Desenvolvimento Mobile

Aula 7

Estudo de Caso

Nesta aula vamos estudar como estilizar as **Views** da aplicação. Para isso, criaremos uma tela estática conforme o protótipo ao lado.

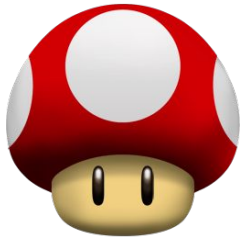
Itens necessários:

- ✅ Retirar *ActionBar*
- ✅ Trabalhar com cores e gradiente
- **Imagens**
- Layouts mais sofisticados e aninhados
- Criação de shapes circulares

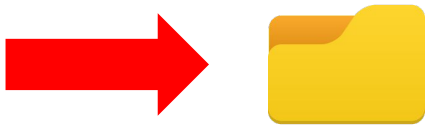
Como exibir imagens na tela?

1- Salvar um arquivo (de preferência PNG) na pasta **`app/src/main/res`** do projeto
2- Em seguida, acrescentar uma *ImageView* apontando para o recurso **`@drawable/icone`** (supondo o nome da imagem seja **icone.png**).

1



`icone.png`



`app/src/main/res`

2

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icone" />
```

Como exibir imagens na tela?

Porém, ao executarmos o projeto em dispositivos diferentes, nos deparamos com o seguinte problema.

Samsung Galaxy Ace

LG Nexus 5X

Como exibir image

Porém, ao ex iferentes, nos deparamos

Paciência precisamos ter ao estudar imagens...

Samsung

Nexus 5X

Informações sobre a tela do dispositivo

Existem uma infinidade de dispositivos Android no mercado, cada com suas características próprias no que toca a configuração da tela. Devemos ter em mente as seguintes características:

Tamanho da Tela

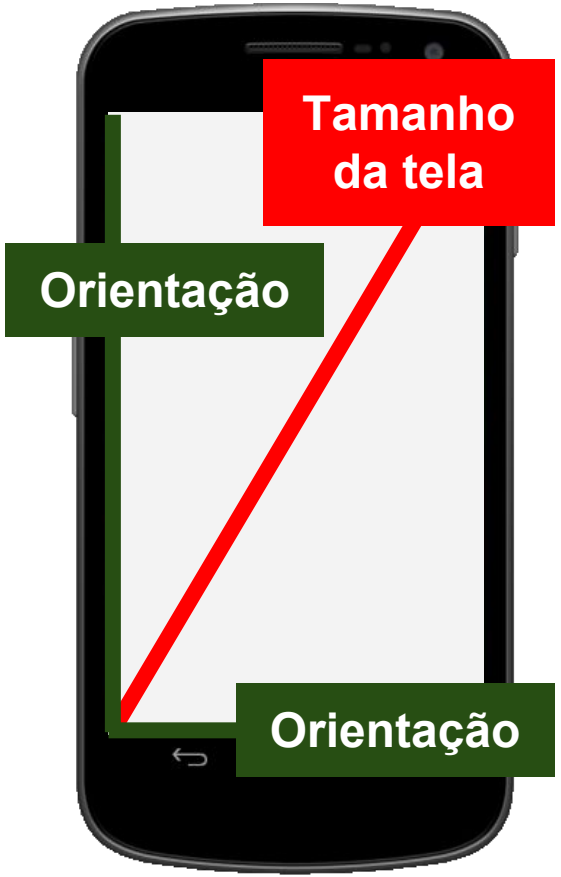
Densidade da Tela

Orientação

Resolução

Informações sobre a tela do dispositivo

Tamanho da Tela	Tamanho físico da tela, medido na diagonal (geralmente em polegadas). A especificação do Android agrupa os tamanhos reais em quatro tamanhos genéricos: `small`, `normal`, `large` e `xlarge`
Orientação	Orientação da tela no ponto de vista do usuário. Há duas opções: retrato (altura maior que comprimento) ou paisagem (comprimento maior que altura).



Informações sobre a tela do dispositivo

Densidade da Tela

Também conhecido como DPI (pontos por polegada), a densidade é uma quantidade de pixels em uma área física da tela.
Por exemplo, uma tela com densidade baixa possui menos pixels em uma determinada área em comparação com telas com média ou alta densidade. A especificação do Android agrupa as densidades reais em seis densidades genéricas:

```
ldpi (baixa) ~ 120 dpi

mdpi (média) ~ 160 dpi

hdpi (alta) ~ 240 dpi

xhdpi (extra-alta) ~ 320 dpi

xxhdpi (extra-extra-alta) ~ 480 dpi

xxxhdpi (extra-extra-extra-alta) ~ 640 dpi
```

Informações sobre a tela do dispositivo

Resolução

Total de pixels físicos em uma tela. Na prática não é muito considerado quando pensamos em layouts portáveis para dispositivos com telas diferentes.

Informações sobre a tela do dispositivo

Tamanho da Tela

Densidade da Tela

Orientação

Resolução

Ok, mas e daí?

Informações sobre a tela do dispositivo

Tamanho da Tela

Densidade da Tela

Orientação

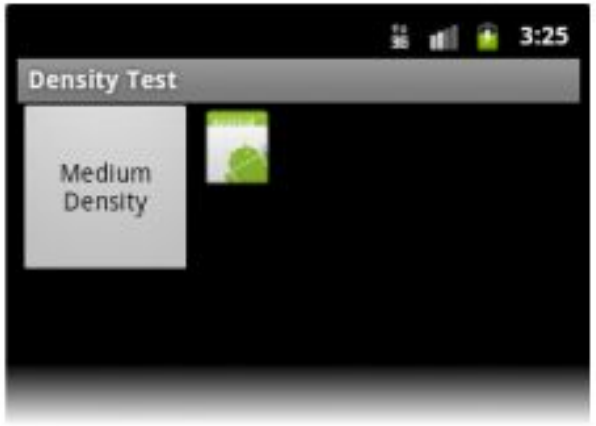
Resolução

Ok, mas e daí?

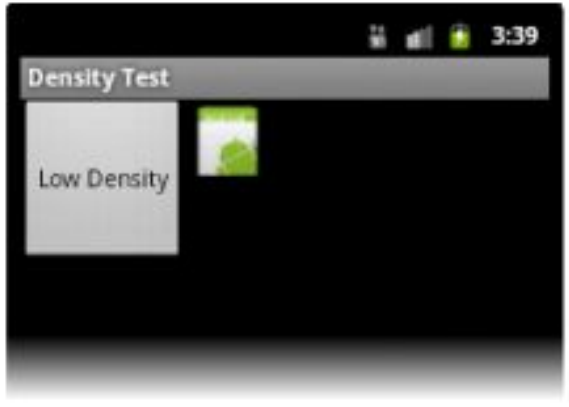
Pixel independente de densidade (dp)

É importante entender esses conceitos para alcançar a *Independência de densidade* (quando o app preserva o tamanho físico dos elementos sob o ponto de vista do usuário).
A falta desta independência faz com que um elemento apareça maior em uma tela de baixa densidade, e menor em uma tela de alta densidade.

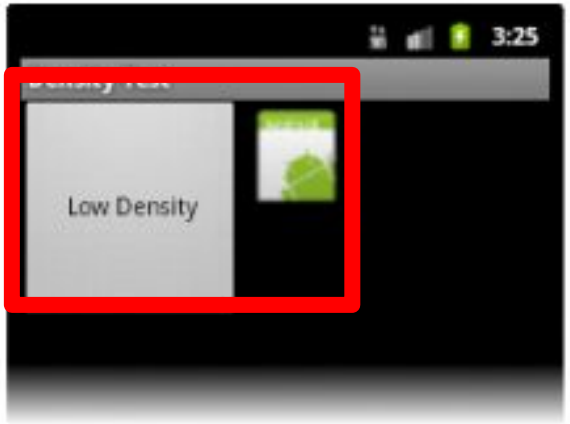
App incompatível com diferentes densidades

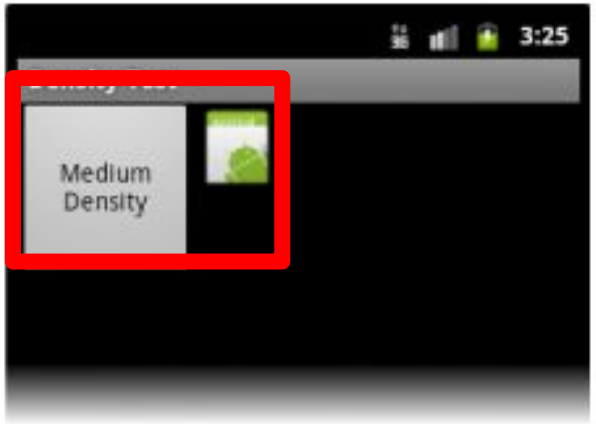


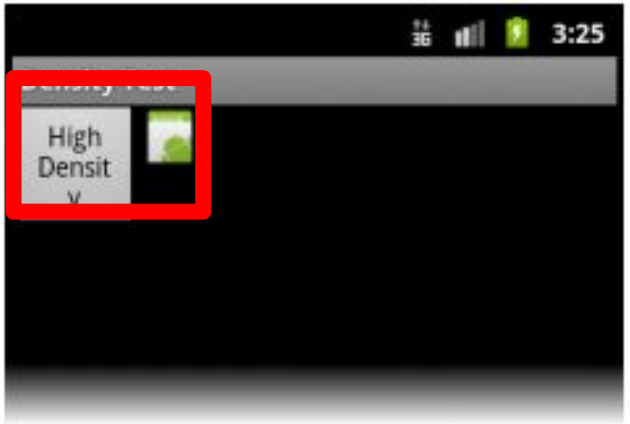
App compatível com diferentes densidades



App incompatível com diferentes densidades





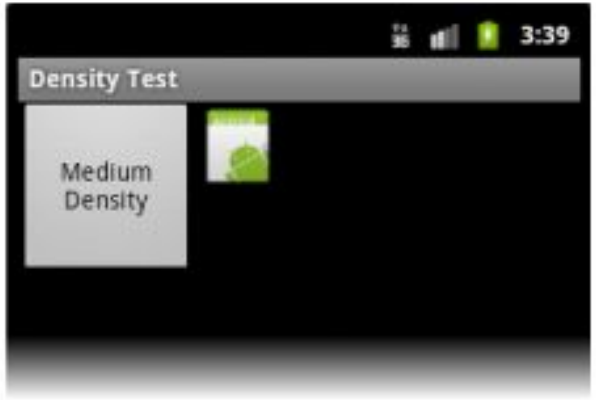


Estes elementos foram definidos com a medida *px*.
⇒ Como em telas de baixa densidade há menos pixels por polegada, o botão e a imagem aparecem maiores.
⇒ Por outro lado, como em telas de maior densidade há mais pixels por polegada, o botão e a imagem aparecem menores.

Informações sobre a tela do dispositivo

Para resolver este problema, temos três práticas:
⇒ Trabalhar com a medida pixels independente de densidades (dp)
⇒ Prover imagens maiores para as densidades mais altas
⇒ Criar layouts específicos por densidade **(veremos no futuro)**

App compatível com diferentes densidades



Informações sobre a tela do dispositivo

Densidade da Tela

`ldpi (baixa) ~ 120 dpi`

`mdpi (média) ~ 160 dpi`

`hdpi (alta) ~ 240 dpi`

`xhdpi (extra-alta) ~ 320 dpi`

`xxhdpi (extra-extra-alta) ~ 480 dpi`

`xxxhdpi (extra-extra-extra-alta) ~ 640 dpi`

O qualificador **`mipmap-xxxhdpi`** é necessário apenas para fornecer um ícone de inicialização

3:4:6:8:12:16

36 x 36 (0,75x) para densidade baixa

48 x 48 (1,0x - configuração básica) para densidade média

72 x 72 (1,5x) para densidade alta

96 x 96 (2,0x) para densidade extra alta

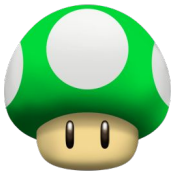
144 x 144 (3,0x) para densidade extra-extra-alta

192 x 192 (4,0x) para densidade extra-extra-extra-alta

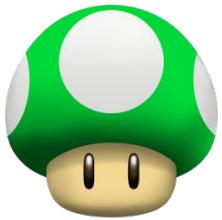
`drawable-ldpi`

`drawable-mdpi`

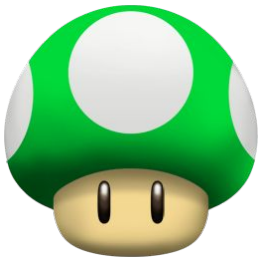
`drawable-hdpi`



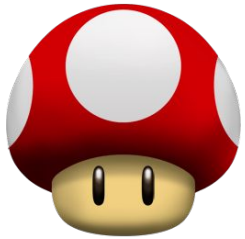
`drawable-xhdpi`



`drawable-xxhdpi`



Exemplo



256 x 256 pixels

Temos um projeto com uma imagem de um único tamanho disponível em *drawable*. Veja o que ocorre ao executá-lo em dispositivos com densidades diferentes.

320x480 mdpi

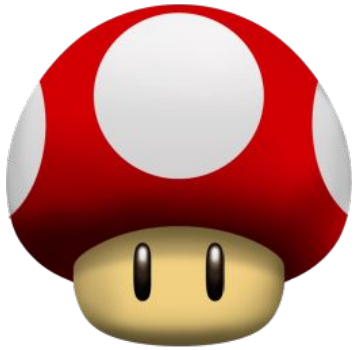
720x1280 xhdpi

1080x1920 xxhdpi

Exemplo

Podemos criar imagens menores a partir da original respeitando a relação indicada na documentação do Android.

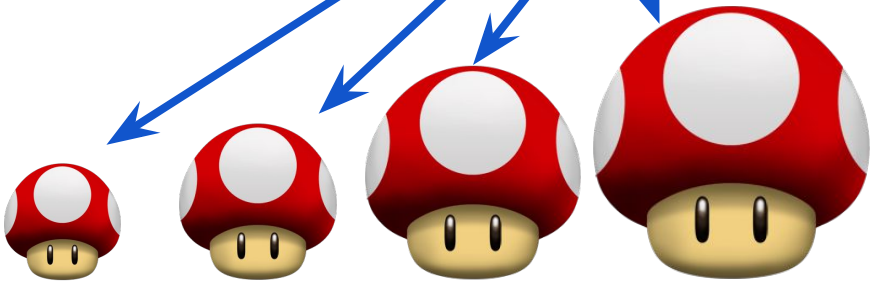
Exemplo:
8 / 12 * 256 = 170,66



256 x 256 pixels

3:4:6:8:12:16

Esta densidade genérica só é necessária para ícones do App.



64x64 85x85 128x128 171x171

Exemplo

Problema resolvido! Note que como a imagem **mdpi** possui 85x85 pixels, ela ainda fica maior no primeiro emulador.

`320x480 mdpi`

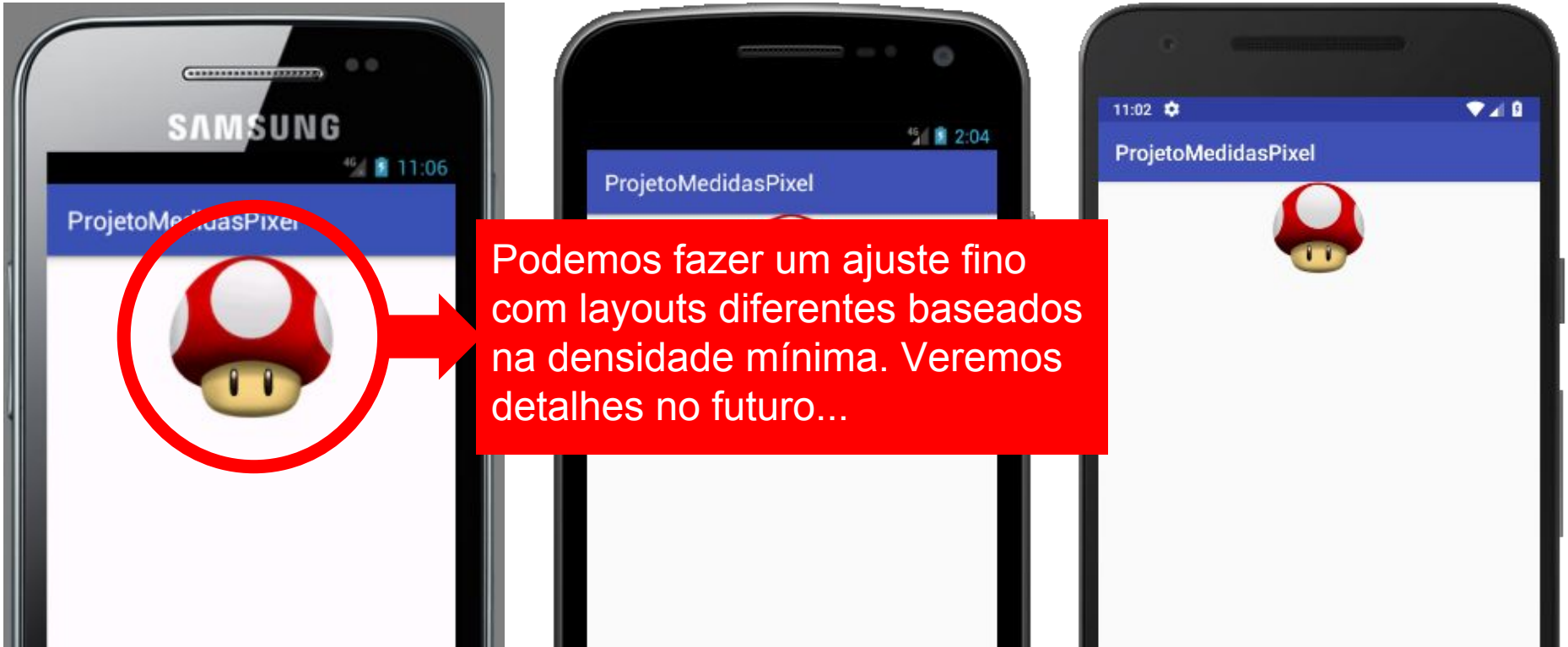
`720x1280 xhdpi`

`1080x1920 xxhdpi`

Exemplo

Problema resolvido! Note que como a imagem **mdpi** possui 85x85 pixels, ela ainda fica maior no primeiro emulador.

`320x480 mdpi` `720x1280 xhdpi` `1080x1920 xxhdpi`

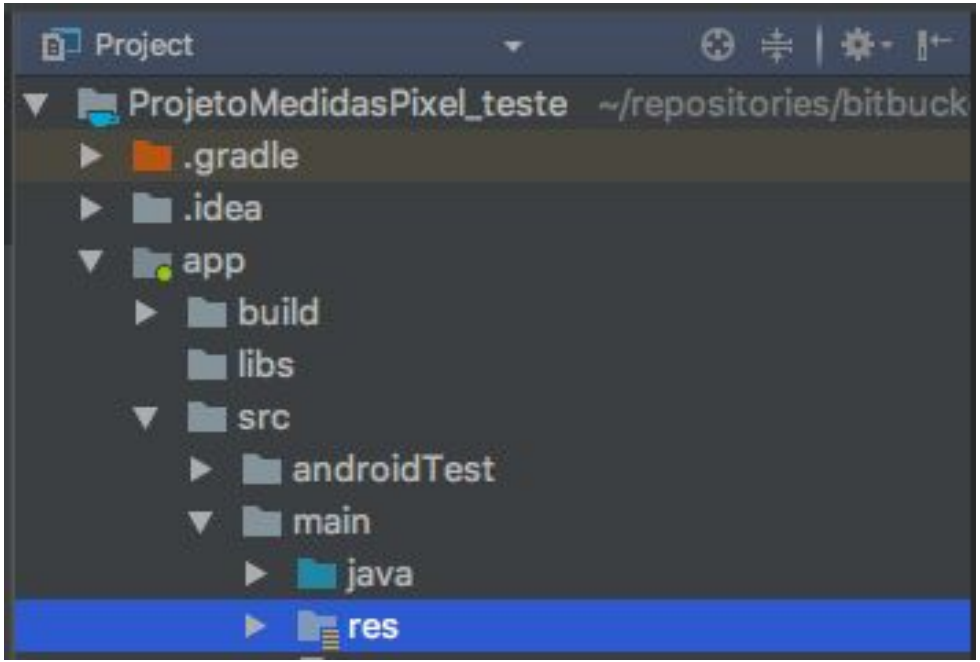


Podemos fazer um ajuste fino com layouts diferentes baseados na densidade mínima. Veremos detalhes no futuro...

Configuração do projeto com imagens distintas

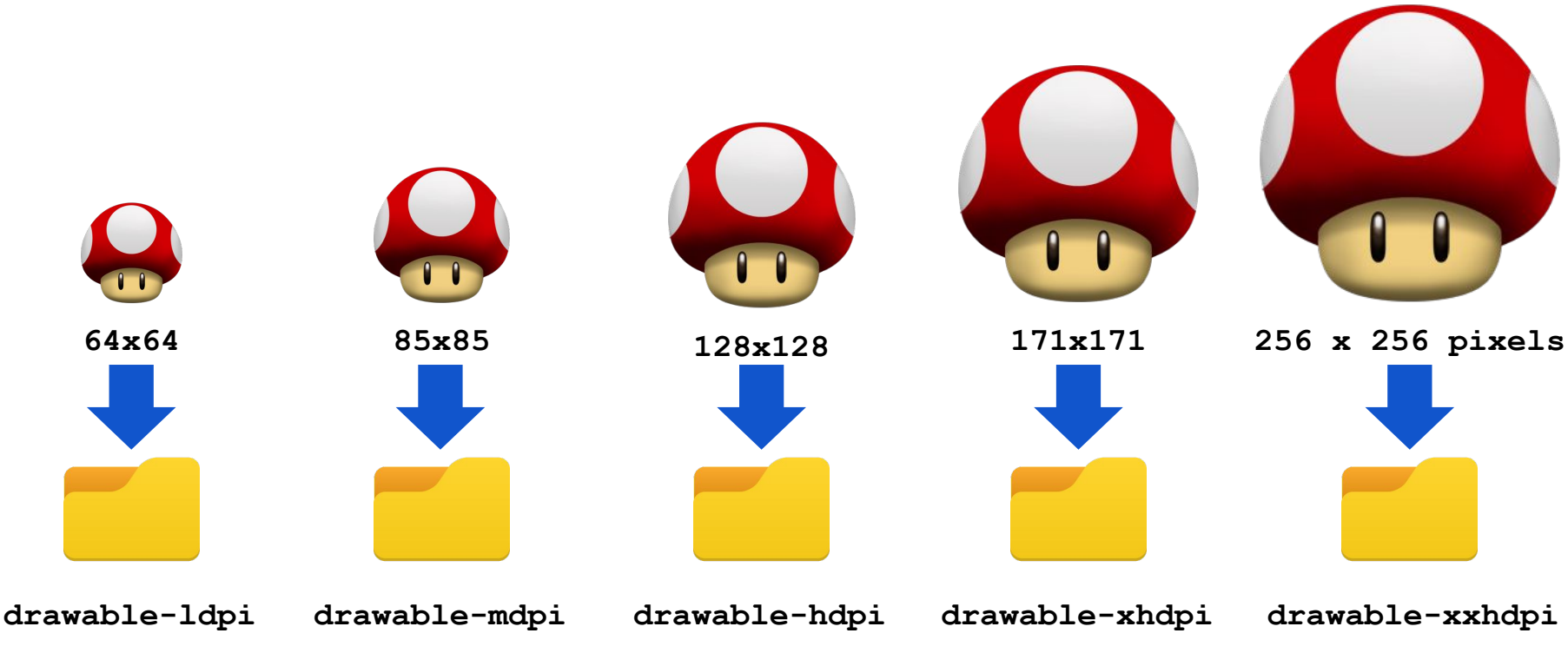
1

Para configurar imagens de diferentes densidades no projeto, crie os diretórios **`drawable-ldpi`**, **`drawable-mdpi`**, **`drawable-hdpi`**, **`drawable-xhdpi`** e **`drawable-xxhdpi`** dentro da pasta **`app/src/main/res`**



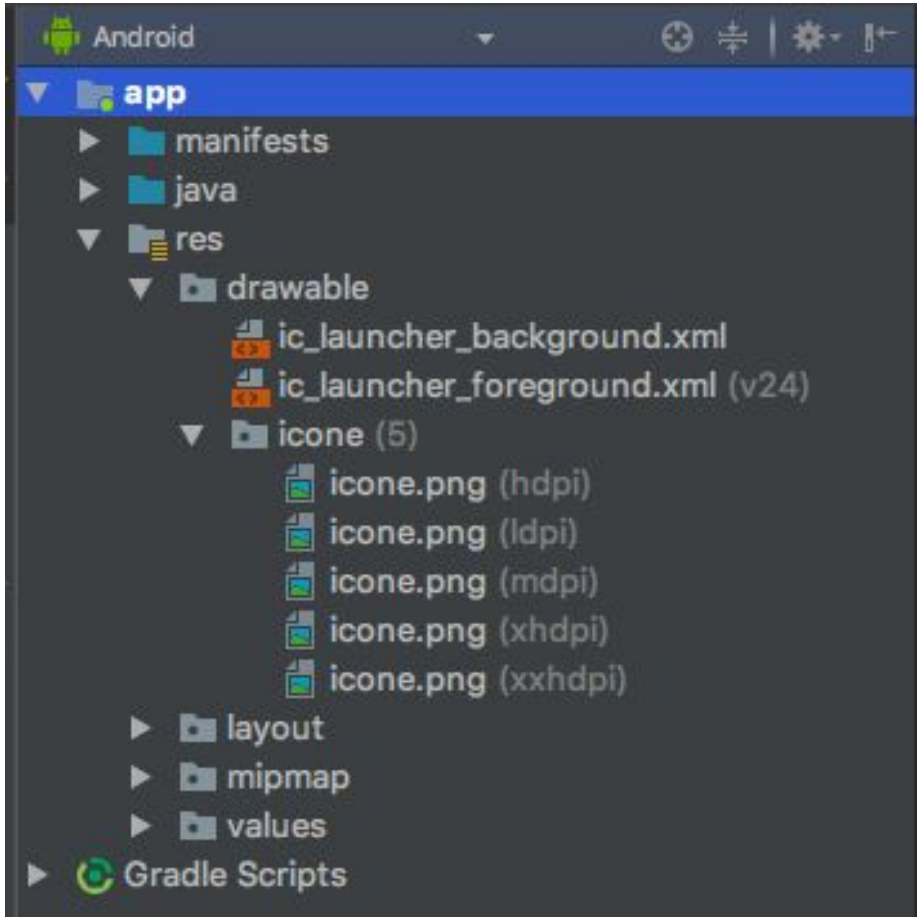
Configuração do projeto com imagens distintas

Em seguida, salve os arquivos de tamanhos diferentes dentro de cada uma destas pastas. Veja a representação a seguir.



Configuração do projeto com imagens distintas

Ao final do processo, o Android Studio vai exibir algo semelhante à imagem ao lado.

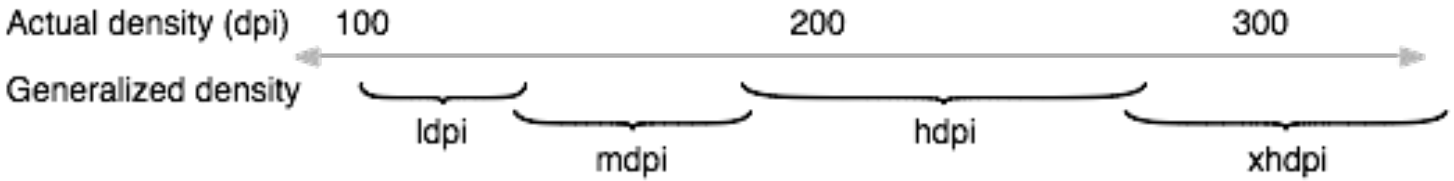


Configuração do projeto com imagens distintas

O código do layout continua idêntico! O *ImageView* deve apontar para o recurso da pasta *drawable* e o Android será capaz de obter a imagem de densidade mais adequada ao dispositivo.

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icone" />
```

Ele faz isso baseando-se em suas densidades genéricas. Veja um exemplo de como os dispositivos são enquadrados nos grupos de densidades.



Dica Extra

Não há uma medida certa para o tamanho físico da tela em pixels de acordo com a densidade. Há, porém, uma média que podemos considerar ao criar as imagens para nossos aplicativos.

Eu gosto de trabalhar com essa tabela:

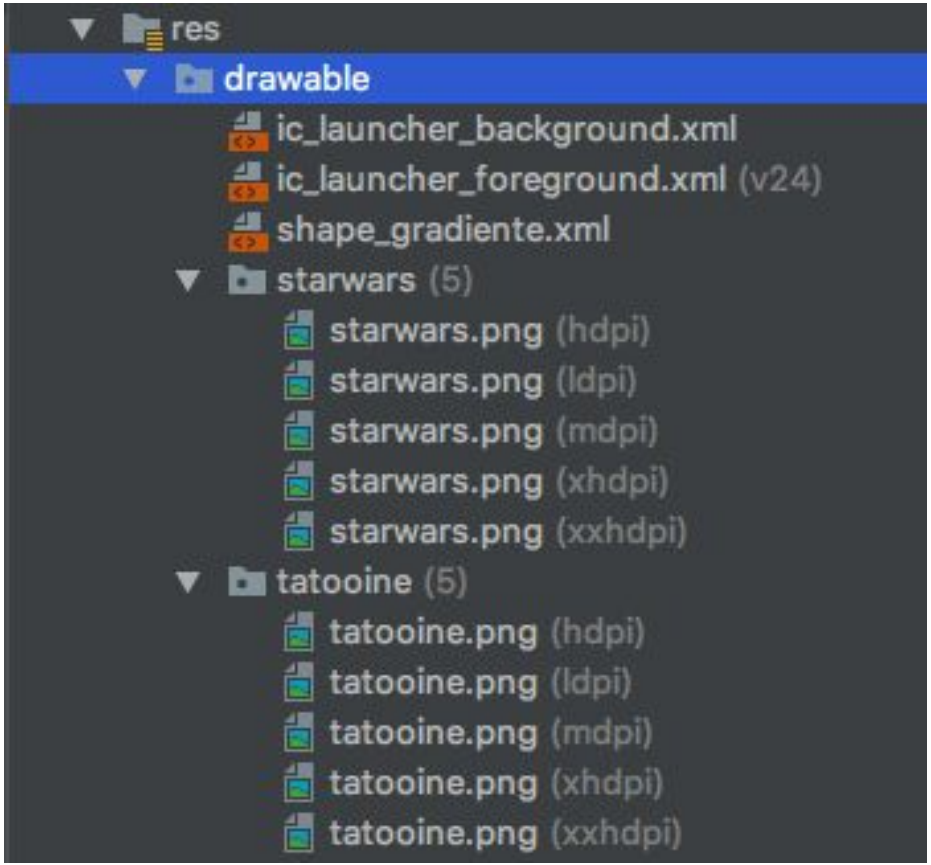
	ldpi	mdpi	hdpi	xhdpi	xxhdpi	xxxhdpi
Background	320*426	320*470	480*640	720*1280	1080*1920	1440*2560

Exercício em Sala

De volta ao estudo de caso!

- Nos *Exercícios da Aula 7* disponível no EAD, há uma pasta imagens/appComEstilo. Dentro desta pasta estão as imagens *tatooine.png* e *starwars.png*, de diferentes densidades. Copie essas imagens para o projeto *AppComEstilo* (em suas respectivas pastas de densidade) e acrescente um *ImageView* no topo do layout.
- Execute em dispositivos com densidades diferentes.

Resolvendo o Exercício



Copie as imagens da pasta *imagens/AppComEstilo* para a pasta *app/src/main/res* do projeto.

Resolvendo o Exercício

Troque o layout para *LinearLayout* e acrescente o *ImageView*. O atributo *layout_width* definido como *match_parent* vai centralizar o conteúdo. Coloque também uma margin de 10dp com o atributo *layout_margin*.

activity_main.xml

```
<?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">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:src="@drawable/starwars" />

</LinearLayout>
```

Resolvendo o Exercício

Executando o projeto em dispositivos com densidades diferentes.

Samsung Galaxy Ace
320x480 mdpi

Samsung Galaxy Nexus
720x1280 xhdpi

LG Nexus 5X
1080x1920 xxhdpi

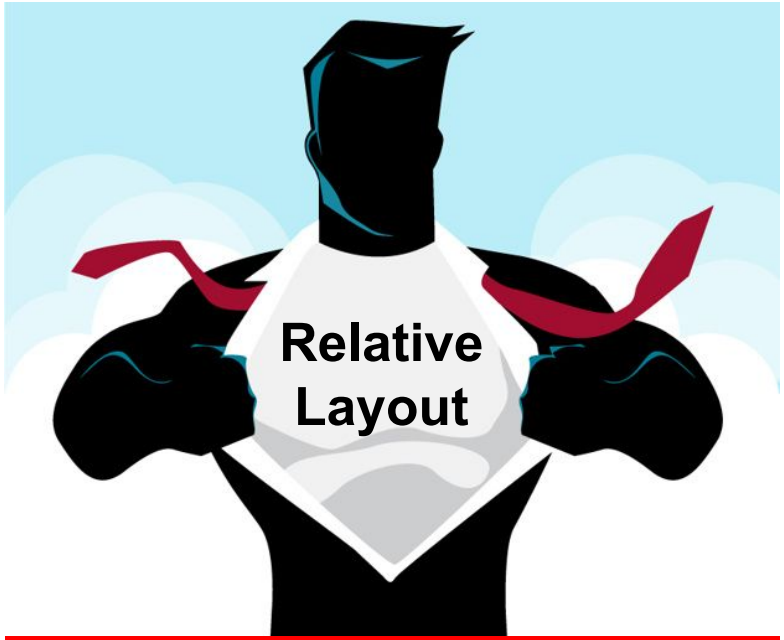
Estudo de Caso

Nesta aula vamos estudar como estilizar as **Views** da aplicação. Para isso, criaremos uma tela estática conforme o protótipo ao lado.

Itens necessários:

- ✅ Retirar *ActionBar*
- ✅ Trabalhar com cores e gradiente
- ✅ Imagens
- **Layouts mais sofisticados e aninhados**
- Criação de shapes circulares

Trabalhando com RelativeLayout

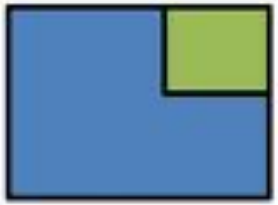


Como vamos posicionar essa imagem no rodapé do dispositivo?

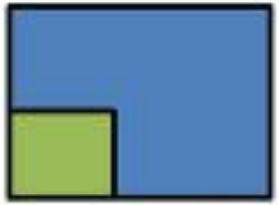
Trabalhando com RelativeLayout

android:layout_alignParentLeft

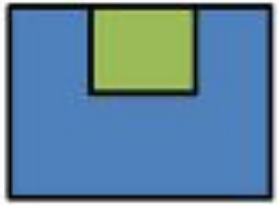
android:layout_alignParentTop



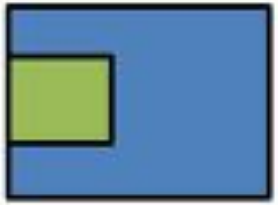
android:layout_alignParentRight



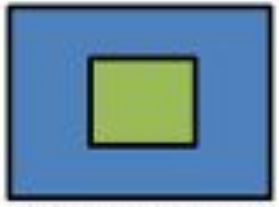
android:layout_alignParentBottom



android:layout_centerHorizontal



android:layout_centerVertical



android:layout_centerInParent

Ao colocar *true* nas propriedades destacadas, o *View* irá se posicionar em relação ao elemento pai.

Trabalhando com RelativeLayout

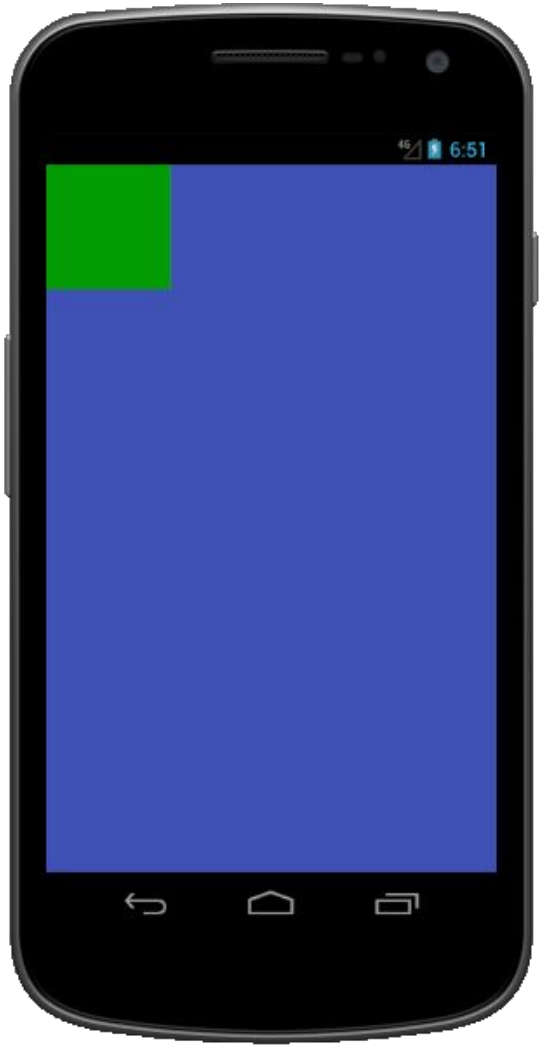
Veja um exemplo em que colocamos a propriedade *layout_alignParentLeft* com o valor *true*.
O elemento filho alinha-se em relação ao pai de acordo com essas propriedades (que por default, são *false*).

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/quadrado"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```



Trabalhando com RelativeLayout

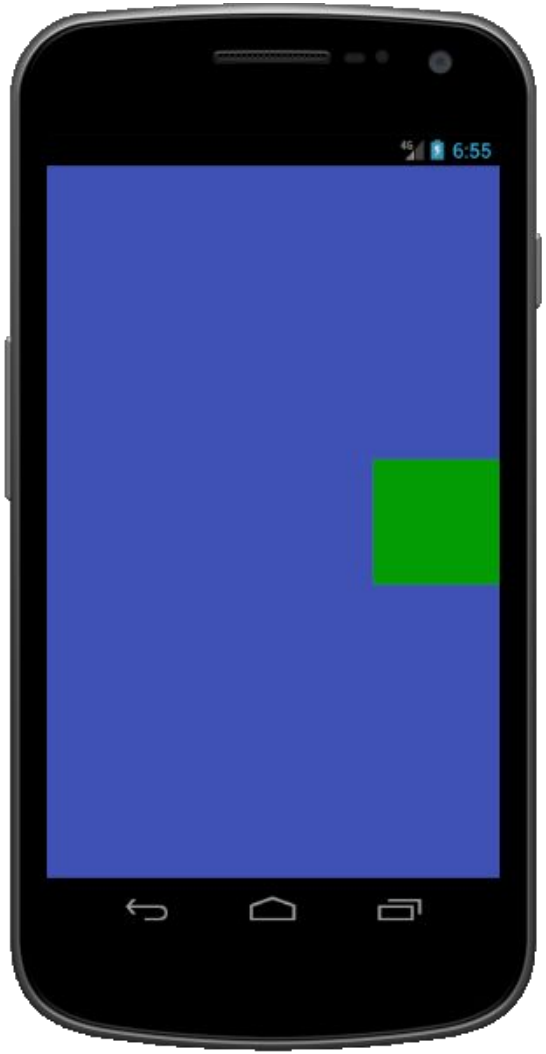
Para alcançar posições específicas em relação ao elemento pai, podemos ainda aplicar essas propriedades conjuntamente, como no exemplo abaixo.

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/quadrado"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"  />

</RelativeLayout>
```



Exercício em Sala

Altere o *AppComEstilo* para trabalhar com *RelativeLayout*, observando:

- Deve-se posicionar o *@drawable/starwars* centralizado no topo (com margen de topo de 10dp).
- Deve-se posicionar o *@drawable/tatooine* ocupando todo o width da parte de baixo da tela
- Execute o projeto em dispositivos com diferentes densidades.

Atenção: como a densidade do aparelho é ligeiramente diferente da densidade genérica do Android, para forçar uma imagem a ocupar toda a área disponível, utilize a propriedade `android:adjustViewBounds`
Use esta propriedade no ImageView de *@drawable/tatooine*

Resolvendo o Exercício

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:src="@drawable/starwars" />

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:adjustViewBounds="true"
        android:src="@drawable/tatooine" />

</RelativeLayout>
```

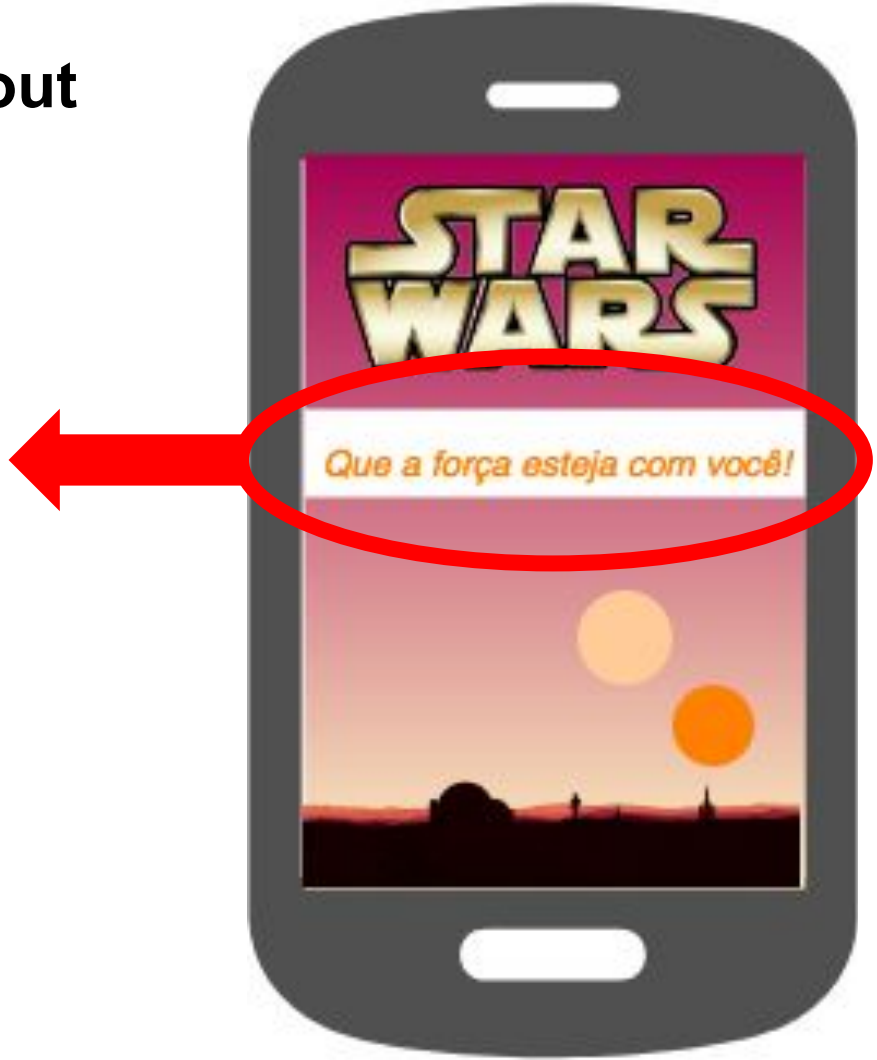
Samsung Galaxy Ace
320x480 mdpi

Samsung Galaxy Nexus
720x1280 xhdpi

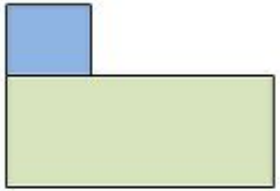
LG Nexus 5X
1080x1920 xxhdpi

Trabalhando com RelativeLayout

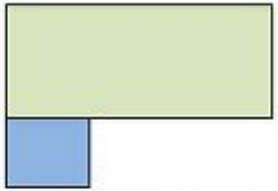
Como este *TextView* ficará posicionado em relação ao pai?



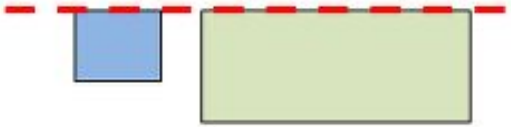
Trabalhando com RelativeLayout



android:layout_above="base"



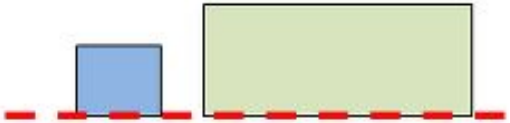
android:layout_below="base"



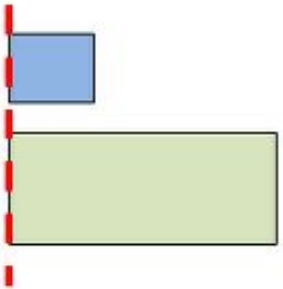
android:layout_alignTop="base"

android:layout_toLeftOf="base"

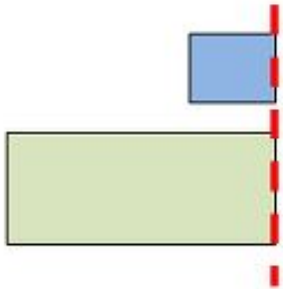
android:layout_toRightOf="base"



android:layout_alignBottom="base"



android:layout_alignLeft="base"



android:layout_alignRight="base"

Um *View* também pode estar posicionado com relação a outro *View*. Geralmente são necessários 2 ou mais atributos para um posicionamento adequado.

Trabalhando com RelativeLayout

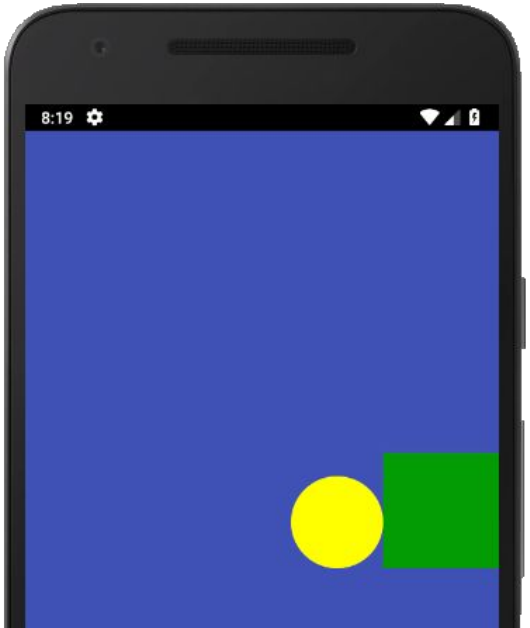
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/quadrado"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:id="@+id/meuQuadrado" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/circulo"
        android:layout_toLeftOf="@id/meuQuadrado"
        android:layout_alignBottom="@id/meuQuadrado" />

</RelativeLayout>
```

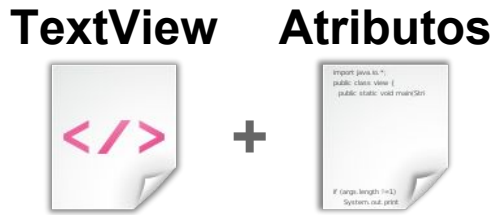
Veja que o *View* que se posiciona em relação ao outro *View* aponta para o ID (círculo apontando para quadrado).



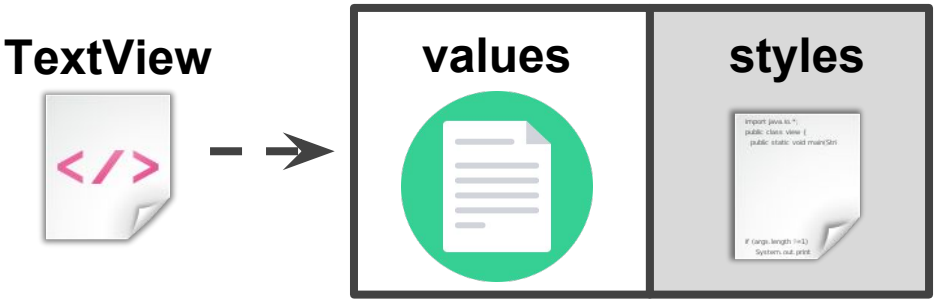
Como vamos estilizar este TextView?

Há duas formas:

⇒ Estilizando direto no *TextView*



⇒ Criando um estilo no arquivo *styles.xml* e importando este tema no *TextView*



Como vamos estilizar este Te

Há duas formas:

⇒ **Estilizando direto no *Text***

TextView **Atr**

⇒ **Criand** **ivo *styles.xml* e importando e** ***tView***

Tex **styles**

⇒ Estilizando direto no *TextView*

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/titulo_starwars"
    android:text="@string/mensagem_legal"
    android:layout_marginTop="20dp"

    android:background="@color/branco"
    android:textColor="@color/laranja"
    android:textStyle="bold"
    android:textSize="22sp"
    android:gravity="center"
    android:padding="5dp" />
```

```
<string name="mensagem_legal">
    Que a força esteja com você!
</string>
```

Lembre-se de criar uma *string* com o texto "Que a força esteja com você!" e referenciá-la no *TextView*

⇒ Criando um estilo no arquivo *styles.xml* e importando este tema no *TextView*

```
<string name="mensagem_legal">
    Que a força esteja com você!
</string>
```

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/fonte_legal"
    android:layout_below="@id/titulo_starwars"
    android:text="@string/mensagem_legal"
    android:layout_marginTop="20dp" />
```

```
<style name="fonte_legal">
    <item name="android:background">@color/branco</item>
    <item name="android:textColor">@color/laranja</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textSize">22sp</item>
    <item name="android:gravity">center</item>
    <item name="android:padding">5dp</item>
</style>
```

Exercício em Sala

Altere o *AppComEstilo* para incluir o *TextView* com o texto "Que a força esteja com você!", observando os seguintes detalhes:

- O texto deve estar no arquivo *res/values/strings.xml*
- O tema deve estar no arquivo *res/values/styles.xml*
- Execute em dispositivos de densidade diferentes

Samsung Galaxy Ace
320x480 mdpi

Samsung Galaxy Nexus
720x1280 xhdpi

LG Nexus 5X
1080x1920 xxhdpi

Estudo de Caso

Nesta aula vamos estudar como estilizar as **Views** da aplicação. Para isso, criaremos uma tela estática conforme o protótipo ao lado.

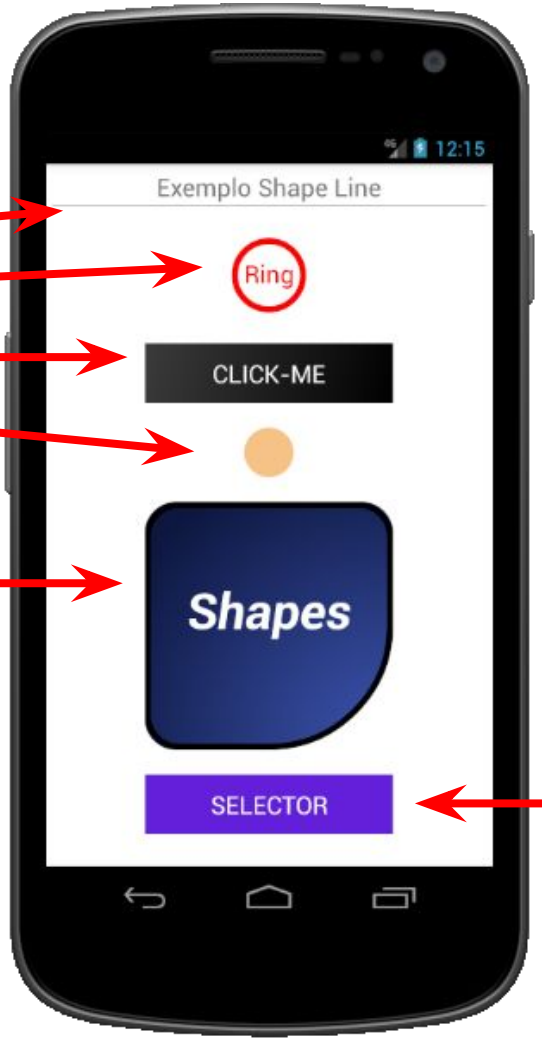
Itens necessários:

- ✅ Retirar *ActionBar*
- ✅ Trabalhar com cores e gradiente
- ✅ Imagens
- ✅ Layouts mais sofisticados e aninhados
- **Criação de shapes circulares**

Trabalhando shapes

```
<shape>
  android:shape="line"
  android:shape="ring"
  android:shape="rectangle"
  android:shape="oval"
```

```
<layer-list>
  <item>
    <shape>
```




```
<selector>
  <item>
    <shape>
```

Trabalhando com shape line

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="line">
    <stroke android:color="@color/cinza" />
</shape>
```

```xml
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Exemplo Shape Line"
    android:textSize="20sp"
    android:textAlignment="center" />
<View
    android:layout_width="match_parent"
    android:layout_height="3dp"
    android:background="@drawable/exemplo_line" />
```

Trabalhando com shape ring

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="ring"
    android:thickness="5dp"
    android:innerRadius="25dp"
    android:useLevel="false">
    <solid android:color="@color/vermelho" />
</shape>
```

```xml
<TextView
    android:layout_width="match_parent"
    android:layout_height="70dp"
    android:layout_marginTop="20dp"
    android:paddingTop="20dp"
    android:text="Ring"
    android:textAlignment="center"
    android:textColor="@color/vermelho"
    android:textSize="20sp"
    android:background="@drawable/exemplo_ring" />
```

Trabalhando com shape rectangle

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <gradient
        android:startColor="@color/cinza"
        android:endColor="@color/preto"
        android:angle="315" />
</shape>
```

```xml
<Button
    android:layout_marginTop="20dp"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:text="Click-me"
    android:textColor="@color/branco"
    android:layout_gravity="center"
    android:background="@drawable/exemplo_rectangle" />
```

Trabalhando com shape oval

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="@color/laranja" />
    <size android:width="40dp" android:height="40dp" />
</shape>
```

```xml
<View
    android:layout_marginTop="20dp"
    android:layout_gravity="center"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:background="@drawable/exemplo_oval" />
```

layer-list

```xml
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape android:shape="rectangle">
            <size android:width="200dp" android:height="200dp" />
            <gradient
                android:type="radial"
                android:startColor="@color/azul1"
                android:endColor="@color/azul2"
                android:gradientRadius="500"
                android:centerX="1"
                android:centerY="1" />
            <corners
                android:topLeftRadius="25dp"
                android:topRightRadius="25dp"
                android:bottomRightRadius="100dp"
                android:bottomLeftRadius="25dp" />
        </shape>
    </item>
    <item>
        <shape android:shape="rectangle">
            <size android:width="200dp" android:heig
            <corners
                android:topLeftRadius="25dp"
                android:topRightRadius="25dp"
                android:bottomRightRadius="100dp"
                android:bottomLeftRadius="25dp" />
            <stroke android:color="@color/preto" and
        </shape>
    </item>
</layer-list>
```


```xml
<!-- Dois ou mais shapes juntos -->
<TextView
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_marginTop="20dp"
    android:paddingTop="60dp"
    android:textStyle="bold|italic"
    android:text="Shapes"
    android:textAlignment="center"
    android:layout_gravity="center"
    android:textColor="@color/branco"
    android:textSize="40sp"
    android:background="@drawable/exemplo_layer_list" />
```

Dica Extra ⇒ ao construir shapes, use o *Preview*

exemplo_layer_list.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape android:shape="rectangle">
            <size android:width="200dp" android:height="200dp" />
            <gradient
                android:type="radial"
                android:startColor="@color/azul1"
                android:endColor="@color/azul2"
                android:gradientRadius="500"
                android:centerX="1"
                android:centerY="1" />
            <corners
                android:topLeftRadius="25dp"
                android:topRightRadius="25dp"
                android:bottomRightRadius="100dp"
                android:bottomLeftRadius="25dp" />
        </shape>
    </item>
    <item>
        <shape android:shape="rectangle">
            <size android:width="200dp" android:height="200dp" />
            <corners
                android:topLeftRadius="25dp"
                android:topRightRadius="25dp"
                android:bottomRightRadius="100dp"
                android:bottomLeftRadius="25dp" />
            <stroke android:color="@color/preto" android:width="5dp" />
        </shape>
    </item>
</layer-list>
```

Preview

Gradle

Preview

```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true">
        <shape>
            <solid android:color="@color/roxo3" />
        </shape>
    </item>
    <item android:state_focused="true">
        <shape>
            <solid android:color="@color/roxo2" />
        </shape>
    </item>
    <!-- Valor default -->
    <item>
        <shape>
            <solid android:color="@color/roxo1" />
        </shape>
    </item>
</selector>
```

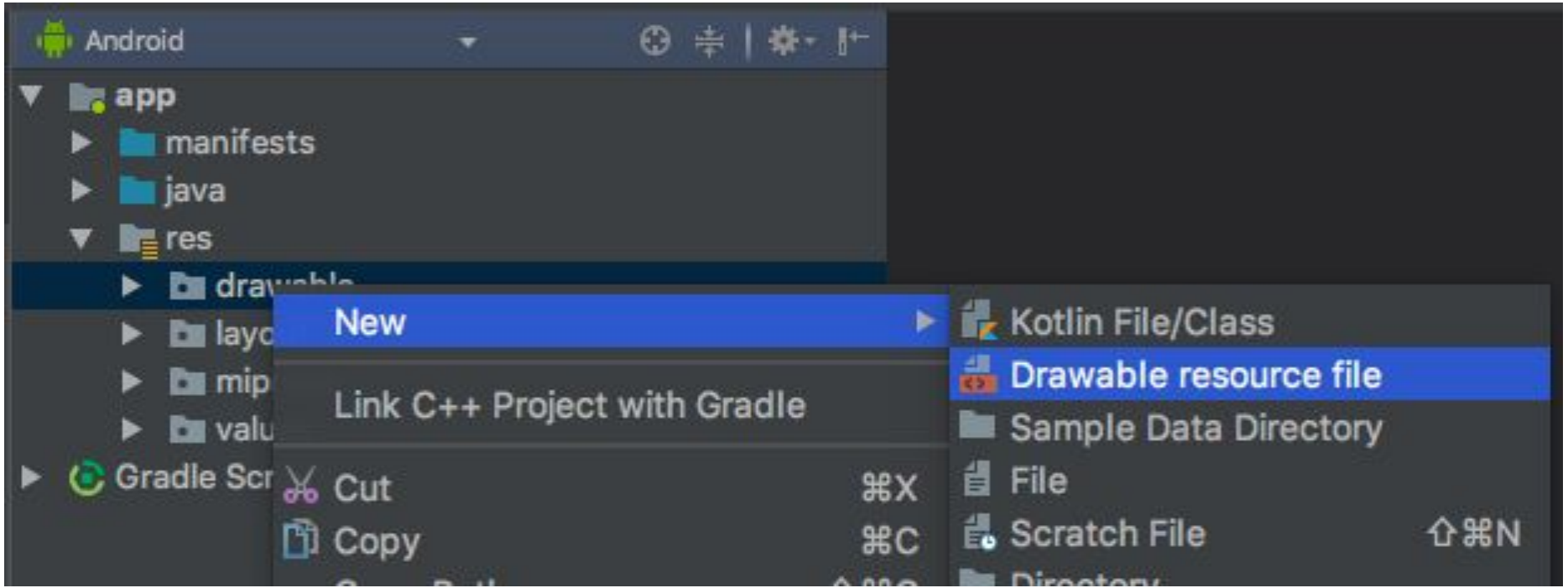
selector


```xml
<!-- Selector -->
<Button
    android:layout_marginTop="20dp"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:text="Selector"
    android:textColor="@color/branco"
    android:layout_gravity="center"
    android:background="@drawable/exemplo_selector" />
```

Veja: https://developer.android.com/guide/topics/resources/color-list-resource

Criando e posicionando o sol laranja

O shapes ficam na pasta *drawable*. Para criá-los, devemos utilizar o seguinte menu:



Criando e posicionando o sol laranja

File name: sol_laranja

Source set: main

Directory name: drawable

Nome

Available qualifiers:

- Country Code
- Network Code
- Locale
- Layout Direction
- Smallest Screen Width
- Screen Width
- Screen Height
- Size
- Ratio
- Orientation
- UI Mode
- Night Mode
- Density
- Touch Screen

Qualificadores
É possível incluir classificadores, que na prática são subpastas para informar ao Android alguma especificidade do drawable.

Cancel OK

Criando e posicionando o sol laranja

sol_laranja.xml

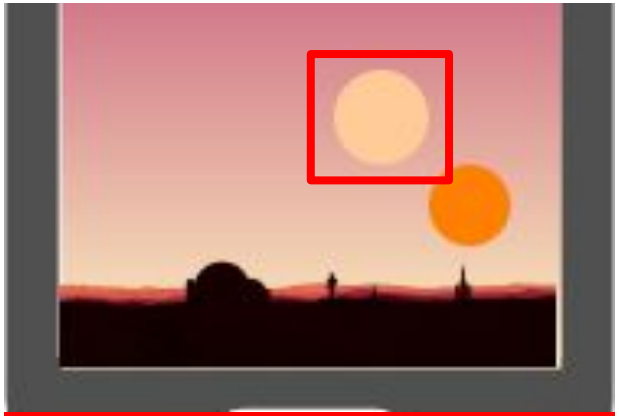
```xml
<?xml version="1.0" encoding="utf-8"?>
<shape android:shape="oval"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@color/laranja"/>
</shape>
```

```xml
<View
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:background="@drawable/sol_laranja"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_marginRight="30dp"
    android:layout_marginBottom="100dp" />
```

Protótipo

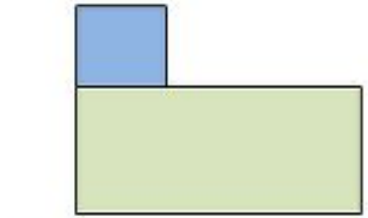
Samsung Galaxy Nexus
720x1280 xhdpi

Criando e posicionando o sol amarelo



Como vamos posicionar o sol amarelo?

Podemos posicioná-lo através das propriedades a seguir, levando em consideração o posicionamento do sol laranja.



android:layout_above="base"

android:layout_toLeftOf="base"

Criando e posicionando o sol amarelo

Para isso precisamos adicionar um ID no sol laranja.

```
<View
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:background="@drawable/sol_laranja"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_marginRight="30dp"
    android:layout_marginBottom="100dp"
    android:id="@+id/view_sol_laranja" />
```

Criando e posicionando o sol amarelo

sol_amarelo.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape android:shape="oval"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@color/amarelo"/>
</shape>
```

```xml
<View
    android:layout_width="70dp"
    android:layout_height="70dp"
    android:background="@drawable/sol_amarelo"
    android:layout_above="@id/view_sol_laranja"
    android:layout_toLeftOf="@id/view_sol_laranja" />
```

Protótipo

Samsung Galaxy Nexus
720x1280 xhdpi

Samsung Galaxy Ace
320x480 mdpi

Samsung Galaxy Nexus
720x1280 xhdpi

LG Nexus 5X
1080x1920 xxhdpi

Exercício em Sala

Conclua o projeto *AppComEstilo* para incluir todos os recursos apresentados no protótipo (texto, imagens, sol laranja e sol amarelo).

Em seguida, execute-o em seu celular.

Obrigado!