Desenvolvimento de Aplicações para Dispositivos Móveis

Aula 03 – Trabalhando Layouts

- XML e Android
- Programação do XML
- Trabalhando Formulários

Prof. Fernando Gonçalves Abadia

XML e o Android

- O propósito principal do XML é a facilidade de compartilhamento de informações através da Internet. Algumas aplicações:
  - Identificação da informação
  - Armazenamento de informação
  - Estruturar informação
  - Publicação
  - Mensagens e tranferência de dados
  - Simplificação da mudança de plataforma
  - Web Services

XML e o Android

- Exemplo de XML:

```
<Pessoa>
    <email>email@pucgoias.br</email>
    <id>1234</id>
    <idade>18</idade>
    <nome>teste</nome>
</Pessoa>
```

XML e o Android

- O layout Android define a estrutura visual para uma interface do usuário, como a IU de uma atividade ou de um widget. É possível declarar um layout de dois modos:
  - Declarar elementos da IU em XML
  - Instanciar elementos do layout em tempo de execução

- Pode-se declarar os layouts padrão do aplicativo em XML, incluindo os elementos da tela que aparecerão neles e em suas propriedades.

XML e o Android

- Usando o vocabulário XML do Android, é possível projetar rapidamente layouts de IU e os elementos de tela intrínsecos do mesmo modo que se cria páginas Web em HTML — com uma série de elementos aninhados.
- Cada arquivo de layout deve conter exatamente um elemento raiz, que deve ser um objeto View ou ViewGroup.
- Com o elemento raiz definido, é possível adicionar objetos ou widgets de layout extras como elementos filho para construir gradualmente uma hierarquia de View que define o layout.

XML e o Android

- Eis um layout XML que usa um LinearLayout vertical para conter uma TextView e um Button

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Programação XML

- Para criar um aplicativo com base em XML, será mostrado nesta aula conceitos básicos de criação de um layout.
- Ao escrever um layout em XML, arquivo com a extensão .xml estará diretório res/layout/ do projeto do Android. Este arquivo pode ser alterado fora do Android Studio.
- O layout XML salvo, como por exemplo o main_layout.xml, deverá estar carregado na Atividade.

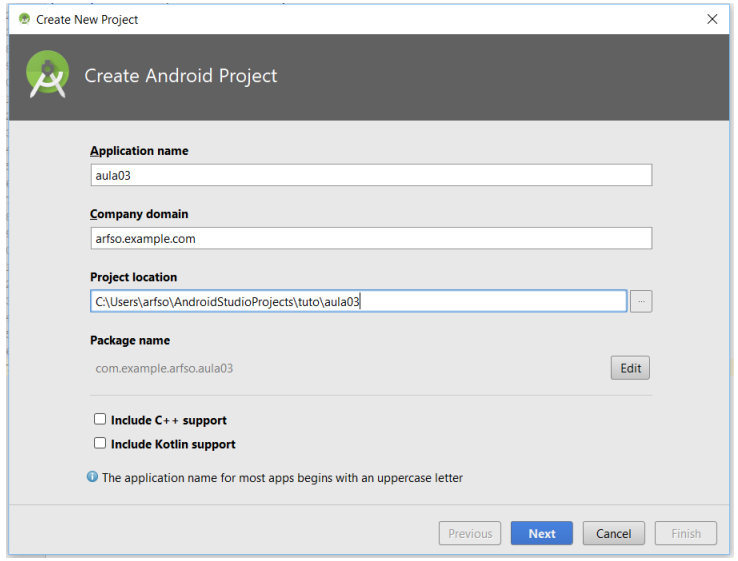
```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

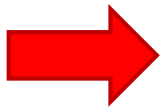
Programação XML

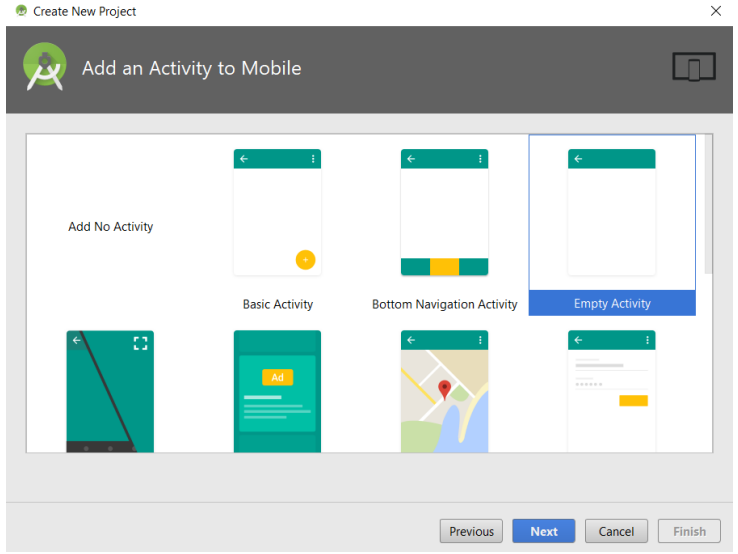
- Proposta: Criar um aplicativo que apresente um formulário e ao clicar com um botão ele apresentará a mensagem do formulário em um campo TextView.
- Para este tipo de aplicativo, usa-se muito mais conceitos de XML do que programação em si.

Programação XML

- Inicialmente precisa-se de um novo projeto no Android Studio com o nome aula03 e com uma atividade em branco.







Programação XML

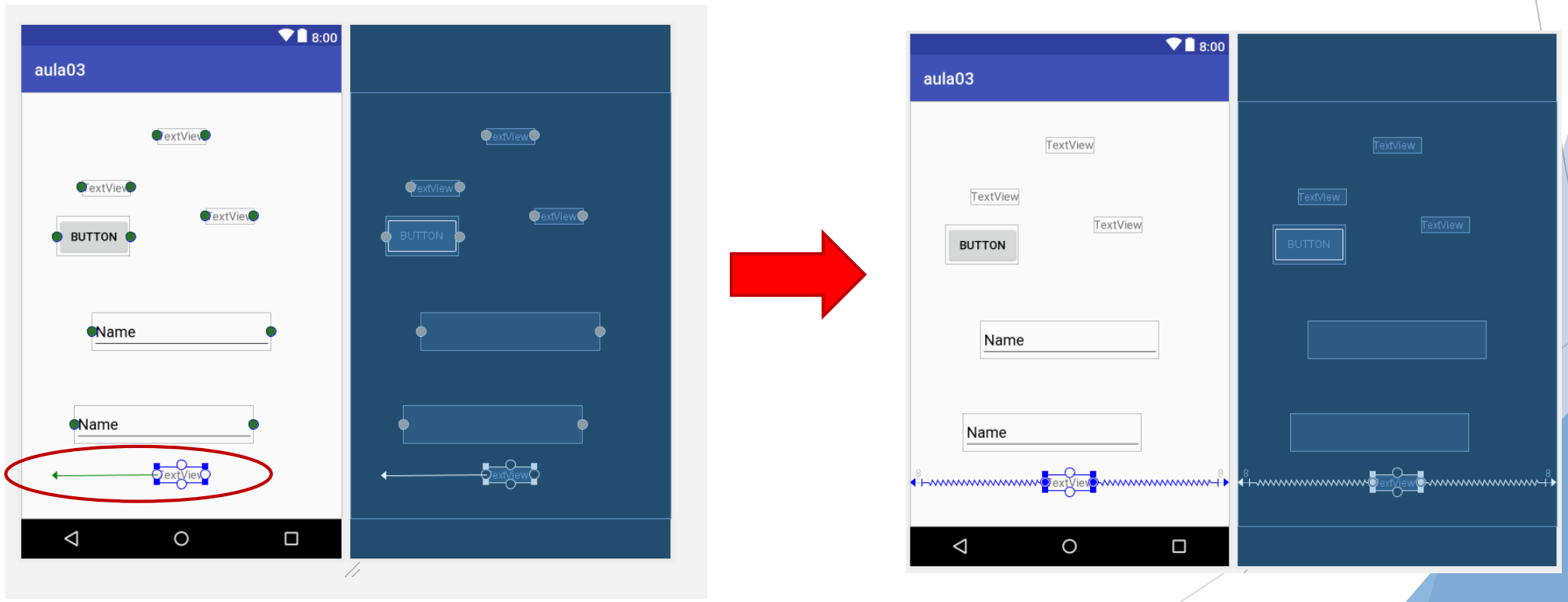
- No activity_main.xml, altere o padrão XML de forma a conter:
  - 4 TextView
  - 2 Plain Text
  - 1 Botão.
- A ordem de cada um não irá influenciar o resultado final.

activity_main.xml | MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.arfso.aula03.MainActivity">

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        tools:layout_editor_absoluteX="221dp"
        tools:layout_editor_absoluteY="139dp" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        tools:layout_editor_absoluteX="163dp"
        tools:layout_editor_absoluteY="63dp" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        tools:layout_editor_absoluteX="64dp"
        tools:layout_editor_absoluteY="375dp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
```

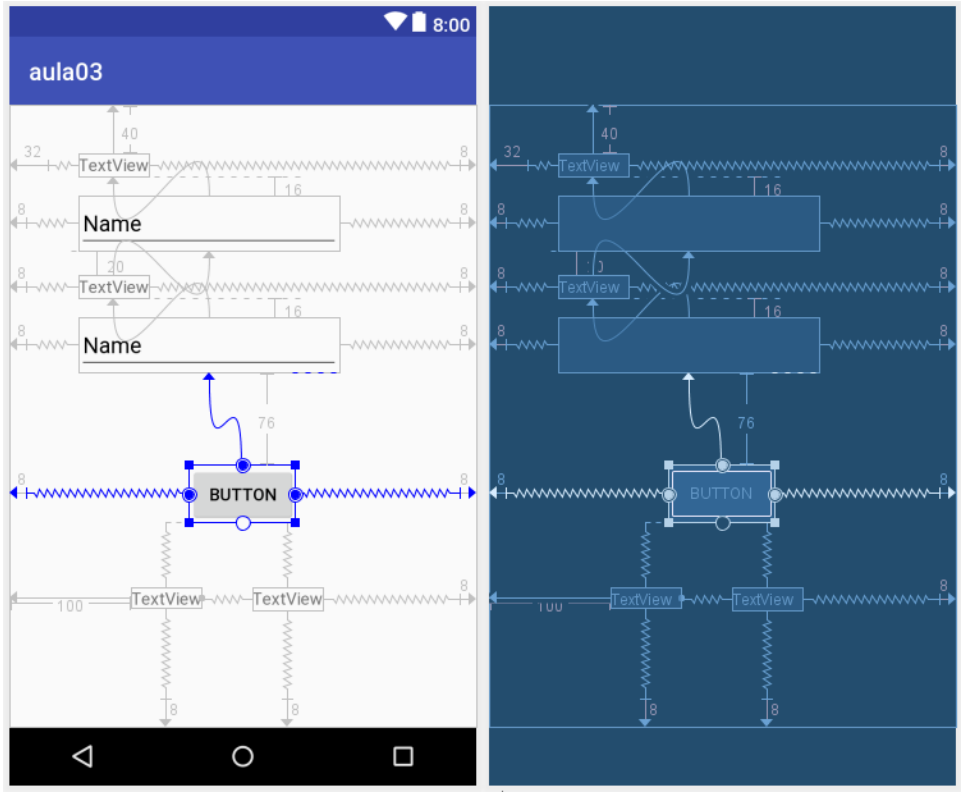
Programação XML

- Para criar um texto responsivo, trabalharemos com o Constyraint Layout. Uma maneira Simplificada seria através da ferramenta de Design do Android Studio.



Programação XML

- Crie o Design Baseando em como tudo ficará adequadamente exposto na versão final, arraste as bordas e fixe cada text view no Layout.



Programação XML

- Desta forma, podemos reduzir todo o processo de escrita de código XML, reduzindo ao máximo o tempo de programação.

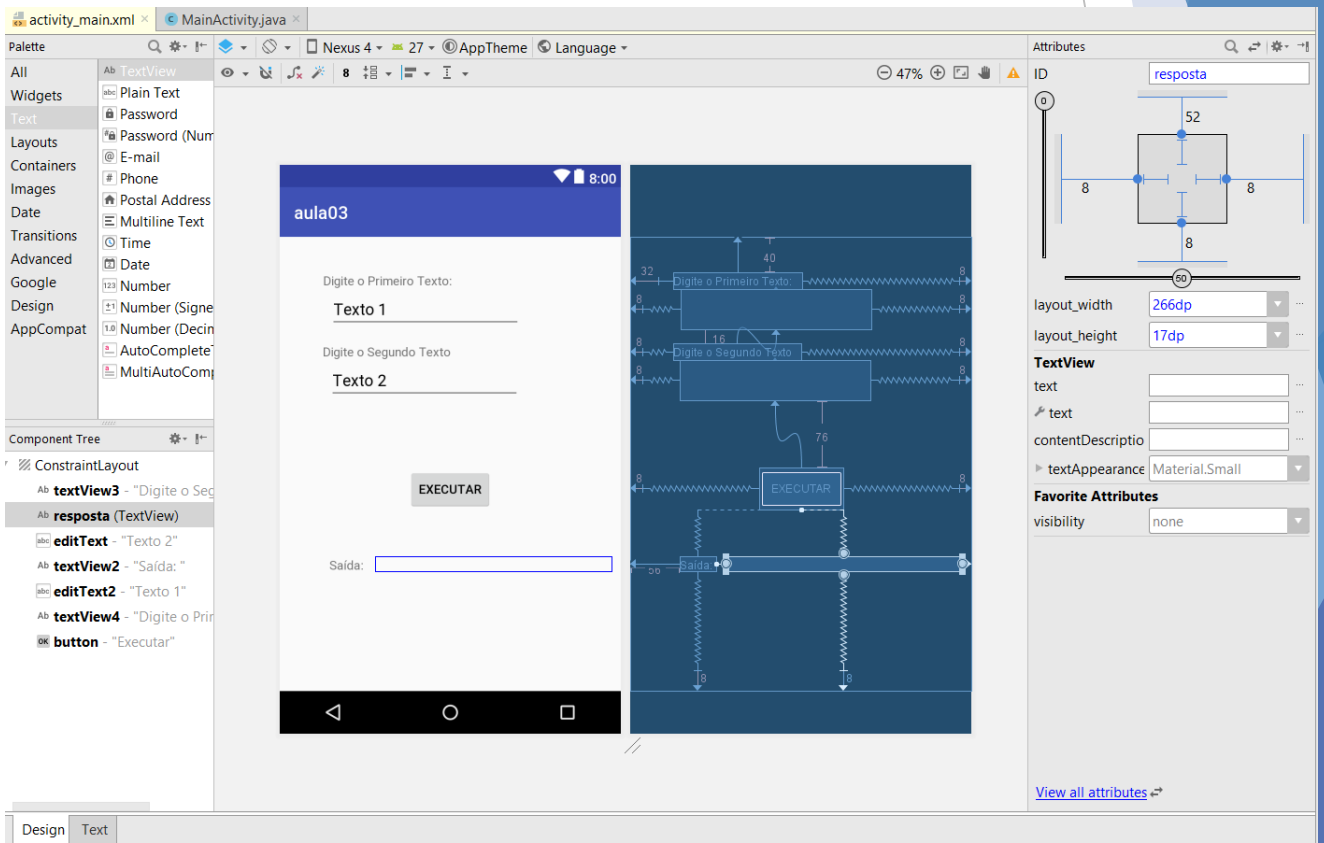
activity_main.xml | MainActivity.java

android.support.constraint.ConstraintLayout | Button

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/sand"
    tools:context="com.example.arfso.aula03.MainActivity">

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="31dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:text="Aula 03 - Exemplo de Layout "
        android:textColor="#000000"
        android:textSize="20dp"
        android:shadowColor="#FFF000"
        android:shadowDx="3"
        android:shadowDy="3"
        android:shadowRadius="2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:text="Digite o Segundo Texto"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.183"
```

Programação XML

- Troque os ids de cada Plain Text e também o texto de cada parte do layout de modo que fique compreensível ao usuário. Deixe o último text view com o texto em branco e o seu id como resposta.



Programação XML

- Caso prefira, modifique o arquivo manifest do Android para evitar rotação de tela.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.arfso.aula03">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="aula03"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportsRtl="true"
11         android:theme="@android:style/Theme.Black.NoTitleBar">
12         <activity android:name=".MainActivity" android:screenOrientation="portrait">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15
16                 <category android:name="android.intent.category.LAUNCHER" />
17             </intent-filter>
18         </activity>
19     </application>
20
21 </manifest>
```

Programação XML

- Caso queira modificar o background, cor do texto e demais configurações, pode ser feito diretamente no XML.

```
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/sand"
    tools:context="com.example.arfso.aula03.MainActivity">

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="31dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:text="Aula 03 - Exemplo de Layout "
        android:textColor="#000000"
        android:textSize="20dp"
        android:shadowRadius="22dp"
        android:shadowColor="#FFF000"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Trabalhando os Formulários

- No arquivo da atividade principal (MainActivity.java) crie duas variáveis do tipo EditText e uma TextView.

activity_main.xml | AndroidManifest.xml | MainActivity.java

MainActivity

```
1  package com.example.arfso.aula03;
2
3  import ...
7
8  public class MainActivity extends AppCompatActivity {
9      EditText texto1 , texto2;
10     TextView resp;
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16     }
17 }
18
```

Trabalhando os Formulários

- Dentro da função onCreate, deve-se setar as variáveis criadas com os campos do formulário:
  - texto1 para o primeiro Plain Text;
  - texto2 para o segundo Plain Text;
  - resp para o TextView resposta sem nada escrito.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    texto1 = (EditText) findViewById(R.id.texto1);
    texto2 = (EditText) findViewById(R.id.texto2);
    resp = (TextView) findViewById(R.id.resposta);
}
```

Trabalhando os Formulários

- Precisamos agora criar uma função para setar o novo texto para o campo resposta do TextView.
- Para isso, podemos criar a função editar logo após a onCreate.
- Esta função deverá passar a view como parâmetro uma vez que esta será alterada.

```
public void editar (View view){

}
```

Trabalhando os Formulários

- Com a função pronta, precisaremos registrá-la no XML

activity_main.xml | AndroidManifest.xml | MainActivity.java

```
1  package com.example.arfso.aula03;
2
3  import ...
8
9  public class MainActivity extends AppCompatActivity {
10     EditText texto1 , texto2;
11     TextView resp;
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17         texto1 = (EditText) findViewById(R.id.texto1);
18         texto2 = (EditText) findViewById(R.id.texto2);
19         resp = (TextView) findViewById(R.id.resposta);
20     }
21
22     public void editar (View view){
23         resp.setText(texto1.getText().toString()+" "+ texto2.getText().toString());
24     }
25 }
26
```

Trabalhando os Formulários

- No XML, deve-se criar o campo onClick e adicionar a função java “editar” que foi criada anteriormente. Note que ela aparecerá nas opções de autocomplete do Android Studio.

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="76dp"
    android:onClick="editar"
    android:text="Executar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/texto2" />
```

Trabalhando os Formulários

- Desta forma pode-se construir o aplicativo e verificar o resultado.

Exercícios

- Criar um aplicativo que entre com 4 valores e mostre-os em ordem lida, ordem crescente e decrescente.
- *Trabalhar a transformação de um texto em objeto XML.*
- *Criar um aplicativo que passe as informações lidas de um formulário para uma nova atividade (use parâmetros).*