FIAP

DESENVOLVIMENTO MOBILE - ANDROID

ACTIVITY

PROF. EDSON A. SENSATO
profedsonsensato@fiap.com.br

Versão 1– 2016

INTRODUÇÃO

A **Activity** é um dos componentes da plataforma Android

Ela representa uma tela onde componentes de interface com o usuário (views) podem ser dispostas

Aplicações podem ter mais de uma Activity, mas somente uma delas deve ser a principal

O desenho da interface pode ser especificado em um arquivo xml na pasta **layout**

Ela pode acionar outras activities tanto da mesma aplicação quanto de outras aplicações

Deve ser declarada no **AndroidManifest.xml**

Quando você disca um número, envia e-mail, tira uma foto está usando uma Activity no seu smartphone!

A CLASSE ACTIVITY

```
package app.primeiro.activityapp;

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

public class MainActivity extends Activity {

}
```

Para que uma classe seja reconhecida como uma Activity ela deve estender (especializar) a classe **Activity**

COMPATIBILIDADE

Com a evolução da plataforma, novos recursos foram agregados à Activity

Isso poderia gerar um problema de compatibilidade com as verões anteriores da API

Sendo assim, é uma boa prática extender a classe **AppCompatActivity** do que Activity diretamente

Contudo, pela hierarquia das Avtivities, temos que Activity é a classe original:

Activity → BaseFragmentActivityDonut → BaseFragmentActivityHoneycomb →FragmentActivity → **AppCompatActivity**

Daqui em diante utilizaremos sempre a **AppCompatActivity**

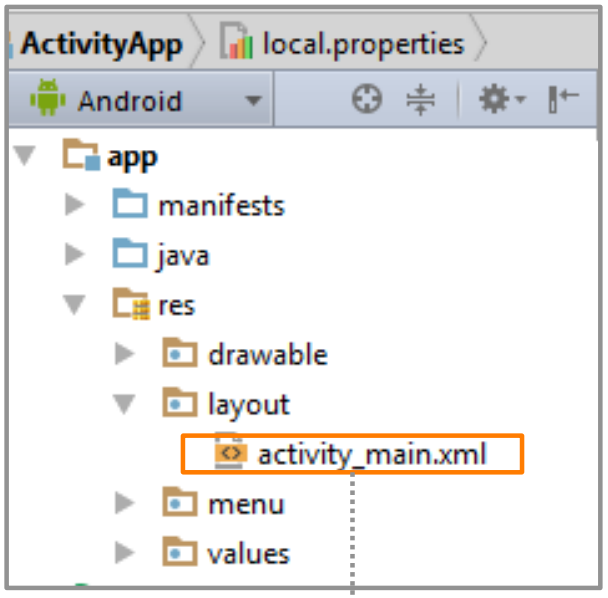
MÉTODO onCreate

```
package app.primeiro.activityapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
   }

}
```

O método **onCreate** é um dos métodos (callback) acionados pelo framework quando uma activity é criada em memória

É o local ideal para iniciar as variáveis globais da aplicação, por exemplo

ASSOCIANDO UM LAYOUT



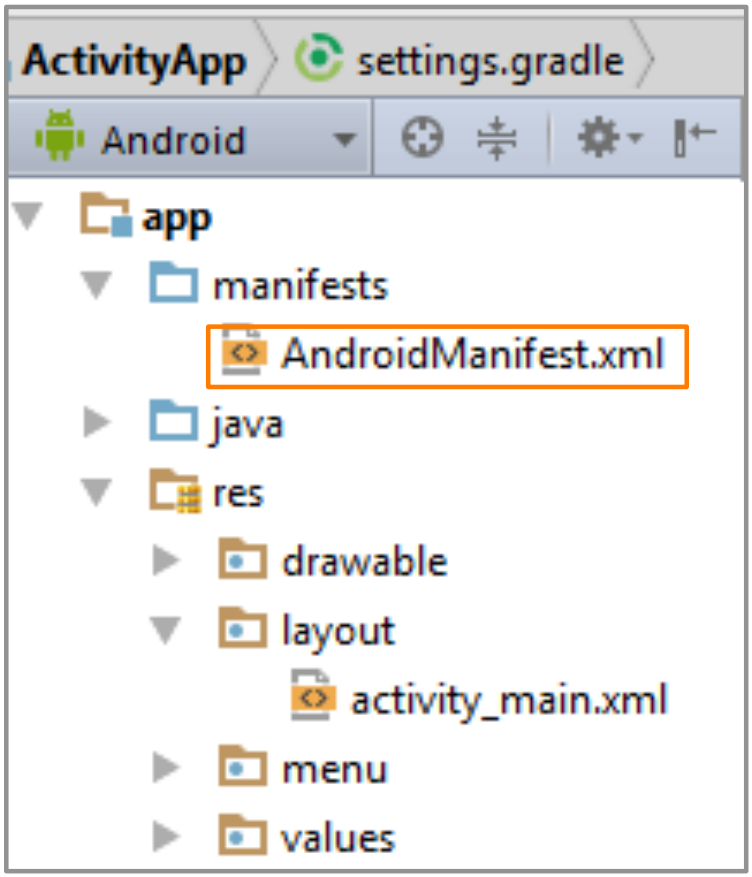

```
package app.primeiro.activityapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class Activity1 extends AppCompatActivity {

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

O método **setContentView** é o responsável por associar um arquivo de layout à Activity

Cada Activity possui um arquivo para definir o seu layout

Os layouts podem ser referenciado por **R.layout.nome_do_layout**

AndroidManifest.xml



O **AndroidManifest.xml** é um aquivo que define algumas configurações da aplicação como um todo

Nele, as Activities da aplicação devem ser declaradas

DECLARANDO A ACTIVITY

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="app.primeiro.activityapp" >

   <application android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity android:name=".Activity1"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>

   </application>

</manifest>
```

As tags **<activity></activity>** definem uma Activity dentro da aplicação

Pode-se declarar várias Activities dentro do AndroidManifest

ALGUNS ATRIBUTOS

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="app.primeiro.activityapp" >

  <application android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity android:name=".Activity1"
      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

  </application>

</manifest>
```

Classe que implementa a Activity
O "." indica que a classe está declarada no mesmo pacote do app

action MAIN → indica que esta é a primeira Activity a ser executada quando o app é acionado
category LAUNCHER → diz que a Activity deve ser exibida na área de apps do dispositivo (Launcher)

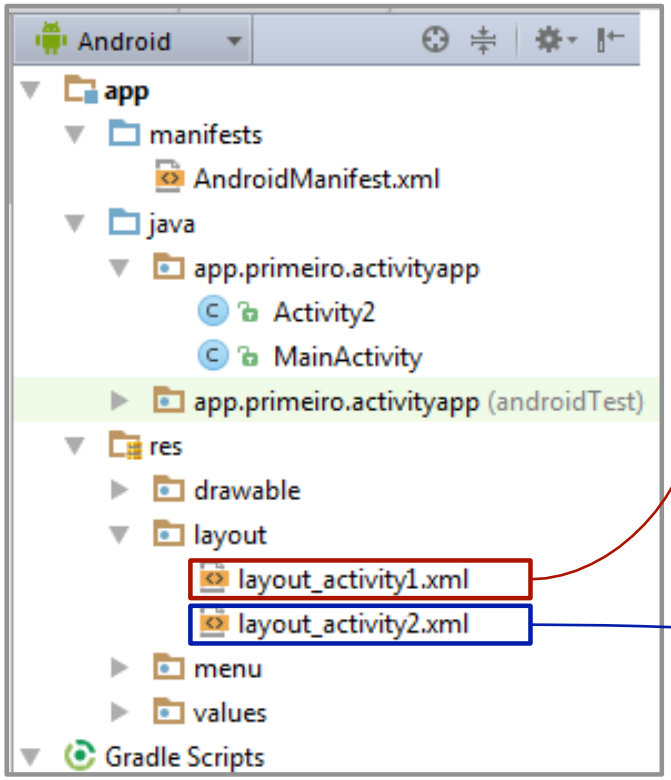
EXERCÍCIO

Criar duas activities conforme acima

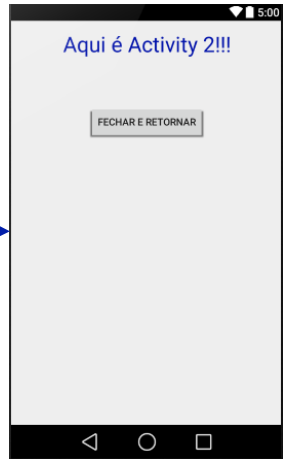
Quando o botão CHAMAR ACTIVITY 2 for acionado, a Activity 2 deve ser exibida

Ao clicar em FECHAR E RETORNAR na Activity 2 ela deve ser fechada e a Activity 1 será exibida novamente

CRIANDO A INTERFACE GRÁFICA



R.layout.layout_activity1



R.layout.layout_activity2

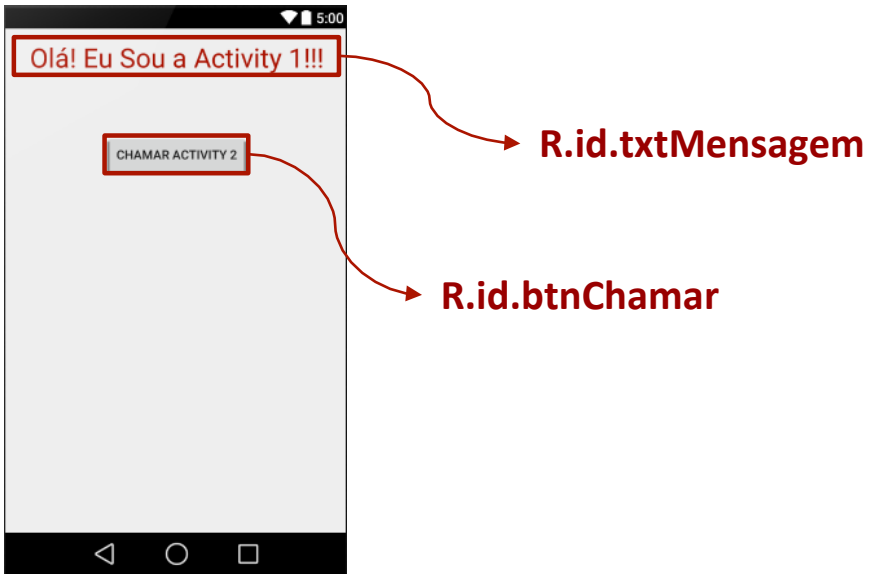
Neste app criaremos 2 arquivos dentro de layout

Cada um definirá a interface gráfica da respectiva activity

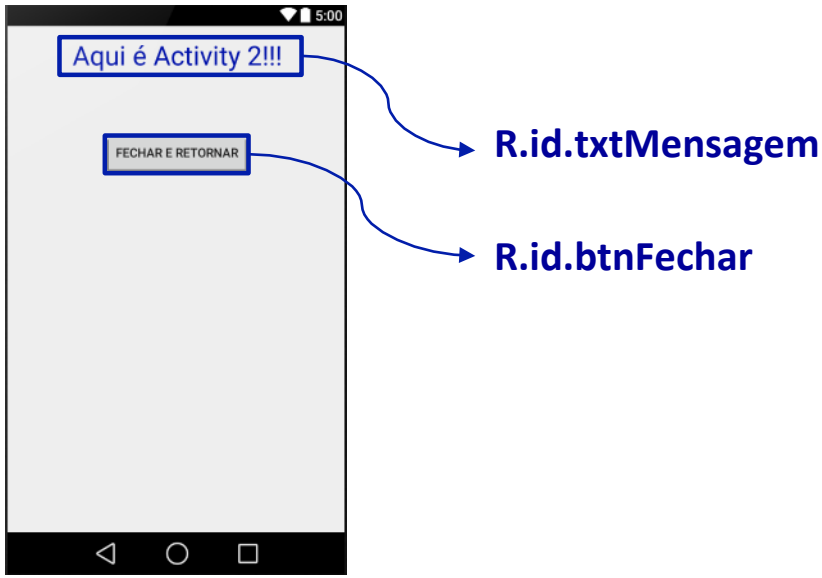
O método **setContentView** deverá ser definido apropriadamente de acordo com o R.layout associado a cada uma das activities

REFERENCIANDO A INTERFACE

layout_activity1.xml



layout_activity2.xml



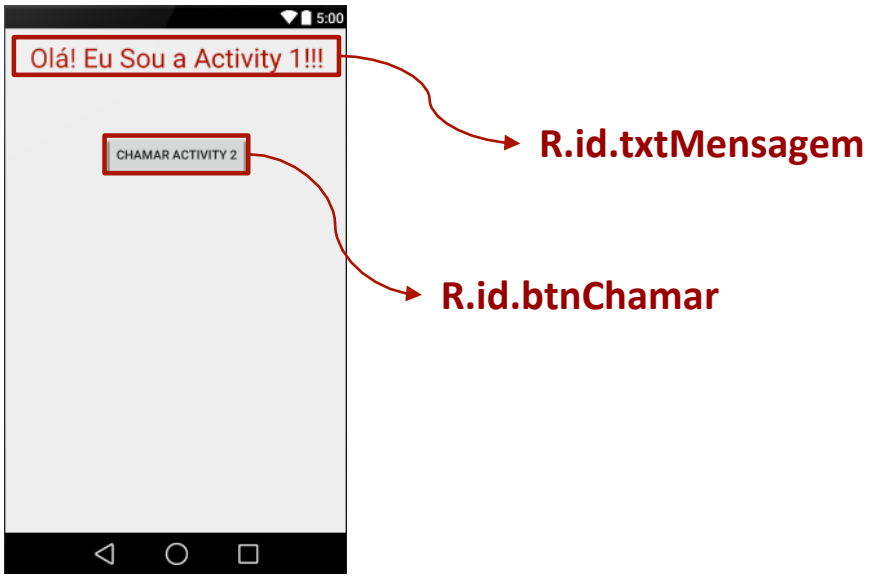
Na Activity utilizamos o método **findViewById** para referenciar elementos da interface:

Exemplo: **Button b = (Button) findViewById(R.id.btnfechar);**

Como findViewById retorna um Object é sempre necessário um cast para o elemento de interface desejado (Button, TextView, etc...)

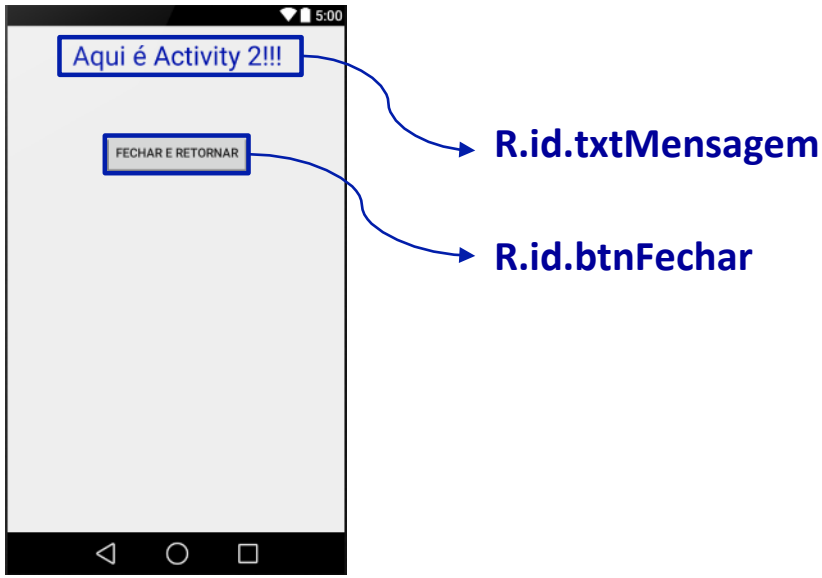
CONTEXTO DA ACTIVITY

layout_activity1.xml



ACTIVITY 1

layout_activity2.xml



ACTIVITY 2

CUIDADO!!! Como cada layout (xml) está associado a uma Activity específica, o findViewById daquela Activity somente tem acesso aos elementos de interface declarados no xml da Activity. Exemplo:

ACTIVITY 1 → Button b = (Button) findViewById (**R.id.btnFechar**); → **ERRO!!!!**
ACTIVITY 2 → Button b = (Button) findViewById (**R.id.btnChamar**); → **ERRO!!!!**

INLCUINDO ACTIVITY NO MANIFEST

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="app.primeiro.activityapp" >

   <application ... >

      <activity android:name=".Activity1" android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>

      <activity android:name=".Activity2" android:label="@string/app_name" >
         <intent-filter>
            <category android:name="android.intent.category.DEFAULT" />
         </intent-filter>
      </activity>

   </application>

</manifest>
```

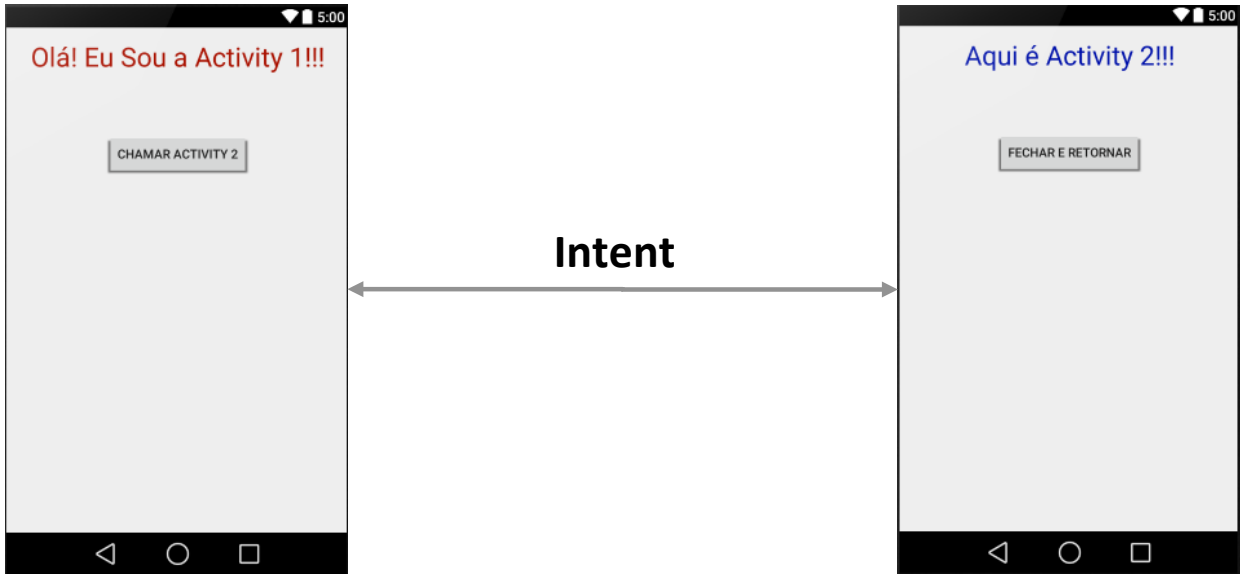
A nova activity deve ser declarada no AndroidManifest.xml
Deve ter a categoria definida como **DEFAULT**

INTENTS

Intents são componentes que fazem parte do protocolo de comunicação entre elementos da plataforma Android (Activity, Service, etc...)

Um **Intent** encapsula uma mensagem que pode ser trocada, por exemplo, entre uma Activity e outra

Os Intents podem conter parâmetros, valores que podem ser trocados entre os elementos envolvidos na comunicação



COMUNICAÇÃO ENTRE ACTIVITY

Uma Activity pode acionar outra Activity seguindo os passos abaixo:

1. Criar um Intent indicando a Activity origem e a de destino

2. Chamar a Activity por meio do método **startActivity**, passando como parâmetro o Intent criado no passo 1

Exemplo:

```
Intent toActivity2 = new Intent(this, Activity2.class);
startActivity(toActivity2);
```

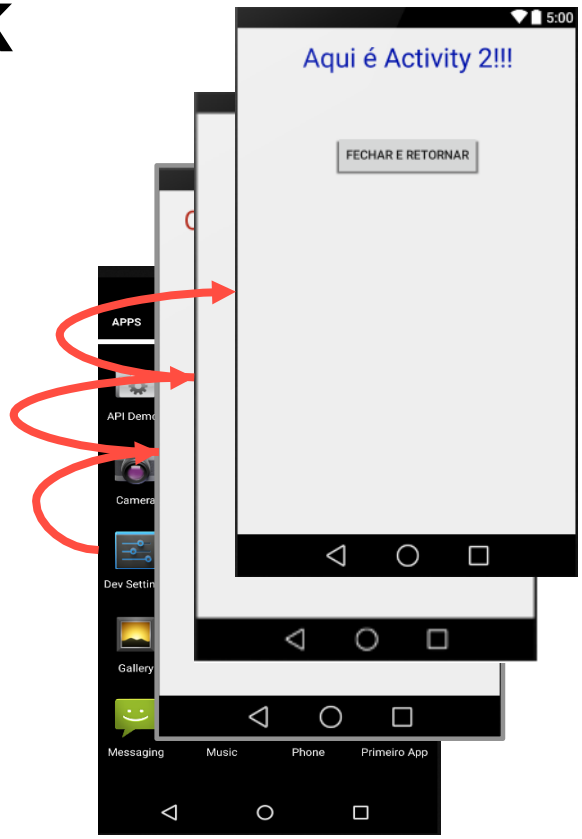
Observe que a Activity2 foi referenciada pelo nome de sua classe (**Activity2.class**)

Já a origem trata-se da própria Activity1 (que está chamando) e, portanto, declaramos como **this**

IMPLEMENTADO STARTACTIVITY

```
package app.primeiro.activityapp;

import android.support.v7.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Activity1 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_activity1);
    }

    public void chamarActivity2(View v) {

        Intent toActivity2 = new Intent(this, Activity2.class);
        startActivity(toActivity2);

    }
}
```

BACKSTACK



Quando a Activity1 aciona a Activity2, a Activity2 é executada sobre a Activity1

Caso a Activity2 acionasse uma outra Activity3, a Activity3 executaria sobre a Activity2...

As Activities vão sendo organizadas em uma pilha, denominada Back Stack

A Activity no topo da pilha está visível ao usuário final

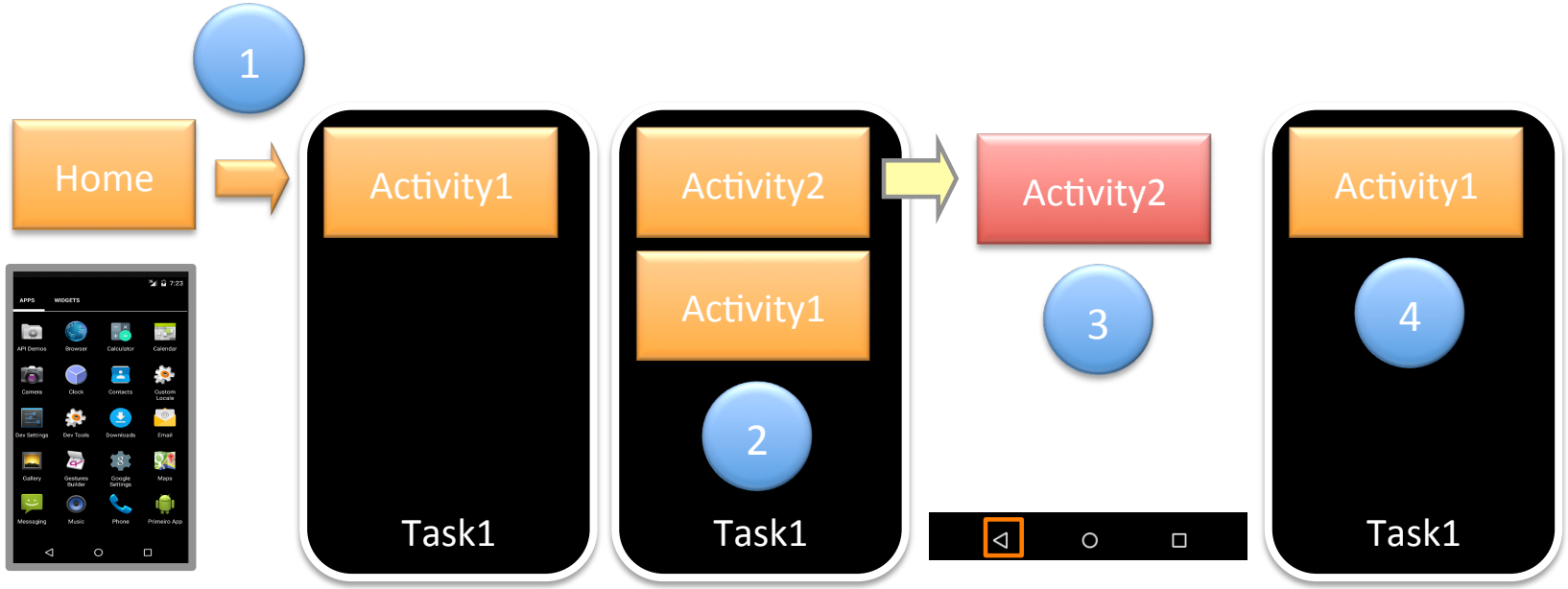
DINÂMICA DO BACKSTACK

Quando a Activity2 é finalizada (botão back), a Activity1 volta a ser visível ao usuário final

A Activity1 volta a executar

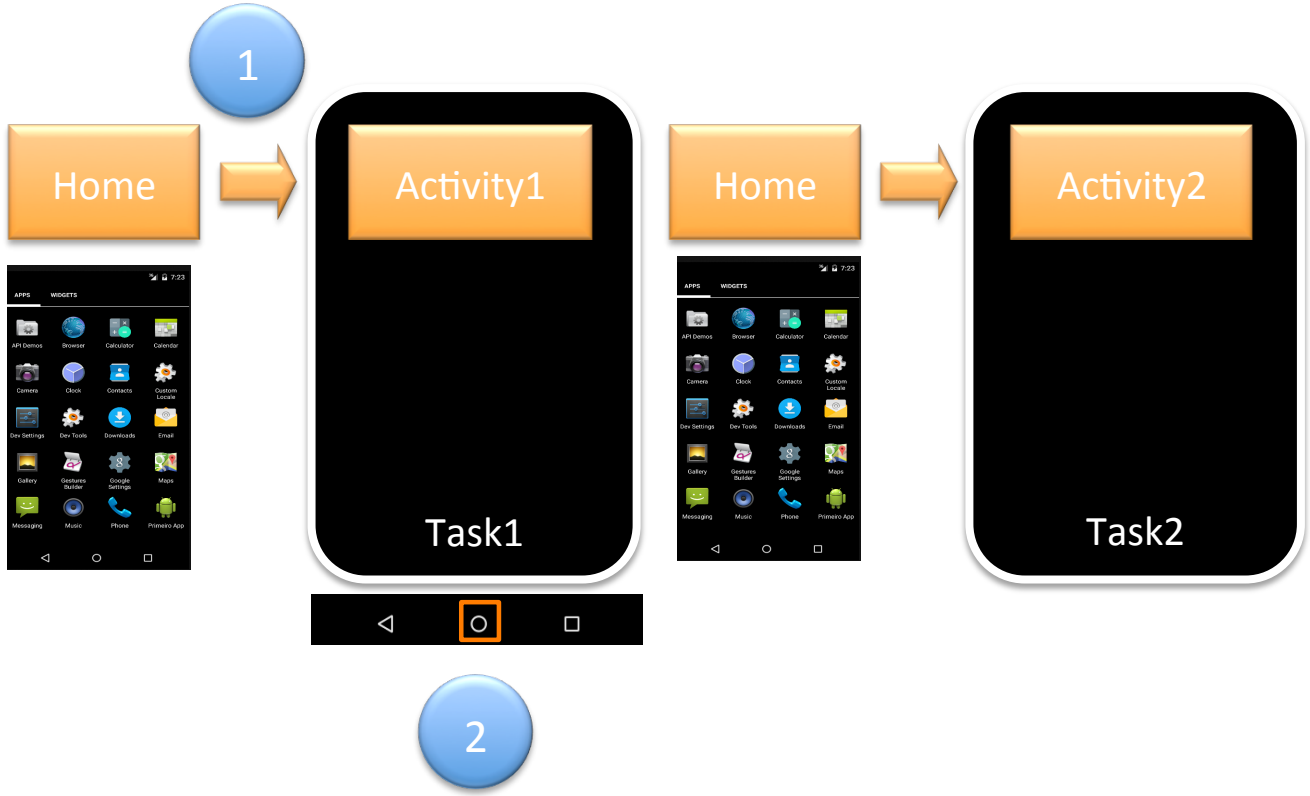
Caso a Activity1 seja fechada (botão back), então a aplicação é finalizada retornando para a Activity home

ACTIVITY TASK



1. O usuário inicia a aplicação a partir da home do dispositivo. Uma *task* é criada, a **Activity1**, que é a principal é iniciada e colocada na pilha (*back stack*).
2. A **Activity1** aciona a **Activity2** que é colocada no topo da pilha e passa a executar. Neste momento, a **Activity1** está *stopped*.
3. O usuário deseja retornar para a **Activity1**. Então ele pressiona o botão *back*, a **Activity2** é destruída e removida da pilha. A **Activity1** passa então a executar (*running*).
4. Quando o usuário encerrar a **Activity1**, como a *back stack* está vazia, então a **Task1** é finalizada e a aplicação encerrada.

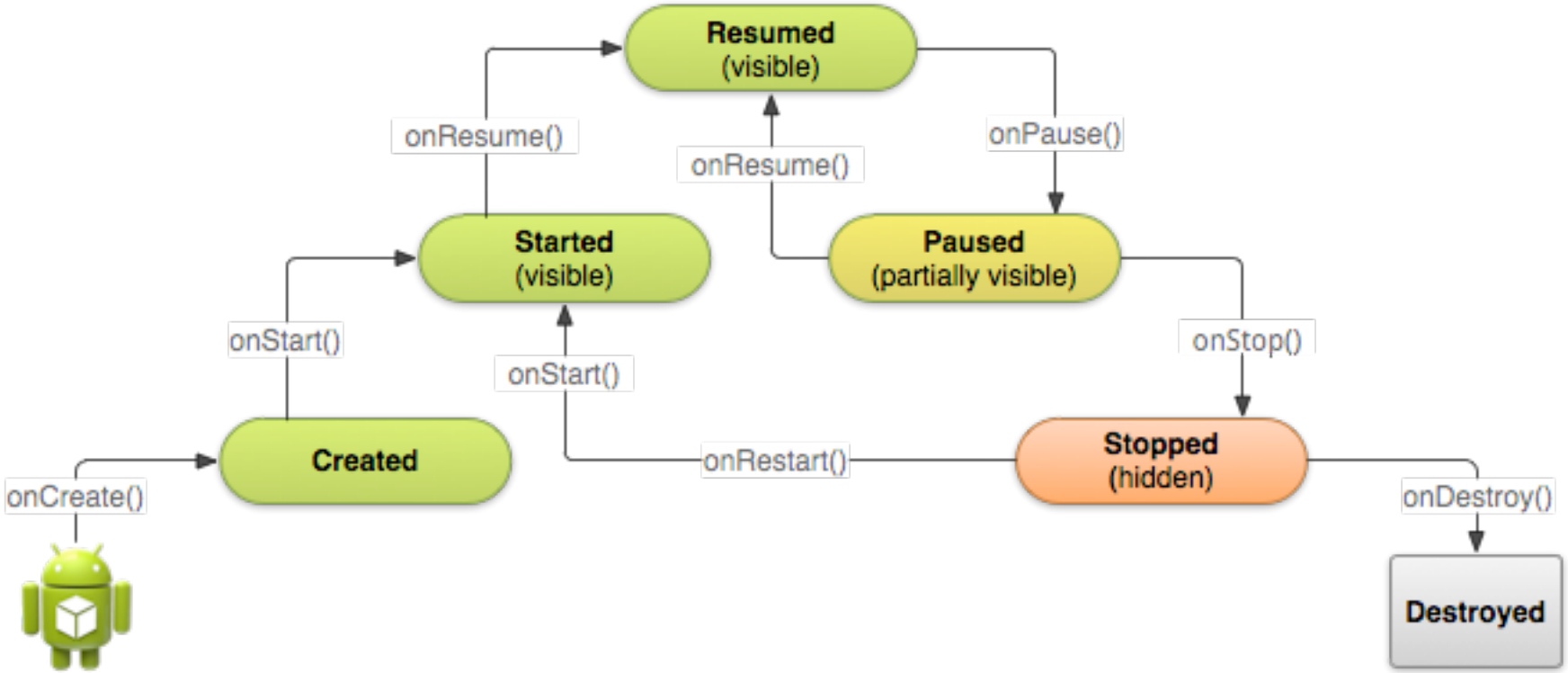
MÚLTIPLAS TASKS



1 O usuário inicia a aplicação a partir da home do dispositivo. Uma *task* é criada, a **Activity1**, que é a principal é iniciada e colocada na pilha (*back stack*).

2 O usuário retorna à Home (pelo botão *home*) sem encerrar a **Activity1** e inicia uma outra aplicação. Uma outra *task* é iniciada e a sua Activity principal (Activity2) passa a executar.

CICLO VIDA ACTIVITY



FIAP