

AULA 04

IMD0509 - DESENVOLVIMENTO PARA **DISPOSITIVOS MOVÉIS**

Prof. Emerson Alencar

emerson@imd.ufrn.br



Componentes do Android

_ X



- **Activities:** Representam uma tela única com uma interface do usuário. Por exemplo, um aplicativo de e-mail pode ter uma atividade que mostra uma lista de novos e-mails, outra atividade que compõe um e-mail e outra ainda que lê e-mails. Embora essas atividades funcionem juntas para formar uma experiência de usuário coesa no aplicativo de e-mail, elas são independentes entre si.
- **Services:** São componentes executados em segundo plano para realizar operações de execução longa ou para realizar trabalho para processos remotos. Eles não apresentam uma interface do usuário. Por exemplo, um serviço pode tocar música em segundo plano enquanto o usuário está em um aplicativo diferente.

- **Content Providers:** Gerenciam um conjunto compartilhado de dados do aplicativo. É possível armazenar os dados no sistema de arquivos, em um banco de dados SQLite ou em qualquer local de armazenamento persistente que o aplicativo possa acessar. Por meio do provedor de conteúdo, outros aplicativos podem consultar ou até modificar os dados (se o provedor de conteúdo permitir). Por exemplo, o sistema Android oferece um provedor de conteúdo que gerencia os dados de contato do usuário.
- **Broadcast Receivers:** São componentes que respondem a anúncios de transmissão por todo o sistema. Muitas transmissões se originam do sistema — por exemplo, uma transmissão que anuncia que uma tela foi desligada, a bateria está baixa ou uma tela foi capturada.

AndroidManifest.xml

_ X

- Arquivo que declara os componentes do aplicativo e outras informações
- Todo aplicativo possui um arquivo AndroidManifest.xml
- formato do arquivo é XML

AndroidManifest.xml

_ X

```
--
20 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
21     package="com.example.android.donebar"
22     android:versionCode="1"
23     android:versionName="1.0">
24
25     <!-- Min/target SDK versions (<uses-sdk>) managed by build.gradle -->
26
27     <application android:label="@string/app_name"
28         android:icon="@drawable/ic_launcher"
29         android:theme="@style/Theme.Sample"
30         android:allowBackup="true">
31
32         <activity android:name=".MainActivity"
33             android:label="@string/app_name">
34             <intent-filter>
35                 <action android:name="android.intent.action.MAIN" />
36                 <category android:name="android.intent.category.LAUNCHER" />
37             </intent-filter>
38         </activity>
39
40         <activity android:name=".DoneBarActivity"
41             android:parentActivityName=".MainActivity" />
42
43         <activity android:name=".DoneButtonActivity"
44             android:parentActivityName=".MainActivity" />
45
46     </application>
47
48 </manifest>
```

- Nomear o pacote Java para o aplicativo. O nome do pacote serve como identificador exclusivo para o aplicativo;
- Descrever os componentes do aplicativo, que abrangem atividades, serviços, receptores de transmissão e provedores de conteúdo que compõem o aplicativo. Ele também nomeia a classe que implementa cada um dos componentes e publica suas capacidades, como as mensagens Intent que podem lidar;
- Declarar as permissões que o aplicativo deve ter para acessar partes protegidas da API e interagir com outros aplicativos. Ele também declara as permissões que outros devem ter para interagir com os componentes do aplicativo;
- Declarar Android API mínima que o aplicativo exige;
- Listar as bibliotecas às quais o aplicativo deve se vincular.

Estrutura do Arquivo

<https://developer.android.com/guide/topics/manifest/manifest-intro.html>

Activities



- Uma Activity representa uma tela da aplicação
- Representada por uma classe que herda de `android.app.Activity`

```
1 package br.com.emerson.myapplication;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.widget.TextView;
6
7
8 public class MainActivity extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13
14         TextView view = new TextView(context: this);
15         view.setText("texto na tela");
16         setContentView(view);
17
18
19     }
20 }
21
```



A classe **Context**

_ X

- Representa o contexto de execução da aplicação do Android
- Possui muitos métodos que são comumente chamados em aplicações
- Um objeto desta classe está normalmente disponível em vários lugares da aplicação
 - A classe **Activity** herda de **Context**
 - O *Context* é fornecido como parâmetro

- O Android tem basicamente dois tipos de contextos:
 - **Contexto da activity:**

Cada activity da aplicação tem um contexto
 - **Contexto da aplicação:**

É único para a aplicação toda
- **Importante:** É preciso tomar cuidado na escolha do contexto para não causar memory leaks na aplicação (vazamento de memória)

A classe Context

_ X

```
8  public class MainActivity extends Activity {  
9  
10     @Override  
11     protected void onCreate(Bundle savedInstanceState) {  
12         super.onCreate(savedInstanceState);  
13         setContentView(R.layout.activity_main);  
14  
15         TextView view = new TextView(context: this);  
16         view.setText("Texto na Tela");  
17         setContentView(view);  
18     }  
19 }  
20 }
```

Contexto da Activity: Essa viu sozinha não tem sentido sem Activity

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    MyManager m = MyManager.getInstance(getApplicationContext());  
}
```

Contexto da Aplicação: Esse componente não tem relação com a Activity

Activities e Views



- **Views** são os componentes que se agrupam para montar a interface gráfica
- As **activities** e as **views** têm uma relação próxima
 - **Activities** representam a tela
 - **Views** representam o que é renderizado na tela

Activities e Views


_ X

- O método **setContentView()** da activity é utilizado para determinar qual view será renderizada

```
1 package br.com.emerson.myapplication;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.widget.TextView;
6
7
8 public class MainActivity extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13
14         TextView view = new TextView(context: this);
15         view.setText("texto na tela");
16         setContentView(view);
17
18     }
19 }
20
21
```

Cria uma caixa de texto

Renderiza a view

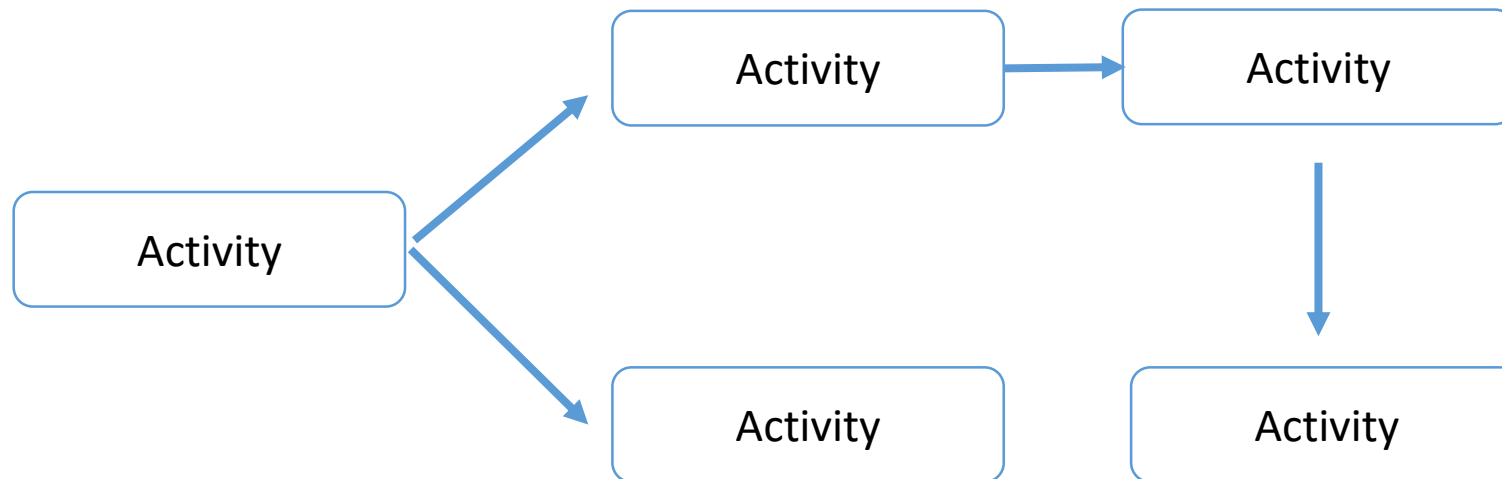


The image shows a smartphone screen displaying the app's interface. The status bar at the top shows the time as 2:20 and various icons. The app's title bar reads "IMD - UFRN". Below the title bar, the text "texto na tela" is displayed on a white background.

Invocando Activities

_ X

- Uma aplicação normalmente é composta por um conjunto de activities

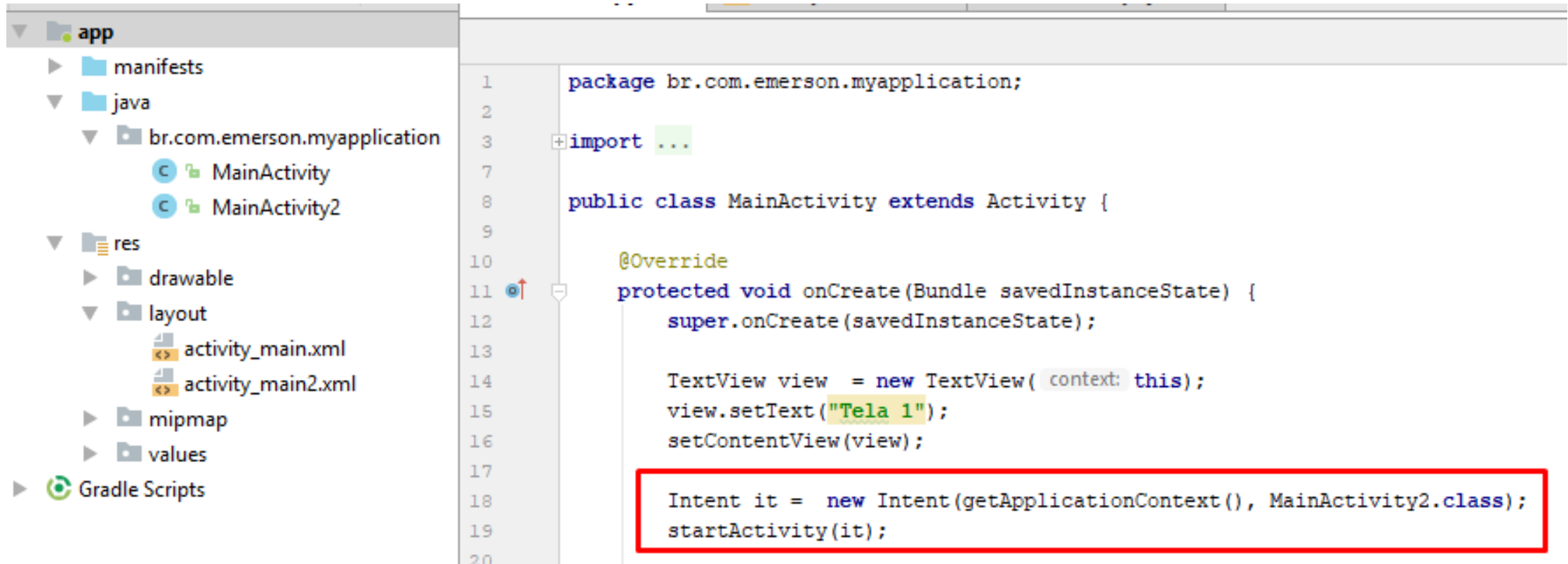


A comunicação entre activities é feita através de uma intent

Invocando Activities

_ X

- Um objeto **Intent** é usada para disparar uma nova activity



The screenshot displays the Android Studio interface. On the left, the 'app' module is expanded, showing the following structure:

- manifests
- java
 - br.com.emerson.myapplication
 - MainActivity
 - MainActivity2
- res
 - drawable
 - layout
 - activity_main.xml
 - activity_main2.xml
 - mipmap
 - values
- Gradle Scripts

The main editor shows the Java code for `MainActivity`:

```
1 package br.com.emerson.myapplication;
2
3 import ...
4
5
6
7
8 public class MainActivity extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13
14         TextView view = new TextView(context: this);
15         view.setText("Tela 1");
16         setContentView(view);
17
18         Intent it = new Intent(getApplicationContext(), MainActivity2.class);
19         startActivity(it);
20     }
21 }
```

The code for creating and starting the `Intent` is highlighted with a red rectangle:

```
Intent it = new Intent(getApplicationContext(), MainActivity2.class);
startActivity(it);
```


Retornando Informações

_ X

- O método **startActivity()** chama outra activity
- Se esta nova activity for finalizada, a activity que fez a chamada não recebe nenhuma informação
- Para notificar a activity chamadora, é possível usar **startActivityForResult()**
- A nova activity carregada é considerada uma **sub-activity**, já que fica atrelada à activity que a chamou

Retornando Informações

_ X

```
8  public class MainActivity extends Activity {  
9  
10     @Override  
11     protected void onCreate(Bundle savedInstanceState) {  
12         super.onCreate(savedInstanceState);  
13  
14         Intent it = new Intent(getApplicationContext(), MainActivity2.class);  
15         startActivityForResult(it, requestCode: 1);  
16     }  
17  
18     @Override  
19     protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
20         super.onActivityResult(requestCode, resultCode, data);  
21  
22         //Aqui tratamos as informações de acordo o requestCode  
23     }  
24 }
```

Informa um código de identificação

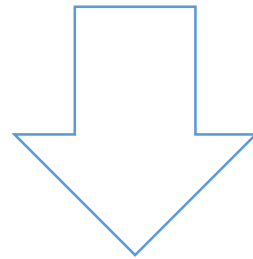
Sobrescreve o método onActivityResult, que é chamado quando a sub-activity é finalizada

Retornando Informações

_ X

```
setResult(10);  
finish();
```

Activity invocada (sub-activity)
Define um código de resposta e
finaliza a activity



onActivityResult(1,10,null)

Método na primeira activity

Passagem de Parâmetros

_ X

- É possível passar parâmetros de uma activity para outra usando um **Bundle (embrulho, pacote)**.

```
8  <> public class MainActivity extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13
14         Intent it = new Intent(getApplicationContext(), MainActivity2.class);
15
16         Bundle params = new Bundle();
17         params.putString("mensagem", "Passando uma mensagem");
18
19         it.putExtras(params);
20
21         startActivity(it);
22     }
23 }
24 }
```

Passagem de Parâmetros

_ X

- A leitura do parâmetro é feita a partir da recuperação da intent que disparou a activity

```
public class MainActivity2 extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        Intent it = getIntent();  
        Bundle params = it.getExtras();  
        String mensagem = params.getString( key: "mensagem");  
    }  
}
```

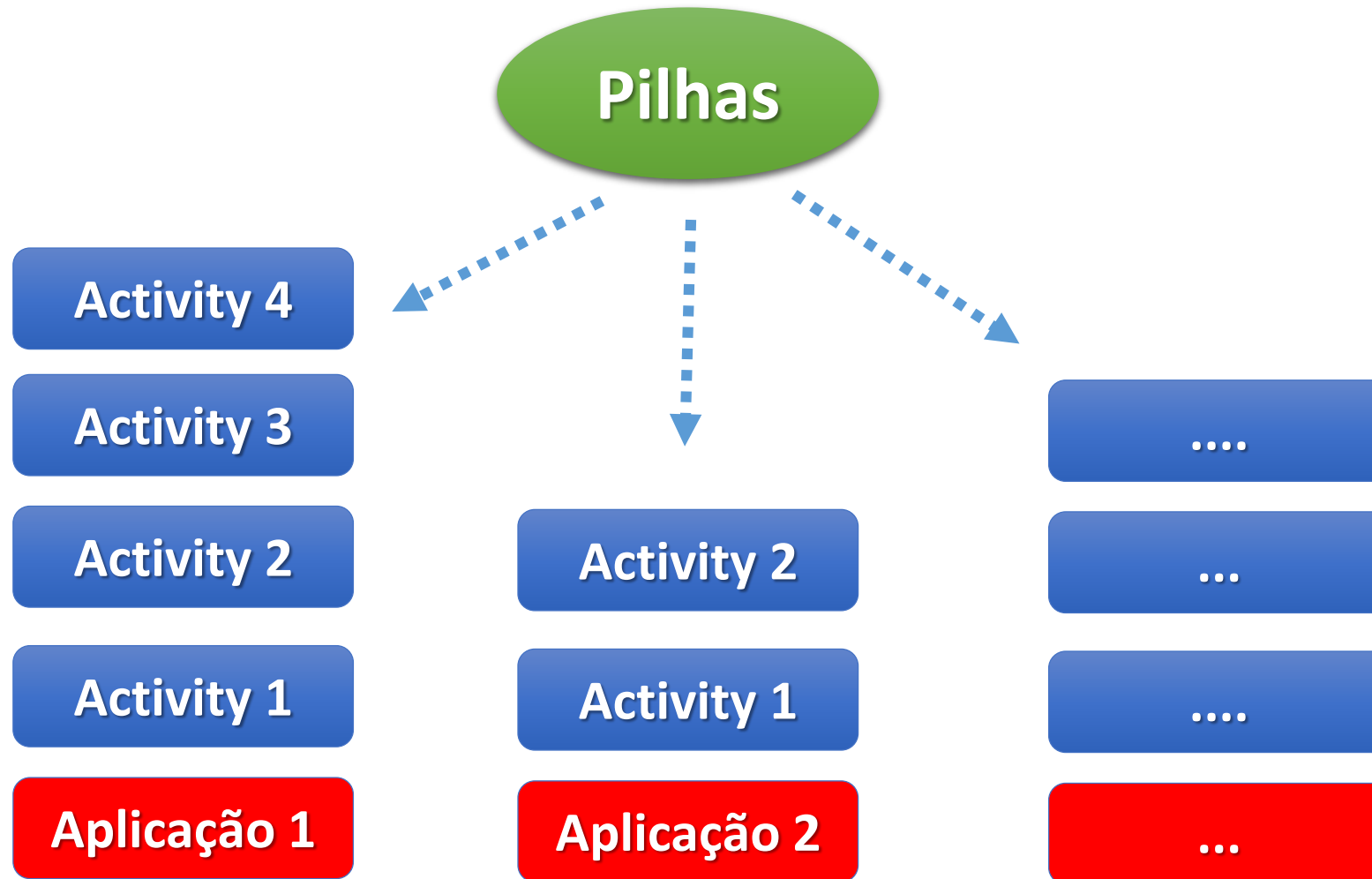
As activities na Memória



- As activities que fazem parte da sua aplicação são organizadas numa **pilha**
- A activity que está no **topo da pilha** é a activity mostrada na **tela**
- Toda vez que uma nova activity é invocada, ela passa a ficar no topo da pilha
- Quando o **botão voltar** é pressionado, a activity atual é **removida da pilha** e dá espaço para a activity que ocupa o topo da pilha ficar ativa

A Pilha de Activities

_ X



Aplicações executando

Acessando o Android Studio



The background is a solid blue color. It is decorated with a complex, abstract geometric pattern. This pattern consists of numerous white dots of varying sizes, some of which are connected by thin white lines to form a network of triangles and other polygons. Scattered throughout the composition are several yellow geometric elements: a circle, a diamond, and several plus signs. Some of the white dots are highlighted with a yellow fill. The overall aesthetic is modern and mathematical.

Perguntas?