



++

45697056

# Análise e Desenvolvimento de Sistemas

## Mobile Development and IOT - Android

45697056

...

### Criação de uma interface gráfica - Parte 01



Prof. Douglas Cabral <[douglas.cabral@fiap.com.br](mailto:douglas.cabral@fiap.com.br)>  
<https://www.linkedin.com/in/douglascabral/>



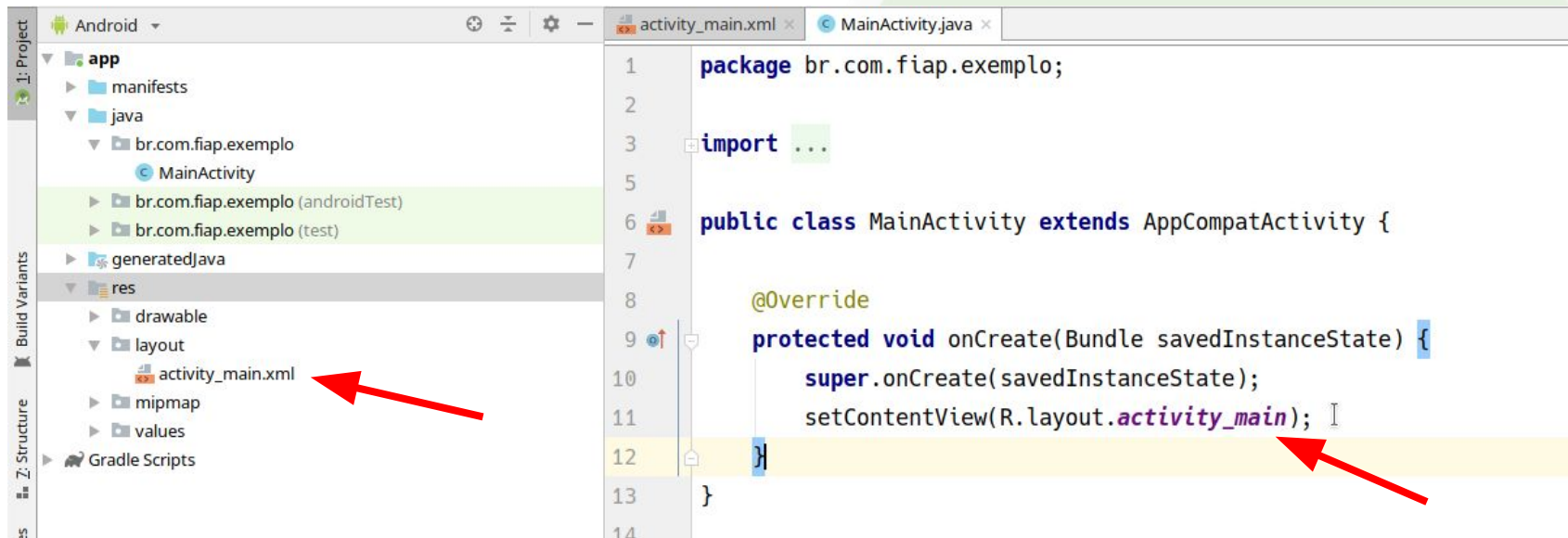
## Agenda

- **Linear Layout**
- **View**
- **ImageView**
- **TextView**
- **EditText**
- **Button**
- **Toast**

FIAP

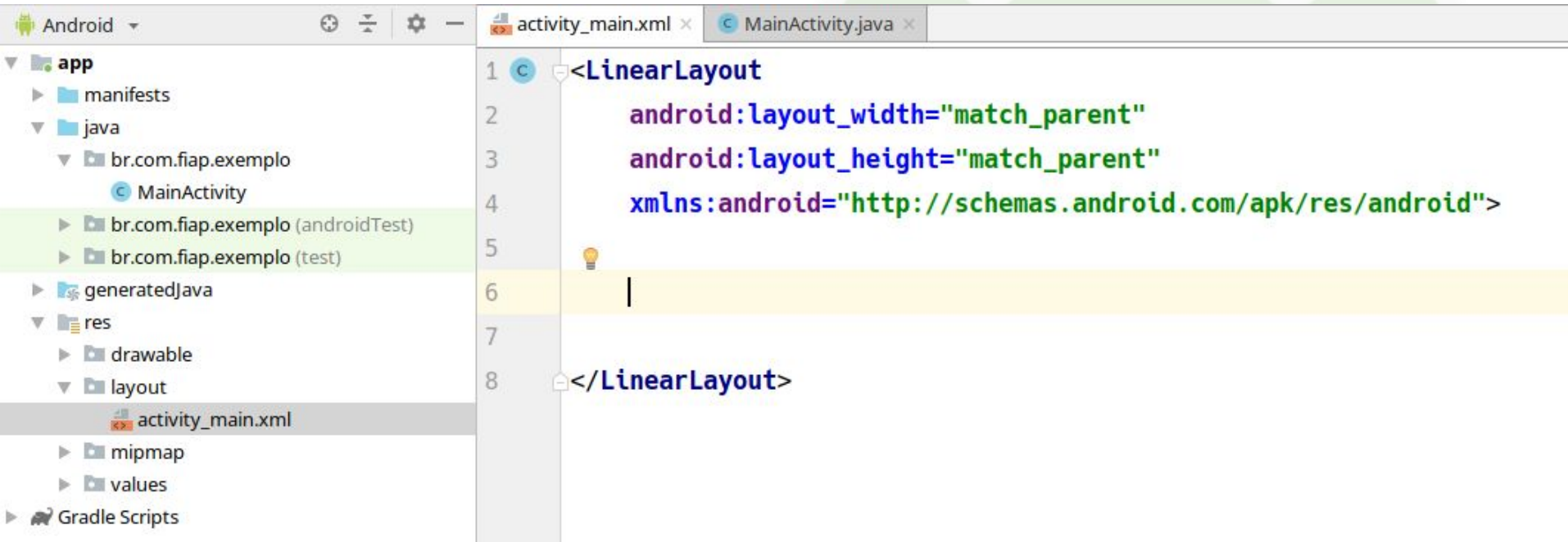
## Criação de uma interface gráfica

Conforme apresentado em aula, toda Activity possui um arquivo XML para representar sua interface gráfica:



## Criação de uma interface gráfica

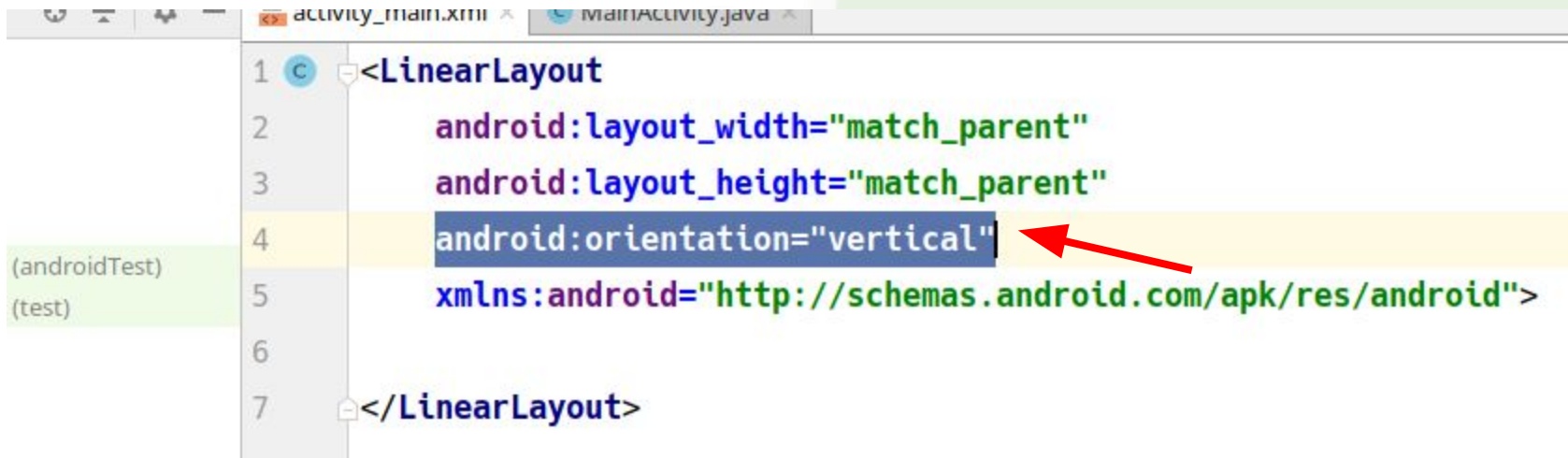
Abrindo o arquivo **activity\_main.xml** no modo texto, vamos apagar todo o conteúdo existente e inserir o seguinte conteúdo:



```
1 <LinearLayout
2     android:layout_width="match_parent"
3     android:layout_height="match_parent"
4     xmlns:android="http://schemas.android.com/apk/res/android">
5
6
7
8 </LinearLayout>
```

## Criação de uma interface gráfica

O **LinearLayout** é um gerenciador de layout simples e organiza os componentes dentro dele de **forma linear**, seja na orientação **horizontal** ou na **vertical**. Para especificar a orientação, vamos inserir o seguinte atributo: **android:orientation="vertical"**

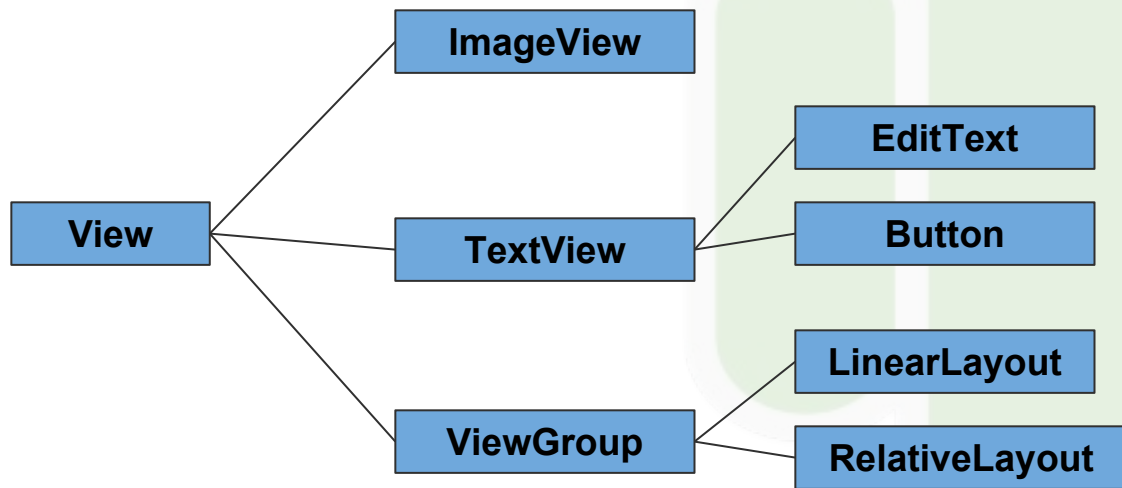


```
1 <LinearLayout
2     android:layout_width="match_parent"
3     android:layout_height="match_parent"
4     android:orientation="vertical"
5     xmlns:android="http://schemas.android.com/apk/res/android">
6
7 </LinearLayout>
```

## Criação de uma interface gráfica

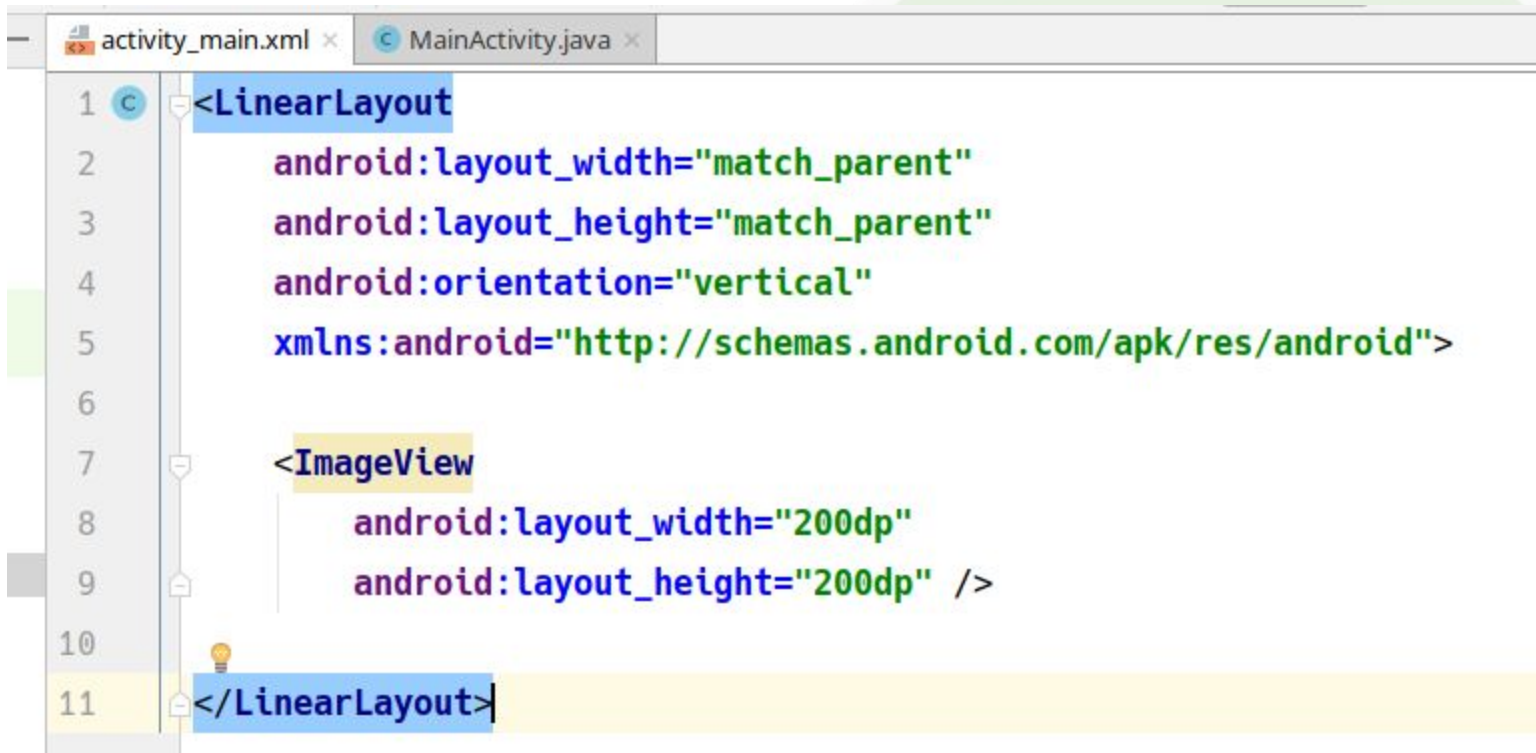
Todo componente gráfico no **Android** herda diretamente ou indiretamente a classe **View**. Outra observação importante é que toda tag XML de um componente no Android possui uma classe de mesmo nome.

Veja abaixo um pequeno pedaço da Hierarquia de classes no Android para os componentes gráficos:



## Criação de uma interface gráfica

Vamos agora inserir uma imagem em nosso layout. Para isso usaremos o **ImageView**.



```
1 <LinearLayout
2     android:layout_width="match_parent"
3     android:layout_height="match_parent"
4     android:orientation="vertical"
5     xmlns:android="http://schemas.android.com/apk/res/android">
6
7     <ImageView
8         android:layout_width="200dp"
9         android:layout_height="200dp" />
10
11 </LinearLayout>
```

## Criação de uma interface gráfica

Para especificar uma imagem no **ImageView**, basta colocar no atributo **android:src=""** o nome de uma imagem presente dentro da pasta **res/drawable** (sem a extensão do arquivo) conforme o exemplo abaixo:



Para inserir imagens dentro da pasta Drawable, basta copiar a imagem (CTRL + C) de algum lugar de seu computador, clicar em cima da pasta Drawable e colar (CTRL + V).

**OBS:** Arrastar a imagem não funciona!



## Criação de uma interface gráfica

Para centralizar a nossa imagem na horizontal de nossa LinearLayout, usaremos o atributo **android:layout\_gravity="center\_horizontal"**:

```
<ImageView  
    android:layout_gravity="center_horizontal"  
    android:layout_width="200dp"  
    android:layout_height="200dp"  
    android:src="@drawable/p1"/>
```



## Criação de uma interface gráfica

Para colocar um “**respiro**” entre os componentes e os cantos da tela, vamos inserir um **padding** em nosso **LinearLayout** conforme a imagem abaixo:

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    android:padding="16dp"  
    xmlns:android="http://schemas.android.com/apk/res/android">
```

## Criação de uma interface gráfica

O resultado de nossas modificações até o momento:

(**Obs:** A imagem pode variar de acordo com a inserida por você! )



## Criação de uma interface gráfica

Vamos inserir agora, logo abaixo da imagem que colocamos anteriormente os seguintes componentes: **TextView**, **EditText** e um **Button**.

```
<TextView
```

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Informe seu nome:"/>
```

```
<EditText
```

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" />
```

```
<Button
```

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Clique-me"/>
```

## Criação de uma interface gráfica

Vamos inserir um **evento** para o clique do botão. Após inserir o nome do evento no **atributo onClick**, basta pressionar **ALT + ENTER** com o cursor do mouse piscando em cima do nome do evento para que o **Android Studio** crie o método necessário para nós em nossa Activity:

`<Button`

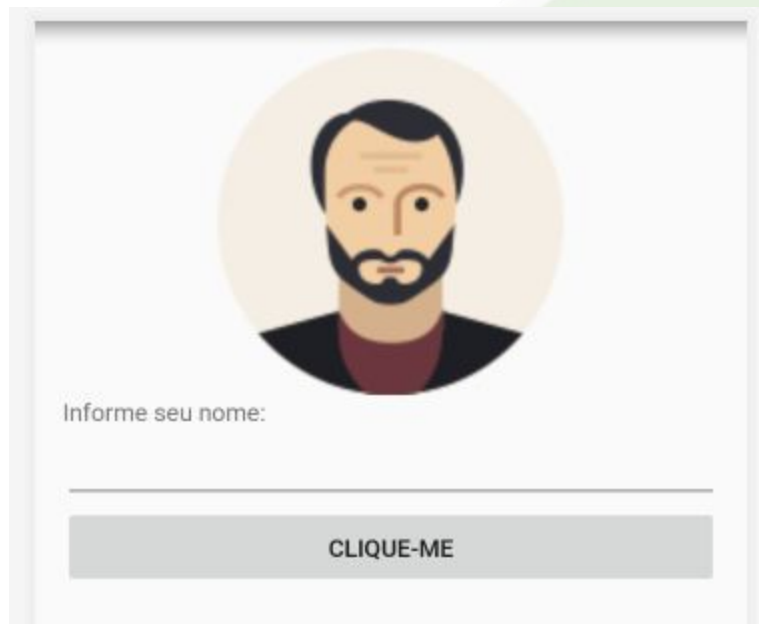
```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Clique-me"
    android:onClick="cliqueBotao"/>
```

Na classe Activity:

```
public void cliqueBotao(View view) {
}
```

## Criação de uma interface gráfica

Resultado:



Informe seu nome:


CLIQUE-ME

## Criação de uma interface gráfica

No Java, pegaremos o texto de nosso **EditText**. Para isso precisamos que no **XML** de nossa Layout, o **EditText** esteja identificado através de um **ID**.

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Informe seu nome:"/>

<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/edtNome"/>
```



## Criação de uma interface gráfica

No Java criaremos um atributo para nossa classe, para representar nosso componente EditText.

```
public class MainActivity extends AppCompatActivity {
```

```
    EditText edtNome;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
        edtNome = findViewById(R.id.edtNome);
```

```
    }
```

inicialização da variável



## Criação de uma interface gráfica

Com a variável inicializada, temos o nosso componente do XML representado agora por um **objeto Java** no qual podemos **manipular seus atributos**.

No método para o botão criado anteriormente vamos **exibir um Toast** dando um olá para o nome informado:

```
public void cliqueBotao(View view) {  
    String nome = edtNome.getText().toString();  
    Toast.makeText(this, "Olá " + nome, Toast.LENGTH_SHORT).show();  
}
```

Pega o atributo text do EditText

Exibe um Toast

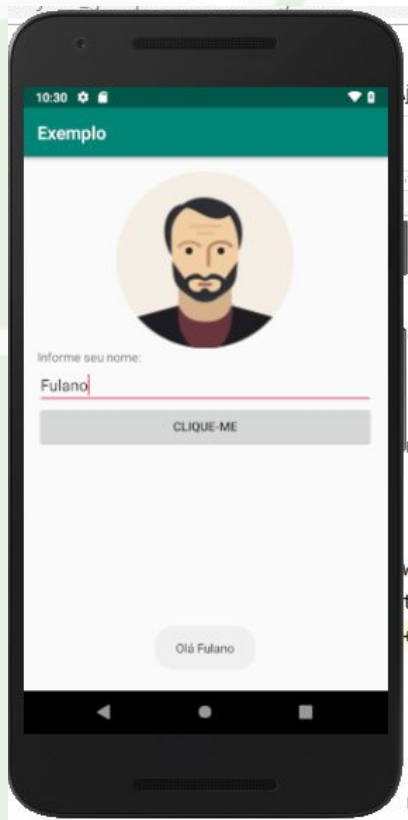
## Criação de uma interface gráfica

O resultado final de nossa aplicação ao executar em um dispositivo será:

**OBS:** O tempo de exibição do Toast poderá ser definido apenas de duas formas, utilizando as seguintes constantes em sua criação:

**Toast.LENGTH\_SHORT** ⇒ para um tempo de exibição mais curto.

**Toast.LENGTH\_LONG** ⇒ para um tempo de exibição mais longo.



+

# Dúvidas?





Copyright © 2019 Prof. Douglas Cabral <[douglas.cabral@fiap.com.br](mailto:douglas.cabral@fiap.com.br)> <https://www.linkedin.com/in/douglascabral/>

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).