Desenvolvimento de Apps para Mobile

Fabrício Tonetto Londero

Aula 05 - Recursos

Finalizar Activity

- Em uma tela secundária, ao utilizar o método `startActivity(Intent it)` para retornar a tela anterior pode não ser a melhor ideia:
  - Na verdade não se está retornando para a tela anterior, e sim criando uma nova instância da mesma.

- Lembre-se: As **Activities/telas** formam uma pilha, sendo que a **Activity** do topo da pilha é a que interage com o usuário naquele momento.

- Portanto, o ideal é utilizar o método `finish()`, que finaliza a Activity do topo e retornar de fato a Activity anterior.

Finalizar Activity

- Em uma tela secundária, ao invés de utilizar isso para retornar:

```
it = new Intent(this, ActivityAnterior.class);
...
public void onClick(View v) {
      startActivity(it);
}
```

- Utilizar isso:

```
public void onClick(View v) {
      finish();
}
```

Organizando os recursos

- A partir de agora, aprenderemos como:
  - ➢ Exteriorizar os recursos da nossa aplicação;
  - ➢ Fornecer alternativas de recursos para diferentes configurações de dispositivos;
  - ➢ Como nomear os diretórios para que a adaptação a diferentes configurações ocorra.
- A partir do momento que exeriorizamos um recursos, poderemos acessá-lo através da classe **`R.java`**.

Agrupando os recursos

- Cada tipo de recurso deverá ser colocado em um diretório específico da pasta res/, como por exemplo:

```
MeuProjeto/
  src/
    MinhaActivity
  res/
    drawable/
      icone.png
    layout/
      principal.xml
      menu.xml
    values/
      strings.xml
```

A pasta **/res** contém todos os recursos da aplicação, separados em subdiretórios: um recurso de imagem na pasta **drawable/**, dois layouts na pasta **layout/**, e um arquivo de recursos de textos na pasta **values/**.

Agrupando os recursos

- Os seguintes diretórios de recursos são suportados dentro da pasta `/res`:

Diretório	Tipo de recurso
`animator/`	Arquivos XML que definem propriedades de animações
`anim/`	Arquivos XML que definem animações de interpolação
`color/`	Arquivos XML que definem listas de estados de cores;
`drawable/`	Arquivos bitmap (.png, .jpg, .gif, etc) ou arquivos XML correspondentes a outros subtipos de recursos drawable: • Listas de estados • Formas/Shapes • Drawables de animação • Outros drawables
`menu/`	Arquivos XML que definem menus da aplicação
`values/`	Arquivos XML que contém valores simples, tais como strings, inteiros e cores. Diferente das outras pastas, os arquivos que estão dentro da pasta values/ descrevem múltiplos recursos. Veremos mais detalhes sobre isso a seguir.

Recursos da pasta `values/`

- A pasta values/ possibilita a definição de múltiplos recursos em um único arquivo XML, tais como:

  - Textos/mensagens da aplicação
  - Cores
  - Listas
  - Dimensões
  - Estilos

Recursos da pasta `values/`

- Cada arquivo dentro da pasta **`values/`** apresenta a seguinte estrutura básica:

```
<?xml version="1.0" encoding="utf-8"?>

<resources>

</resources>
```

- Cada nó filho de **`<resources>`** define um recurso único:
  - Por exemplo, o elemento **`<string>`** cria um recurso de texto, e o elemento **`<color>`** cria um recurso de cor.

Recursos da pasta `values/`

- Os arquivos da pasta values/ que irão conter os recursos poderão ser nomeados de qualquer forma;

- No entanto, de forma a padronizar e manter a clareza do projeto, sugere-se algumas convenções para a nomeção dos arquivos de acordo com os recursos que os mesmos definirão dentro deles:

  - **`arrays.xml`** para definição de listas
  - **`colors.xml`** para definição de cores
  - **`dimens.xml`** para definição de dimensões
  - **`strings.xml`** para definição de textos
  - **`styles.xml`** para definição de estilos

Recursos de texto

- Recursos do tipo string definem textos que nossa aplicação irá utilizar, classificados em 3 tipos:
  - **String**
    - Recurso que fornece uma única string;
  - **String Array**
    - Recurso que fornece uma lista de strings;
  - **Quantity Strings (Plurals)**
    - Recursos que carregam diferentes strings para pluralização;

String

- Recurso correspondente a uma única string, que pode ser referenciado pela nossa aplicação ou por outros recursos.

- Localizção:
  - `res/values/strings.xml`
    - O nome do arquivo não importa.

- Acesso:
  - No java: `R.string.nome_do_recurso`
  - No XML: `@string/nome_do_recurso`

String

- Sintaxe:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="nome_da_string">Valor da String</string>
</resources>
```

- Em que:
  - **<string>**:elemento que define um recurso do tipo string, contendo os seguintes atributos:
    - **name**:o nome da String;

String - Exemplo

- Arquivo XML salvo na pasta **`res/values/strings.xml`**:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="texto_botao">Enviar</string>
        <string name="texto_hello">Olá!</string>
</resources>
```

String - Exemplo

- Recurso sendo referenciado por componentes no layout da aplicação:

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

<Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/texto_botao" />
```

String - Exemplo

- Recurso sendo recuperado no Java:

```
String string = getString(R.string.texto_hello);
```

String Array

- Recurso correspondente a listas de strings que podem ser referenciadas pela nossa aplicação.

- Localizção:
  - `res/values/arrays.xml`
    - O nome do arquivo não importa.

- Acesso:
  - No java: `R.array.nome_do_recurso`
  - No XML: `@array/nome_do_recurso`

String Array

- Sintaxe:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
          <string-array name="nome_da_lista">
                    <item>Item1</item>
                    <item>Item2</item>
                    <item>Item3</item>
          </string-array>
</resources>
```

- Em que:
  - **<string-array>**:elemento que define um array de strings, que írá conter um ou mais elementro do tipo **<item>** e os seguintes atributos:
    - **name**: o nome do array de strings;
    - **<item>** elemento correspondente a uma string do array de strings. O valor pode ser estático ou referenciar um outro recurso do tipo string.

String Array - Exemplo

- Arquivo XML salvo na pasta **`res/values/arrays.xml`**:

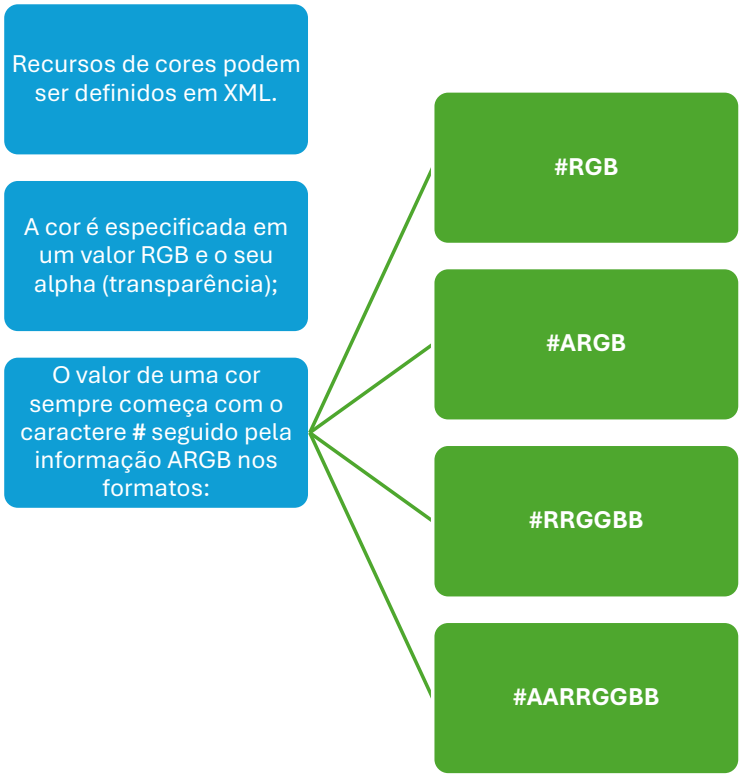
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string-array name="lista_de_planetas">
                <item>Mercúrio</item>
                <item>Venus</item>
                <item>Terra</item>
                <item>Marte</item>
        </string-array>
</resources>
```

String Array - Exemplo

- Recurso sendo recuperado no Java:

```
Resources res = getResources();
String[] planetas = res.getStringArray(R.array.lista_de_planetas);
```

Cores


Recursos de cores podem ser definidos em XML.
A cor é especificada em um valor RGB e o seu alpha (transparência);
O valor de uma cor sempre começa com o caractere # seguido pela informação ARGB nos formatos:
#RGB
#ARGB
#RRGGBB
#AARRGGBB

Cores

- Localização:
  - **res/values/*colors*.xml**
    - O nome do arquivo não importa.

- Acesso:
  - No java: **R.color.*nome_do_recurso***
  - No XML: **@color/*nome_do_recurso***

- Sintaxe:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <color name="nome_da_cor">valor_da_cor_em_hexadecimal</color>
</resources>
```

Cores - Exemplo

- Arquivo XML salvo na pasta **`res/values/colors.xml`**:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <color name="vermelho_opaco">#f00</color>
        <color name="vermelho_translucido">#80ff0000</color>
</resources>
```

Cores - Exemplo

- Recurso sendo referenciado por componentes no layout da aplicação:

```
<TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello"
        android:textColor="@color/vermelho_translucido" />
```

Cores - Exemplo

- Recurso sendo recuperado no Java:

```
Resources res = getResources();
int cor = res.getColor(R.color.vermelho_opaco);
```

Dimensões

- É o valor de uma dimensão definido em XML. A dimensão é um valor especificado com um número seguido de sua unidade de medida
    - Por exemplo, 20px, 32in, 15sp

Dimensões

Opção	Descrição
px (pixels)	Correspondente ao número de pixels da tela
In (polegadas)	Baseado no tamanho físico da tela
Mm (milímetros)	Baseado no tamanho físico da tela
pt (pontos)	1/72 de uma polegada, baseado no tamanho físico da tela
dp ou dip	(Density-independent Pixels) Essa unidade é relativa à densidade dos píxeis da tela. Por exemplo, se a densidade da tela é de 160 dpi, significa que um dp representa 1 pixel em um total de 160.
sp	(Scale-independent Pixels) Idem ao dp, mas também considera o tamanho da fonte que o usuário está utilizando. É recomendado que use essa unidade quando especificar o tamanho de uma fonte, para que esta seja automaticamente ajustada conforme as preferências da tela do usuário.

Dimensões

- Localização:
  - **res/values/*dimens*.xml**
    - O nome do arquivo não importa.

- Acesso:
  - No java: **R.dimen.*nome_do_recurso***
  - No XML: **@dimen/*nome_do_recurso***

- Sintaxe:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <dimen name="nome_da_dimensao">valor_da_dimensao</dimen>
</resources>
```

Dimensões - Exemplo

- Arquivo XML salvo na pasta **`res/values/dimens.xml`**:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <dimen name="textview_altura">25dp</dimen>
        <dimen name="textview_largura">150dp</dimen>
        <dimen name="raio_bola">30px</dimen>
        <dimen name="tamanho_fonte">16sp</dimen>
</resources>
```

Dimensões - Exemplo

- Recurso sendo referenciado por componentes no layout da aplicação:

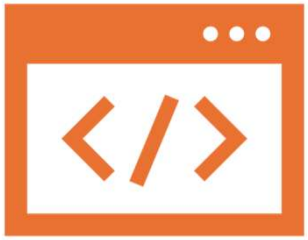
```
<TextView
        android:layout_width="@dimen/textview_largura"
        android:layout_height="@dimen/textview_altura"
        android:text="Hello"
        android:textSize="@dimen/tamanho_fonte" />
```

Dimensões - Exemplo

- Recurso sendo recuperado no Java:

```
Resources res = getResources();
float fontSize = res.getDimension(R.dimen.tamanho_fonte);
```

Estilos



Recursos de estilos definem o formato e a aparência de uma interface gráfica;

Um estilo pode ser aplicado a um determinado componente ou a uma tela inteira;

Estilos

- Localização:
  - **res/values/*styles*.xml**
    - O nome do arquivo não importa.

- Acesso:
  - No XML: **@style/*nome_do_recurso***

- Sintaxe:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <style name="nome_do_estilo">
                <item name="nome_da_propriedade">valor_da_propriedade</item>
        </style>
</resources>
```

Estilos - Exemplo

- Arquivo XML salvo na pasta **`res/values/styles.xml`**:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <style name="texto_personalizado">
                <item name="android:textSize">20sp</item>
                <item name="android:textColor">#0000ff</item>
        </style>
</resources>
```

Estilos - Exemplo

- Recurso sendo referenciado por componentes no layout da aplicação:

```
<TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello"
        style="@style/texto_personalizado" />
```

COMPUTAÇÃO MÓVEL

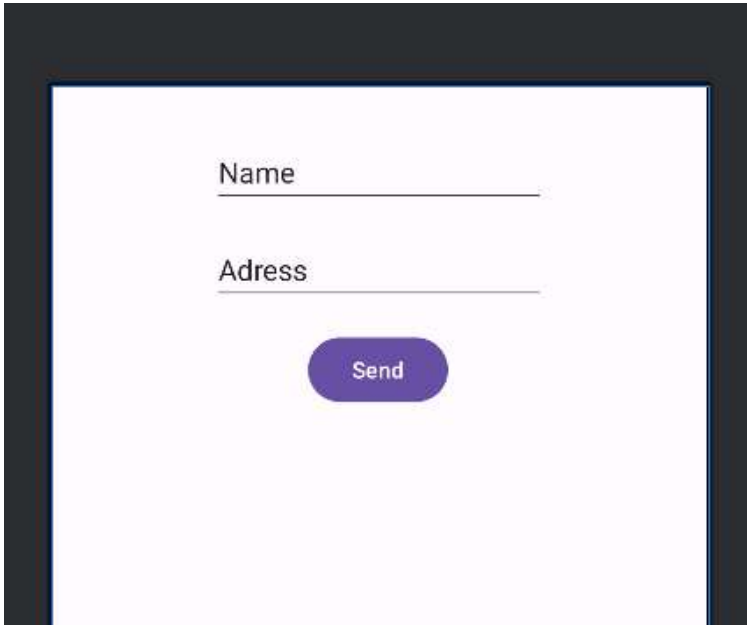
Recursos de texto e Internacionalização de Software

Recursos de texto

- Conforme visto, a pasta **`res/values`** contém as mensagens da aplicação, de forma a organizar os textos em um único arquivo centralizado e exteriorizar os recursos deste tipo;
- É considerada uma boa prática de programação;
- Desta forma, podemos facilmente traduzir este arquivo para diversos idiomas e tornar uma aplicação internacionalizável;

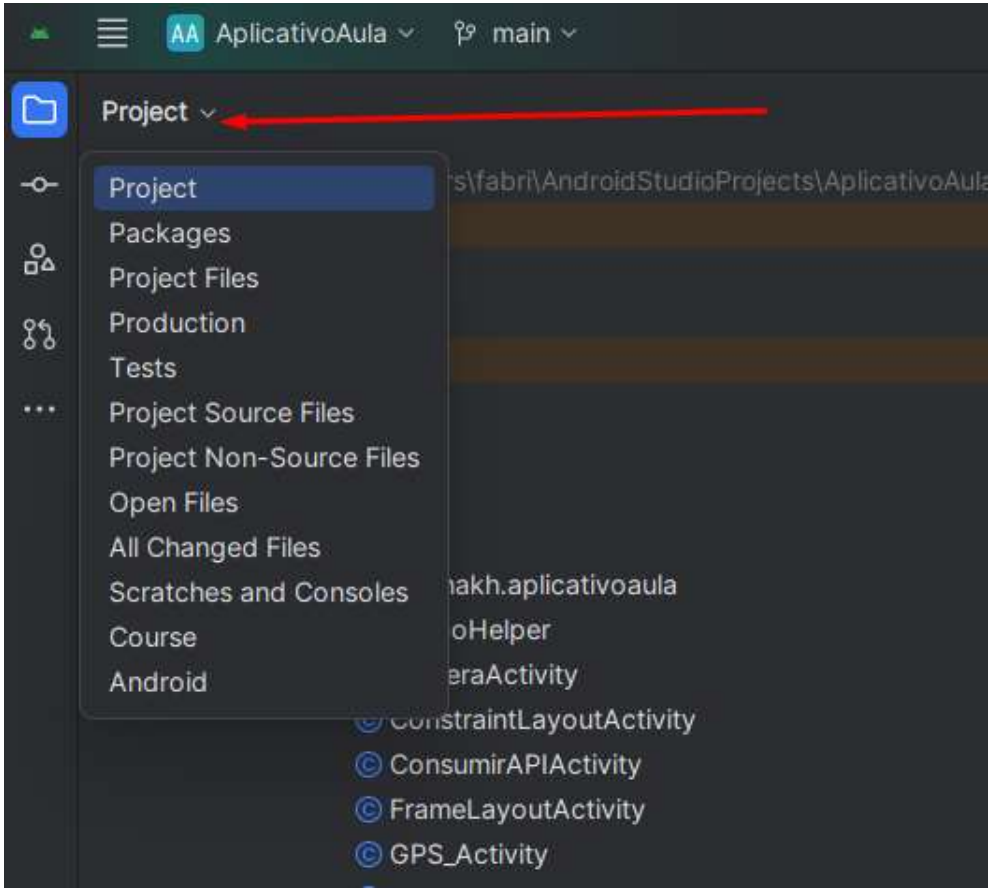
Internacionalização – Objetivo

- O objetivo desta aula é implementar a interface gráfica ao lado de maneira que a mesma se adapte a diferentes configurações de idiomas do dispositivo



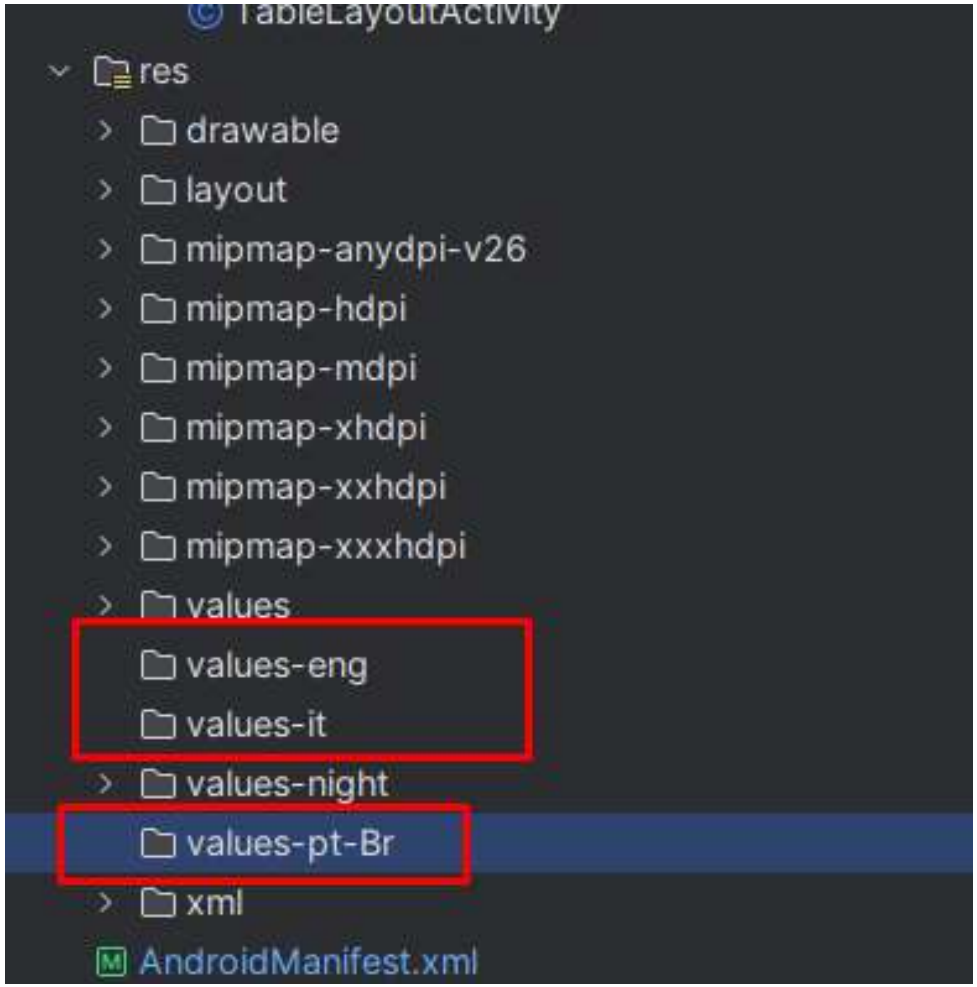
Internacionalização – 1º passo

- Primeiramente, precisamos modificar a nossa visão do projeto de “Android” para “Project”



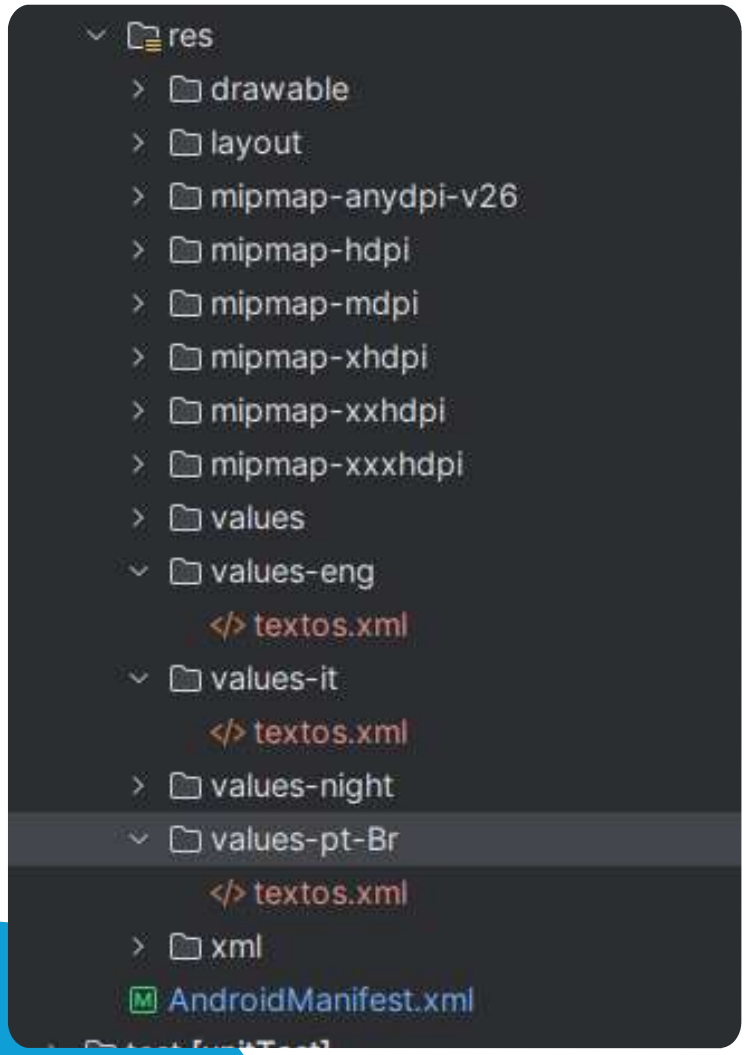
Internacionalização – 2º passo

- Em seguida, devemos criar, dentro da pasta **/res**, novas pastas values, estas com o sufixo referente aos idiomas para os quais desejamos traduzir nossa aplicação:



Internacionalização – 2º passo

- A pasta **`/res/values`** é a pasta padrão, ou seja, se o dispositivo estiver configurado em um idioma cuja aplicação não foi traduzida, ele irá buscar os recursos de texto nesta pasta;
    - E o idioma padrão de qualquer dispositivo é o inglês.
- As outras pastas criadas são específicas para cada idioma;



Internacionalização – 3º passo

- O próximo passo é adicionar um arquivo .xml (**`textos.xml`**) dentro de cada uma das pastas values criadas anteriormente:

Internacionalização – 4º passo

- Em seguida devemos adicionar os mesmos recursos de textos em todos os arquivos criados nas diferentes pastas values, observando os seguintes detalhes:
    - Os recursos de texto deverão ter o mesmo **nome**, independente de qual pasta **values** ele estiver;
    - O que varia é o **valor** do recurso, que dependerá do idioma.

Internacionalização – 4º passo

- Pasta **/res/values/textos.xml** – inglês (idioma padrão)

```
<resources>
    <string name="txt_nome">Name</string>
    <string name="txt_endereco">Address</string>
    <string name="txt_enviar">Send</string>
</resources>
```

- Pasta **/res/values-pt/textos.xml** – português

```
<resources>
    <string name="txt_nome">Nome</string>
    <string name="txt_endereco">Endereço</string>
    <string name="txt_enviar">Enviar</string>
</resources>
```

- Pasta **/res/values-pt-rBR/textos.xml** – português – região do Brasil

```
<resources>
    <string name="txt_nome">Nome</string>
    <string name="txt_endereco">Endereço</string>
    <string name="txt_enviar">Enviar</string>
</resources>
```

Internacionalização – 4º passo

- Pasta **/res/values-es/textos.xml** – espanhol

```
<resources>
    <string name="txt_nome">Nombre</string>
    <string name="txt_endereco">Dirección</string>
    <string name="txt_enviar">Enviar</string>
</resources>
```

- Pasta **/res/values-it/textos.xml** – italiano

```
<resources>
    <string name="txt_nome">Nome</string>
    <string name="txt_endereco">Indirizzo</string>
    <string name="txt_enviar">Inviare</string>
</resources>
```

- Pasta **/res/values-fr/textos.xml** – francês

```
<resources>
    <string name="txt_nome">Nom</string>
    <string name="txt_endereco">Addresse</string>
    <string name="txt_enviar">Envoyer</string>
</resources>
```

Internacionalização – 5º passo

- Mudar o texto dos componentes com base nos xmls

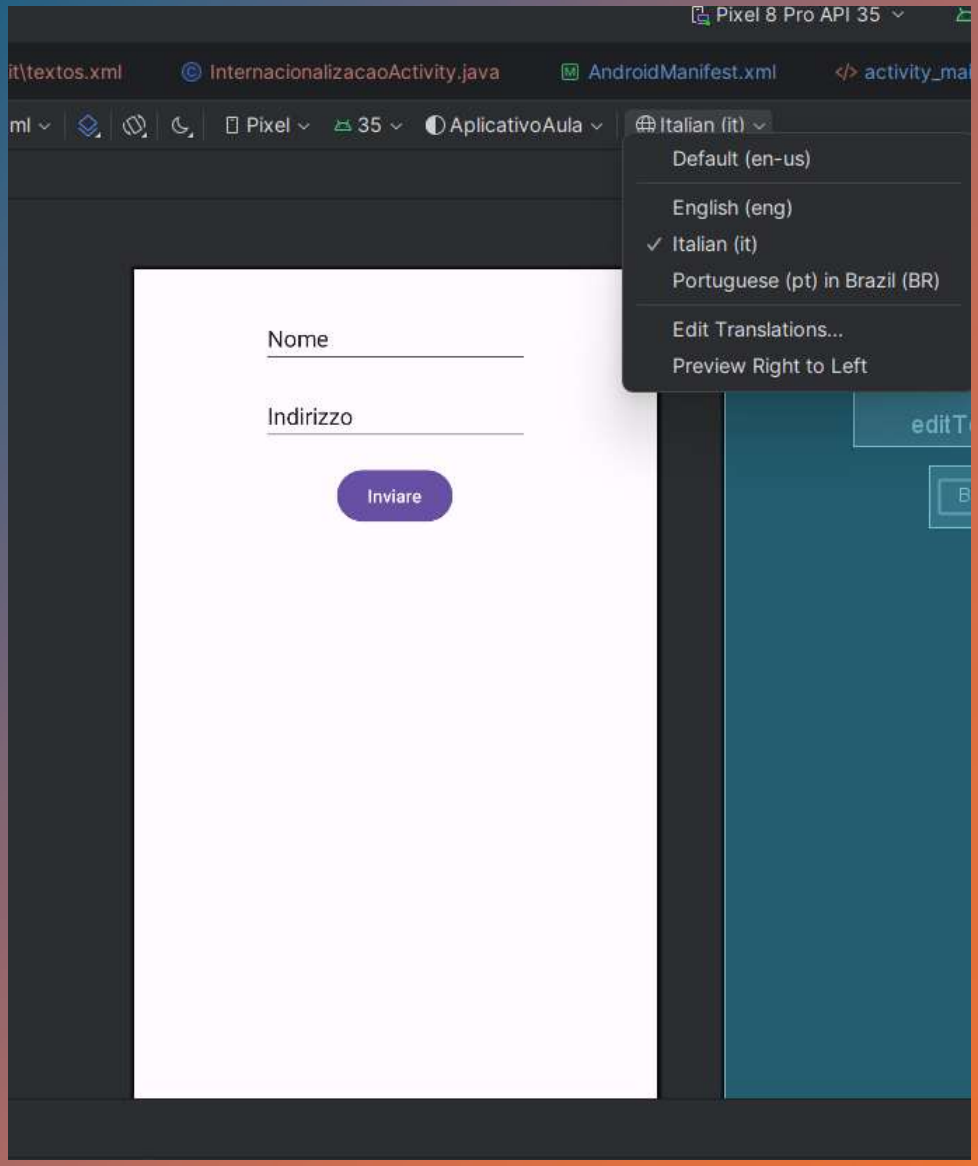
```
android:text="@string/txtNome"
```

```
android:text="@string/txtEnd"
```

```
android:text="@string/btnEnviar"
```

Observações

- O código referente ao idioma para o qual desejamos traduzir os recursos de texto (sufixos das pastas **values**) seguem a **ISO 639-1**:
  - http://www.loc.gov/standards/iso639-2/php/code_list.php
- Já os códigos referentes as regiões seguem o que foi definido pela **ISO 3166-1-alpha-2**:
  - https://www.iso.org/obp/ui/



Testando

- Para testarmos, não precisamos nem executar o emulador, basta mudar o idioma no próprio editor de interfaces:

Internacionalizando uma aplicação

- Se alterarmos o idioma nativo do emulador para italiano, ao executar a aplicação automaticamente ela alterará o idioma