FIAP

Introdução a Programação Java para Google Android

FIAP

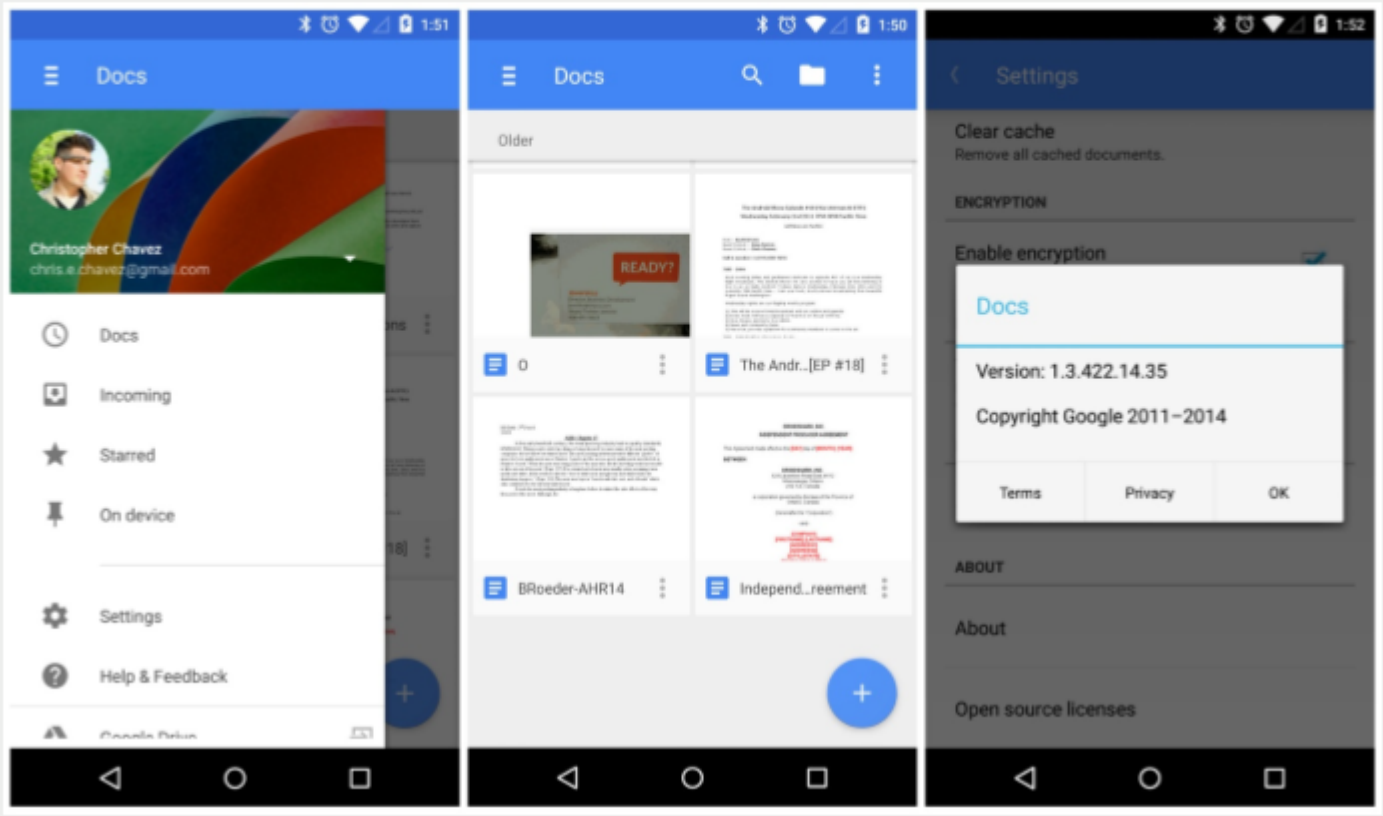
DESENVOLVIMENTO ANDROID

LAYOUTS E TEMAS

ÍNDICE

PARA SABER

Crie o layout que se adapte ao design nativo do sistema operacional



DIMENSÕES DO ANDROID

FIAP

Dimensão	O que é?	Como funciona?
px	Pixels	Número de pixels na tela
in	Polegadas	Tamanho físico da tela
mm	Milímetros	Tamanho físico da tela
pt	Pontos	1/72 de uma polegada, baseado no tamanho físico da tela
dp ou dip	Density-Independent Pixels	Relativo a resolução da tela. Exemplo: em uma tela de 160dpi, um pixel representa 1 em um total de 160
sp	Scale-Independent Pixels	Parecido com o dp porém também considera o tamanho da fonte. Recomendado a ser utilizado para especificar tamanho da fonte, para que seja ajustada automaticamente de acordo com a preferência da tela do usuário.

DIMENSÕES DO ANDROID

FIAP

- **EMS**: medida utilizada em “tipografia” (em relação a fonte utilizada), onde 1em representa o comprimento da letra m.

Ex: “android:ems=15”, então limita o tamanho de uma TextView a 15 letras m.

- **wrap_content**: o componente ocupa a medida necessária do que estiver contido nele

- **match parent**: o componente ocupa o maior espaço possível de acordo com o componente pai

CORES

FIAP

Tente usar cores da mesma escala ou mesma tabela de cores, tendo sempre tom sobre tom

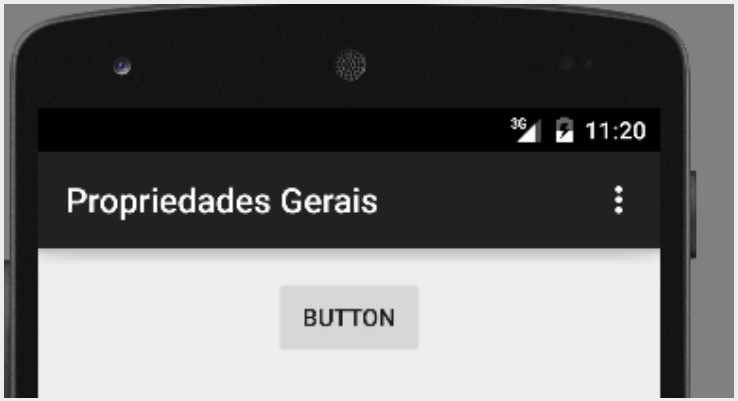
Combine as cores do logo com o layout da aplicação

ORGANIZAÇÃO DOS ASSETS

FIAP

Tipo	Prefixo	Exemplo
Ícone	ic_	ic_fiap.png
Launcher	ic_launcher_	ic_launcher_nomedoapp.png
Ícones de Action Bar	ic_menu	ic_menu_adicionar.png
Ícones de Barra de Status	ic_stat_notificar_	ic_stat_notificar_call.png
Ícones das Tabs	ic_tab	ic_tab_dados.png
Ícones de dialogo	ic_dialog	ic_dialog_atencao.png

PROPRIEDADES GERAIS



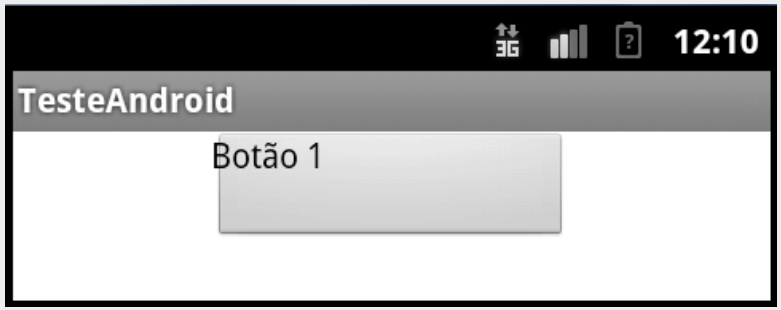
Pode-se também especificar o posicionamento do texto dentro da view por meio das propriedades padding:

1: padding_left
2: padding_top
3: padding_right
4: padding_bottom

EXEMPLO

FIAP

```
<Button
    android:id="@+id/button1"
    android:paddingLeft="0dp"
    android:paddingTop="0dp"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_alignParentTop="true"
    android:text="@string/btn1"
    android:gravity="left"/>
```



ORGANIZADORES DE LAYOUTS

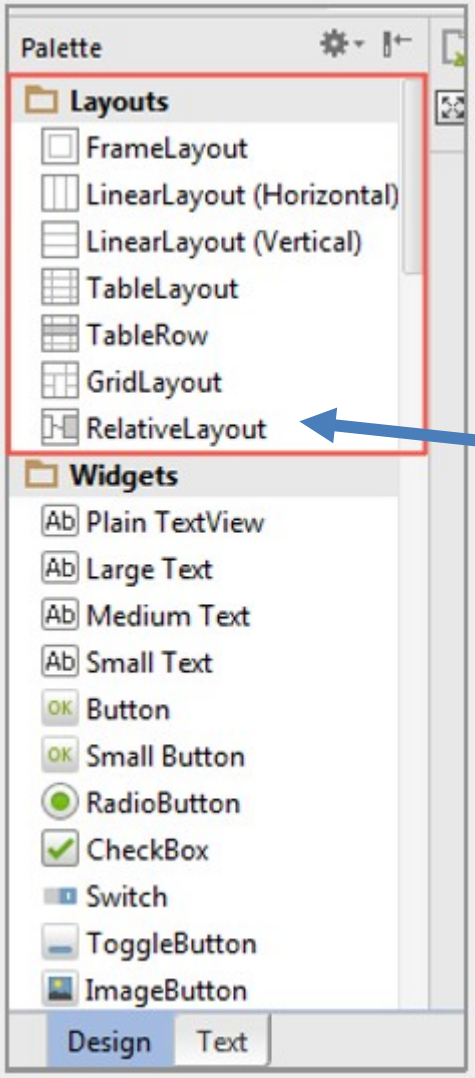
FIAP

Existem muitos tipos de dispositivos nos quais as aplicações Android podem ser executados. Eles variam em muitas características, mas a dimensão da área de display e resolução podem afetar a forma na qual as views das aplicações são exibidas. Desta forma, para tornar a adequação das views mais flexíveis entre os diversos tipos de dispositivos, existem os organizadores de layout.

Os organizadores de layout têm esta principal função: organizar dinamicamente as views que fazem parte da interface de usuário. Contudo, temos um pequeno preço a pagar: **não é só "clicar" e "arrastar"** uma view para a posição que queremos e pronto... Ao invés disso, a "**entregamos**" para gerenciamento de um organizador de layout.

Podemos combinar mais de um organizador de layout por interface.

ORGANIZAÇÃO DE LAYOUTS



Eles podem ser combinados para produzir interfaces complexas e dinâmicas

Mais complexo porém mais flexível

FRAME LAYOUT

Organizador bastante simples, pois apresenta apensar uma única view

O alinhamento da view pode ser realizado através da propriedade android:layout_gravity e que pode ser center_vertical, center_horizontal etc

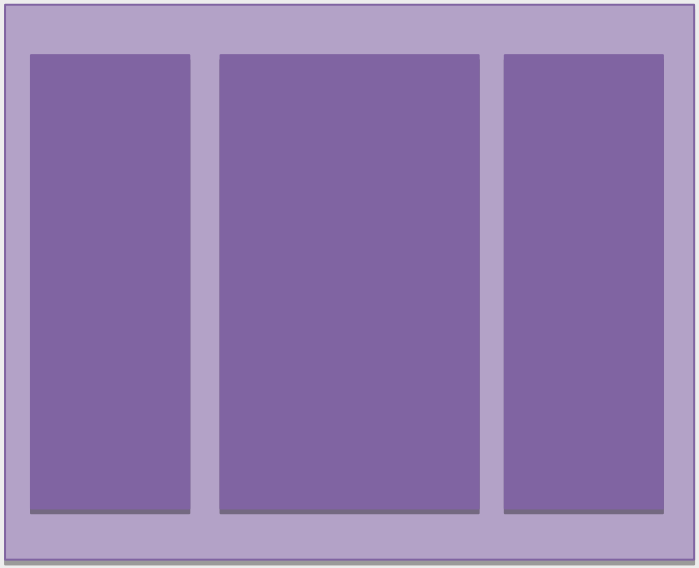
Exemplo:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#FF0000"
   android:layout_gravity="center_vertical">

   <ImageView
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:id="@+id/imageView2"
      android:src="@drawable/ic_launcher" />

</FrameLayout>
```

LINEAR LAYOUT

Organiza as views em uma única direção, em linhas (vertical) ou colunas (horizontal);

A propriedade **android:orientation** define a orientação: vertical ou horizontal;



LINEAR LAYOUT

VERTICAL

HORIZONTAL

http://developer.android.com/guide/topics/ui/layout/linear.html

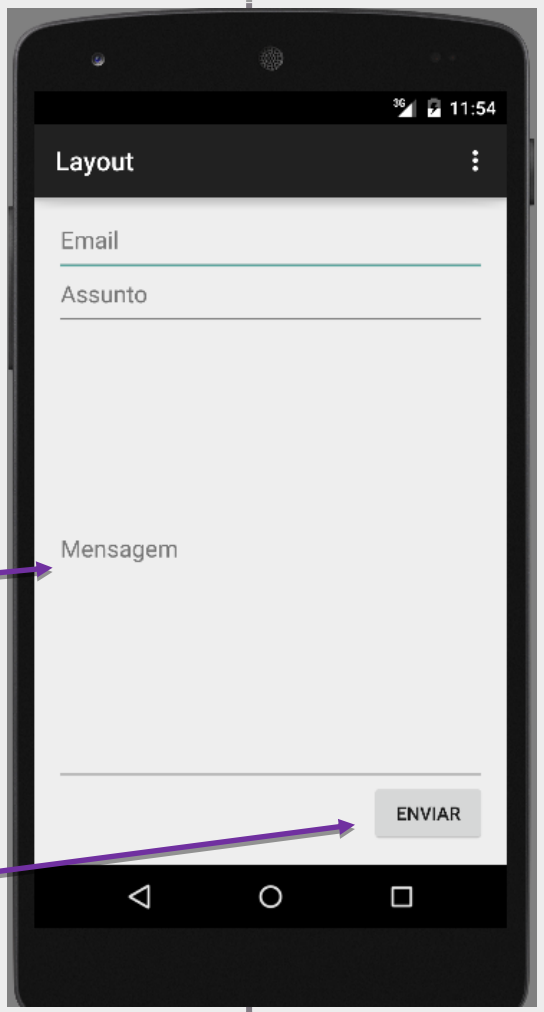
LINEAR LAYOUT

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity"
   android:orientation="vertical">

   <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/email"
      android:hint="@string/email"/>

   <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/assunto"
      android:hint="@string/assunto"/>

   <EditText
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:id="@+id/mensagem"
      android:layout_weight="1"
      android:hint="@string/mensagem"/>

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/btn_enviar"
      android:id="@+id/enviar"
      android:layout_gravity="right" />

</LinearLayout>
```

Alinha os elementos na vertical

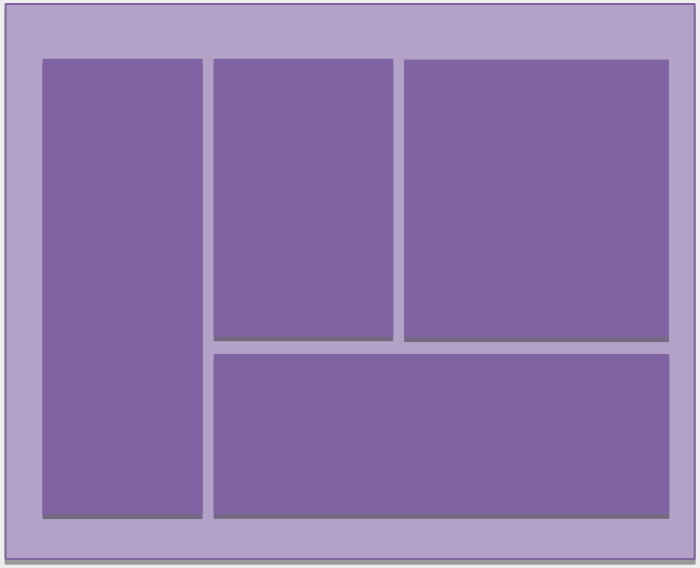
Ocupa todo o espaço disponível

Posicionamento da view



RELATIVE LAYOUT

Organiza as views em posições relativas. A posição de cada elemento pode ser definido em relação ao outro elemento da tela;



RELATIVE LAYOUT

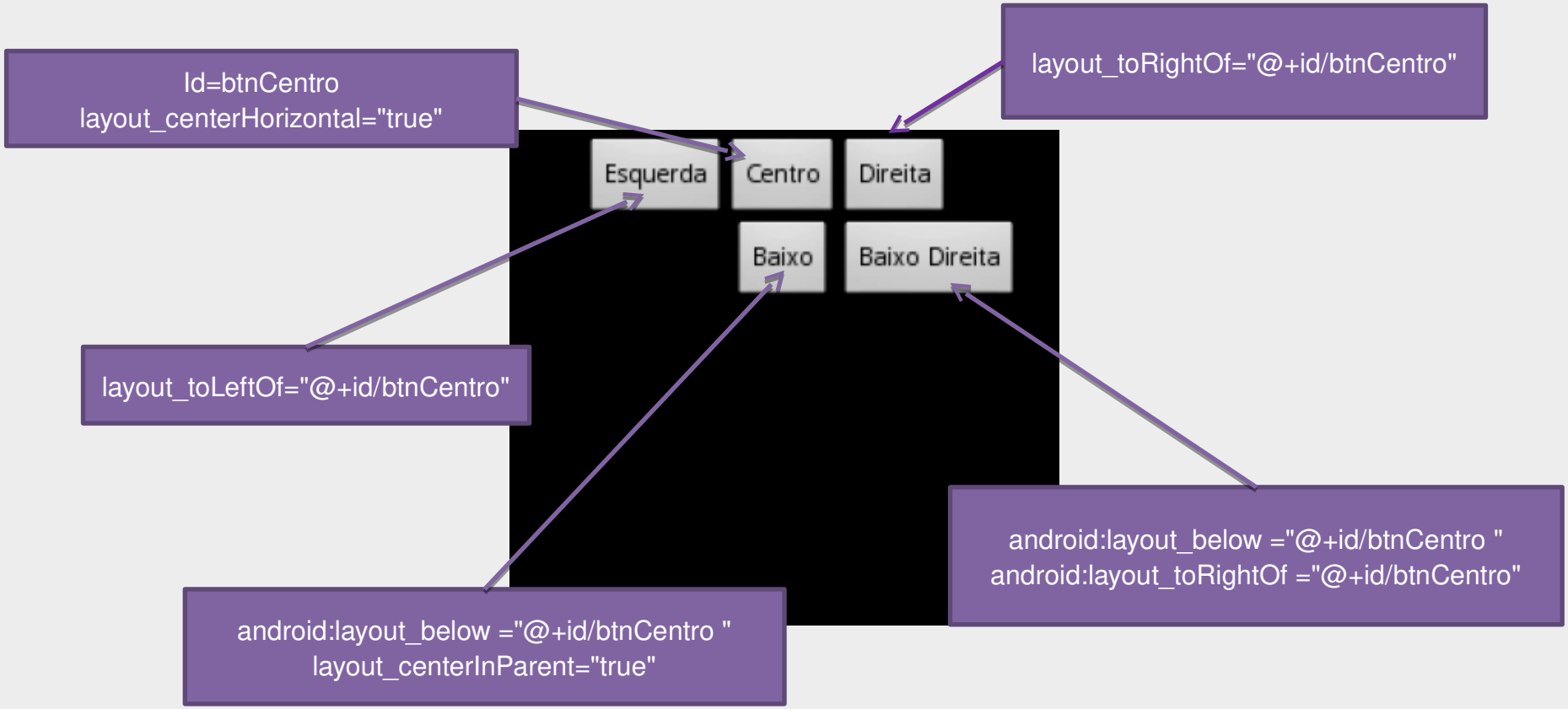
Propriedades:

- **android:layout_alignParentTop:** alinha no topo, se true;
- **android:layout_centerVertical:** alinha no centro, se true;

- **android:layout_below:** abaixo da view;
- **android:layout_toRightOf:** a direita da view;
- **android:layout_toLeftOf:** a esquerda da view;
- **android:layout_above:** acima da view;

RELATIVE LAYOUT

Exemplo:



http://developer.android.com/guide/topics/ui/layout/relative.htm

TABLE LAYOUT

FIAP

Organiza as views em uma tabela de modo semelhante às tabelas HTML;

Cada linha da tabela é representada por uma **TableRow**;

Existe uma propriedade Stretch Colums utilizada para indicar quando uma coluna deve ocupar o espaço restante não utilizado

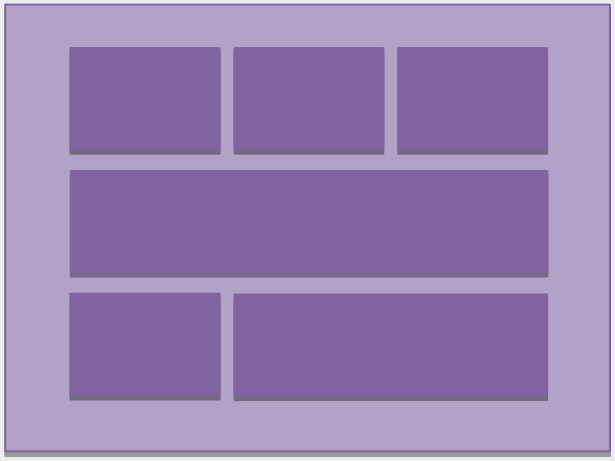
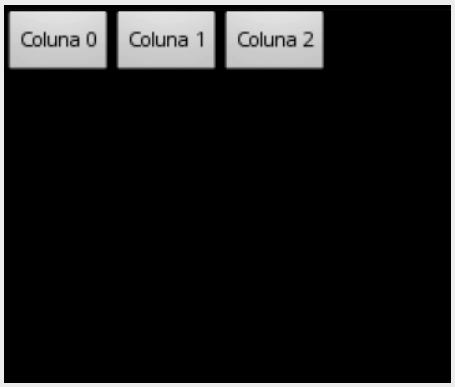
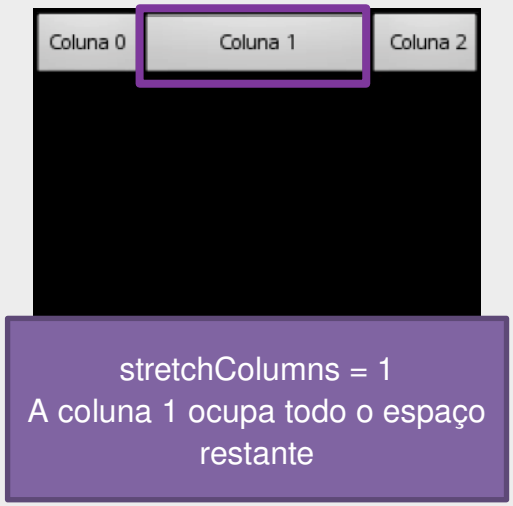
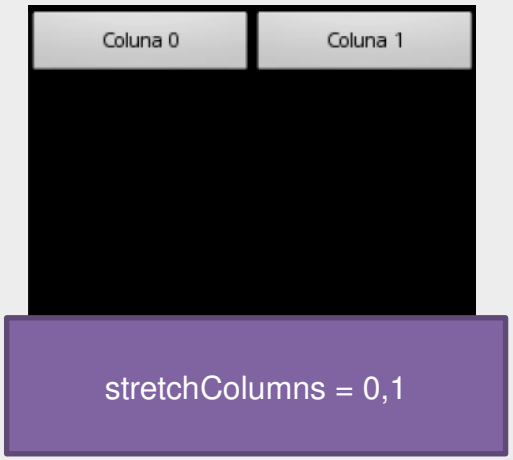


TABLE LAYOUT







https://developer.android.com/guide/topics/ui/layout/grid.html

ESTILOS E TEMAS

FIAP

- Podemos definir estilos para serem aplicados às nossas views.
- É um processo semelhante ao utilizado em HTML com CSS.
- Um conjunto de estilos podem ser aplicados a uma Activity ou aplicação inteira recebendo aí o nome de Tema.

CRIANDO UM ESTILO

FIAP

Para criar um estilo é necessário:

1. Criar um arquivo nome_do_seu_estilo.xml na pasta res/values
2. O conteúdo do arquivo de estilo deve ser:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="meuestilo“>
        <item name="android:layout_width">
            fill_parent
        </item>
    </style>
</resources>
```

style name: identifica unicamente o seu estilo

style parent: (opcional) estilo pai – somente válido para estilos herdados da plataforma (@android)

item name: nome da propriedade (do exemplo: layout_width) – o seu respectivo valor fica entre <item></item> (do exemplo: fill_parent)

http://developer.android.com/guide/topics/ui/themes.html

ESTILO

FIAP

Uma vez que o estilo foi criado ele pode ser aplicado a qualquer View que suporte as propriedades definidas nele:

```
<Button
        android:id="@+id/button1"
        android:paddingLeft="0dp"
        android:paddingTop="0dp"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_alignParentTop="true"
        android:text="@string/btn1"
        android:gravity="left"
        style="@style/meuestilo"/>
```

ESTILO

FIAP

É possível utilizar estilos já existentes e acrescentar ou sobrescrever suas propriedades por meio de herança. Isso pode ser feito para estilos da própria plataforma ou para estilos definidos pelo próprio usuário.

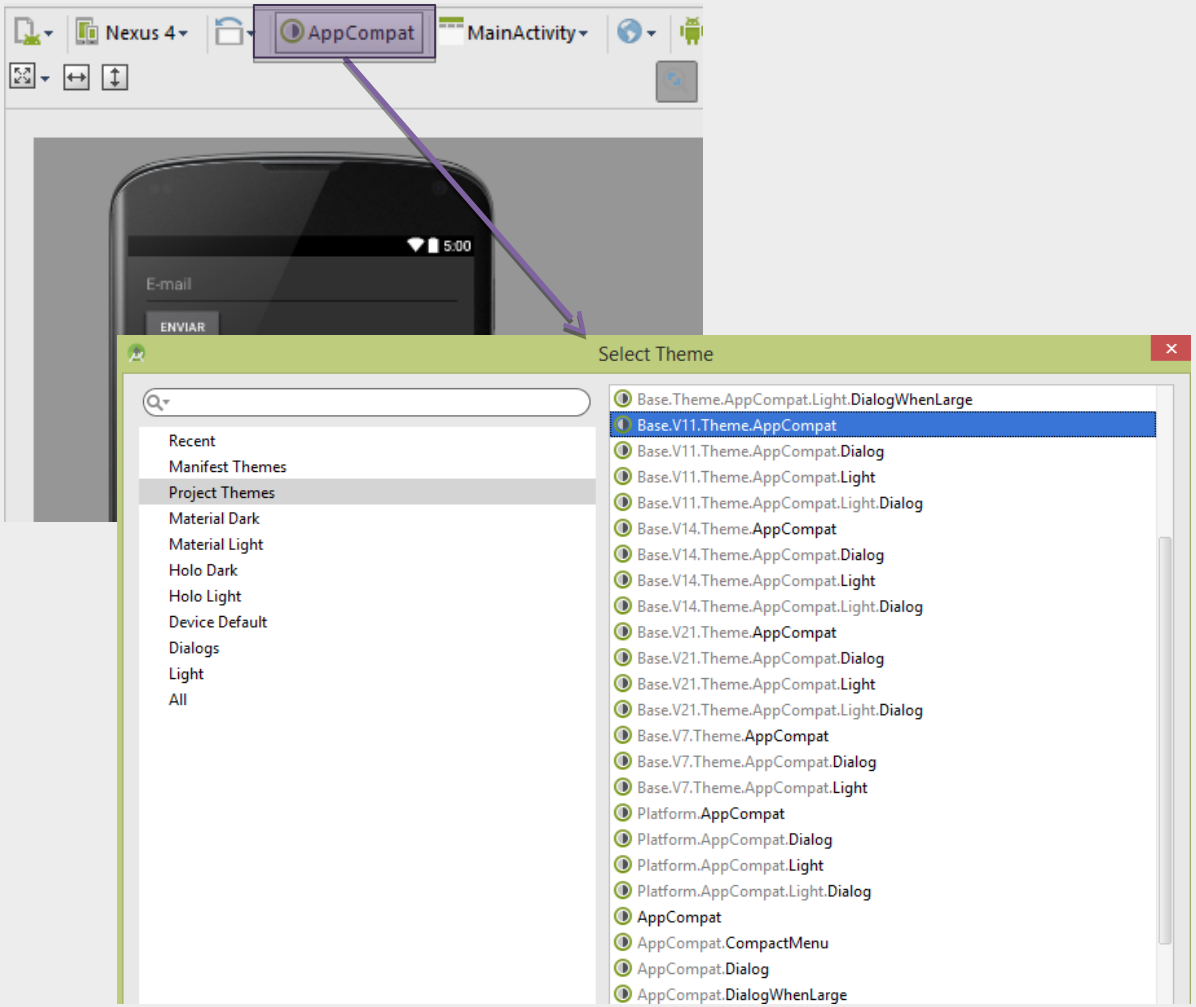
Para o segundo caso, temos o exemplo:

```
<style name="meuestilo"> … </style>

<style name="meuestilo.especifico"> … </style>
```

Neste exemplo, **meuestilo.especifico** herda todas as propriedades de **meuestilo** e pode sobrescrever ou acrescentar novas propriedades

TEMAS

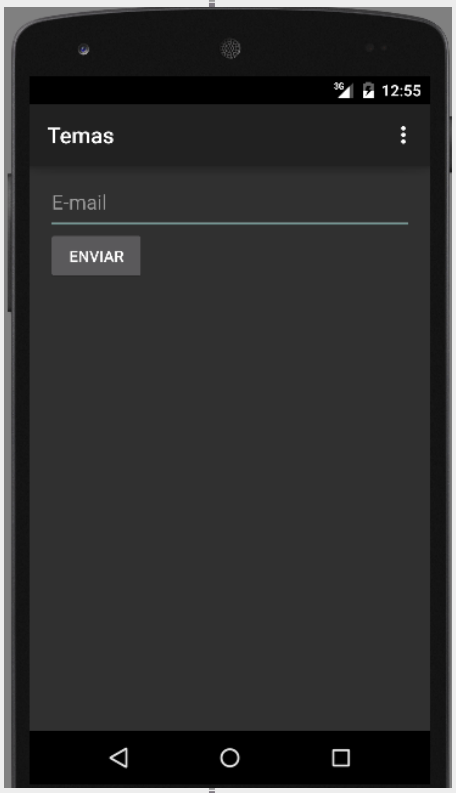
Pode-se utilizar temas da própria plataforma (verifique quais existem na própria janela do editor):



TEMAS

Os temas devem ser aplicados no AndroidManifest.xml para a aplicação ou para uma Activity específica – basta definir o atributo theme:

```
<application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/Base.V11.Theme.AppCompact" >

      <activity
         android:theme="@android:style/Theme.AppCompact"
         android:name="fiap.testeandroid.TesteActivity"
         android:label="@string/app_name" >
         ...
      </activity>
  </application>
```



EXERCíCIO

FIAP

No TableLayout:

1. **android:stretchColumns="1“**
significa que a segunda coluna irá ocupar todo o espaço restante.

2. **android:layout_marginTop="50dp“** distância entre o topo do TableLayout e a parte superior da área de display

3. **android:padding="5dp“** margens internas do TextView

No LinearLayout – horizontal (botões):

1. **android:layout_span="2“** ocupa as duas colunas da tabela
2. **android:layout_gravity="center_horizontal“** alinhado ao centro na horizontal
3. **android:padding="40dp“** margem interna

FIAP