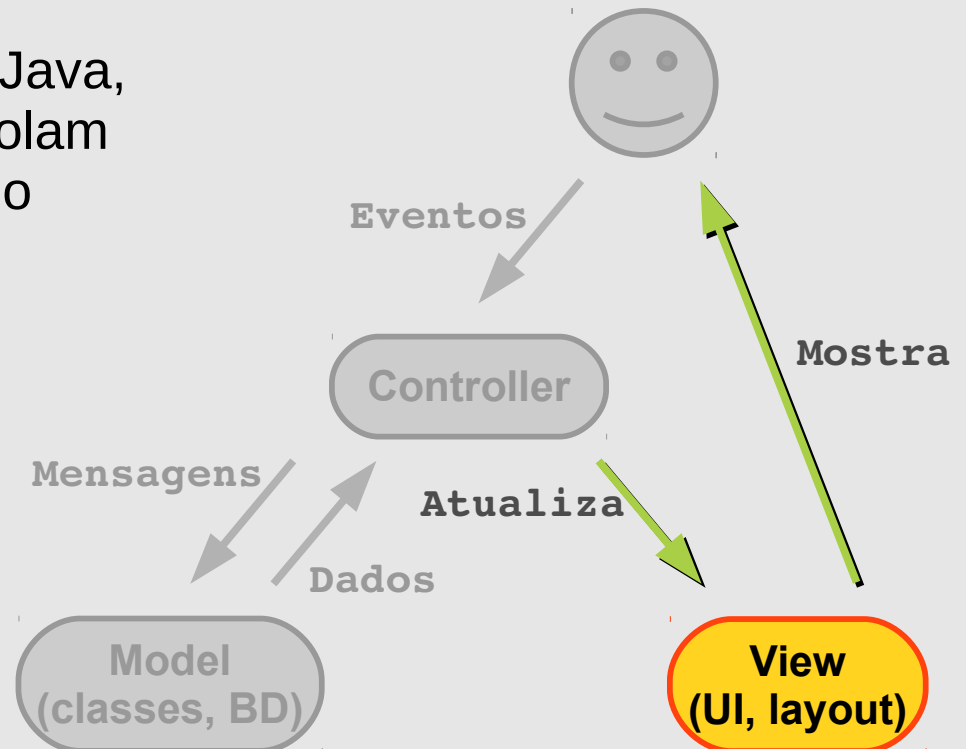
- O Android usa o modelo MVC – *Model, View, Controller*
 - Modelo: classes implementadas em Java
 - View: componentes da Interface normalmente feitos em XML
 - Controller: implementados em Java, instanciam os modelos e controlam as views. São conhecidas como “*activities*” no Android



Android

Interface Gráfica

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



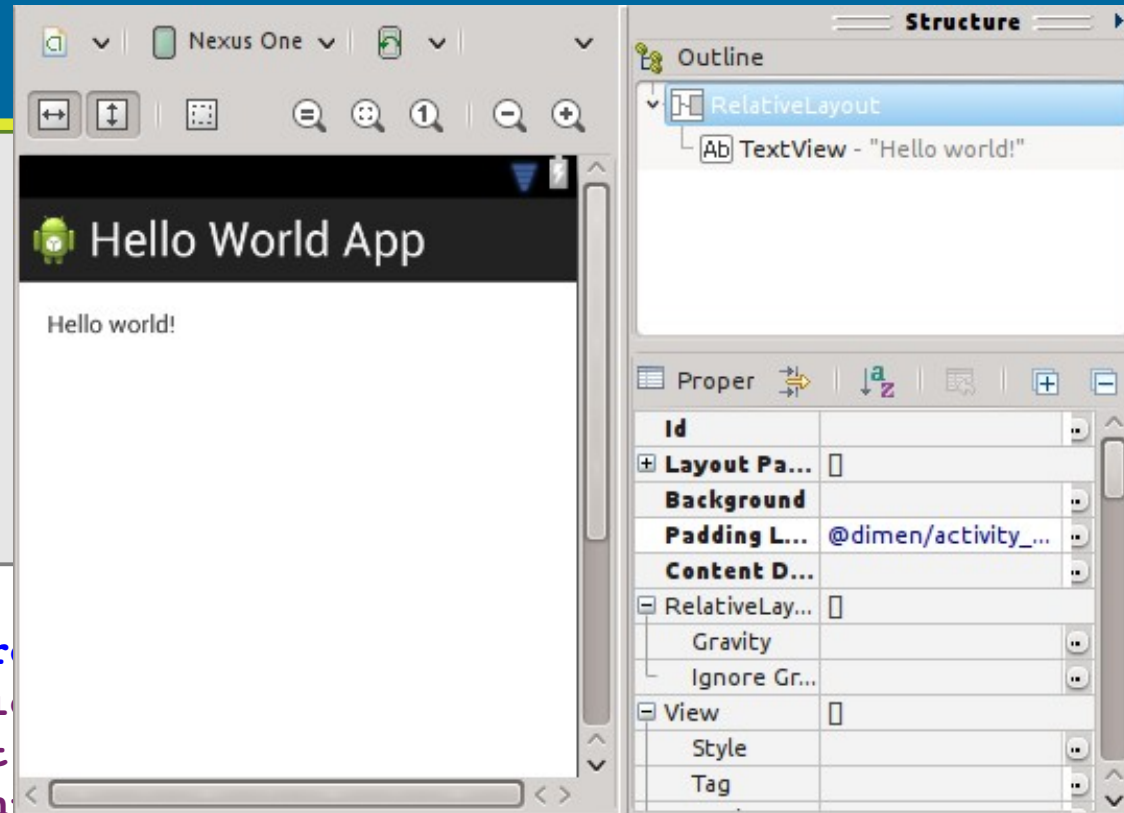
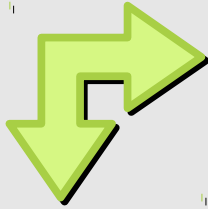
Android

Interface Gráfica

- A interface gráfica pode ser feita de três formas:
 - Usando o Editor do Eclipse (*Graphical Layout*)
 - *O editor irá gerar um XML do Layout automaticamente*
 - Editando o XML do Layout manualmente
 - *O Android faz um parse do XML do Layout e gera internamente os objetos dos componentes (Views). Este processo é conhecido como “inflate”.*
 - *Toda tag XML tem uma classe Java correspondente (subclasse da classe View)*
 - Gerando Códigos em Java manualmente
 - *Instanciando objetos de componentes gráficos (Views) manualmente e adicionando-os à interface (similar ao feito no Swing)*
 - *Não recomendado pelo Google. Exceção: nos casos em que as propriedades dos componentes são calculadas dinamicamente*

Android Interface Gráfica

- Do *Graphical Layout* ao XML



<RelativeLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
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=".MainActivity" >
```

<TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/hello_world" />
```

</RelativeLayout>

Android

Interface Gráfica – Layouts

- Todo componente UI (botões, textos, etc) deve estar dentro de um *ViewGroup*
- *ViewGroup*:
 - Agrupa componentes UI (Views) relacionados
 - *Pode ter outros ViewGroups internamente (que agruparão outras Views)*
 - Define o layout dos elementos internos
 - *Como ficarão dispostos na tela*
- Principais Layouts:
 - RelativeLayout
 - LinearLayout

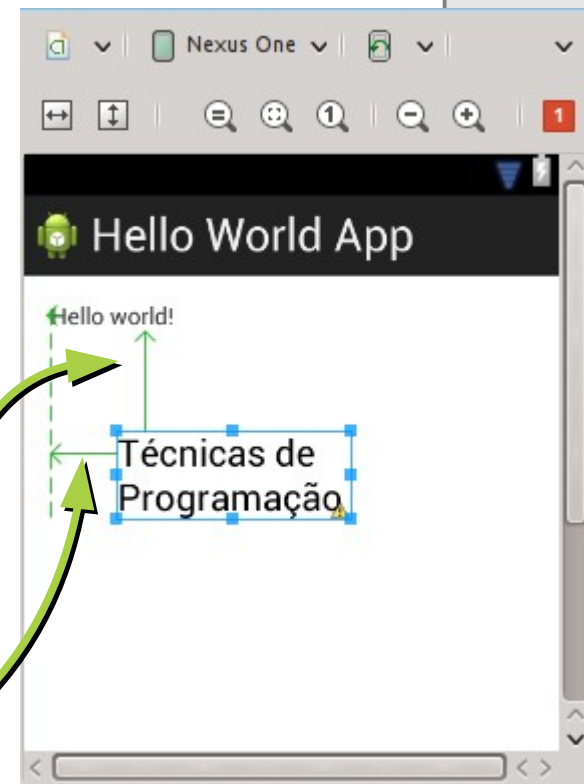
Android Interface Gráfica – RelativeLayout

- Posiciona os elementos de forma relativa

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    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=".MainActivity" >
```

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="140dp"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/textView2"
    android:layout_below="@+id/textView2"
    android:layout_marginTop="60dp"
    android:layout_marginLeft="40dp"
    android:text="Técnicas de Programação"
    android:textAppearance="?android:attr/textAppearanceLarge" />
```



Android

Interface Gráfica – Recursos

- Recursos:
 - Imagens, Strings, XML de Layouts, Menus, Dimensões, Estilos, etc
 - Recursos são “externalizados” do seu código para serem mantidos independentemente e mais facilmente. Os recursos ficam na pasta “res”
 - Exemplo de recursos de Strings (res/values/strings.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Hello World App</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
</resources>
```

- São acessados pelo nome usando o “@”:

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

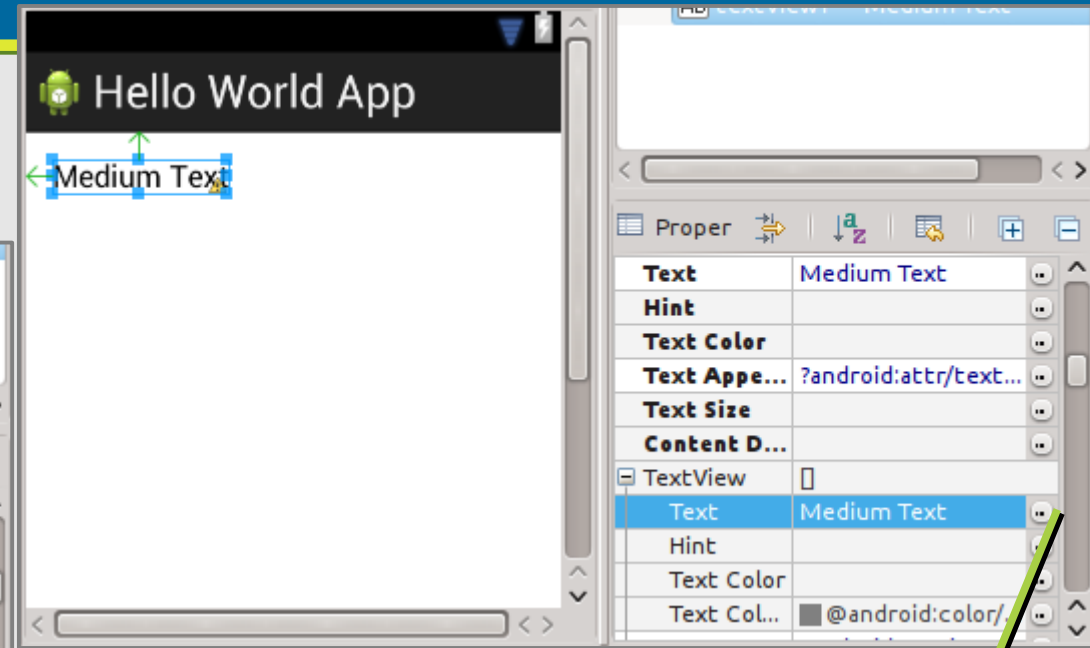
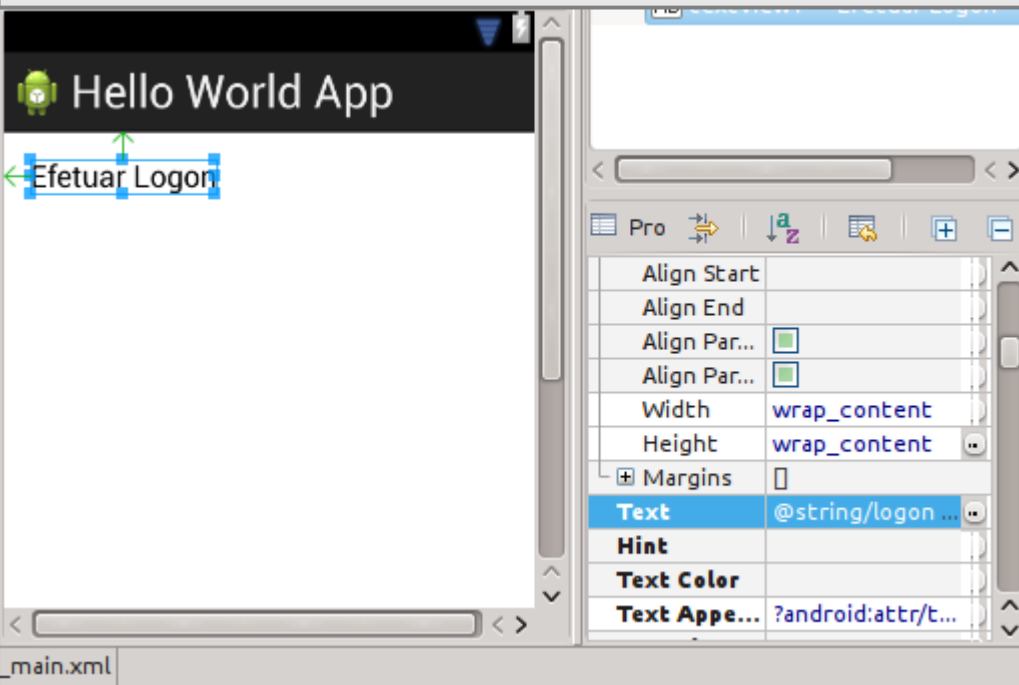
Android

Interface Gráfica – Elementos UI

- Componentes UI são chamados de *Views*, pois todas as classes que as implementam herdam a classe *View*
- Algumas Views:
 - *TextView*
 - *EditText*
 - *RadioGroup / RadioButton*
 - *Button*
 - *CheckBox*
 - *ImageView*

Android Interface Gráfica – TextView

- Adiciona um texto (*label*)



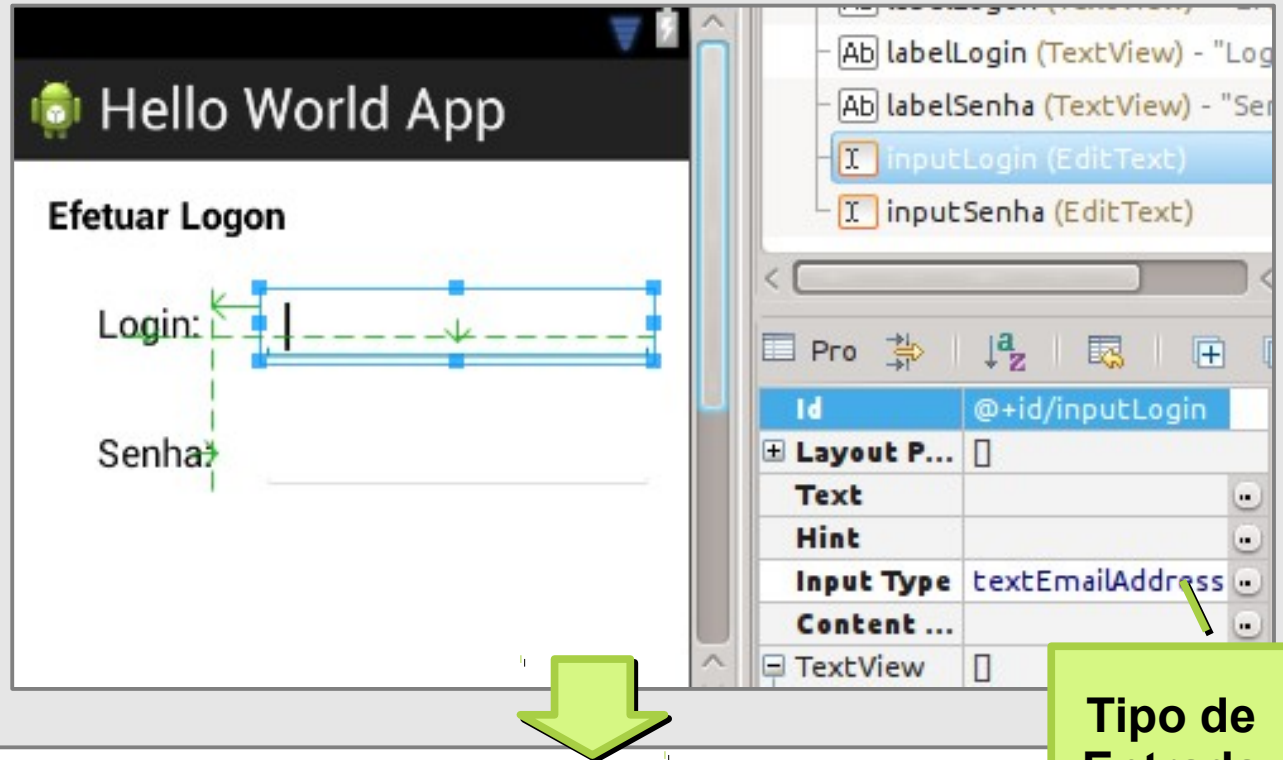
Adicionar ou usar
um recurso de
String



```
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentTop="true"  
    android:text="@string/logon"  
    android:textAppearance="?android:attr/textAppearanceMedium" />
```

Android Interface Gráfica – EditText

- Adiciona um campo de texto de entrada editável
- Diversos tipos:
 - textEmailAddress
 - textPassword
 - textUri
 - number
 - phone
 - datetime



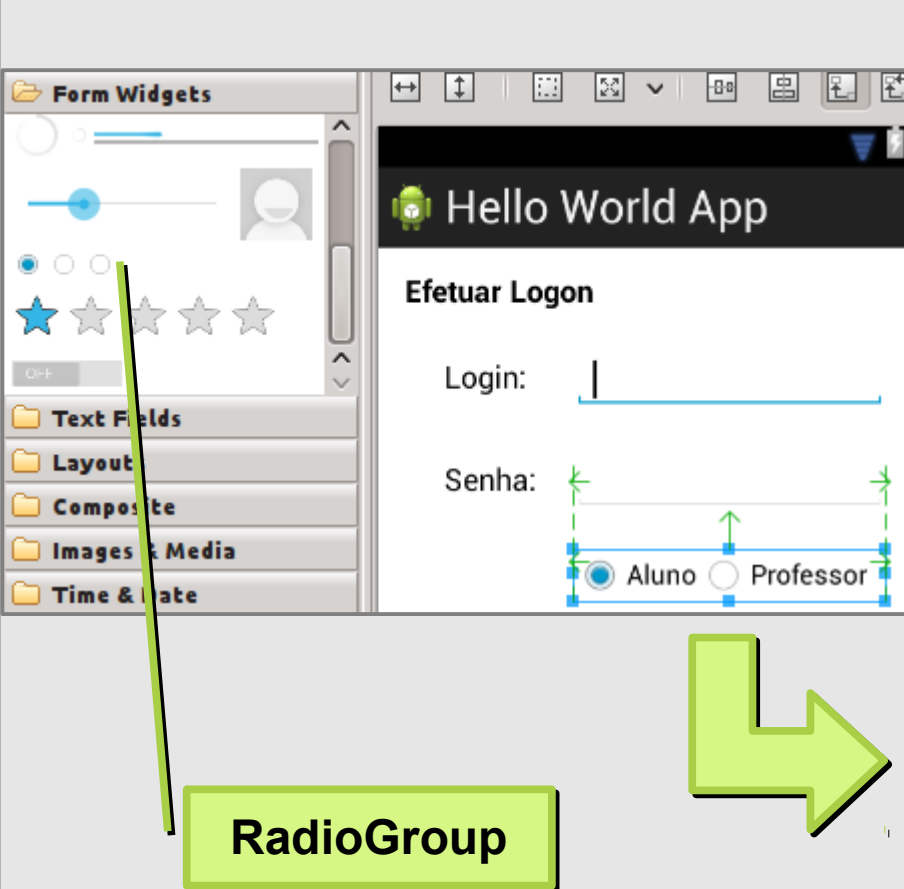
Id: identificador do campo, será usado para acessar o valor dentro do Java

```
<EditText
    android:id="@+id/inputLogin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/labelLogin"
    android:layout_alignBottom="@+id/labelLogin"
    android:layout_marginLeft="22dp"
    android:layout_toRightOf="@+id/labelSenha"
    android:ems="10"
    android:inputType="textEmailAddress" />
```

Tipo de Entrada

Android Interface Gráfica – RadioGroup / RadioButton

- RadioGroup agrupa RadioButtons relacionados



<RadioGroup

```
android:id="@+id/radioGroupTipo"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_alignLeft="@+id/inputSenha"  
android:layout_alignRight="@+id/inputSenha"  
android:layout_below="@+id/inputSenha"  
android:layout_marginTop="30dp"  
android:orientation="horizontal" >
```

<RadioButton

```
android:id="@+id/tipoAluno"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:checked="true"  
android:text="@string/aluno" />
```

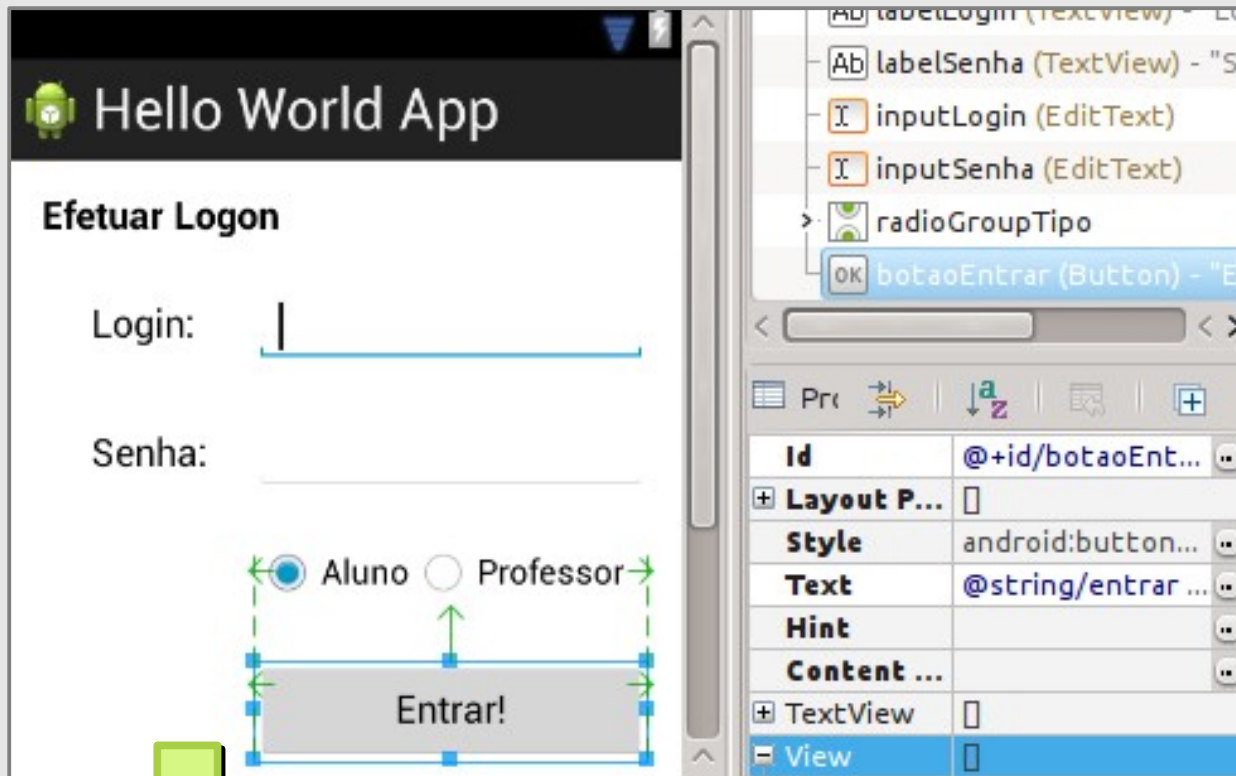
<RadioButton

```
android:id="@+id/tipoProfessor"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="@string/professor" />
```

</RadioGroup>

Android Interface Gráfica – Button

- Button

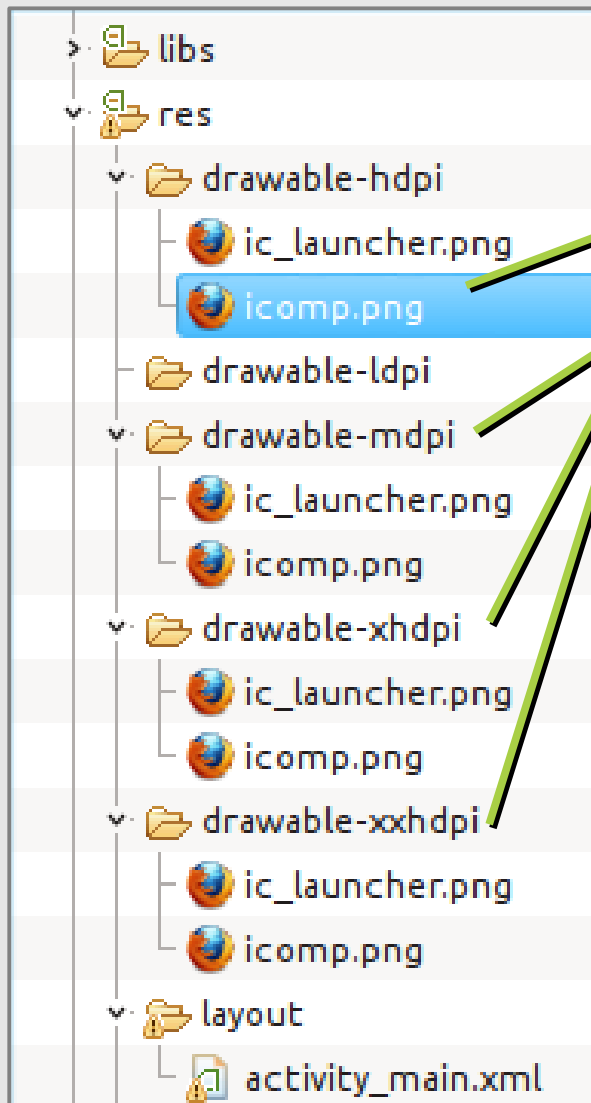


<Button

```
android:id="@+id/botaoEntrar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/radioGroupTipo"
android:layout_alignRight="@+id/radioGroupTipo"
android:layout_below="@+id/radioGroup1"
android:layout_marginTop="25dp"
android:text="@string/entrar" />
```

Android Interface Gráfica – ImageView

- Imagens



Copiar imagem

Mesma imagem em outras resoluções. Será escolhida de acordo com o dispositivo.

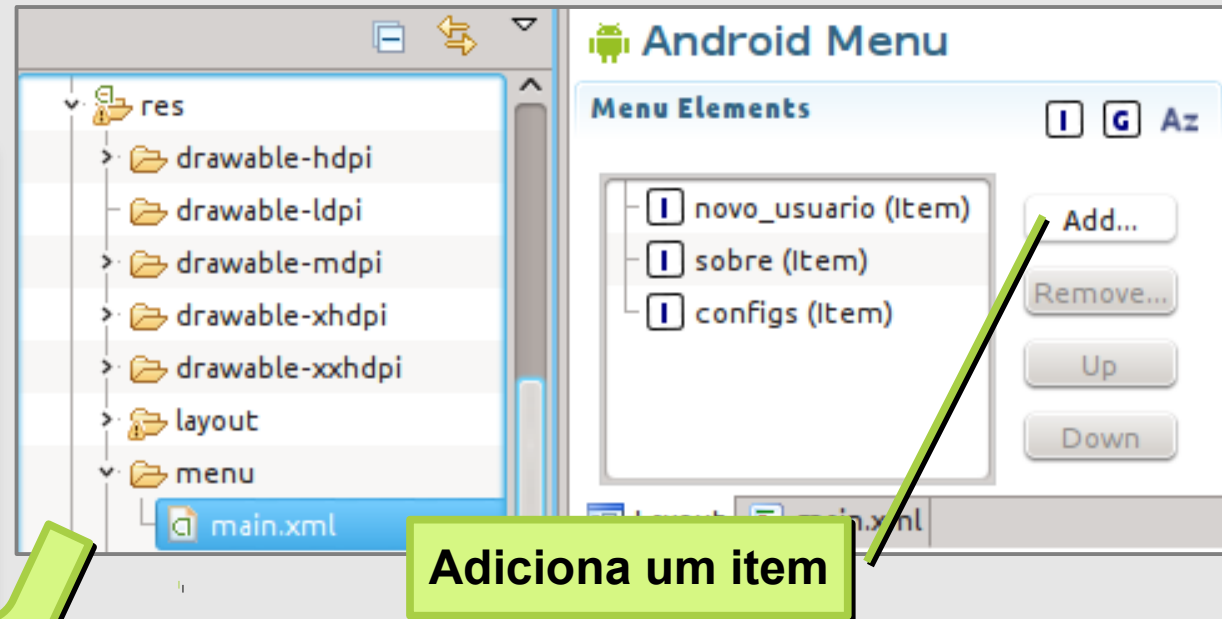
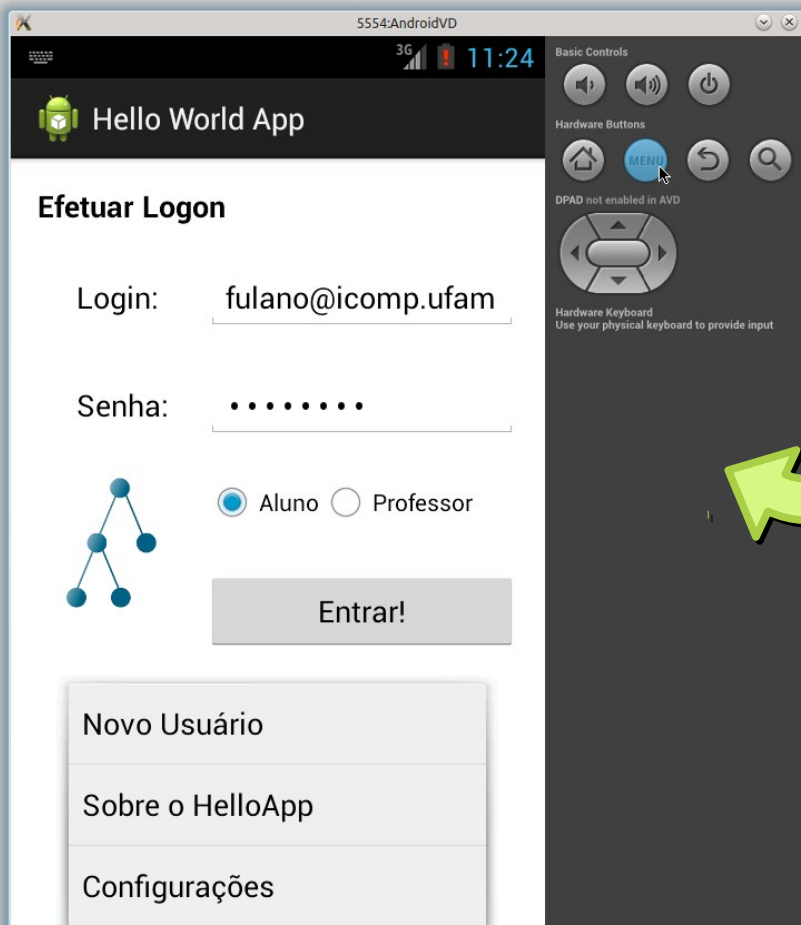


<ImageView

```
android:id="@+id/imgIcomp"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_alignRight="@+id/labelLogin"  
android:layout_alignTop="@+id/radioGroupTipo"  
android:src="@drawable/icomp"  
android:contentDescription="IComp" />
```

Android Interface Gráfica – Menu

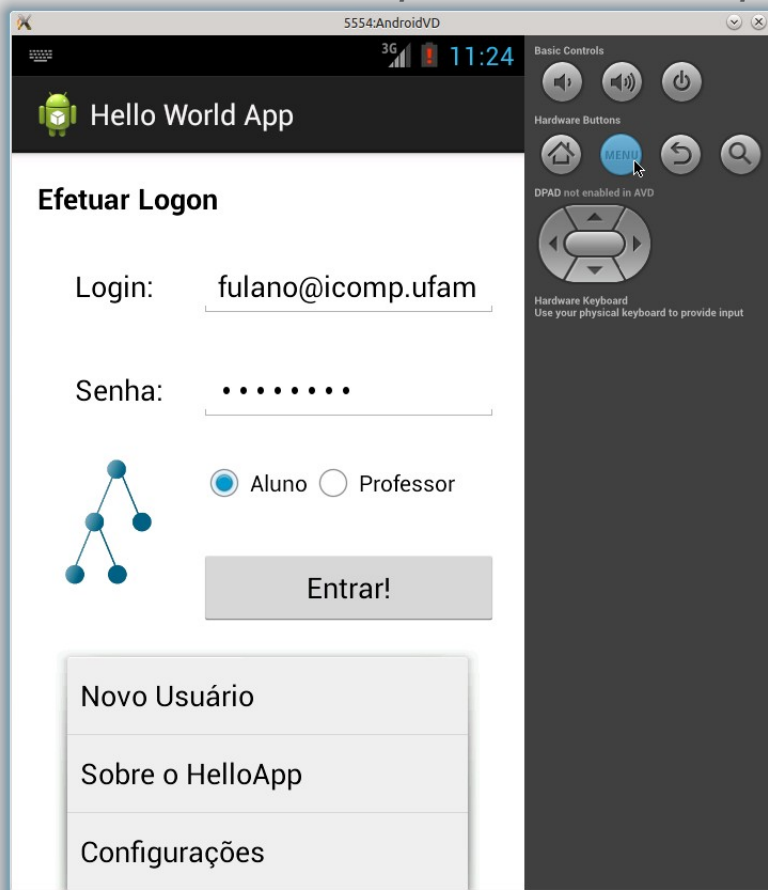
- Ao clicar no botão “Menu” do celular, a view do Menu é mostrada
 - `res/menu/main.xml`



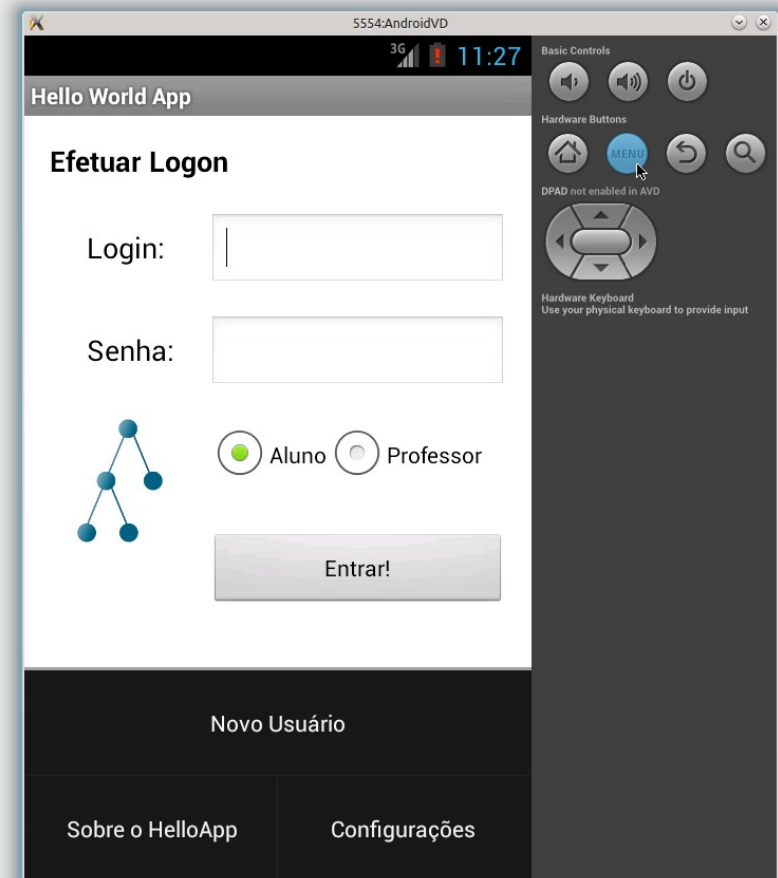
```
<menu xmlns:android="...">
  <item
    android:id="@+id/sobre"
    android:title="@string/sobre" />
  <item android:id="@+id/configs"
    android:title="@string/configs" />
</menu>
```

Android Interface Gráfica – Temas

- Temas permitem mudar os estilos dos componentes da App
 - Definição geral do tema: `res/styles/styles.xml` (pouco usado)
 - *Def. dependente de dispositivo: `res/styles-vXX/styles.xml` (recomendado)*



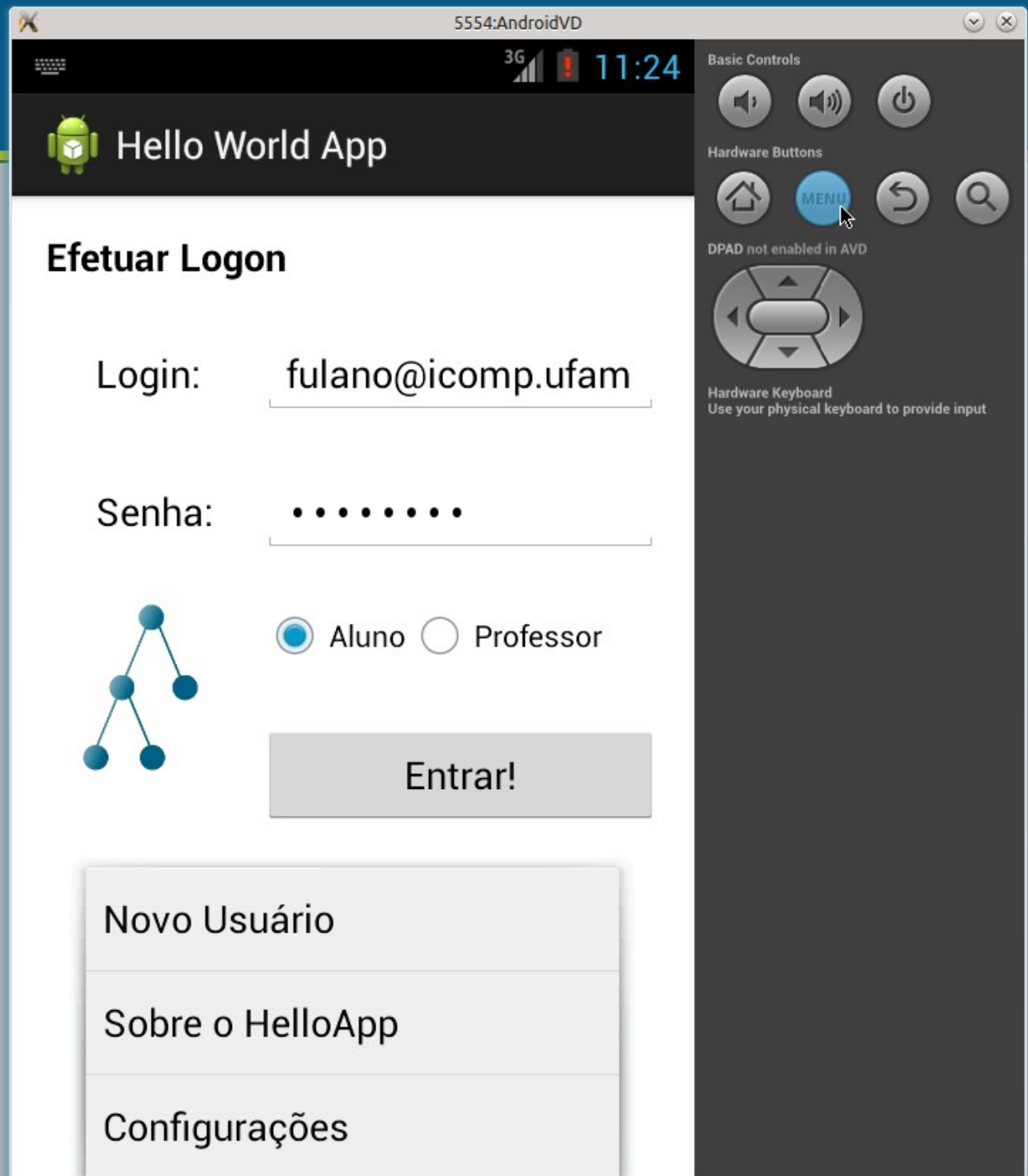
Theme.Holo.Light.DarkActionBar



Theme.Light

Android Interface Gráfica

- Resultado Final
- MVC
 - Note como geramos a interface (*View*) sem necessidade de códigos Java
- Vamos agora
 - Programar ações
 - *Controllers*
 - Acessar BD
 - *Models*
 - Gerar outras telas
 - Outros



Android

Passos para Desenvolvimento

Configuração

Instalação
do ADT Bundle

Hello World
App

Configuração
do AVD

Execução
do App

Desenvolvimento

Interface
Gráfica

***Activities e
Intents***

Banco de
Dados

Tópicos
Avançados

Android

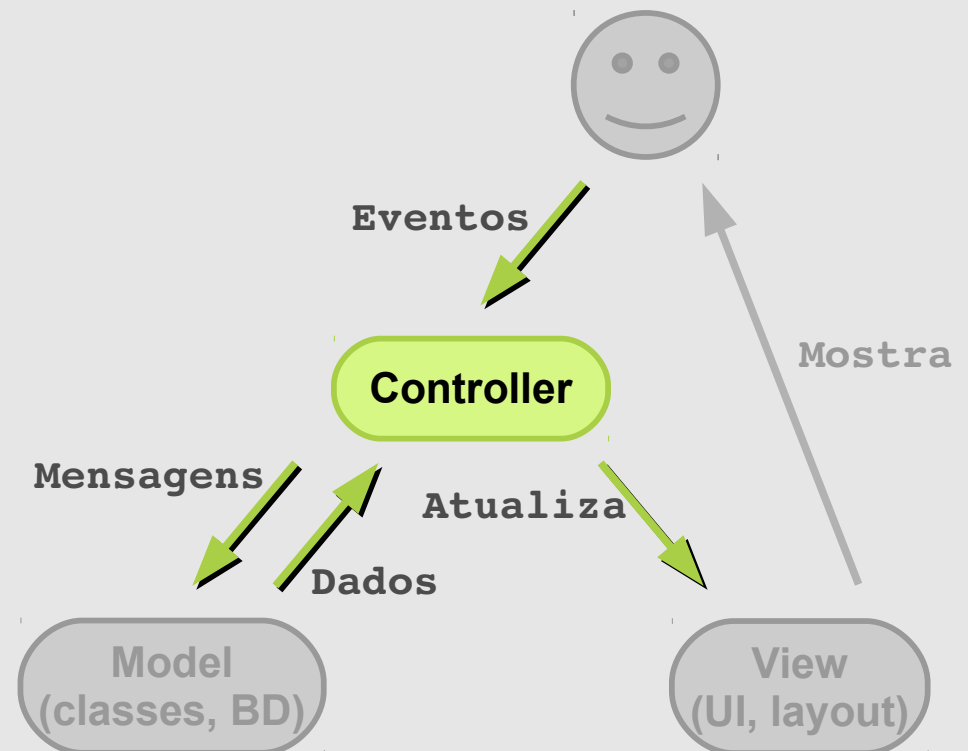
Activities

- *Activity* é um componente da aplicação que permite:
 - acesso a uma instância da tela (interface) em que os componentes UI podem ser acessados e controlados
 - interação do usuário com o aplicativo e resposta a eventos
 - instanciar classes (criar objetos) dos modelos implementados
- Normalmente, uma aplicação em Android é composta por diversas *Activities*
 - Cada nova tela da aplicação é uma *activity*
 - Uma das *activities* será a “main”, aberta quando a aplicação inicia
 - Uma *activity* vai chamando a outra
 - *Ao fazer isso, a activity anterior para de executar, mas seu estado é guardado e, ao se pressionar o botão de “voltar”, ela volta ao estado anterior*

Android

Interface Gráfica

- No modelo MVC, *activity* provê a parte do *controller*
 - implementados em Java, instanciam os modelos a controlam as *views*.
São conhecidas como “*activities*” no Android.



Android *Activities*

- Toda *activity* estende a classe Activity

- Principais métodos:

- onCreate
- onStart
- onCreateOptionsMenu
- onPause
- onResume
- onStop

Mostra a tela
activity_main,
que criamos na
parte anterior

Cria o menu

```
package ufam.icomp.helloworldapp;

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

public class MainActivity extends Activity {

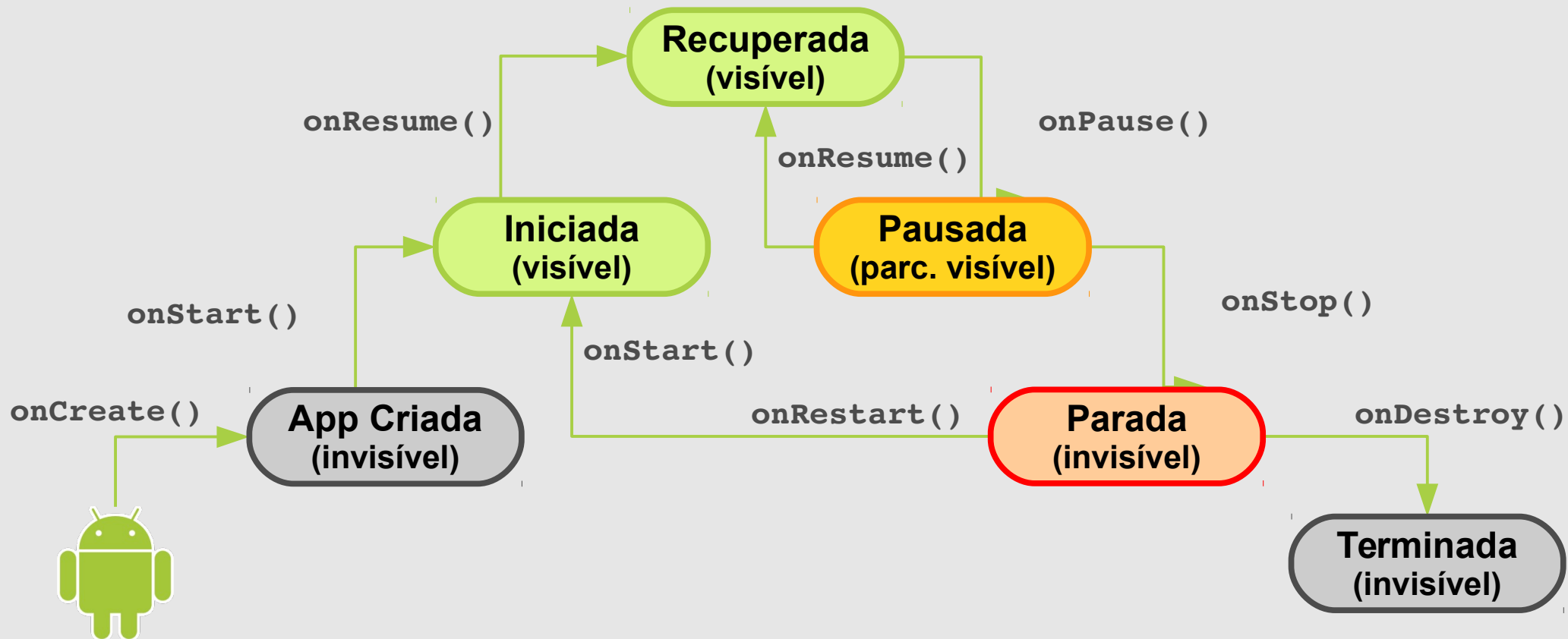
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

Android

Activities – *Ciclo de Vida*

- Ciclo de vida de uma *Activity*
 - Este ciclo se repete para cada uma das *activities*

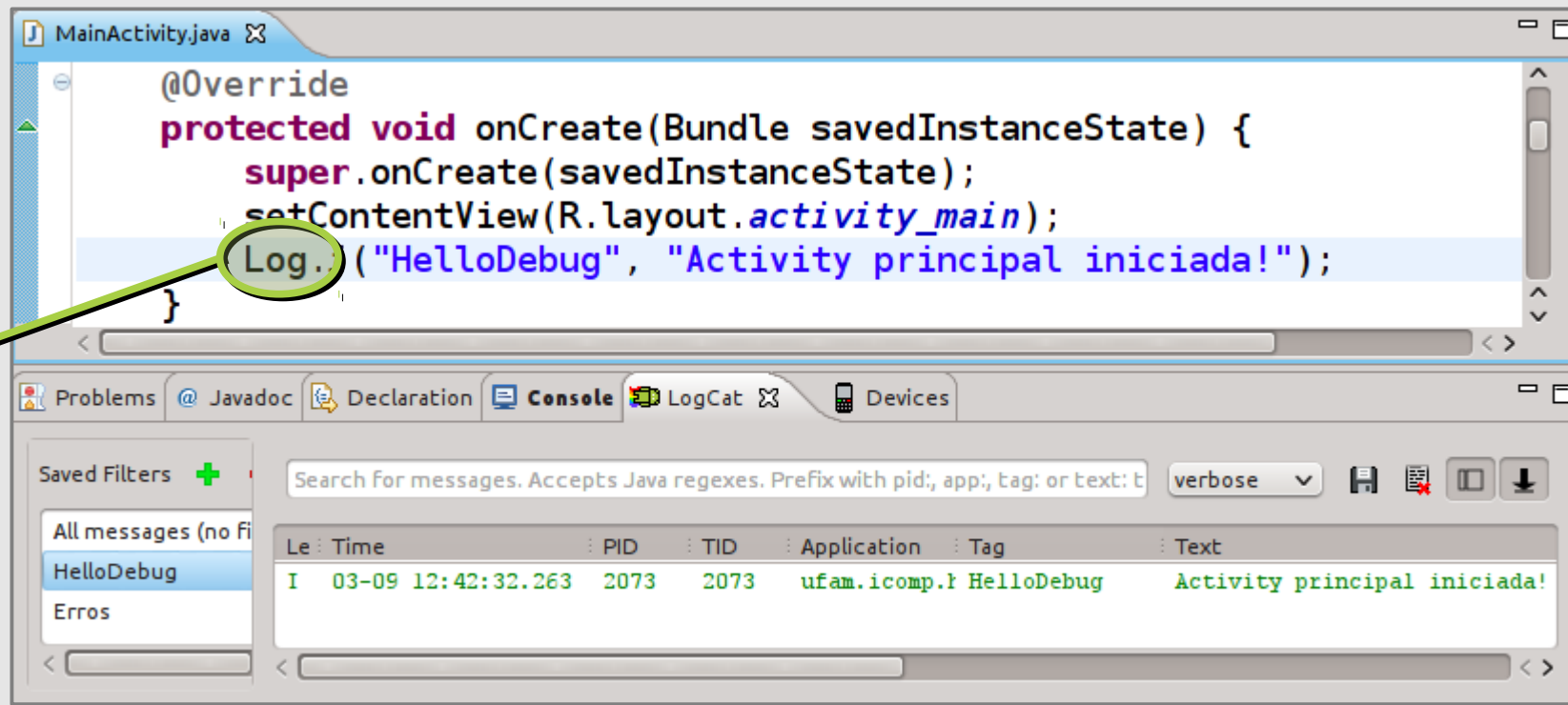


Android Activities – Debug

- O LogCat, do Android, permite receber mensagens de debug vindas da sua aplicação
 - Para escrever: `Log.i(String tag, String msg);`
 - Para ler: *Window → Show View → Other → Android → LogCat*
 - Para facilitar a leitura, crie um filtro para mostrar apenas as suas Tags
 - Os erros do Java (incluindo as Exceções), são também mostradas no LogCat, mas é recomendável também criar um filtro para os erros

d = debug
e = error
i = info
v = verb
w = warn

Nota: pressione
Ctrl + 1 para
importar a classe
Log automatic.



Android Activities – Debug e Ciclo de Vida

- Mensagens de Debug nas diversas fases do ciclo de vida

```
MainActivity.java
protected void onStart() {
    super.onStart();
    Log.i("HelloDebug", "Método onStart executado ..");
}

@Override
protected void onResume() {
    super.onResume();
    Log.i("HelloDebug", "Método onResume executado ..");
}
```

Problems | Javadoc | Declaration | Console | LogCat | Devices

Search for messages. Accepts Java regexes. Prefix with pid;, app;, tag; or text: to

verbose

Le	Time	PID	TID	Application	Tag	Text
I	03-09 12:51:51.251	2152	2152	ufam.icompl	HelloDebug	Activity principal iniciada!
I	03-09 12:51:51.271	2152	2152	ufam.icompl	HelloDebug	Método onStart executado ..
I	03-09 12:51:51.271	2152	2152	ufam.icompl	HelloDebug	Método onResume executado ..
I	03-09 12:52:20.792	2152	2152	ufam.icompl	HelloDebug	Método onPause executado ..
I	03-09 12:52:25.852	2152	2152	ufam.icompl	HelloDebug	Método onStop executado ..
I	03-09 12:52:34.642	2152	2152	ufam.icompl	HelloDebug	Método onRestart executado ..
I	03-09 12:52:34.642	2152	2152	ufam.icompl	HelloDebug	Método onStart executado ..
I	03-09 12:52:34.642	2152	2152	ufam.icompl	HelloDebug	Método onResume executado ..

App iniciada (onCreate)

Botão 'Home' Pressionado

Retornando à App

Android

Activities – Manipulando Eventos

- Cada um dos componentes UI possui uma série de eventos
 - Botões podem ser clicados, campos de texto podem ser modificados, etc
- Tais eventos são especificados no XML do Layout e executam um determinado método implementado na *Activity*.
 - Interface do método: `public void nomeDoMetodo(View view)`
 - *View view contém o objeto do componente UI que gerou o evento*

Android Activities – Manipulando Eventos

- Exemplo: mostra “Olá” ao clicar no botão “Entrar”

```
<Button android:id="@+id/botaoEntrar"  
        android:layout_width="wrap_content"  
        android:onClick="entrarClicado" />
```

```
package ufam.icomp.helloworldapp;  
import android.os.Bundle;  
import android.app.Activity;  
import android.view.Menu;  
import android.view.View;  
import android.widget.Toast;  
  
public class MainActivity extends Activity {  
    // onCreate, onCreateOptionsMenu  
  
    public void entrarClicado(View view) {  
        Toast.makeText(getApplicationContext(), "Olá!",  
            Toast.LENGTH_SHORT).show();  
    }  
}
```

Efetuar Logon

Login:

Senha:



☒ Aluno ☐ Professor

Entrar!

Olá!

Android

Activities – Eventos do Menu

- Quando uma opção do Menu é clicada, o método `onOptionsItemSelected` é executado com a opção como arg.

```
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.sobre:  
            AlertDialog.Builder alert = new AlertDialog.Builder(this);  
            alert.setMessage("Hello World App v1.0")  
                .setNeutralButton("Ok", null).show();  
            return true;  
        default:  
            return super.onOptionsItemSelected(item);  
    }  
}
```

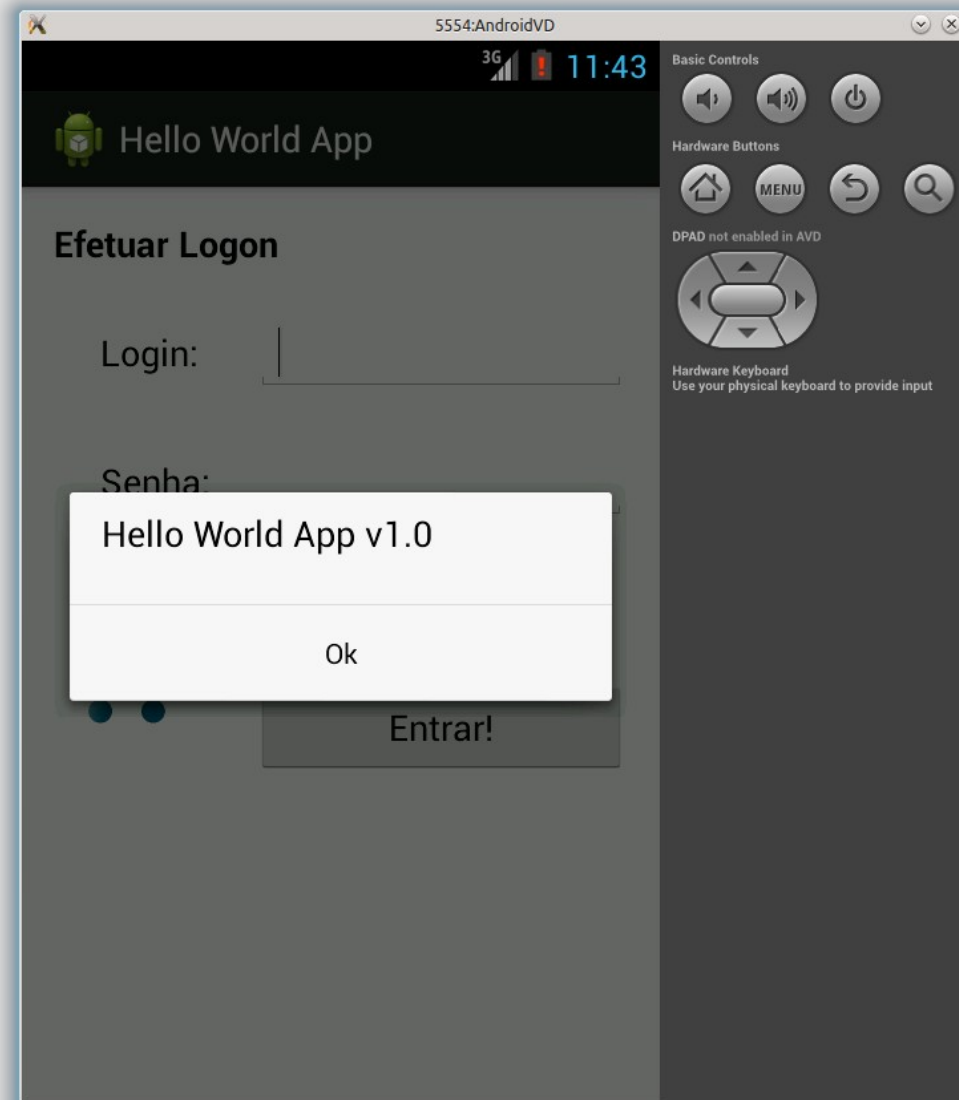
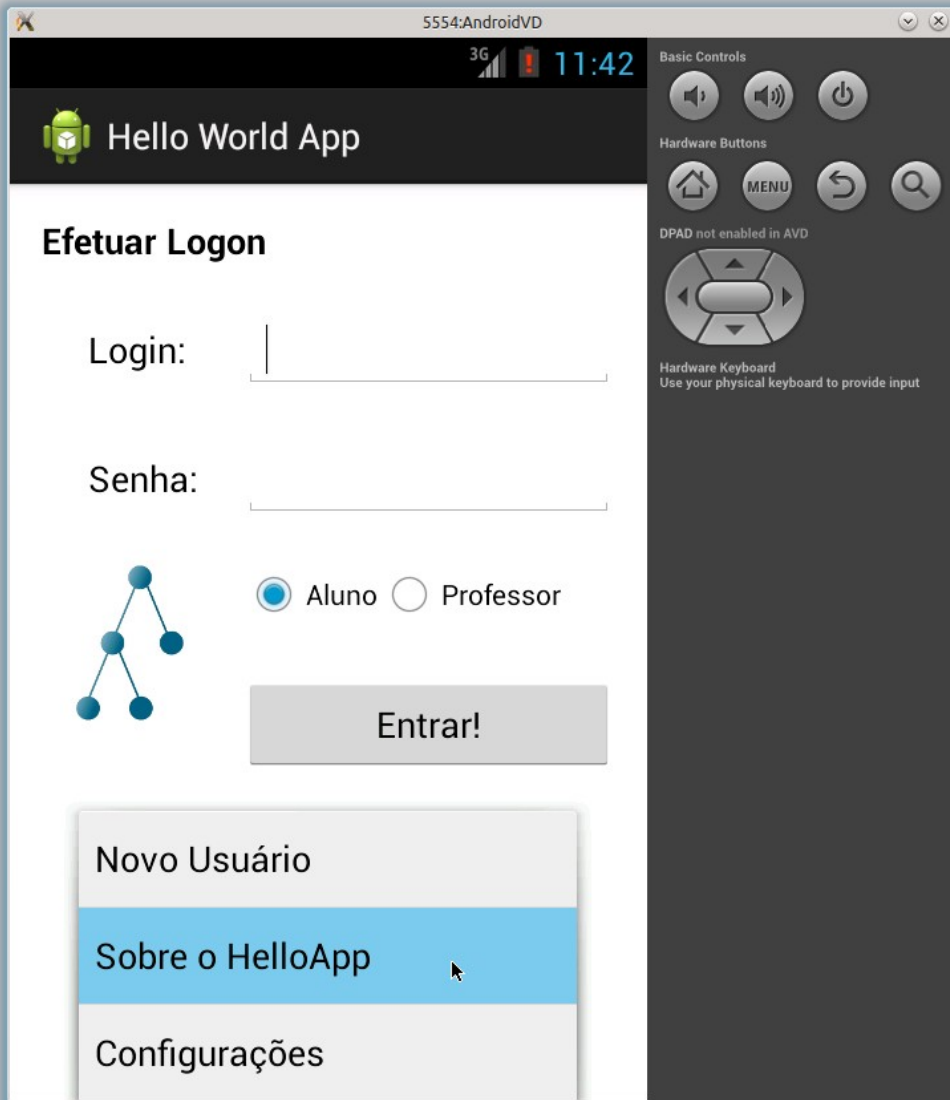
Qual item?

Item clicado

Item
'Sobre'

Constói uma
Caixa de
Alerta

Android *Activities* – Eventos do Menu



Activities – Acessando Dados da View

- Conforme mencionado, no XML dos Layouts os recursos (incluindo as *tags* dos componentes UI) são acessados usando o “@”. Mas para se acessar tais componentes dentro do Java, usa-se a classe “R”.
- A classe R é gerada automaticamente e contém uma constante estática (atributo de classe) para cada um dos recursos (strings, componentes UI, figuras, etc).
- Exemplo (parcial) de uma classe R gerada automaticamente:

```
package ufam.icomp.helloworldapp;

public final class R {
    /* ... */
    public static final class id {
        public static final int botaoEntrar=0x7f080008;
        public static final int imgIcomp=0x7f080009;
        public static final int inputLogin=0x7f080003;
        public static final int inputSenha=0x7f080004;
        public static final int labelLogin=0x7f080001;
        /* ... */
    }
}
```

Id do recurso
campo de
Login

Android

Activities – Acessando Dados da View

- Uma vez que temos o ID do recurso no Java, podemos conseguir uma referência ao seu objeto na memória através do método `findViewById`.

```
package ufam.icomp.helloworldapp;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {
    // onCreate, onCreateOptionsMenu
    public void entrarClicado(View view) {

        EditText editText =
            (EditText) findViewById(R.id.inputLogin);

        String login = editText.getText().toString();

        String msg = "Olá " + login + " !!";

        Toast.makeText(getApplicationContext(), msg,
            Toast.LENGTH_SHORT).show();

    } }
```

Irá apontar
para o objeto
da classe
EditText

É necessário
fazer um cast

Acessa o valor
do campo



Hello World App

Efetuar Logon

Login:

Senha:

☒ Aluno ☐ Professor



Entrar!

Olá fulano@icomp.ufam.edu.br !!

Intents – Abrindo outras *Activities*

- Um *Intent* provê uma ligação entre dois componentes que, em geral, são duas *activities*
 - Indica uma intenção de fazer alguma coisa

Intenção

```
public void entrarClicado(View view) {  
    Intent intent = new Intent(this, BemVindoActivity.class);  
    startActivity(intent);  
}
```

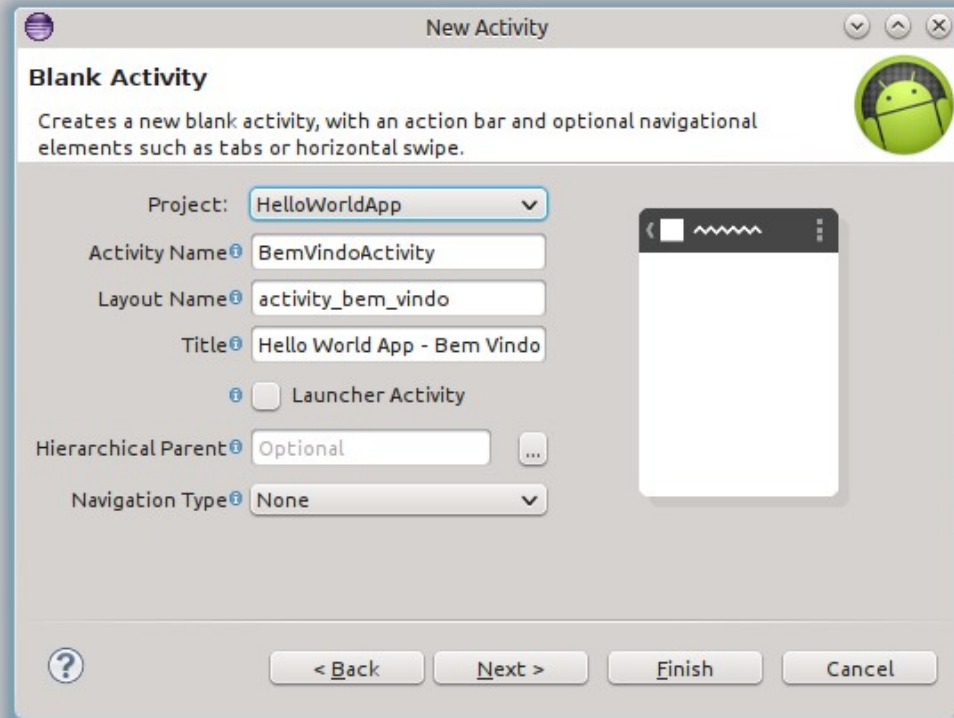
Execução

- A partir de agora, o botão “Entrar”, irá abrir uma nova *activity*
 - *Entretanto, precisamos criar essa nova activity (BemVindoActivity)*

Android

Intents – Criando uma nova Activity

- No Eclipse
 - *File → New → Other → Android → Android Activity → Blank Activity*

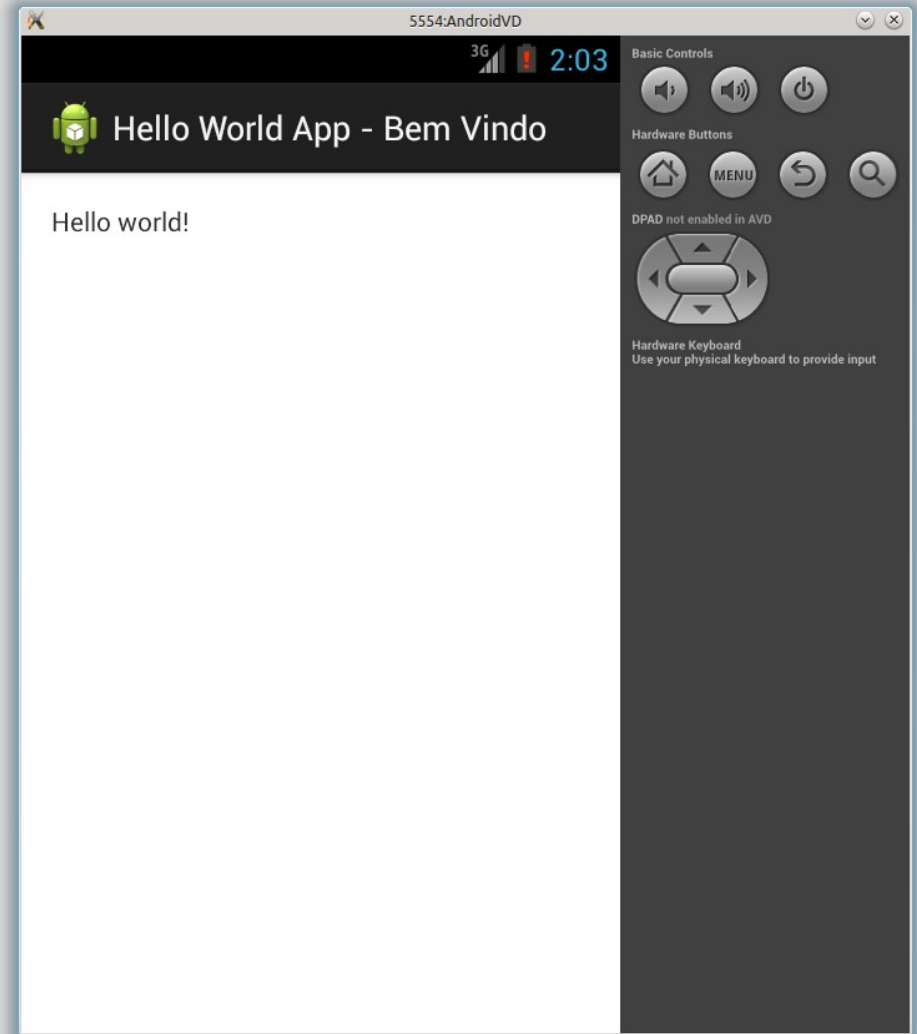
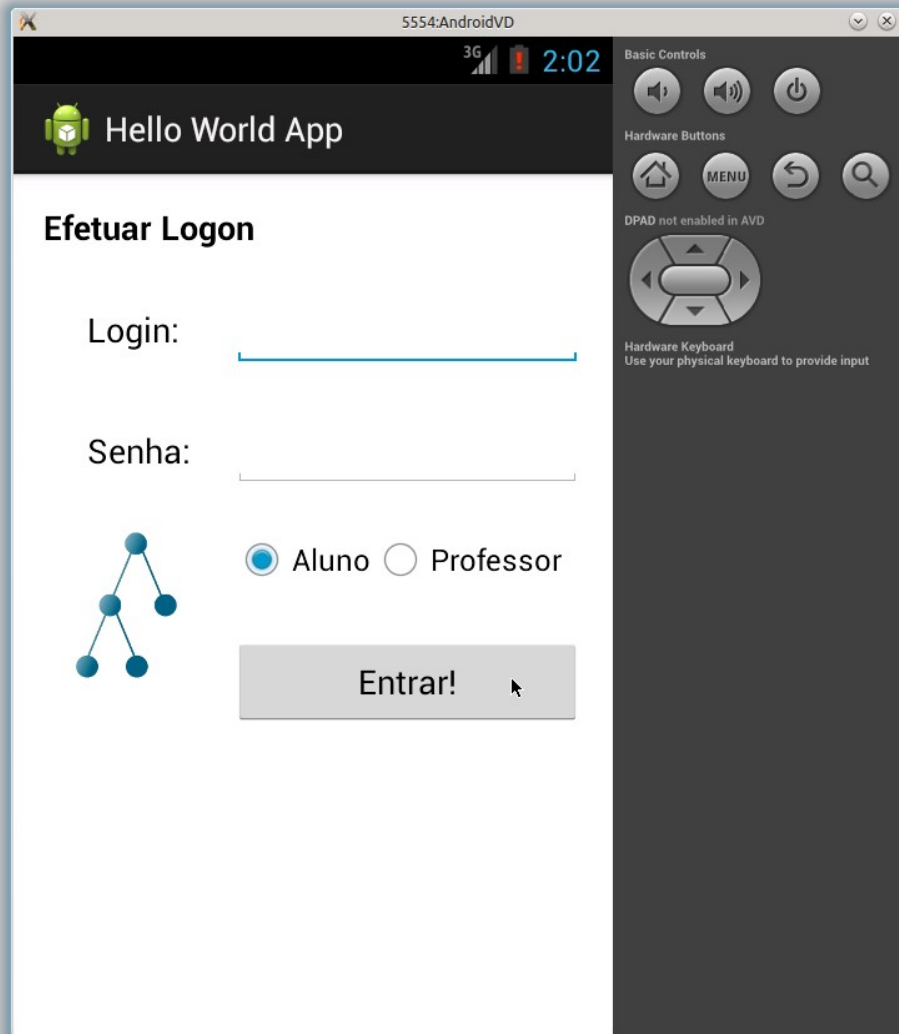


- O Eclipse irá criar e modificar os arquivos necessários para o funcionamento da nova *activity* no Android
 - *AndroidManifest.xml, res/values/strings.xml, res/menu/bem_vindo.xml, res/layout/activity_bem_vindo.xml, src/.../BemVindoActivity.java*

Android

Intents – Teste

- Testando o botão “Entrar”:



Intents – Enviando Dados para a Activity

- Para passar dados para a *activity* que será aberta, usa-se o método `putExtra` da classe `Intent`:

Intent

```
public void entrarClicado(View view) {  
    Intent intent = new Intent(this, BemVindoActivity.class);
```

**Acessando
dados de
texto**

```
    EditText inputLogin = (EditText) findViewById(R.id.inputLogin);  
    EditText inputSenha = (EditText) findViewById(R.id.inputSenha);
```

**Acessando
dados do
RadioGroup**

```
    RadioGroup groupTipo =  
        (RadioGroup) findViewById(R.id.radioGroupTipo);  
    RadioButton radioSelected =  
        (RadioButton) findViewById(groupTipo.getCheckedRadioButtonId());  
    String tipo = (String) radioSelected.getText();
```

**Passando
dados para
a próxima
Activity**

```
    intent.putExtra("login", inputLogin.getText().toString());  
    intent.putExtra("senha", inputSenha.getText().toString());  
    intent.putExtra("tipo", tipo);
```

Executa

```
    startActivity(intent);  
}
```

Android

Intents – Acessando Dados Enviados

- Para acessar os dados na nova *activity*, usa-se o método `getStringExtra` da classe `Intent`:

onCreate da
nova *Activity*

Acessa o
Intent

Acessa os
dados enviados

Acessa o *label*
de bem vindo

Muda o seu
conteúdo

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

    Intent intent = getIntent();
    String login = intent.getStringExtra("login");
    String senha = intent.getStringExtra("senha");
    String tipo = intent.getStringExtra("tipo");

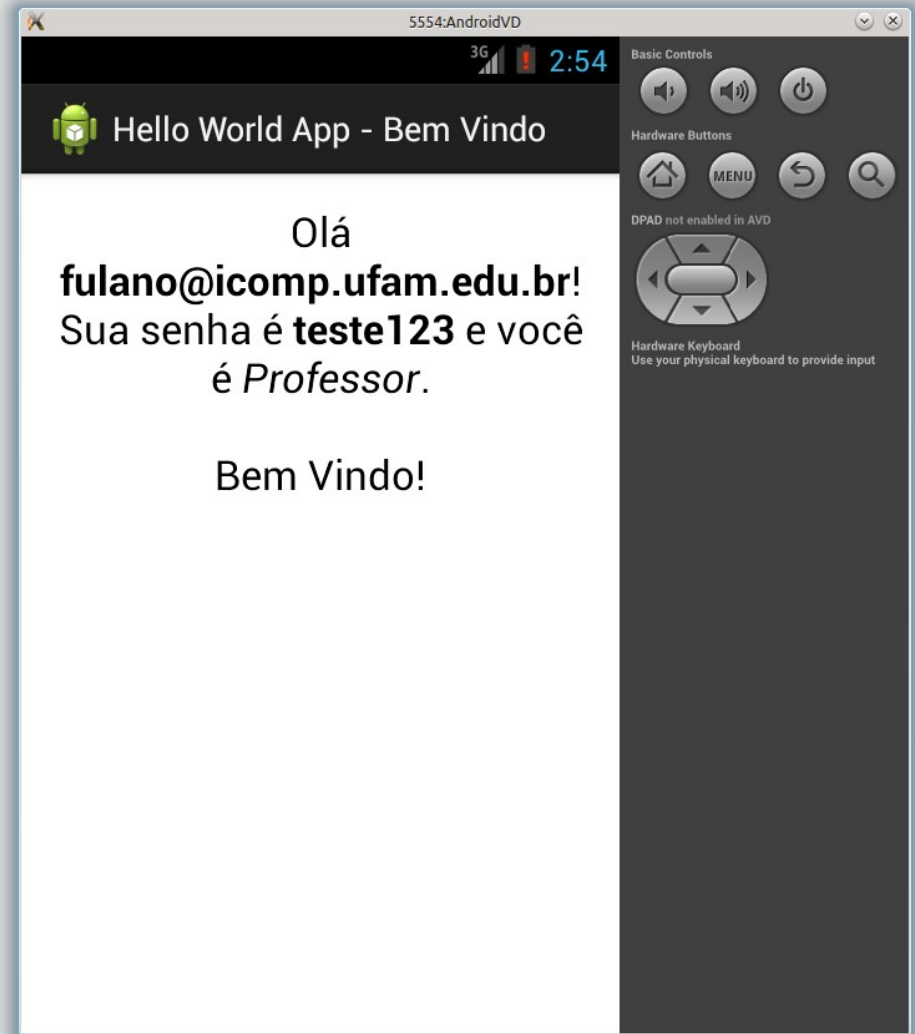
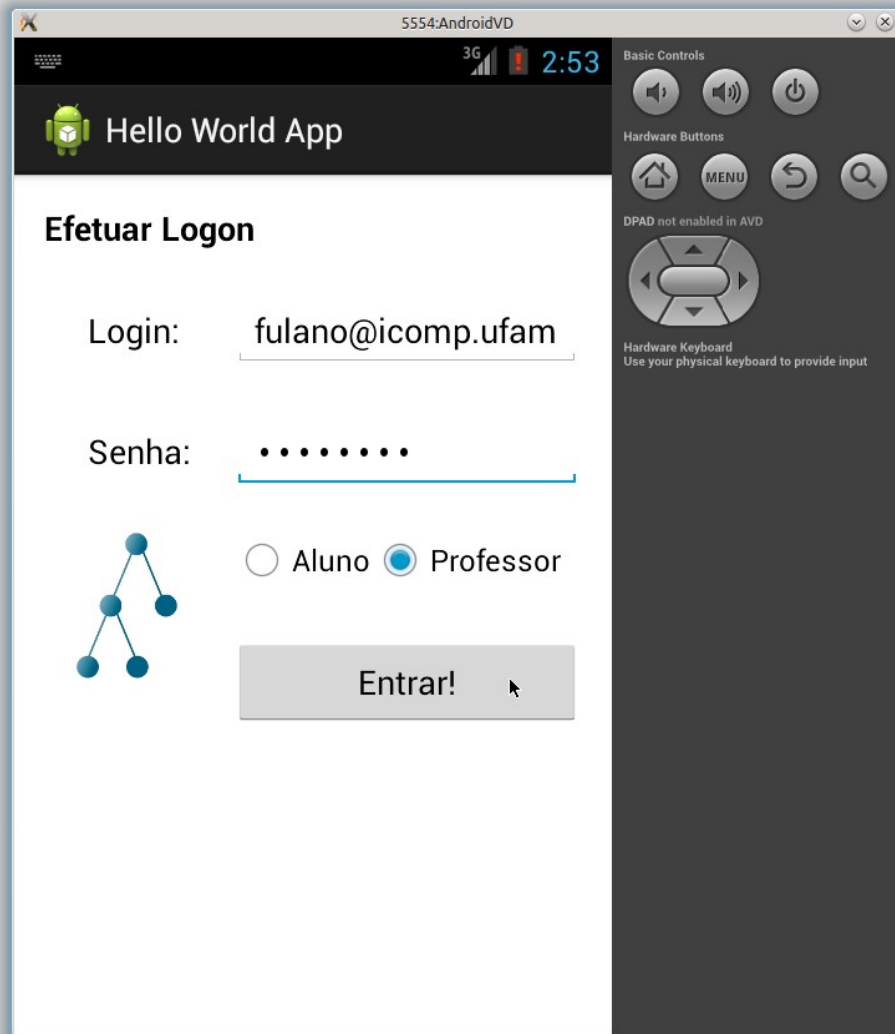
    TextView textBemVindo =
        (TextView) findViewById(R.id.textBemVindo);

    textBemVindo.setText(Html.fromHtml(
        "Olá <b>" + login +
        "</b>! Sua senha é <b>" + senha +
        "</b> e você é <i>" + tipo +
        "</i>.<br><br>Bem Vindo!"));
}
```


Android

Intents – Teste

- Testando novamente o botão “Entrar”:



Android

Passos para Desenvolvimento

Configuração

Instalação
do ADT Bundle

Hello World
App

Configuração
do AVD

Execução
do App

Desenvolvimento

Interface
Gráfica

*Activities e
Intents*

**Banco de
Dados**

Tópicos
Avançados

Android

Banco de Dados

- O Banco de Dados do Android é o SQLite
 - Suporta a comandos SQL, Leve
 - Dados são salvos em um único arquivo local
 - Já vem instalado
 - *Sem necessidade de usuário e senha. Entretanto, seus dados estão seguros, pois só serão acessíveis por sua App.*
 - <http://www.sqlite.org/>

Android

Banco de Dados

- Hello World App

Usuario	
String	login
String	nome
String	senha
int	tipo

Usuario
(modelo)

Livro
(modelo)

BancoDeDados
(Auxiliar)

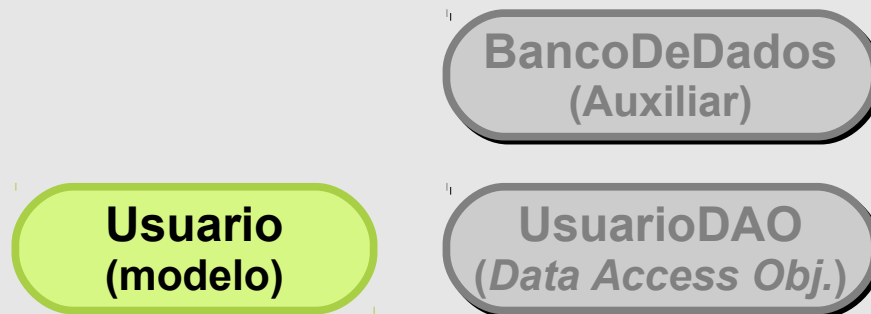
UsuarioDAO
(Data Access Obj.)

LivroDAO
(Data Access Obj.)

Android

Banco de Dados

- Hello World App



Android

Banco de Dados – Modelo

**Usuario
(modelo)**

Usuario

String login
String nome
String senha
int tipo

- Modelo Usuario

- src/ufam.icomp.helloworldapp/Usuario.java

```
package ufam.icomp.helloworldapp;
```

```
public class Usuario implements Serializable {  
    private static final long serialVersionUID = 1L;  
    private String login, nome, senha;  
    private int tipo;
```

Para passar um
objeto da classe
Usuario para
outra Activity

```
    public Usuario(String login, String nome, String senha, int tipo) {  
        this.login = login;  
        this.nome = nome;  
        this.senha = senha;  
        this.tipo = tipo;
```

```
    }
```

```
    // Getters e Setters
```

```
    public String getTipoString() {  
        if (this.tipo == 1) return "Aluno";  
        else return "Professor";
```

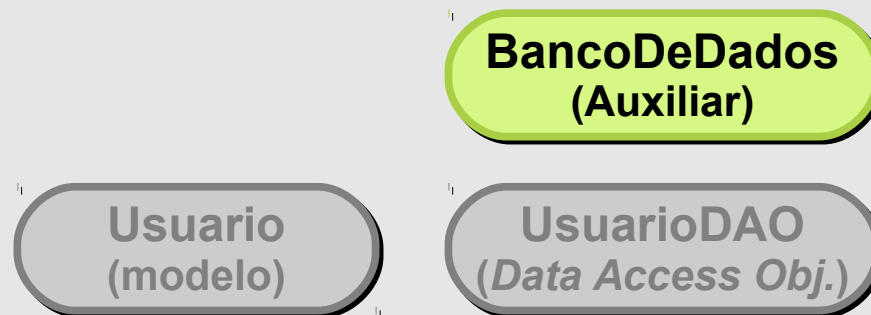
```
    }
```

```
}
```

Android

Banco de Datos

- Hello World App



Android

Banco de Dados – Auxiliar

**BancoDeDados
(Auxiliar)**

- Auxiliar de Banco de Dados
 - Cria o banco (quando necessário)
 - Atualiza o banco (quando necessário)
 - `src/ufam.icomp.helloworldapp/BancoDeDados.java`


```
package ufam.icomp.helloworldapp;  
import android.content.Context;  
import android.database.sqlite.*;
```

Versão do BD.
Incr. ao modificar

SQL para criar
o BD

SQL para
popular o BD

```
public class BancoDeDados extends SQLiteOpenHelper {  
    public static final int DATABASE_VERSION = 1;  
    public static final String DATABASE_NAME = "HelloWorldApp.db";  
  
    private static final String SQL_CREATE_TABLES = "CREATE TABLE Usuarios(" +  
        "login TEXT PRIMARY KEY, nome TEXT, senha TEXT, tipo INT)";  
  
    private static final String SQL_POPULATE_TABLES = "INSERT INTO Usuarios " +  
        "VALUES ('fulano@icomp.ufam.edu.br', 'Fulano de Tal', 'teste123', 2)";  
  
    private static final String SQL_DELETE_TABLES = "DROP TABLE IF EXISTS Usuarios";  
  
    public BancoDeDados(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
  
    public void onCreate(SQLiteDatabase db) {  
        db.execSQL(SQL_CREATE_TABLES);  
        db.execSQL(SQL_POPULATE_TABLES);  
    }  
  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        db.execSQL(SQL_DELETE_TABLES);  
        onCreate(db);  
    }  
}
```

SQL para
limpar o BD

Contexto é a
activity

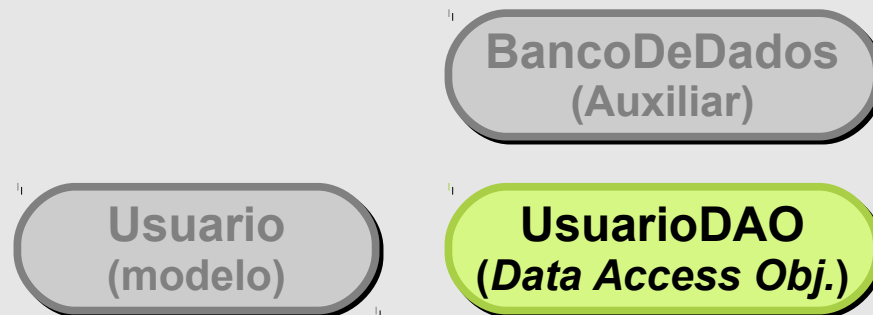
Executado ao
criar o BD

Executado ao
atualizar o BD

Android

Banco de Dados

- Hello World App



- UsuarioDAO
 - Construtor
 - *abre o BD*
 - `public Usuario getUsuario(String login, String senha)`
 - *Pega os dados de um usuário do BD usando login/senha*
 - `public boolean addUsuario(Usuario u)`
 - *Adiciona um novo usuário no BD*
 - `public Cursor getUsuarios()`
 - *Retorna um cursor do BD com uma consulta aos usuários do sistema*

Android

Banco de Dados – UsuarioDAO

UsuarioDAO
(Data Access Obj.)

```
// package, imports ...
```

```
public class UsuarioDAO {  
    private SQLiteDatabase bancoDeDados;  
  
    public UsuarioDAO(Context context) {  
        this.bancoDeDados = (new BancoDeDados(context)).getWritableDatabase();  
    }
```

Abre conexão
com o BD

```
    public Usuario getUsuario(String login, String senha) {  
        Usuario usuario = null;  
        String sqlQuery = "SELECT * FROM Usuarios WHERE " +  
                           "login='" + login + "' AND senha='" + senha + "'";  
        Cursor cursor = this.bancoDeDados.rawQuery(sqlQuery, null);
```

Consulta SQL

Executa Consulta

```
        if (cursor.moveToNext()) {  
            usuario = new Usuario(cursor.getString(0), cursor.getString(1),  
                                  cursor.getString(2), cursor.getInt(3));  
        }
```

Se retornou
uma linha

```
        cursor.close();  
        return usuario;  
    }
```

Retorna um
objeto Usuario

```
// Continuação no próximo slide ...
```

Android

Banco de Dados – UsuarioDAO

UsuarioDAO
(Data Access Obj.)

```
public boolean addUsuario(Usuario u) {  
    try {  
        String sqlCmd = "INSERT INTO Usuarios VALUES ('" +  
            u.getLogin() + "', " + u.getNome() + "', " +  
            u.getSenha() + "', " + u.getTipo() + ")";  
        this.bancoDeDados.execSQL(sqlCmd);  
        return true;  
    } catch (SQLException e) {  
        Log.e("HelloAppBD", e.getMessage());  
        return false;  
    }  
}  
  
public Cursor getUsuarios() {  
    return this.bancoDeDados.rawQuery("SELECT rowid AS _id, " +  
        "login, nome, " +  
        "CASE WHEN tipo=1 THEN 'Aluno' ELSE 'Professor' END AS tipo " +  
        "FROM Usuarios ORDER BY login", null);  
}  
  
} // Fim da classe UsuarioDAO
```

SQL para adicionar usuário

Executa comando

Retorna um cursor de uma consulta no BD

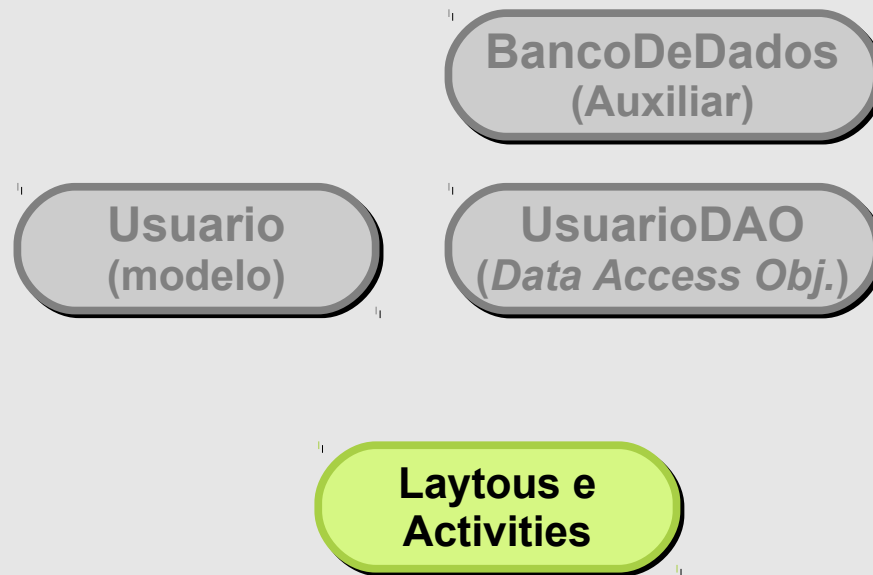
Consulta

Usado pelo ListView (mais adiante)

Android

Banco de Dados

- Hello World App



- Agora que criamos as classes necessárias para manipular os modelos e o banco de dados
- Vamos mudar/criar os *Layouts e Activities* necessários para:
 - Verificar login/senha e acessar dados do usuário
 - Adicionar um novo usuário
 - Listar usuários cadastrados

Android

Banco de Dados – Verificar login/senha

- Mudando a MainActivity para checar o login/senha

```
public void entrarClicado(View view) {  
  
    Intent intent = new Intent(this, BemVindoActivity.class);  
    EditText inputLogin = (EditText) findViewById(R.id.inputLogin);  
    EditText inputSenha = (EditText) findViewById(R.id.inputSenha);  
  
    // Verifica a senha  
    UsuarioDAO usuarioDAO = new UsuarioDAO(this);  
    Usuario usuario = usuarioDAO.getUsuario(inputLogin.getText().toString(),  
                                             inputSenha.getText().toString());  
  
    if (usuario != null) {  
        intent.putExtra("usuario", usuario);  
        startActivity(intent);  
    } else {  
        Toast.makeText(this, "Usuário e/ou Senha inválidos!",  
                       Toast.LENGTH_SHORT).show();  
    }  
}
```

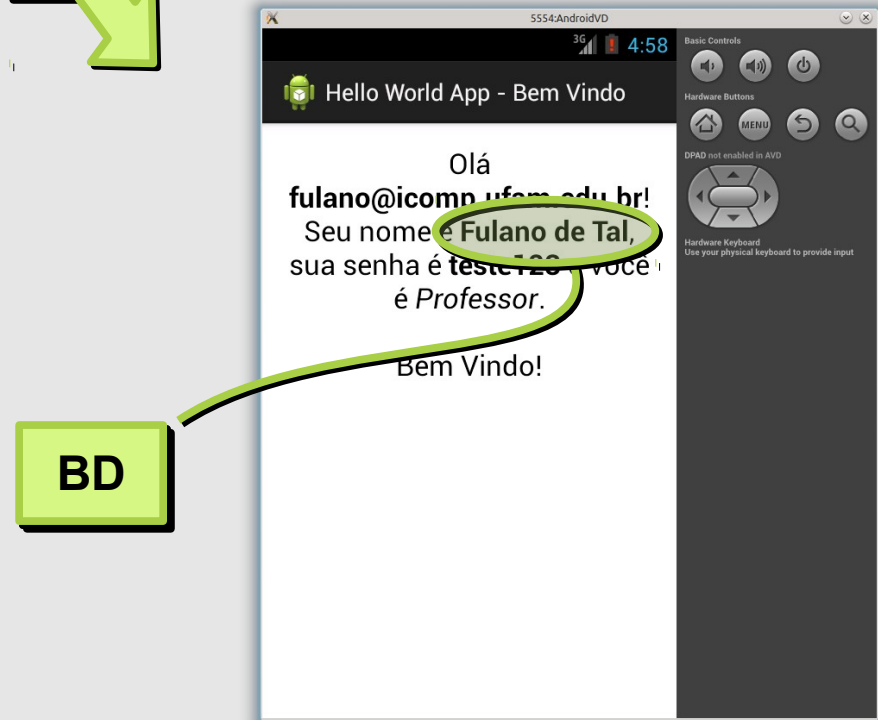
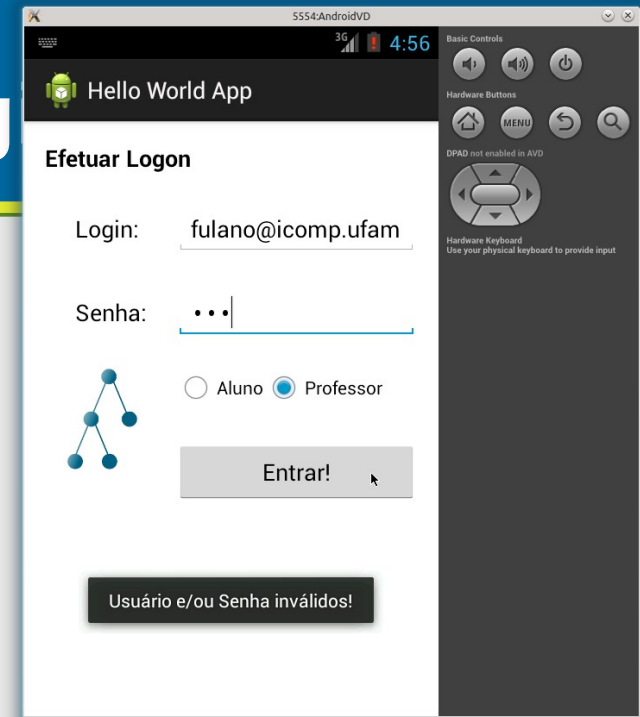
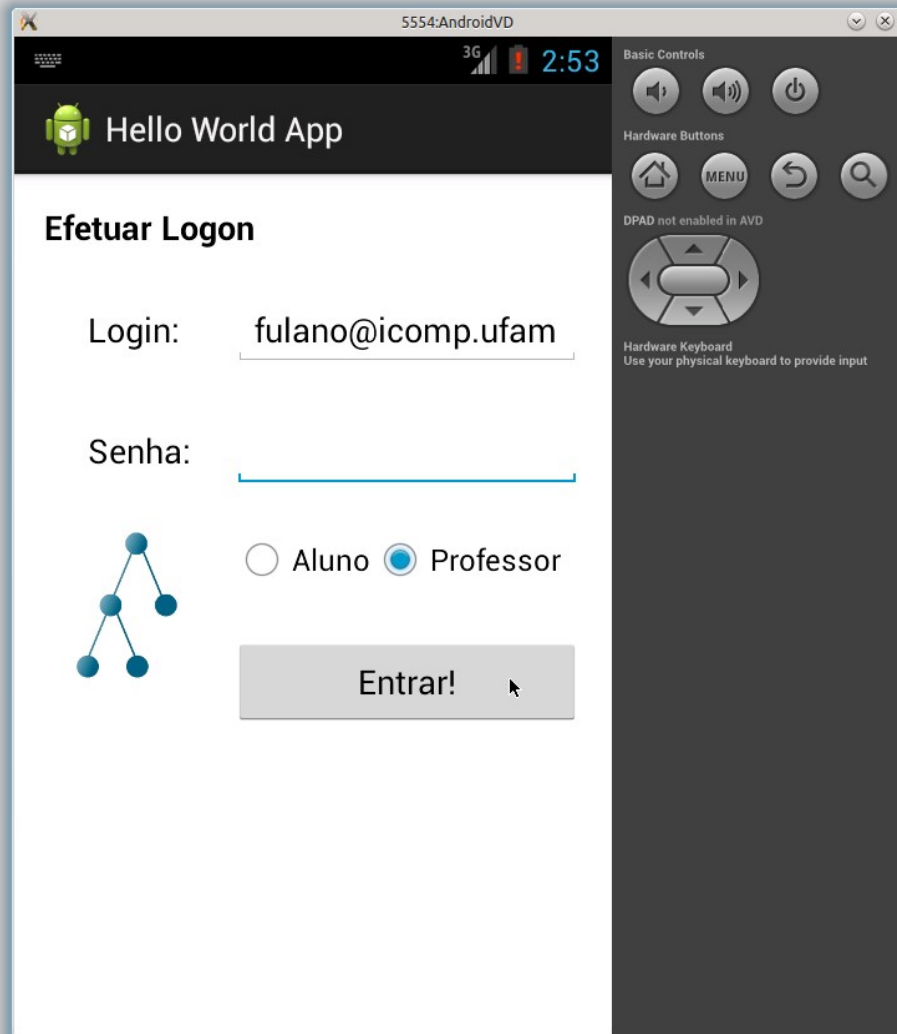
Pega o usuário com o login/senha

Login/senha corretos

Passa o usuário retornado para a nova Activity

Android Banco de Dados – Verificar log

- Teste da nova MainActivity



- Mudando a MainActivity para capturar o evento de pressionar o Menu → Novo Usuário

```
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
  
        case R.id.novo_usuario:  
            Intent intent = new Intent(this, NovoUsuarioActivity.class);  
            startActivity(intent);  
            return true;  
  
        case R.id.sobre:  
            AlertDialog.Builder alert = new AlertDialog.Builder(this);  
            alert.setMessage("Hello World App v1.0")  
                .setNeutralButton("Ok", null).show();  
            return true;  
        case R.id.configs:  
            return true;  
        default:  
            return super.onOptionsItemSelected(item);  
    }  
}
```

Inicia a Activity
NovoUsuarioActivity

- Layout da *Activity* NovoUsuario:

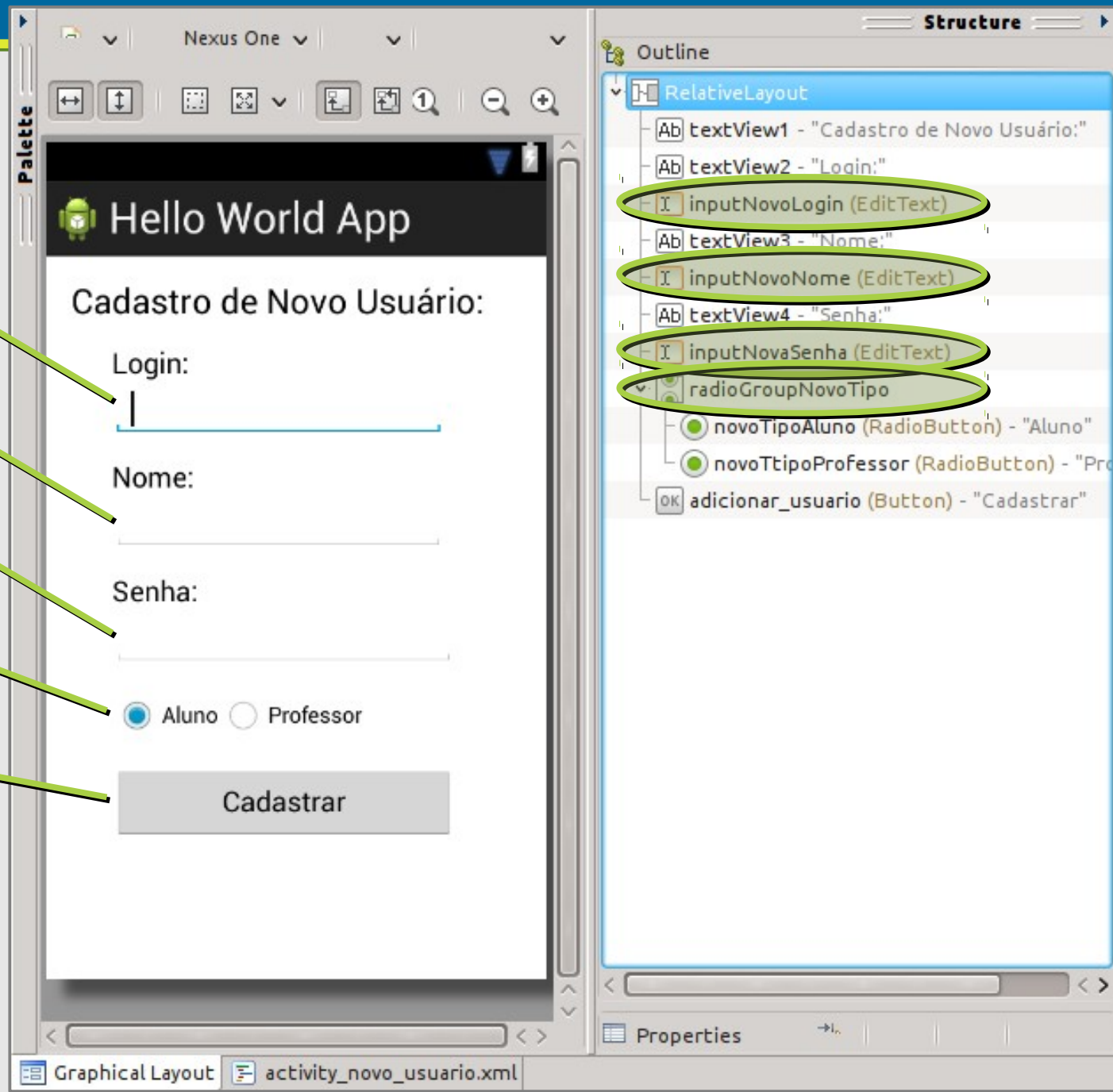
inputNovoLogin

inputNovoNome

inputNovaSenha

radioGroupNovoTipo

android:onClick="cadaststrarUsuario"



Android

Banco de Dados – Adicionar um novo usuário

```
// package, imports, public class
```

```
public class NovoUsuarioActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_novo_usuario);  
    }
```

Ao clicar no
botão cadastrar

Acessa os
inputs

Cria o objeto
do usuário

```
public void cadastrarUsuario(View view) {  
    EditText inputLogin = (EditText) findViewById(R.id.inputNovoLogin);  
    EditText inputNome = (EditText) findViewById(R.id.inputNovoNome);  
    EditText inputSenha = (EditText) findViewById(R.id.inputNovaSenha);  
    RadioGroup groupTipo = (RadioGroup) findViewById(R.id.radioGroupNovoTipo);  
    int tipo = groupTipo.getCheckedRadioButtonId() == R.id.novoTipoAluno ? 1:2;  
  
    Usuario usuario = new Usuario(inputLogin.getText().toString(),  
        inputNome.getText().toString(), inputSenha.getText().toString(), tipo);  
  
    UsuarioDAO usuarioDAO = new UsuarioDAO(this);  
    if (usuarioDAO.addUsuario(usuario))  
        Toast.makeText(this, "Usuário criado!", Toast.LENGTH_SHORT).show();  
    else  
        Toast.makeText(this, "Erro ao criar usuário!", Toast.LENGTH_SHORT).show();  
    finish();  
}
```

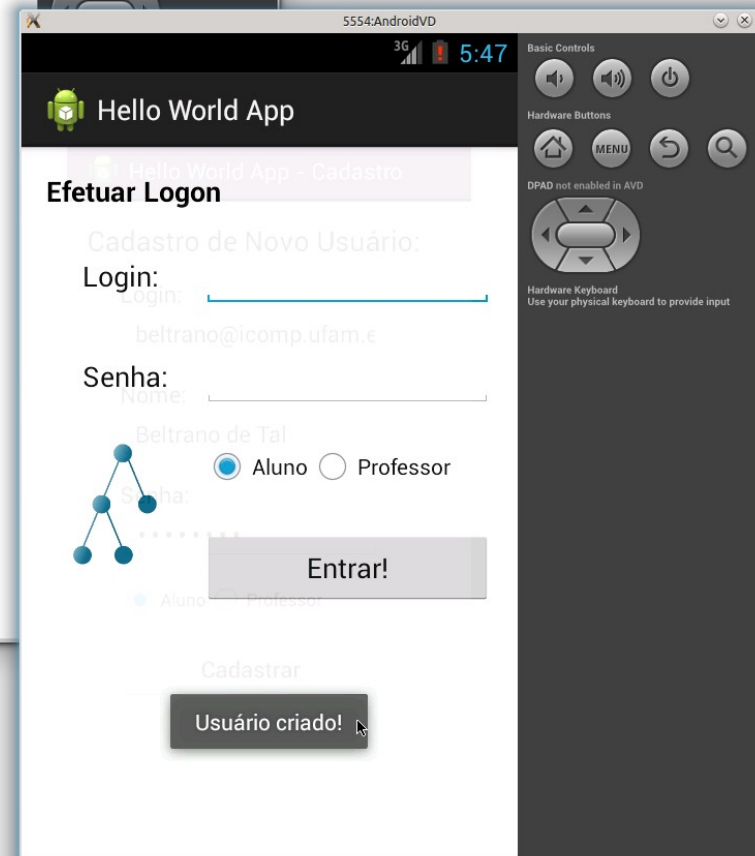
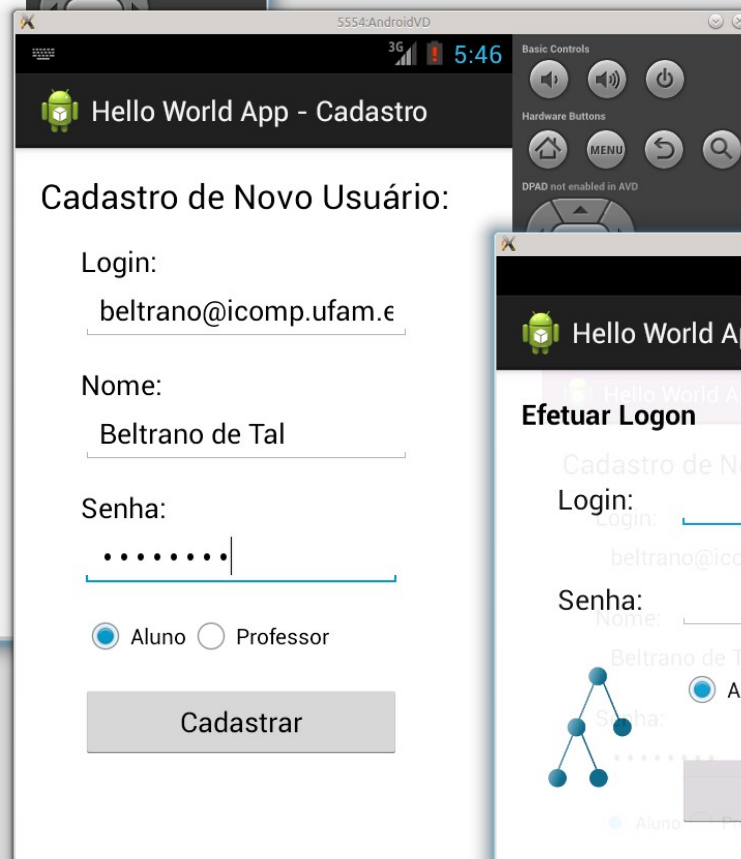
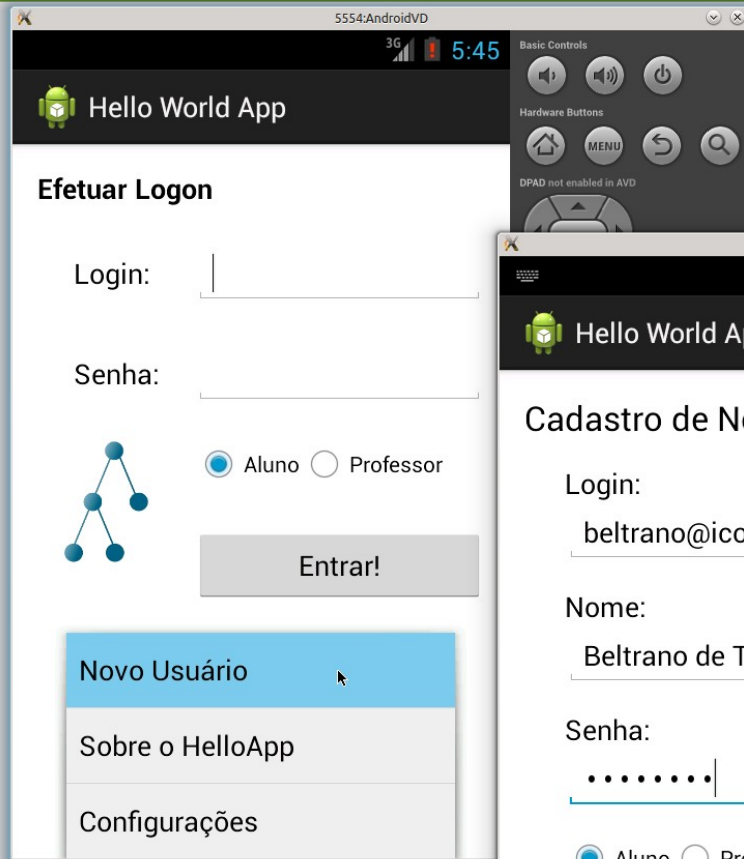
Adiciona o
usuário no BD

Volta à Activity
anterior

Android

Banco de Dados – Adicionar um novo usuário

- Teste

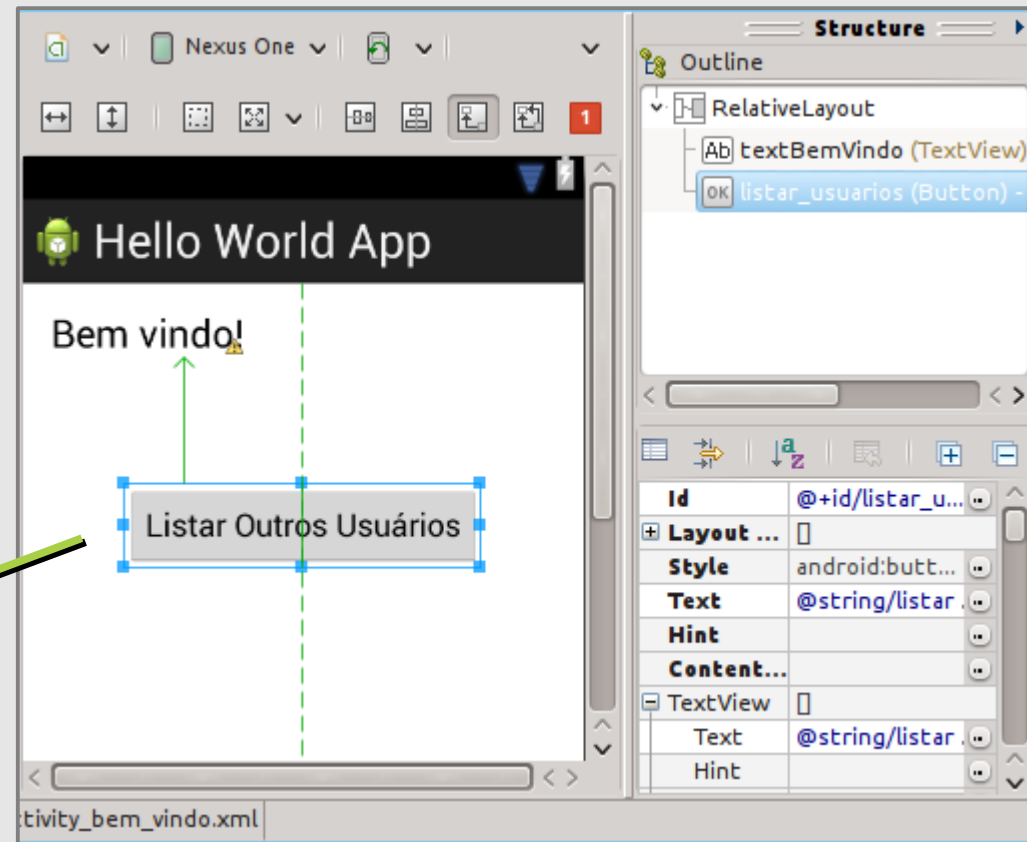


Android

Banco de Dados – Listando Usuários

- Mudando a *view* da BemVindoActivity para adicionar um botão para “Listar Usuários”

onClick



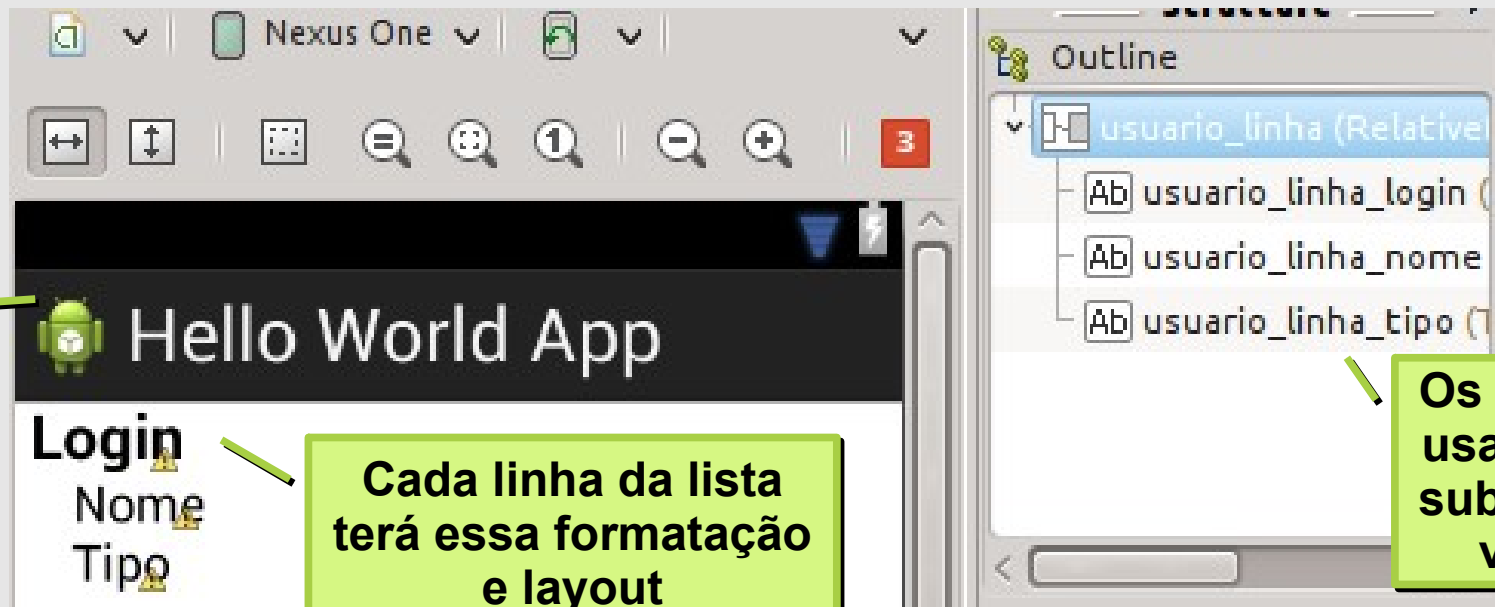
```
// BemVindoActivity.java
public void abrirListarUsuarios(View view) {
    Intent intent = new Intent(this, ListarUsuariosActivity.class);
    startActivity(intent);
}
```

- Criando a `ListarUsuariosActivity`
 - Android possui uma classe de *Activity* específica para os casos em que a *view* contém uma “lista”: a `ListActivity`
- Para criar uma nova `ListActivity`:
 - Crie uma *Activity* normal
 - *File* → *New* → *Other ...* → *Android* → *Android Activity* → *Blank Activity*
 - Faça-a estender a classe `ListActivity`
 - A classe *ListActivity* estende a classe *Activity*

Android

Banco de Dados – Listando Usuários

- View:
 - Não há necessidade de modificar a view da *activity*, pois a `ListActivity` irá criar automaticamente o componente UI para a lista
 - Entretanto, precisamos definir a *view* que cada uma das linhas da lista terá
 - *File* → *New* → *Other ...* → *Android* → *Android XML Layout File*
 - *File:* *usuario_linha.xml*
 - *Root Element:* *RelativeLayout*




```
public class ListarUsuariosActivity extends ListActivity {  
    private UsuarioDAO usuarioDAO;  
    private SimpleCursorAdapter dados;  
  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        this.usuarioDAO = new UsuarioDAO(this);  
  
        dados = new SimpleCursorAdapter(this,  
            R.layout.usuario_linha,  
            usuarioDAO.getUsuarios(),  
            new String[] { "login", "nome", "tipo" },  
            new int[] { R.id.usuario_linha_login,  
                R.id.usuario_linha_nome,  
                R.id.usuario_linha_tipo }, 0);  
  
        setListAdapter(dados);  
    }  
  
    public void onItemClick(ListView l, View v, int pos, long id) {  
        TextView textLogin =  
            (TextView) v.findViewById(R.id.usuario_linha_login);  
        Toast.makeText(this, "Usuário " + textLogin.getText().toString(),  
            Toast.LENGTH_SHORT).show();  
    }  
}
```

Acesso aos dados do BD

Fonte de dados da Lista

Ligação entre o BD e a Lista

Layout de cada Linha

Dados do BD

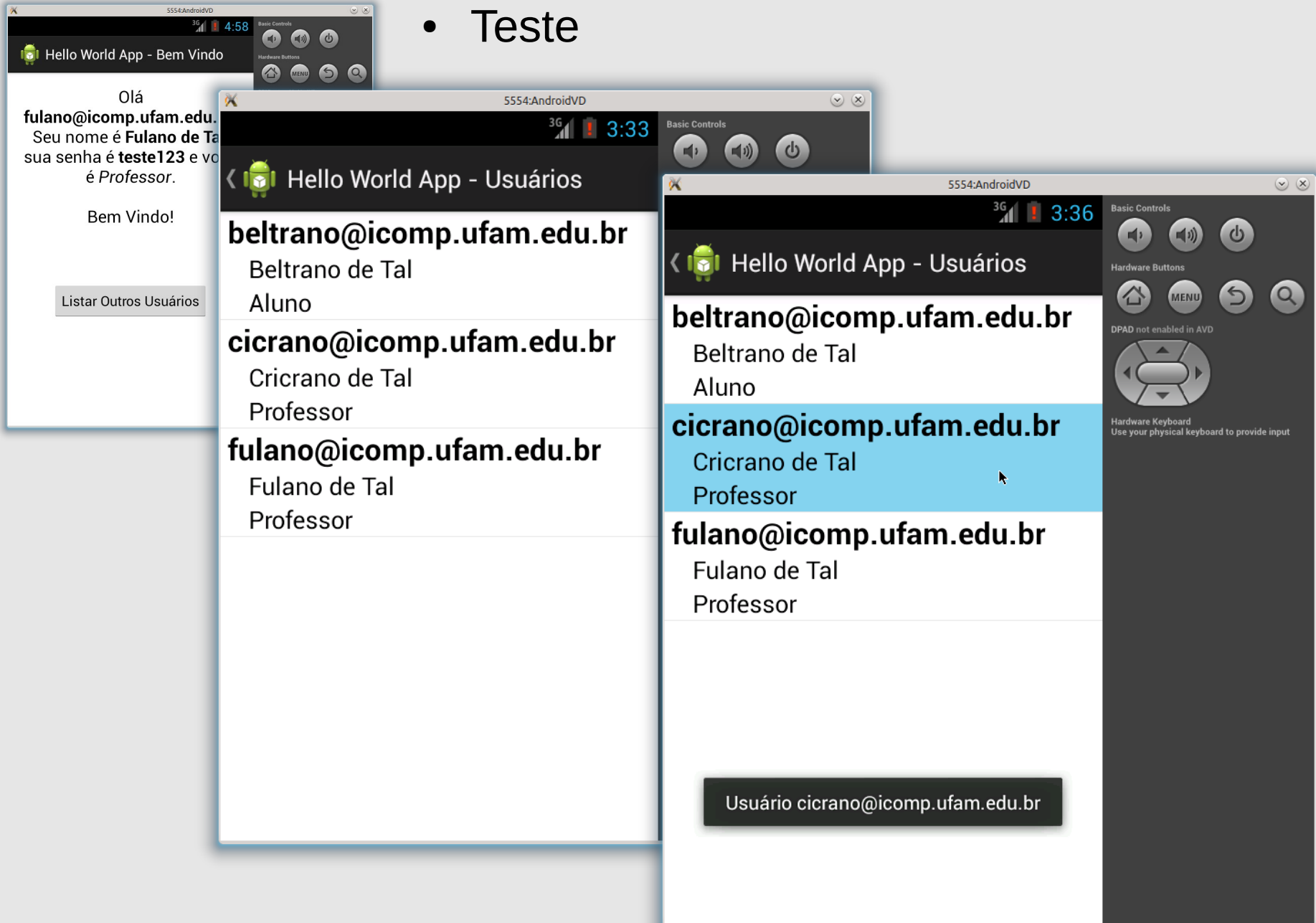
Mapeamento das colunas do BD para os campos da View

Executado ao clicar em uma Linha

Android

Banco de Dados – Listando Usuários

- Teste



Android

Passos para Desenvolvimento

Configuração

Instalação
do ADT Bundle

Hello World
App

Configuração
do AVD

Execução
do App

Desenvolvimento

Interface
Gráfica

*Activities e
Intents*

Banco de
Dados

**Tópicos
Avançados**

Android

Tópicos Avançados

- Fragmentos
 - Criar aplicativos em uma só tela com “fragmentos” dos lados
- Configurações e Preferências
- Serviços Baseados em Localização
 - GPS, Mapas
- Sensores
 - Acelerômetro, Giroscópio, Luminosidade, Magnetômetro
- Multimídia
 - Câmera, Vídeo, Voz
- Telefonia
 - Contatos, SMS

Android

Referências

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