

INICIANDO COM A PLATAFORMA ANDROID



“Não basta adquirir sabedoria; é preciso, além disso, saber utilizá-la.” (Cicero , 106-43 a.C.)

CONTEÚDO

- Instalação do Android Studio e Android SDK
- Emuladores do Android
- Minha primeira aplicação
- Entendendo o que foi gerado
- Componentes GUI



INSTALAÇÃO E USER GUIDE

- <https://developer.android.com/studio/install.html>

The screenshot shows the Android Studio User Guide page for installing the IDE. The top navigation bar includes the Android Studio logo, 'FEATURES' and 'USER GUIDE' tabs, and a search bar. The left sidebar has a 'User Guide' section with links like 'Meet Android Studio', 'Workflow Basics', 'Manage Your Project', 'Write Your App', 'Build and Run Your App', 'Configure Your Build', 'Debug Your App', and 'Test Your App'. The main content area is titled 'Instalar o Android Studio' and provides instructions for Windows users. It includes steps for executing the .exe file and configuring the JDK path. A dropdown menu on the right allows selecting 'Windows' or 'Mac OS X'.

A configuração do Android Studio pode ser feita com apenas alguns cliques. (Você precisa antes [baixar o Android Studio](#).)

Para instalar o Android Studio no Windows, faça o seguinte:

1. Execute o arquivo `.exe` que você baixou.
2. Siga o assistente de configuração para instalar o Android Studio e todas as ferramentas do SDK necessárias.

Em alguns sistemas Windows, o script de inicialização não encontrará o local de instalação do JDK. Se ocorrer esse problema, será preciso definir uma variável de ambiente indicando o local correto.

Selecione Start menu > Computer > System Properties > Advanced System Properties. Em seguida, abra a guia Advanced > Environment Variables e adicione uma nova variável de sistema, `JAVA_HOME`, que aponta para a pasta do JDK. Por exemplo, `C:\Program Files\Java\jdk1.8.0_77`.



ANDROID STUDIO X ECLIPSE ADT

Android Studio vs. Eclipse ADT Comparison

The following table lists some key differences between Android Studio Beta and [Eclipse with ADT](#).

Feature	Android Studio	ADT
Build system	Gradle 	Ant 
Maven-based build dependencies	Yes	No
Build variants and multiple-APK generation (great for Android Wear)	Yes	No
Advanced Android code completion and refactoring	Yes	No
Graphical layout editor	Yes	Yes
APK signing and keystore management	Yes	Yes
NDK support	Coming soon	Yes



AVDS DO ANDROID

- São criados/definidos via **AVD Manager**, recurso disponível no Android SDK;
- Permite definir parâmetros de hardware personalizados na criação dos emuladores;
- Podem ser baixados e integrados alguns modelos de emuladores prontos.



AVDS DO ANDROID

Android Virtual Device Manager

Your Virtual Devices

Android Studio

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
Tablet	Nexus 4 API 10	768 x 1280: xhdpi	10	Android 2.3.3	arm	103 MB	▶️ 🖊️ 🔍
Tablet	Nexus 5 API 22 x86	1080 x 1920: xxhdpi	22	Google APIs	x86	750 MB	▶️ 🖊️ 🔍

Máquinas virtuais para executar as aplicações.

Create Virtual Device...



ANDROID DEVICES

Virtual Device Configuration

Android Virtual Device (AVD)

Verify Configuration

Latency:

Emulated Performance

Use Host GPU (Requires API > 15)
 Store a snapshot for faster startup
You can either use Host GPU or Snapshots

Memory and Storage

RAM: MB

VM heap: MB

Internal Storage: MB

SD card: Studio-managed MB
 External file ...

Custom skin definition

[How do I create a custom hardware skin?](#)

Keyboard

Enable keyboard input

Nothing Selected



Prof. Roberson Alves

7

EMULADORES DO ANDROID

- Utilizados para desenvolver aplicações no Android, que tem um emulador para simular o celular, ferramentas utilitárias e uma API completa para a linguagem Java, com todas as classes necessárias para desenvolver as aplicações;
- Também é possível plugar um dispositivo(smartphone ou tablet) na porta USB do computador e executar os aplicativos diretamente no dispositivo.



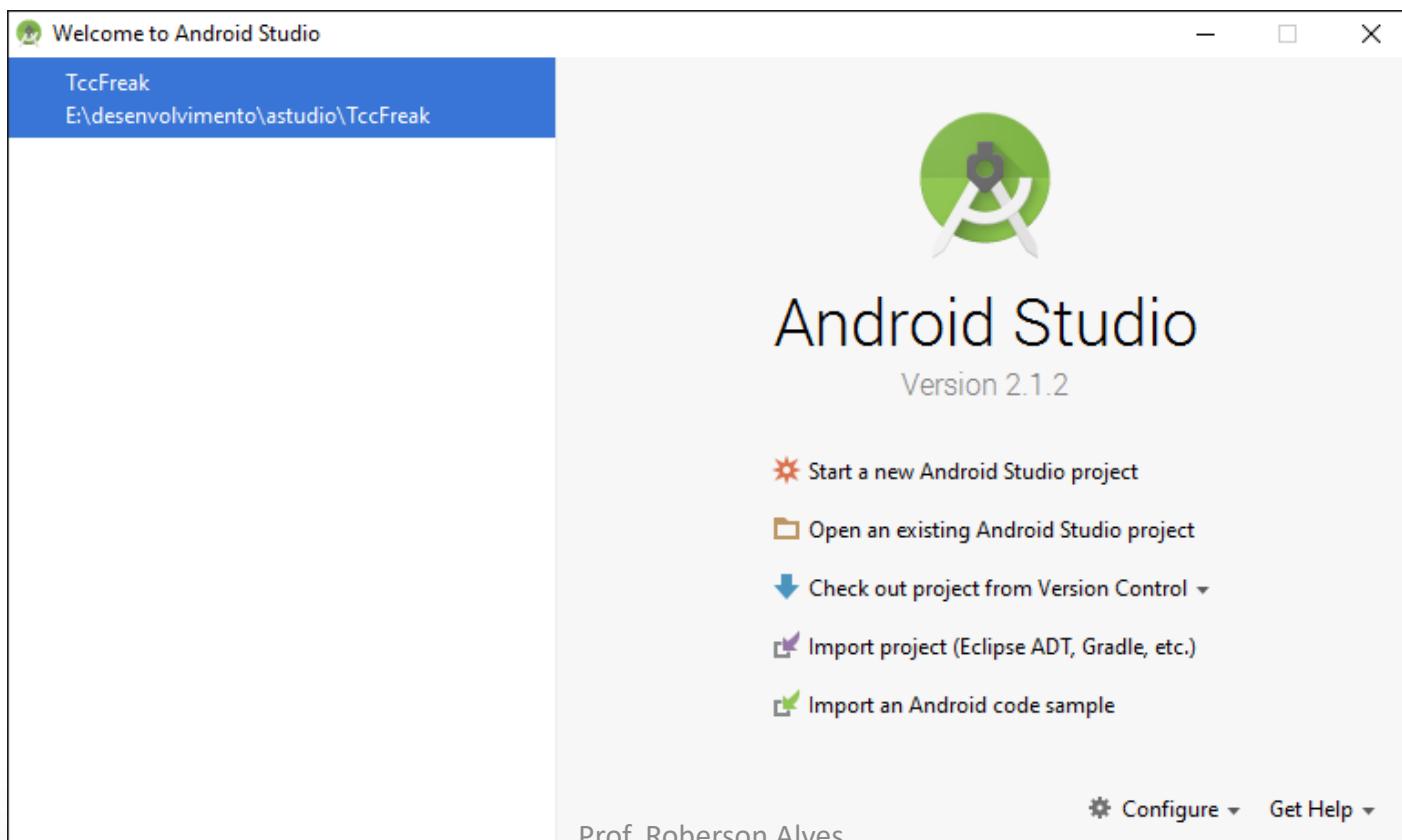
MINHA PRIMEIRA APLICAÇÃO: CONHECENDO O ANDROID STUDIO

- Vamos criar nosso primeiro programa no Android Studio!



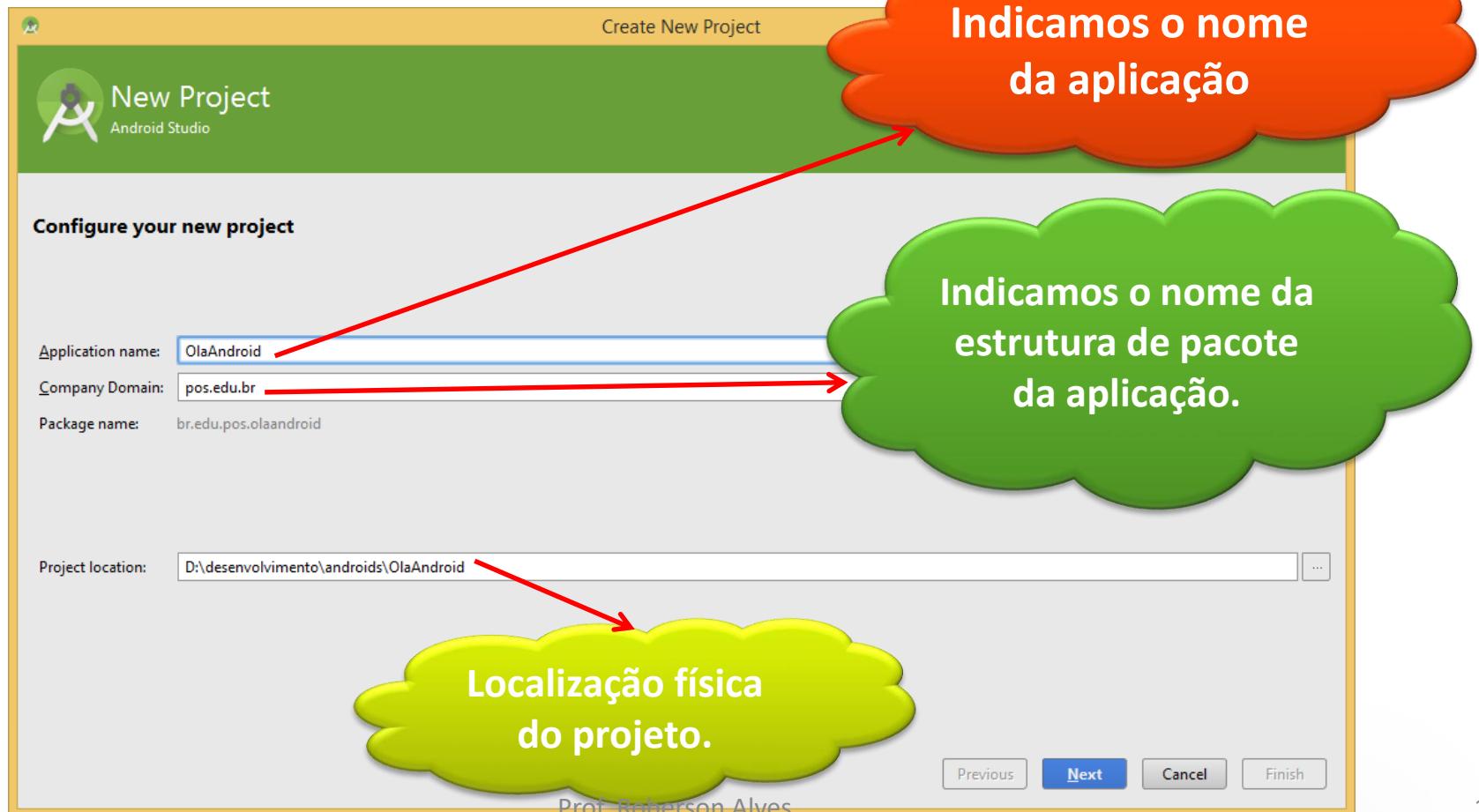
MINHA PRIMEIRA APLICAÇÃO: CONHECENDO A IDE ANDROID STUDIO

- Para criar o projeto selecione: *Start a new Android Studio project*



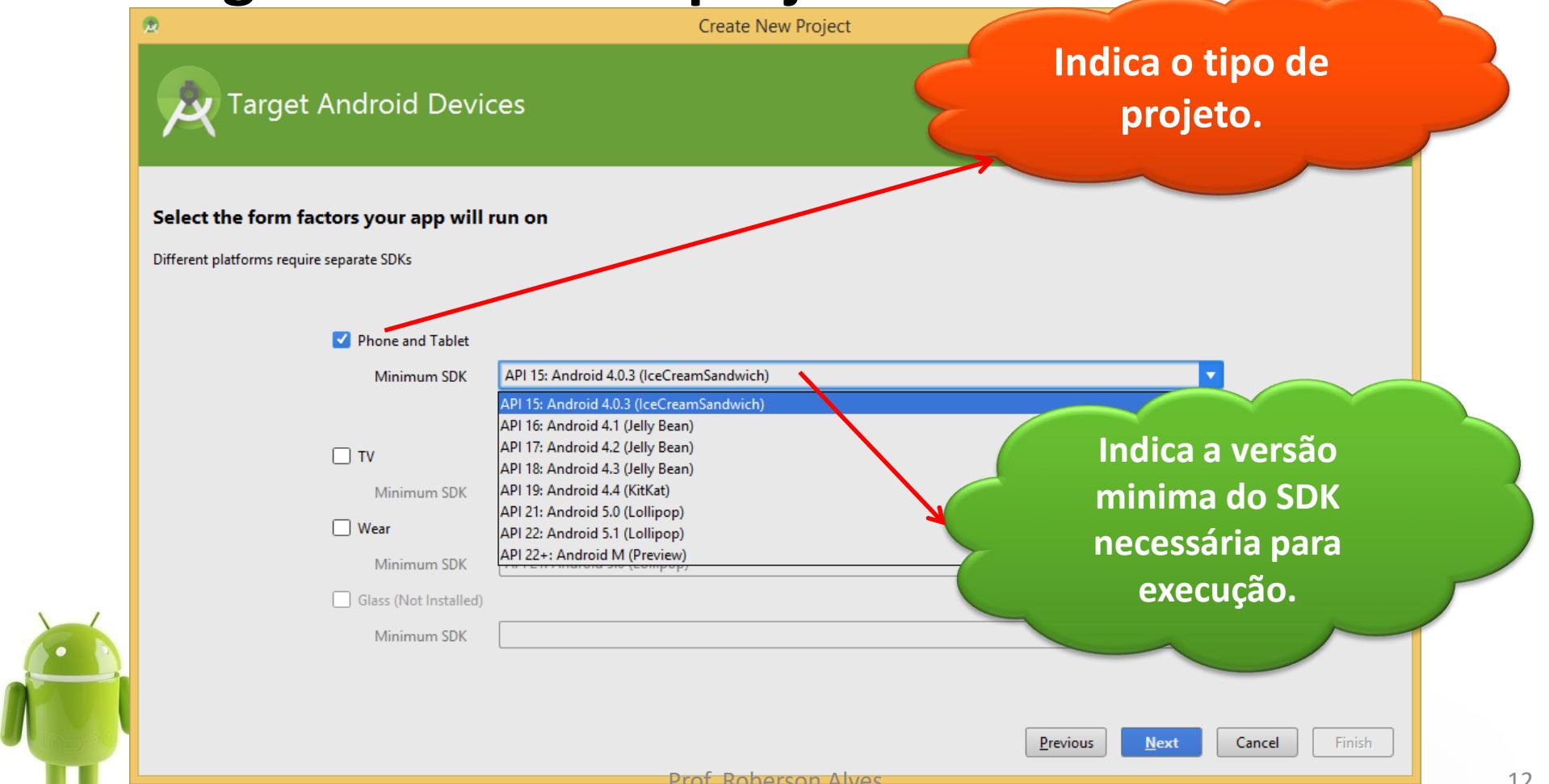
MINHA PRIMEIRA APLICAÇÃO: OLAANDROID

- Configurando o novo projeto



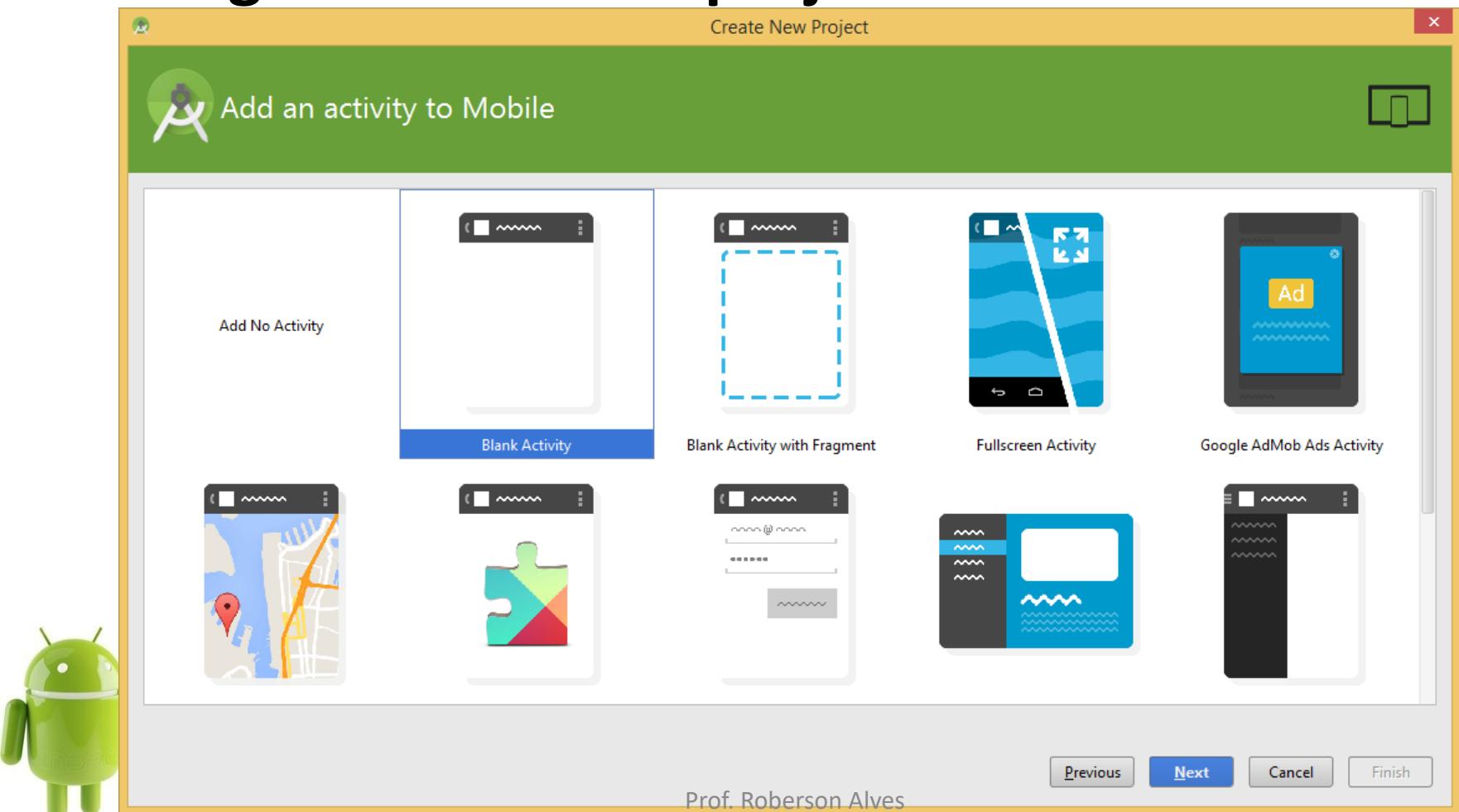
MINHA PRIMEIRA APLICAÇÃO: OLAANDROID

- Configurando o novo projeto



MINHA PRIMEIRA APLICAÇÃO: OLAANDROID

- Configurando o novo projeto: modelo de atividade



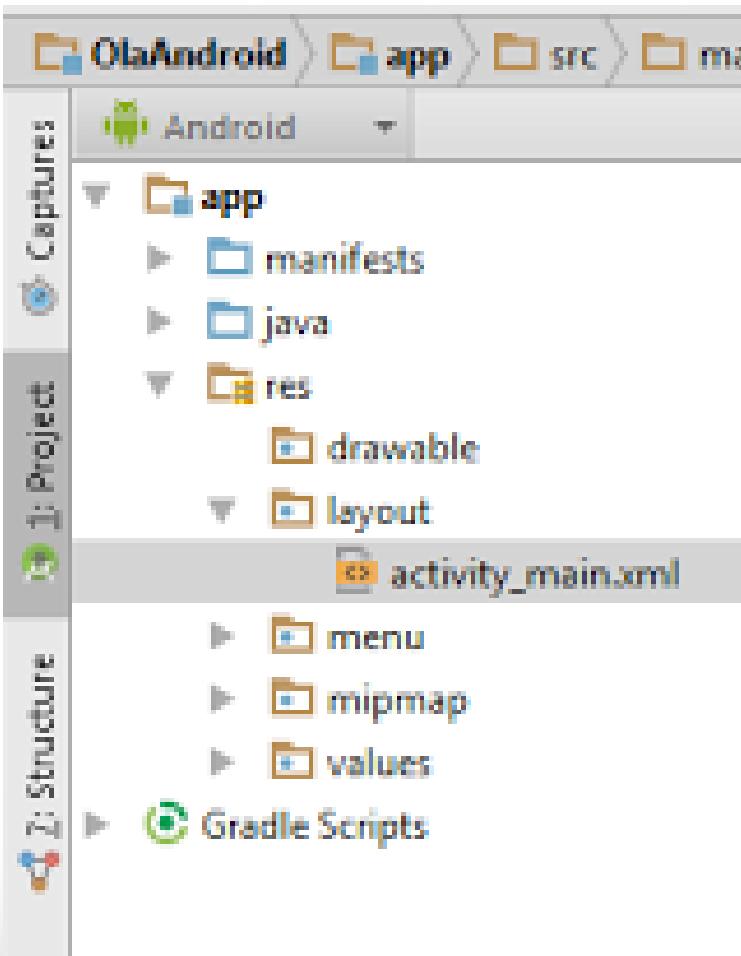
MINHA PRIMEIRA APLICAÇÃO: OLAANDROID

- Configurando o novo projeto: dados da atividade



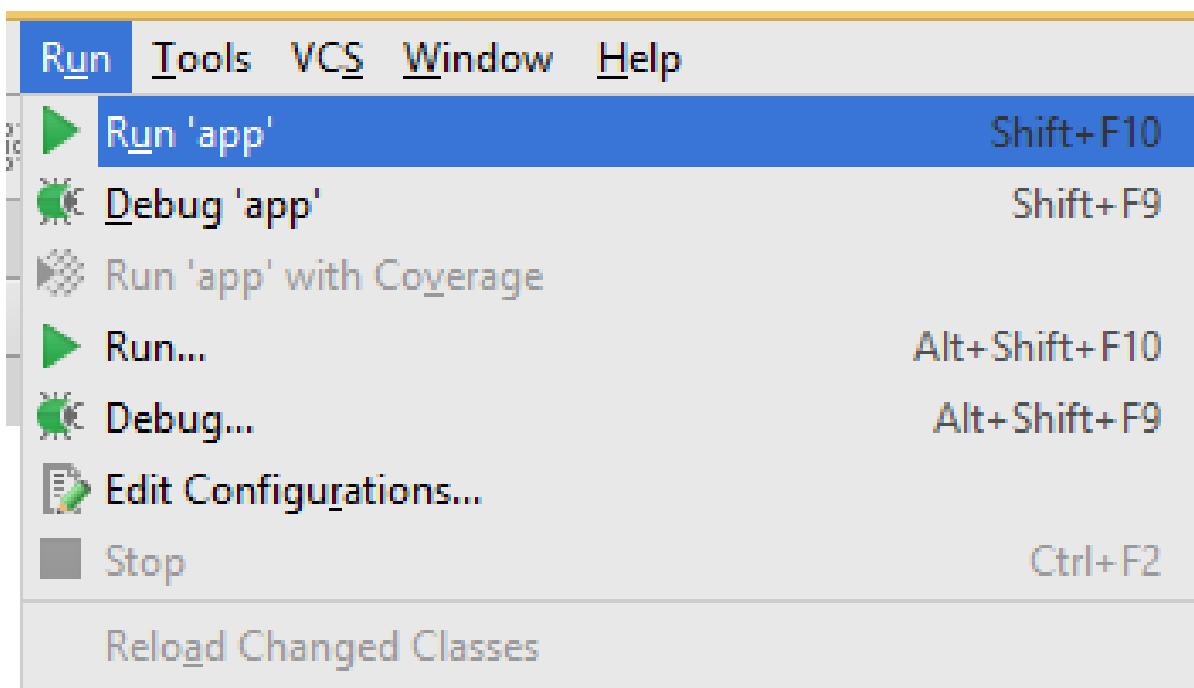
MINHA PRIMEIRA APLICAÇÃO: OLAANDROID

- Aguarde a criação do projeto! Após a criação teremos ...



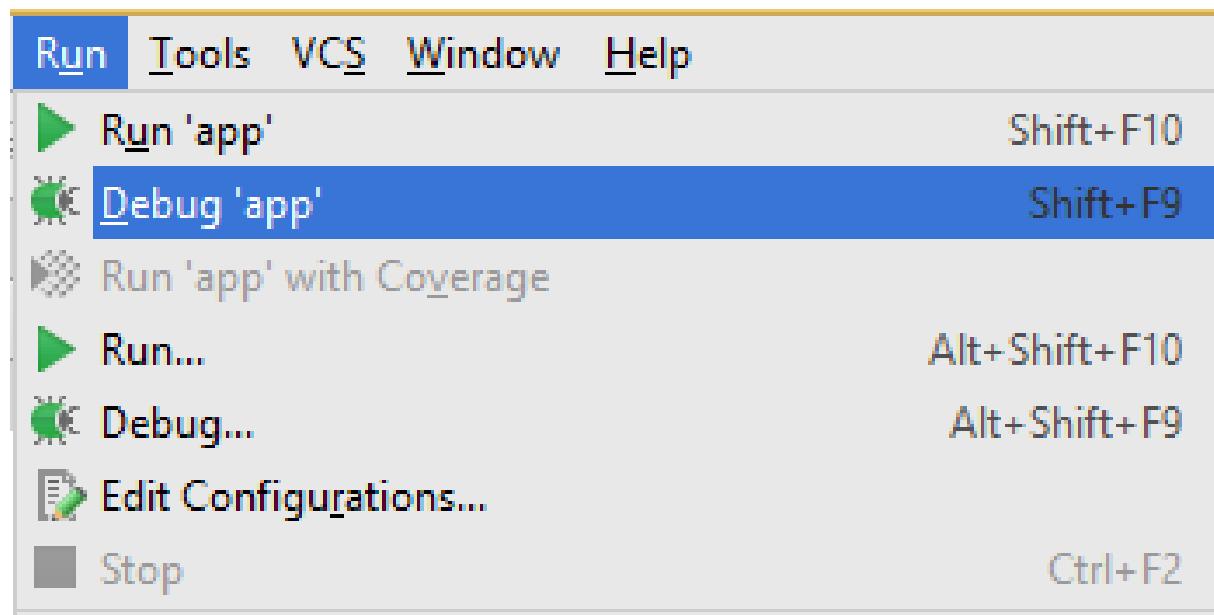
MINHA PRIMEIRA APLICAÇÃO: EXECUTANDO A APLICAÇÃO NO EMULADOR OU DISPOSITIVO

- Menu Run -> Run ‘...’ ou Shift+F10



MINHA PRIMEIRA APLICAÇÃO: DEPURANDO A APLICAÇÃO NO EMULADOR OU DISPOSITIVO

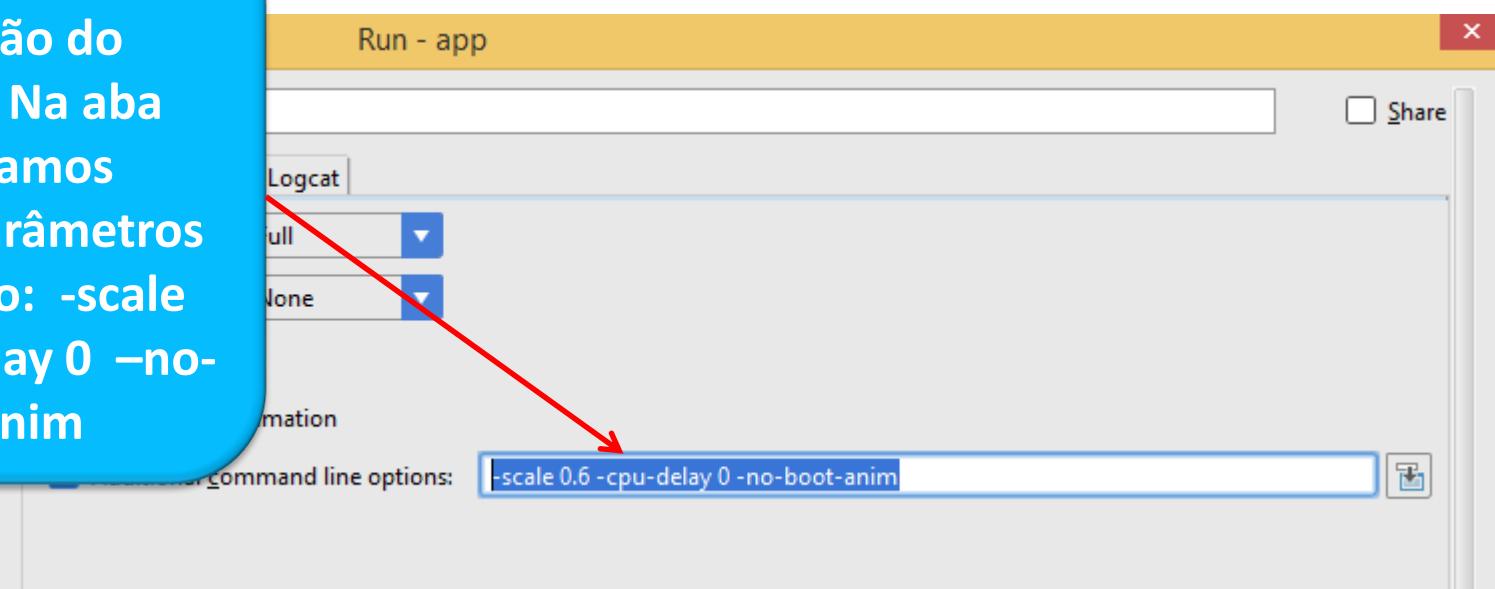
- Menu **Run -> Debug '...' ou Shift+F9**



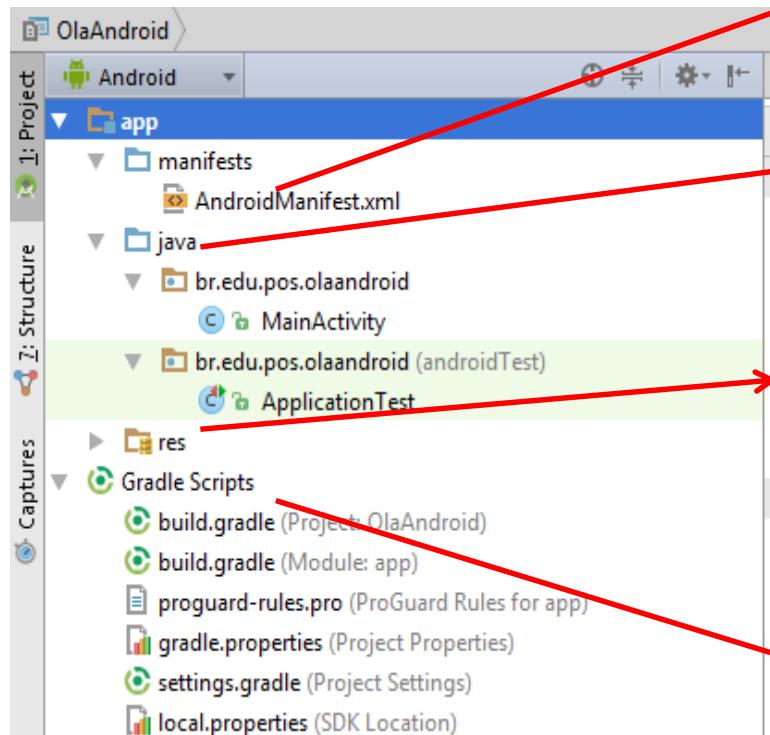
MINHA PRIMEIRA APLICAÇÃO: EXECUTANDO A APLICAÇÃO NO EMULADOR OU DISPOSITIVO

- Indicando alguns parâmetros para execução da aplicação

Vamos tentar otimizar a execução do emulador. Na aba Target vamos adicionar parâmetros de execução: -scale 0.6 -cpu-delay 0 -no-boot-anim



ENTENDENDO O QUE FOI GERADO



Arquivo de configuração da aplicação.

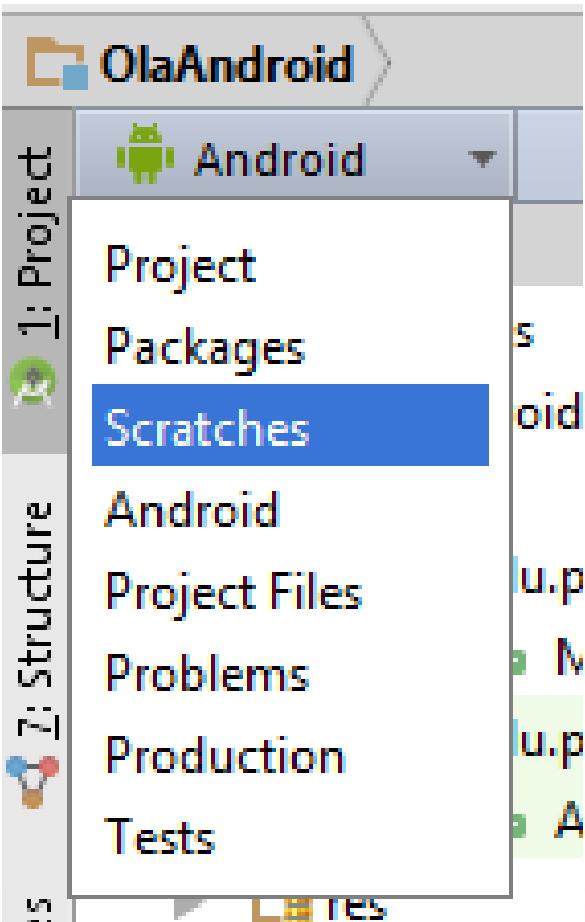
Diretório com os códigos fonte da aplicação.

Diretório que contém os recursos da aplicação, como imagens, menus, layouts de telas e arquivos de internacionalização. Tem três subpastas: drawable-*, layout e values.

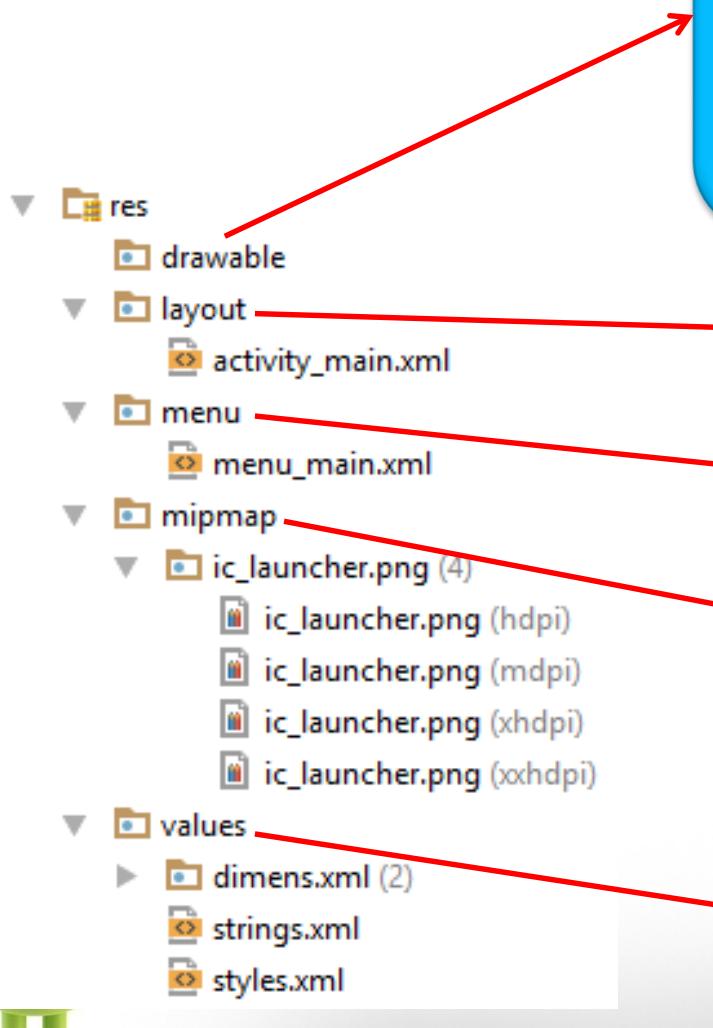
Scripts de build do Gradle.



ENTENDENDO O QUE FOI GERADO



ENTENDENDO O QUE FOI GERADO



Pasta com as imagens da aplicação. Atualmente como existem diversos celulares Android com resolução de tela diferentes, é possível customizar as imagens para ficar com o tamanho exato em cada resolução automaticamente. Para isso existem 3 pastas para as imagens: **drawable-ldpi**, **drawable-mdpi**, **drawable-hdpi** e **drawable-xhdpi**.

Contém os arquivos XML de layouts para construir as telas da aplicação.

Contém os arquivos XML de menus.

Diretório específico para os ícones da app.

Contém os arquivos XML utilizados para internacionalização da aplicação e outras configurações. O XML é composto de um layout.

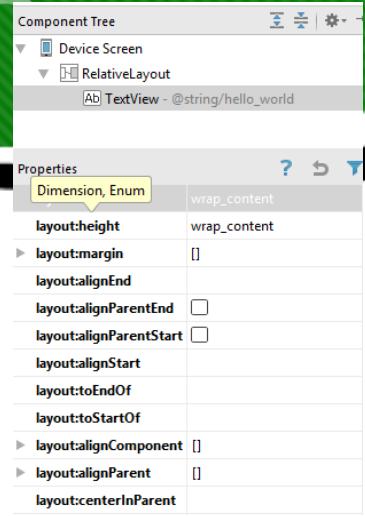
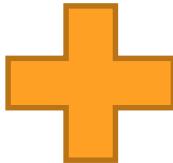
ANATOMIA DA APLICAÇÃO



```
<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"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world" />

</RelativeLayout>
```

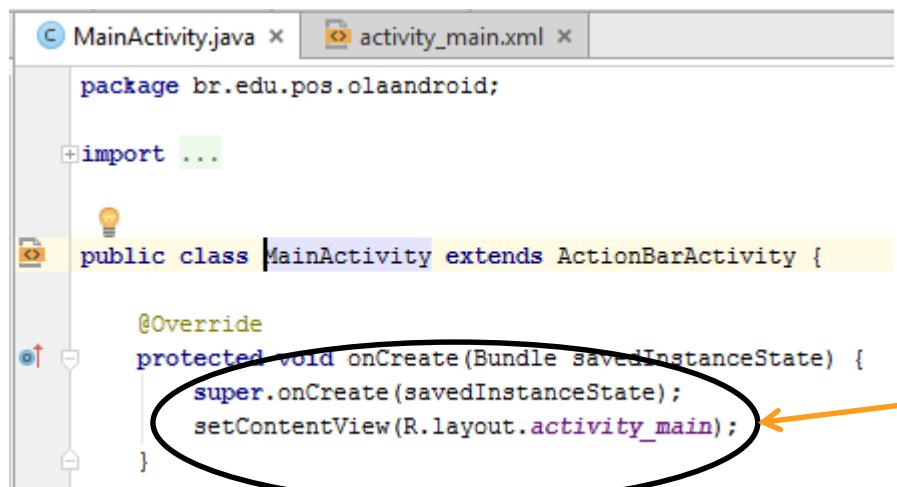


Component Tree

- Device Screen
- RelativeLayout
 - TextView - @string/hello_world

Properties

Dimension, Enum	wrap_content
layout:height	wrap_content
layout:margin	[]
layout:alignEnd	
layout:alignParentEnd	<input type="checkbox"/>
layout:alignParentStart	<input type="checkbox"/>
layout:alignStart	
layout:toEndOf	
layout:toStartOf	
layout:alignComponent	<input type="checkbox"/>
layout:alignParent	<input type="checkbox"/>
layout:centerInParent	



```
package br.edu.pos.olaandroid;

import ...

public class MainActivity extends ActionBarActivity {

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



ACTIVITY(ou atividade)

- Representa uma tela da aplicação
- Recurso do Android que oferece uma tela com a qual os usuários podem interagir
- Pode-se fazer uma analogia com uma janela ou caixa de diálogo de sistemas desktop
- A implementação de activity deve herdar características da classe **android.app.Activity**

```
public class MainActivity extends Activity {
```

- Deve implementar o método **onCreate(Bundle)**
- Cada activity deve ser declarada dentro do **AndroidManifest.xml**



ACTIVITY – ANDROIDMANIFEST.XML

The screenshot shows the AndroidManifest.xml file in an IDE. The code is color-coded: blue for XML tags, green for attributes, and red for values. A yellow highlight covers the theme definition. The left sidebar has icons for application, activity, intent-filter, category, and manifest.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.edu.pos.olaandroid" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="OlaAndroid"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="OlaAndroid" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

ACTIVITY

- Dentro do método `onCreate`, é invocado o método `onCreate` da super classe passando o mesmo parâmetro (**Bundle savedInstanceState**);
- Logo após esta chamada deste método, vem o método **`setContentView`**, responsável por exibir a tela da aplicação, baseada nos layouts XML(eXtensible Markup Language).



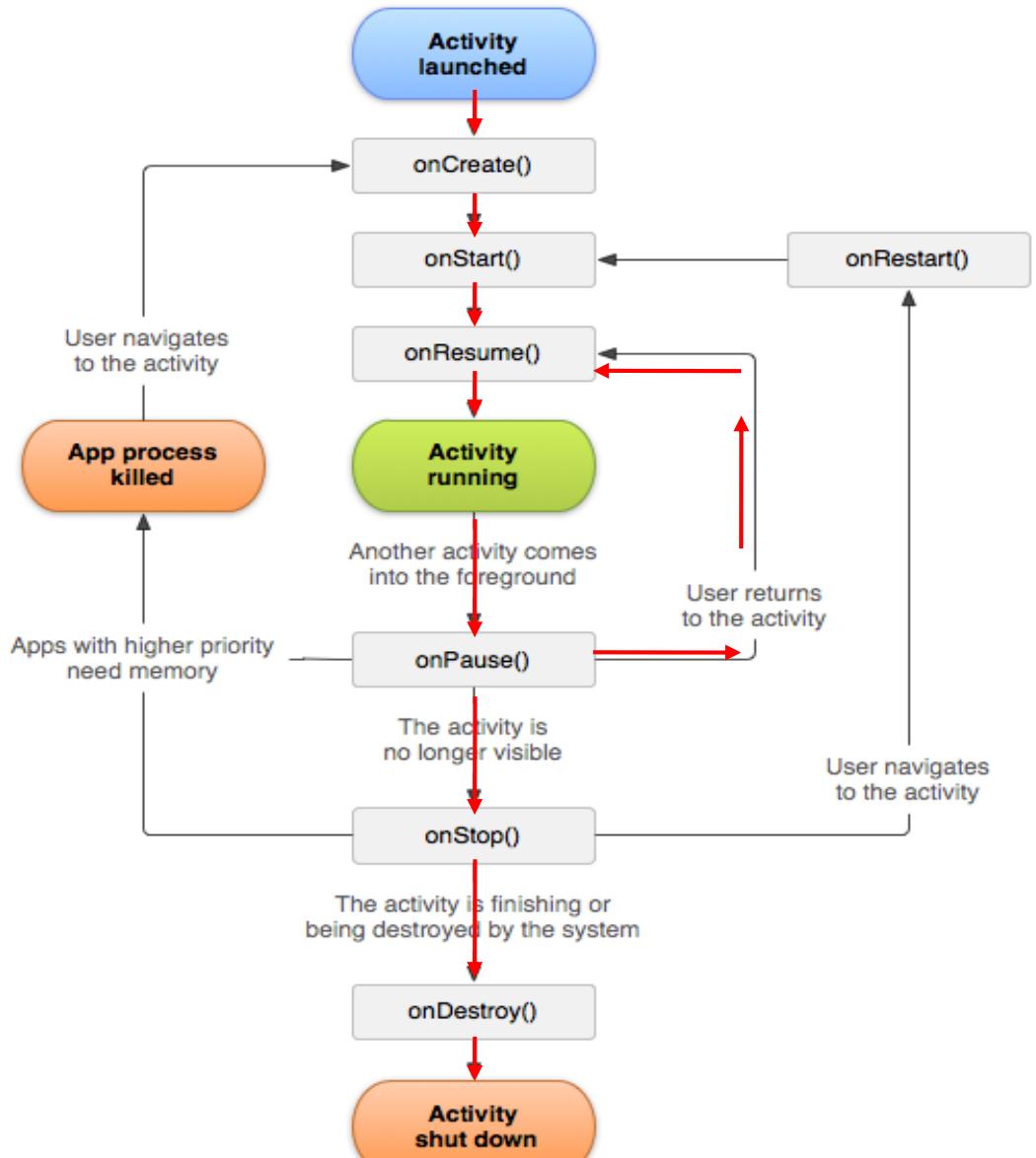
ACTIVITY - IMPLEMENTAÇÃO

Método(callback) onCreate é redefinido.

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



CICLO DE VIDA DE UMA ACTIVITY



Fonte: Android Developers(2012a).



CICLO DE VIDA DE UMA ACTIVITY

- Possíveis estados em que se encontra:
 - executando;
 - temporariamente interrompida;
 - em segundo plano;
 - completamente destruída.
- O sistema operacional cuida desse ciclo de vida, mas ao desenvolver aplicações é importante levar cada estado possível em consideração.
- Cada activity iniciada é inserida no topo de uma pilha de atividades. A activity que está no topo da pilha é a que está em execução no momento, as demais podem estar executando em segundo plano, estar no estado pausado ou totalmente paradas.



CICLO DE VIDA DE UMA ACTIVITY

- Há métodos(callbacks) da classe Activity para controlar os estados de uma atividade:
 - **onCreate(Bundle)**
 - **onStart()**
 - **onRestart()**
 - **onResume()**
 - **onPause()**
 - **onStop()**
 - **onDestroy()**

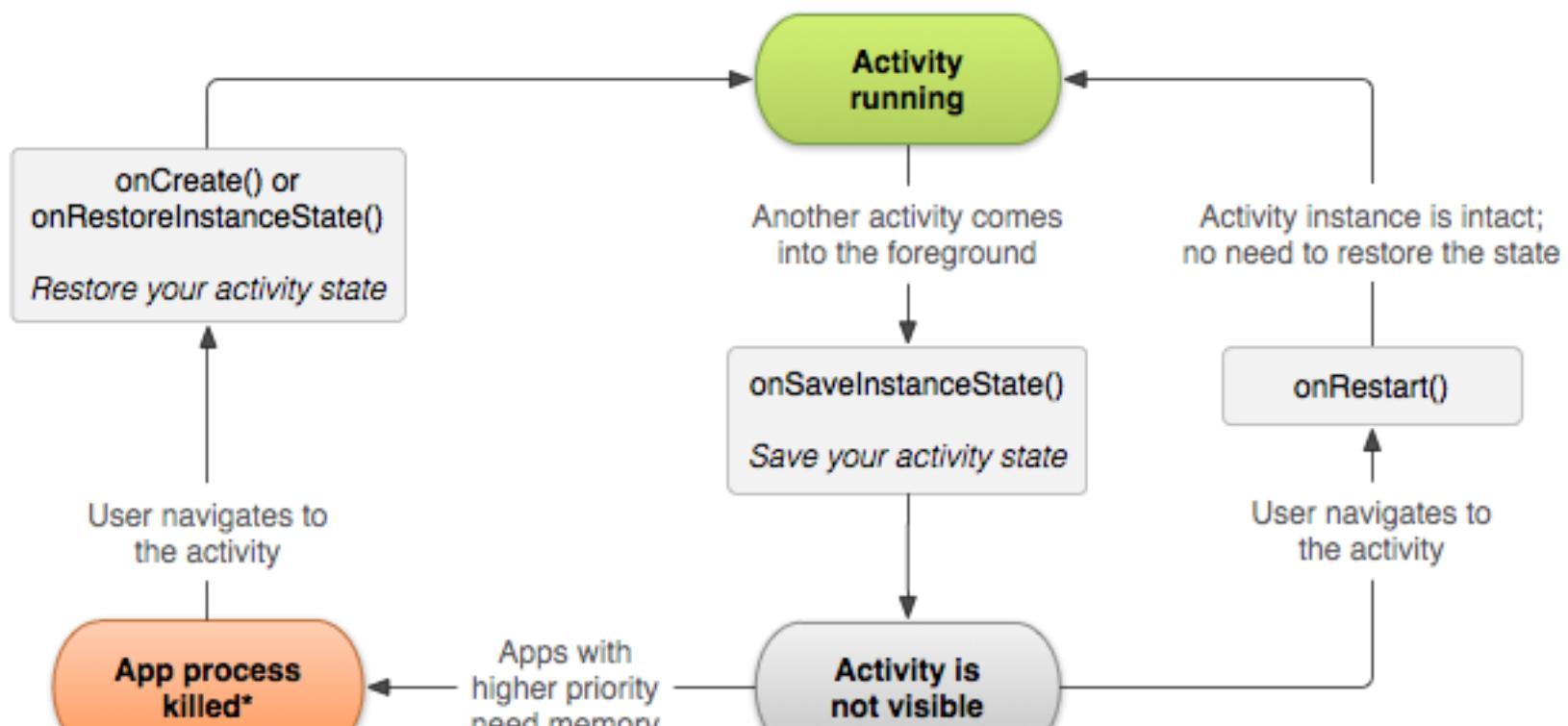


CICLO DE VIDA DE UMA ACTIVITY

- Existem três subníveis do ciclo de vida principal, que por sua vez ficam se repetindo(estão encadeados) durante a execução da aplicação:
 - **entire lifetime:** ciclo de vida completo entre o início e destruição de uma activity
 - `onCreate(Bundle)`
 - `onDestroy()`
 - **visible lifetime:** activity está iniciada, mas pode estar no topo da pilha interagindo com o usuário ou temporariamente parada em segundo plano
 - `onStart()`
 - `onStop()`
 - `onRestart()`
 - **foreground lifetime:** activity está no topo da pilha e interagindo com o usuário
 - `onResume()`
 - `onPause()`



CICLO DE VIDA DE UMA ACTIVITY: salvando o estado da activity



*Activity instance is destroyed, but the state from `onSaveInstanceState()` is saved

Fonte: Android Developers(2012a).



CICLO DE VIDA DE UMA ACTIVITY

Método	Descrição	Destruível depois?	Método seguinte
onCreate(Bundle)	É obrigatório e é chamado uma única vez. Dentro dele deve-se chamar o método para construir a view e passar para o método onStart() para iniciar o ciclo visível da activity. Bundle é um tipo de objeto que permite guardar o estado anterior de uma activity.	Não	onStart()
onRestart()	É chamado quando a activity foi parada e está sendo iniciada outra vez. Chama o método onStart() automaticamente.	Não	onStart()
onStart()	É chamada quando activity está ficando visível para o usuário. Pode ser chamada depois do método onCreate(Bundle) ou onRestart() .	Não	onResume() ou onStop()
onResume()	É chamado quando a activity está no topo da pilha, é o método que representa o estado de que a activity está executando. É chamado sempre depois de onStart().	Não	onPause()
onPause()	Se algum evento ocorrer, como uma ligação durante a execução da activity, a aplicação chama o método onPause(): para salvar o estado da aplicação para posteriormente quando a activity voltar a executar, tudo possa ser recuperado.	Sim	onResume() ou onStop()
onStop()	É chamado quando a activity está sendo encerrada, e não está mais visível ao usuário.	Sim	onRestart() ou onDestroy()
onDestroy()	Encerra a execução de uma activity. Após ser executado a activity é removida completamente da pilha e seu processo no sistema operacional também é completamente encerrado.	Sim	-

CICLO DE VIDA DE UMA ACTIVITY: exemplo

```
public class CicloVidaActivity extends Activity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        Log.i("CicloVidaActivity", "Passando pelo método onCreate ...");  
        setContentView(R.layout.main);  
    }  
  
    public void onStart() {  
        super.onStart();  
        Log.i("CicloVidaActivity", "Passando pelo método onStart ...");  
    }  
  
    public void onRestart() {  
        super.onRestart();  
        Log.i("CicloVidaActivity", "Passando pelo método onRestart ...");  
    }  
}
```



CICLO DE VIDA DE UMA ACTIVITY: iniciando uma activity

- Para chamar outra tela precisamos criar outra activity. Para isso existem dois métodos:
 - **startActivity(intent)**: inicia a próxima activity sem nenhum vínculo.
 - **startActivityForResult(intent,codigo)**: recebe um parâmetro que identifica essa chamada, para que posteriormente essa segunda activity possa retornar alguma informação para a activity que a chamou.
- Esse método utiliza a classe **Intent**.



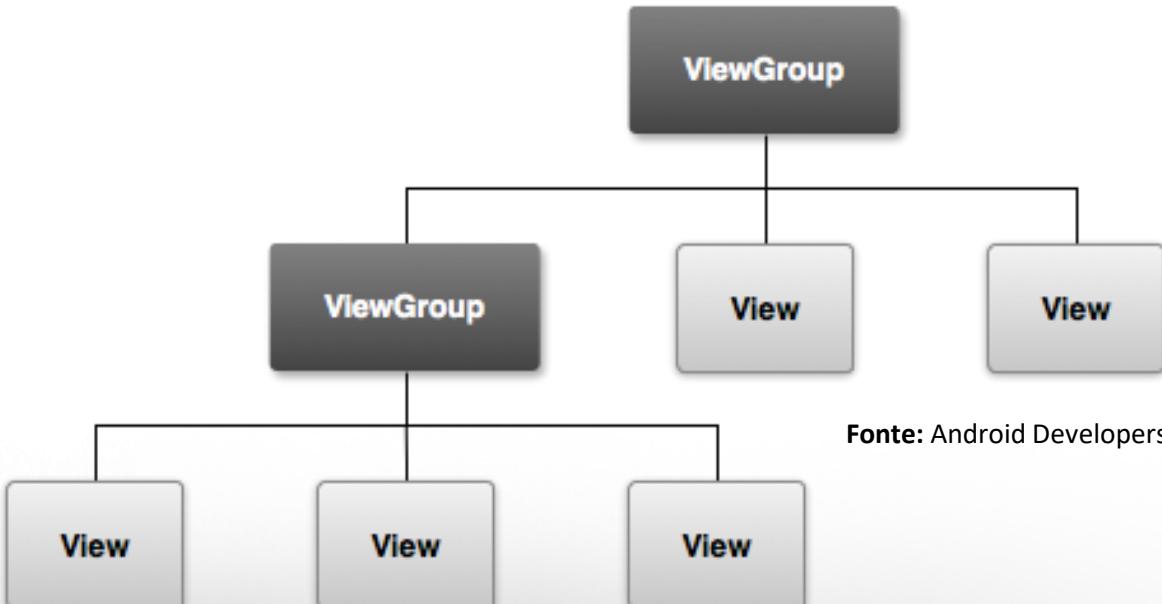
CICLO DE VIDA DE UMA ACTIVITY: finalizando uma activity

- Para finalizarmos uma activity podemos utilizar os métodos:
 - **finish():** finaliza e fecha uma activity.
 - **finishActivity():** Força o fechamento de uma activity previamente iniciada com o método **startActivityForResult(Intent, int)**.



VIEWS E VIEWGROUPS

- A interface com o usuário em uma aplicação Android é construída utilizando objetos **View** e **ViewGroup**;
- Basicamente a estrutura da aplicação é uma composição hierárquica de View e ViewGroup.

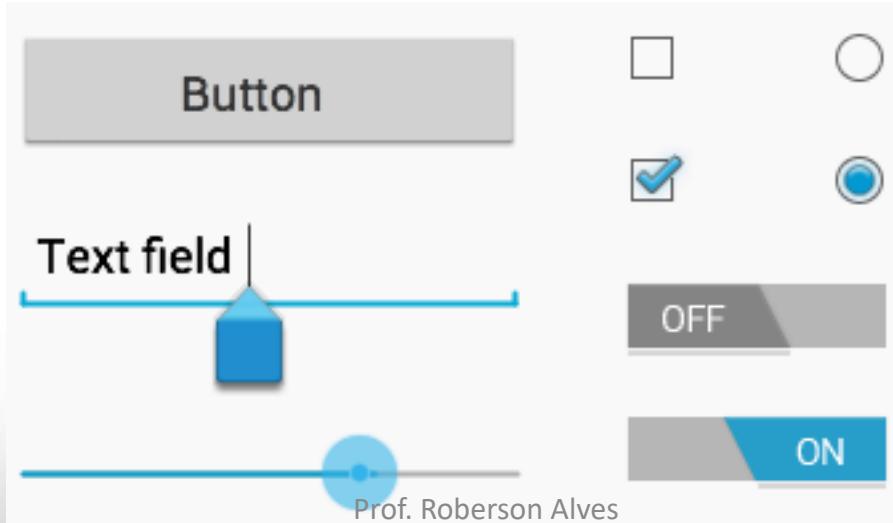


Fonte: Android Developers(2012b).



VIEW

- Todos os componentes que permitem interação do usuário com a tela são representados por subclasses de **android.view.View**
- Representam os componentes gráficos (os chamados widgets) como **TextView**, **Button**, **TextEdit**, **RadioButton**, **Checkbox**, **ToggleButton**, **ImageButton**, **Spinner**, etc;



VIEWGROUPS

- A classe **android.view.ViewGroup**, que representa um container de Views e também ViewGroups;
- Ela é a classe base para layouts, como **LinearLayout**, **FrameLayout**, **AbsoluteLayout**, **RelativeLayout**, **TableLayout**, entre outros;
- O fato de um ViewGroup também ser composto por um ou mais ViewGroups é o fator que permite que layouts complexos (layouts aninhados) sejam desenvolvidos;
- Quando utilizamos o método **setContentView()**, passamos uma referência ao nó raiz da hierarquia.



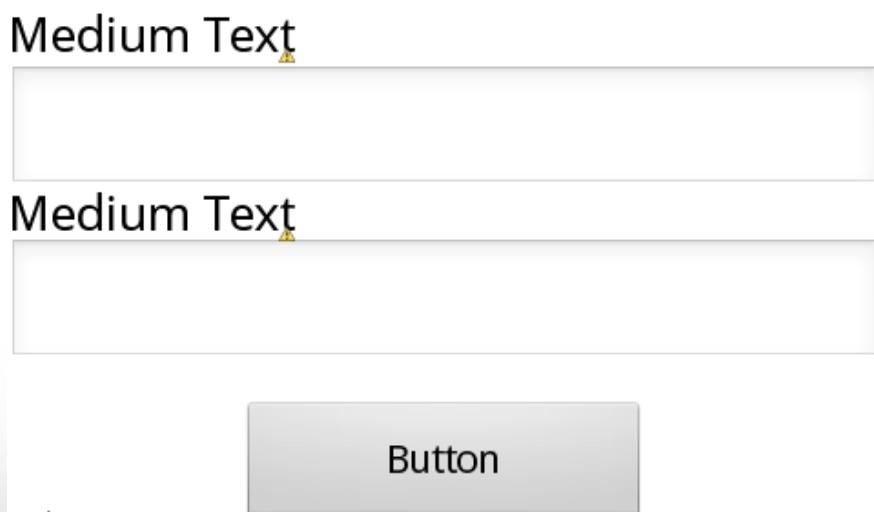
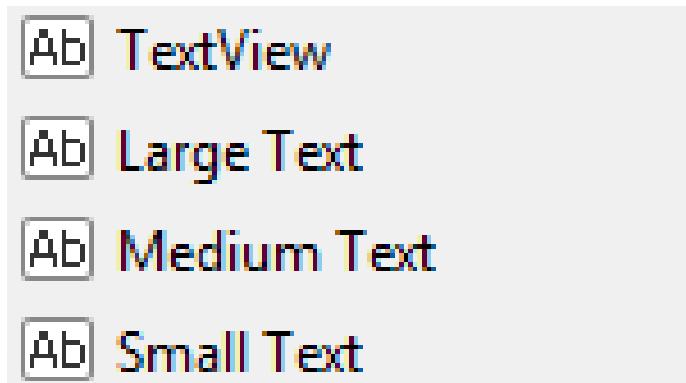
INPUT CONTROLS

- O Android tem disponíveis uma série de Views, ou componentes, para interação do usuário, incluindo: rótulos, botões, campos de entrada de dados, listas de valores, entre outros;
- Cada um destes componentes possui suas próprias propriedades, várias herdadas de View e outras específicas;
- Os componentes também possuem um conjunto de eventos aos quais podem responder, como cliques, entradas de texto ou mesmo toques na



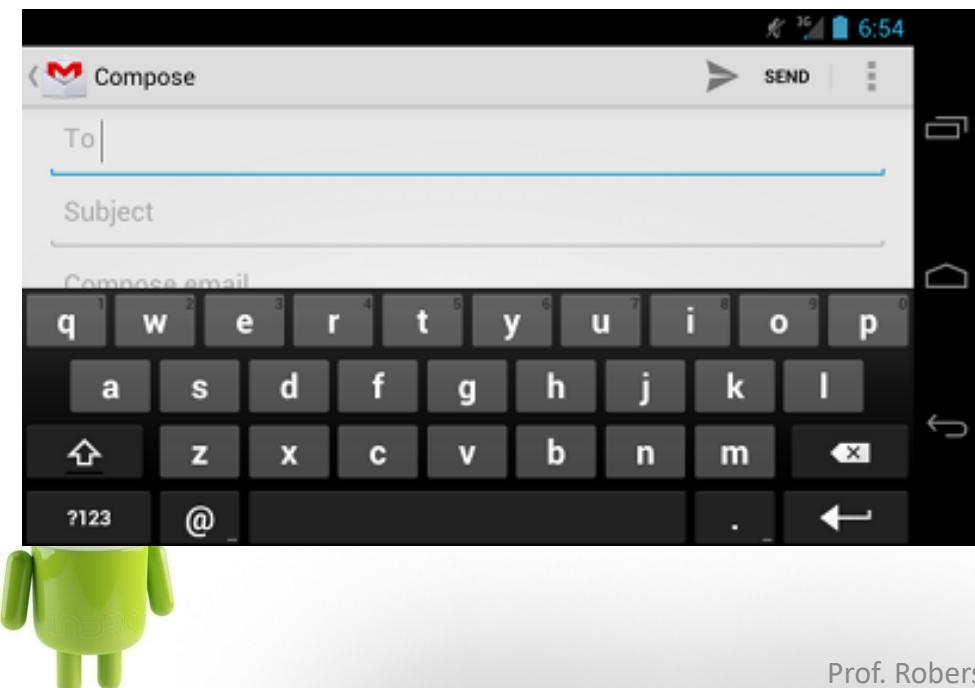
INPUT CONTROLS - TEXTVIEW

- Permite a exibição de um texto na tela;
- Pode ser utilizado para identificar/rotular campos de entrada de texto;
- Objetos **TextView** por padrão não permitem entradas de texto por parte do usuário;
- **EditText** é uma classe especializada de **TextView** que permitirá ao usuário entrar seus textos.



INPUT CONTROLS - EDITTEXT

- Permite que o usuário entre com textos na aplicação;
- Pode ser de uma linha ou multilinha;
- Permite diferentes tipos de entradas;



Text Fields		
abc	Firstname Lastname
1...2...3	user@domain	(555) 0100
Address	Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor	12:00am
1/1/2011	42	-42
42.0		

INPUT CONTROLS - BUTTON

- Consiste de um texto ou ícone(ou ambos) que comunica uma ação do usuário quando pressionado;



```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/button_text"  
    ... />
```

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/button_text"  
    android:drawableLeft="@drawable/button_icon"  
    ... />
```

```
<ImageButton  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/button_icon"  
    ... />
```



INPUT CONTROLS - TOGGLEBUTTON

- Permite modificar um determinado estado através do botão. Ex.: ligado/desligado, on/off, ativo/inativo;



Toggle buttons

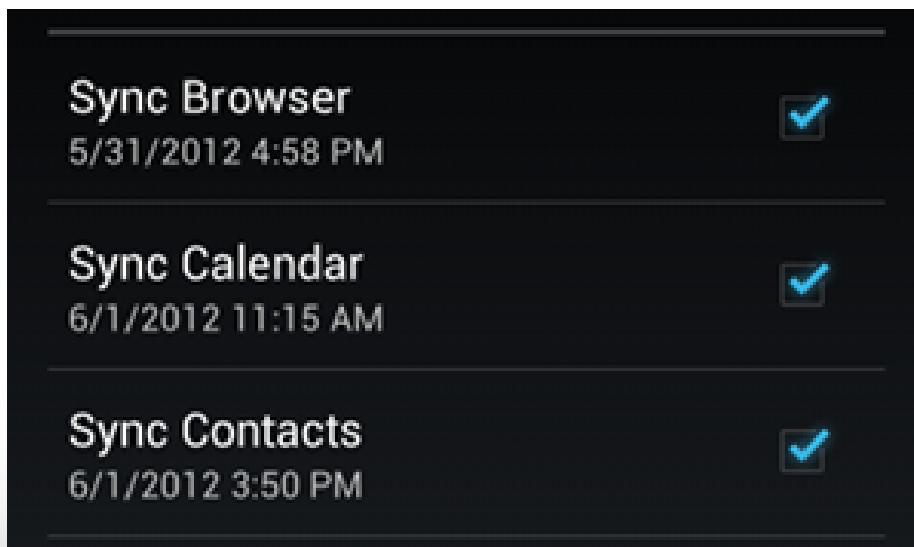


Switches (in Android 4.0+)



INPUT CONTROLS – CHECKBOX

- Permite ao usuário selecionar/marcar uma ou mais opções;
- Normalmente as opções para checagem são exibidas em uma lista vertical.



INPUT CONTROLS - RADIOBUTTON

- Similar ao CheckBox, no entanto, o usuário pode selecionar somente uma opção;
- Pode ser adicionado na tela de forma individual **RadioButton** ou em grupo **RadioGroup**.

ATTENDING?

Yes Maybe No



MÉTODO FINDVIEWBYID

- Método utilizado para recuperar a instância de uma view(widget);
- A recuperação é feita pelo ID gerado no arquivo R.java;
- Utiliza-se o formato:
R.id.algumWidget
- Exemplo: **findViewById(R.id.btnSair)**



EVENT LISTENERS – ONCLICKLISTENER

- No Android, existe mais de uma maneira de interceptar eventos de interação de usuário em sua aplicação. Quando consideramos eventos dentro de sua interface de usuário, a abordagem deve ser a de capturar os eventos do objeto de View específica na qual o usuário está interagindo;
- **Event Listeners:** Um event listener é uma interface da classe View que contém um método simples de callback. Esse método pode ser chamado pelo framework Android quando a View sofre uma interação do usuário com um item da interface.



EVENT LISTENERS – ONCLICKLISTENER

- **OnClickListener:** Interface implementada por **android.view.View;**
- Permite capturar eventos de clique por exemplo em um botão;
- Exemplo:

```
final Button btnSomar = (Button)findViewById(R.id.btnSoma);
btnSomar.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        //implementação da resposta ao usuário
    }
});
```

EXIBINDO MENSAGENS COM TOAST

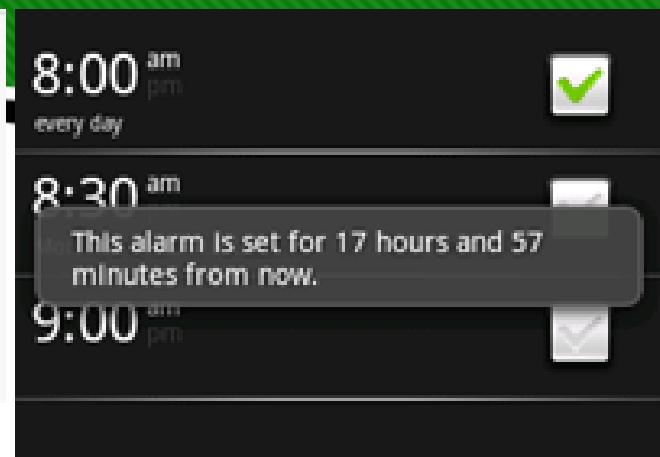
- Utilizando a classe **Toast** podemos criar notificações ou mensagens rápidas para serem exibidas ao usuário;
- O tamanho da janela gerada é de acordo com o texto que desejamos exibir;
- Possui as constantes: **LENGTH_SHORT** e **LENGTH_LONG** para determinar o tempo de exibição;
- A mensagem aparece e desaparece automaticamente;
- A mensagem de **Toast** pode ser customizada utilizando-se um layout XML.



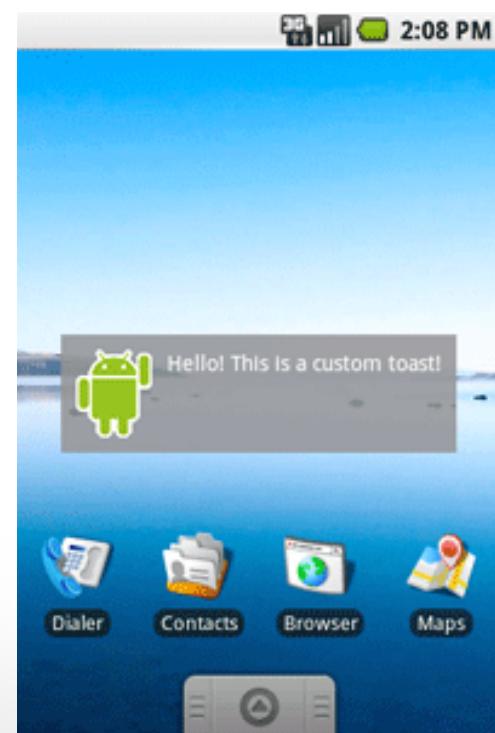
EXIBINDO MENSAGENS COM TOAST

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```



```
Toast.makeText(context, text, duration).show();
```



```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```



REFERÊNCIAS

- **Android Developers. Publishing Overview.** 2012. Disponível em: <http://developer.android.com/guide/publishing/publishing_overview.html>. Acesso em: 15 jun. 2012
- **Eu Android.** 2011. Disponível: <<http://www.euandroid.com.br/iniciante-android/2011/05/maquina-virtual-dalvik/>>. Acesso em: 03 fev. 2012
- **LECHETA, Ricardo R.** Google android: aprenda a criar aplicações para dispositivos móveis com o android SDK. 2. ed., rev. e ampl. São Paulo: Novatec, 2010. 608 p. ISBN 9788575222447.
- **What is Android.** 2012. Disponível em: <<http://developer.android.com/guide/basics/what-is-android.html>> . Acesso em: 03 fev. 2012

