Programação para dispositivos móveis

Aula 7: Estilos e Temas

INTRODUÇÃO

É essencial projetar interfaces gráficas de alta qualidade em aplicativos móveis. Esta aula visa apresentar desenvolvimento de telas implementando os conceitos de estilo e tema, bem como a configuração de dimensões, cores, strings e valores em Android.

OBJETIVOS

Descrever a configuração de dimensões, de cores, de strings, de valores, de estilos e de temas em Android.

RECURSOS DO ANDROID

O desenvolvimento de interfaces para dispositivos móveis de qualidade é um fator determinante para o sucesso de qualquer aplicativo.

Conceitos como usabilidade, User Interface(UI) e User Experience(UX) nunca estiveram tão em pauta quanto nos dias de hoje. Porém, operacionalizar este desenvolvimento não é algo tão trivial assim.

Para tanto, o Android nos oferece uma série de recursos que facilitam não só o desenvolvimento, como a manutenção dessas interfaces.

Entre eles podemos destacar o tratamento, em arquivo XML, de:

• Dimensões;
• Cores;
• Strings;
• Valores;
• Estilos;
• Temas.

DIMENSÕES

Em grande parte de nossos exemplos, configuramos as dimensões de nossos componentes usando os parâmetros fill_parent, match_parent e wrap_content.

Também podemos configurar através de um número seguido de uma unidade de medida, como, por exemplo, 10sp.

Abaixo é demonstrada as **unidades de medidas suportadas pelo Android:**

DP

É muito usada no desenvolvimento de layouts. Esta unidade abstrata baseia-se na densidade física da tela.

A proporção de dp pode mudar com a densidade da tela, mas não a sua proporção.

O conceito de dp é bastante complexo. Para maiores informações consulte o link:
https://developer.android.com/guide/practices/screens_support.html (https://developer.android.com/guide/practices/screens_support.html)

SP

É muito usada para especificar tamanho da fonte em um aplicativo.

Esta unidade é bastante similar à unidade de medida dp, mas é dimensionada de acordo com a preferência do tamanho de fonte do usuário.

PX

É uma unidade de medida não recomendada, salvo em casos bem específicos.

Aqui um pixel corresponde a um pixel da tela.

Ao desenvolvermos para um dispositivo específico, pode funcionar bem, mas em outros, devido à diferença de tamanho e densidade das telas, o resultado normalmente não é o esperado.

PT

Pelo mesmo motivo da px, não é uma unidade de medida recomendada, pois tem como base o tamanho físico da tela.

Um pt corresponde a 1/72 polegadas.

MM

Assim como px e pt, também não é recomendada, pois trabalha com o tamanho físico da tela em milímetros.

IN

Assim como px, pt e mm, também não é recomendada, pois trabalha com o tamanho físico da tela em polegadas.
O Android permite que tratemos as medidas de qualquer componente via código Java, porém devemos defini-las em um arquivo chamado *dimens.xml*.

Você poderá encontrá-lo na pasta *res/values* de nosso projeto Android.

O exemplo abaixo demonstra o conteúdo do arquivo *res/values* de um dos nossos exemplos.

EXEMPLO

Nele, estamos definindo três dimensões, sendo elas margem_horizontal(16dp), margem_vertical(16dp) e tamanho_textView(100dp):

```
<resources>
  <dimen name="margem_horizontal">16dp</dimen>
  <dimen name="margem_vertical">16dp</dimen>
  <dimen name="tamanho_textView">100dp</dimen>
</resources>
```

Em nosso código Java, ao invés de definirmos a propriedade android:layout_width diretamente com o valor 100dp, estamos obtendo esta dimensão configurada no arquivo *res/values*, através da dimensão *tamanho_view*, que se encontra destacada em vermelho.

android:layout_width="@dimen/ tamanho_textView"

CORES

Assim como as dimensões, também podemos definir a cor diretamente em nossas propriedades ou, como recomendado, em um arquivo chamado **colors.xml**. Este também se localiza na pasta *res/values*.

Em sua documentação oficial, a Google recomenda o uso da paleta de cores que pode ser encontrada no link:
http://www.google.com/design/spec/style/color.html (glossário)

O **padrão RGB (Red/Green/Blue)** é usado para definir cores nas telas Android.

Para implementá-las, usamos a notação hexadecimal que é baseado no sistema de contagem com base 16.

Precisamos de seis caracteres precedidos pelo carater # para representar a cor, onde os dois primeiros são referentes à cor vermelha, os dois seguintes à cor verde e os dois últimos à cor azul.

Veja o exemplo abaixo:

Como já discutimos e exemplificamos em várias aulas anteriores o componente Button, abaixo apenas ilustramos um trecho de código que configura as cores de fundo e do texto de um Botão.

```
<Button
      android:id="@+id/botao"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:background="#00F"
      android:textColor="#FFFFFF"
      android:text="Exemplo" />
```

Podemos usar a representação reduzida da cor em hexadecimal.

Quando é representada por dois dígitos iguais para cada cor, podemos reduzir para três dígitos apenas.

Observe que definimos a *android:background="#00F"* com apenas três caracteres. Nesse caso, equivale ao *android:background="#0000FF"*.

O exemplo abaixo demostra a configuração das cores azul e branco no arquivo colors.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="azul">#00F</color>
   <color name="branco">#FFFFFF</color>
</resources>
```

Nosso mesmo exemplo do botão agora ficaria assim:

```
<Button
      android:id="@+id/botao"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:background="@color/azul"
      android:textColor="@color/branco"
      android:text="Exemplo" />
```

STRINGS

As Strings também devem ser definidas em um arquivo chamado ***string.xml***, localizado na pasta ***res/values***. Isso facilita, inclusive, a internacionalização de nosso aplicativo, pois podemos substituir o arquivo pelo que possui o idioma desejado.

Exemplo 1

```
<?xml version="1.0"
encoding="utf-8"?>
<resources>
    <string name="saudacao">Seja
bem-vindo!</string>
</resources>
```

Exemplo 2

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

VALORES

Além de strings, podemos também definir valores constantes para vários tipos.

Para isso, precisamos criar o xml dentro da pasta res/values, atribuindo um nome para o arquivo.

Entre os mais comuns, podemos destacar:

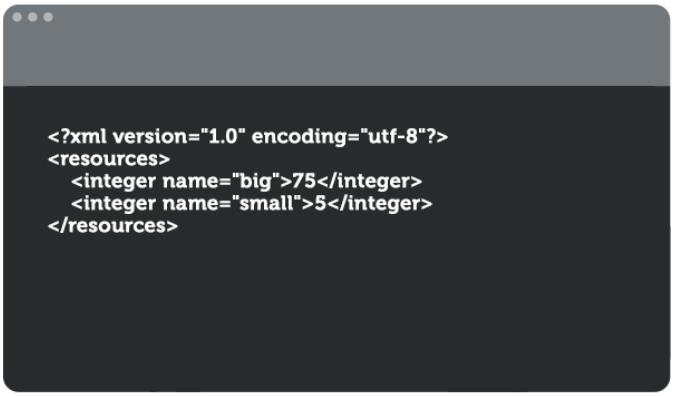
Booleano

Valores booleanos:
Exemplo bools.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="screen_small">true</bool>
    <bool name="adjust_view_bounds">true</bool>
</resources>
```

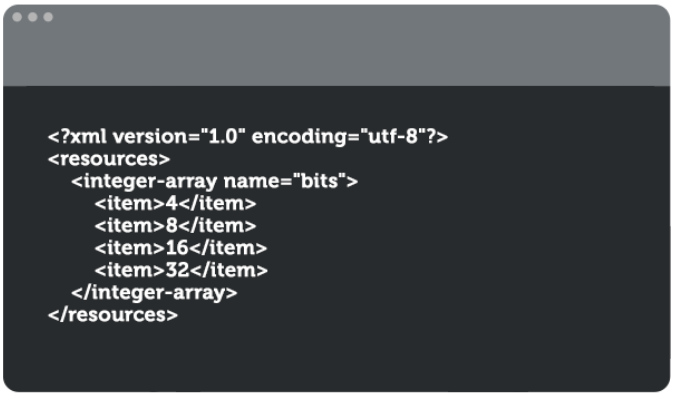
Inteiro

Valores Inteiros.
Exemplo integers.xml:



Arrays de Inteiros

Vetor de inteiros.
Exemplo integers.xml:



Array tipado

Vetor com tipos misturados.
Exemplo:

Estilos - Conceito de estilo em Android

Um estilo em Android corresponde a um conjunto de propriedades que especificam a aparência e o formato para uma View. Este pode especificar propriedades, tais como altura, preenchimento, cor de fonte, tamanho de fonte, cor de fundo e muito mais.

O estilo é definido em um arquivo XML que é separado do XML que especifica o layout.

O estilo Android permite que sejam separados de acordo com a concepção do conteúdo. Isso pode reduzir a duplicação da especificação de styles, pois podemos efetuar esta definição em um arquivo central, podendo ser aplicado em outras Views de nossa aplicação.
É bastante simples definir um conjunto de estilos. Basta criar um arquivo XML no diretório res/values/. Apenas não podemos esquecer que a tag deve ser a raiz.

O código abaixo define a especificação de um estilo chamado MeuEstilo no arquivo styles.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<style name="MeuEstilo">
   <item name="android:textColor">#FF0000</item>
   <item name="android:textSize">40sp</item>
</style>
</resources>
```

Após definido um estilo, basta associar a um componente.

Veja o código abaixo:

```
<TextView
 android:id="@+id/tv_text"
 style="@style/MeuEstilo"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/hello_world" />
```

A partir de agora, o textView possui a cor de fonte vermelha e o tamanho de texto de 40sp, conforme foi definido no estilo chamado MeuEstilo.

HERANÇA DE ESTILO

O Android permite aplicar o conceito de herança da orientação a objetos para estilos.

Com isso, podemos:

• Fazer uso das propriedades definidas em um estilo pai;
• Modificar ou adicionar novas propriedades;
• Herdar estilos pré-definidos;
• Herdar estilos criados.

É bastante simples implementar a herança de estilo. Para herdar estilos pré-definidos do Android, devemos usar o atributo parent seguido de "@android:style/" + o estilo desejado.

No exemplo abaixo, destacamos em vermelho esse tipo de herança de estilo:

```
<style name="TextoVerde" parent="@android:style/TextAppearance">
<item name="android:textColor">#00FF00</item>
</style>
```

Já para herdar estilos definidos na própria aplicação, não usamos o atributo patent. Apenas usamos o nome do estilo herdado e acrescentamos o "." + o novo nome.

O exemplo abaixo destaca em vermelho o emprego desse tipo de herança:

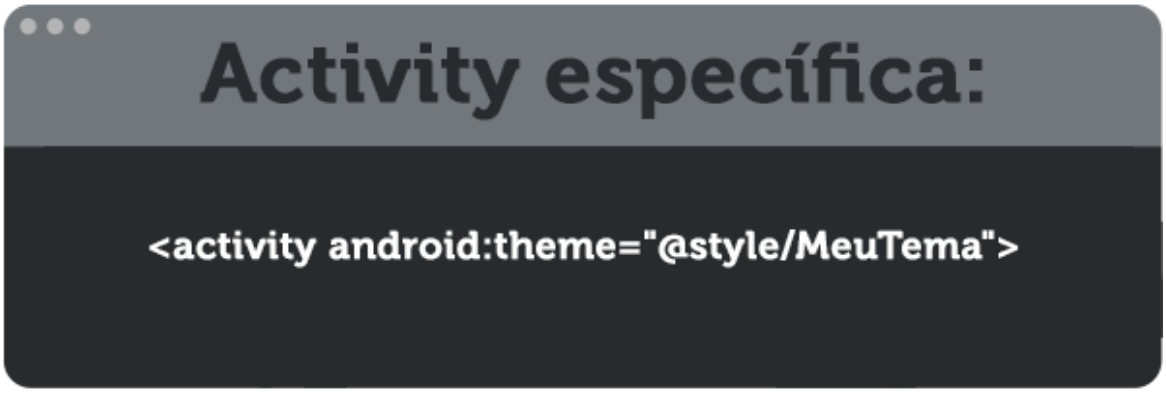
```
<style name="TextoVermelho">
<item name="android:textColor">#FF0000</item>
</style>

<style name="TextoVermelho.Grande">
<item name="android:textSize">40sp</item>
</style>
```

TEMAS

Um tema é um estilo aplicado a uma Activity ou aplicação inteira, ao invés de uma View individual.

Quando um estilo é aplicado como um tema, todas as Views na Activity ou aplicação irão usar todas as propriedades de estilo por ele definidas.

Para definir um tema para aplicação ou uma atividade específica, precisamos editar o arquivo AndroidManisfest.xml, conforme exemplo abaixo:



Saiba mais

, A plataforma Android oferece uma grande coleção de estilos e temas que você pode usar em seus aplicativos.

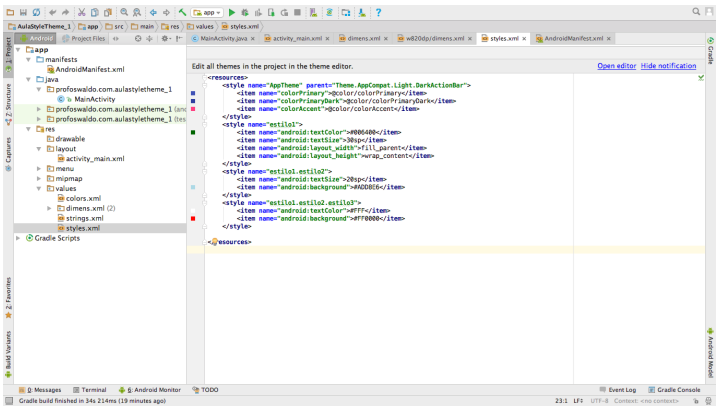
Você pode encontrar uma referência de todos os estilos disponíveis no link:, , • aquihttps://developer.android.com/reference/android/R.style.html (https://developer.android.com/reference/android/R.style.html), , Essa documentação não descreve minuciosamente os estilos e temas. Caso precise de mais detalhes, acesse os seguintes links:, , • https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/res/values/styles.xml (https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/res/values/styles.xml)

• https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/res/values/themes.xml (https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/res/values/themes.xml)

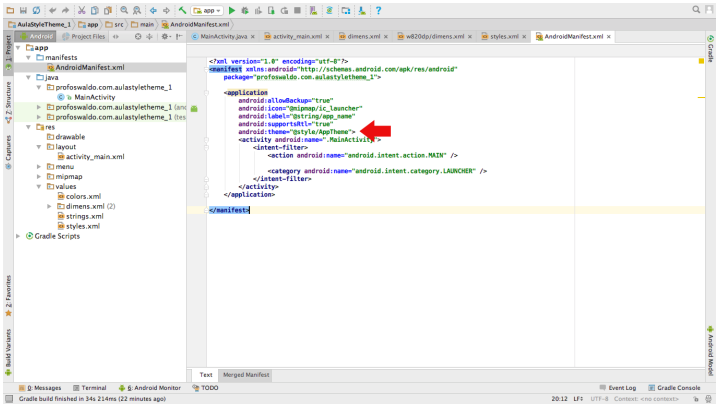
EXEMPLO EMPREGANDO ESTILO E TEMA

O primeiro arquivo que vamos desenvolver é o styles.xml.

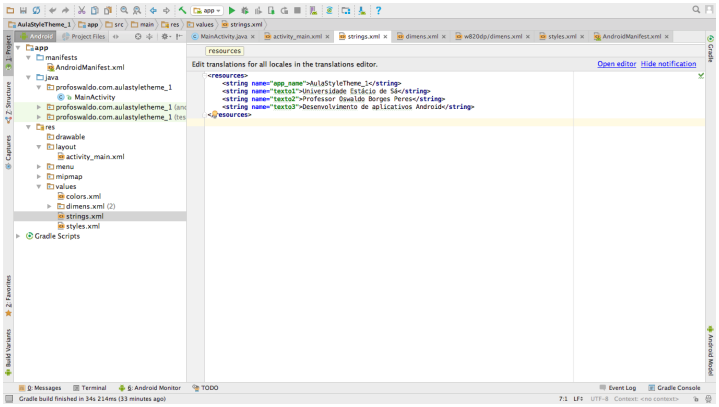
Nele, definiremos vários estilos, como ilustrado na tela abaixo:



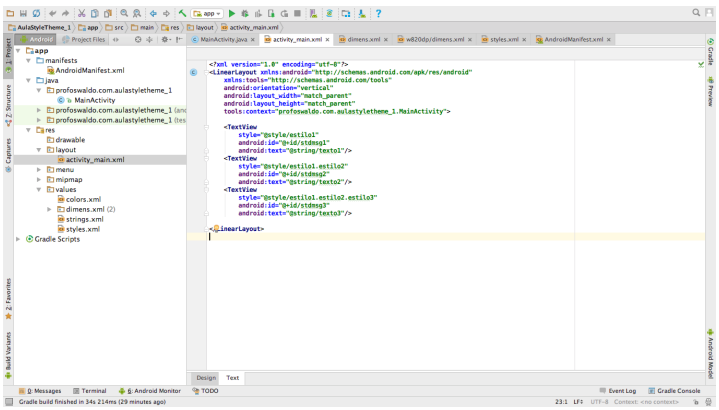
Estamos também definindo o tema de nosso aplicativo no arquivo **AndroidManifest.xml:**



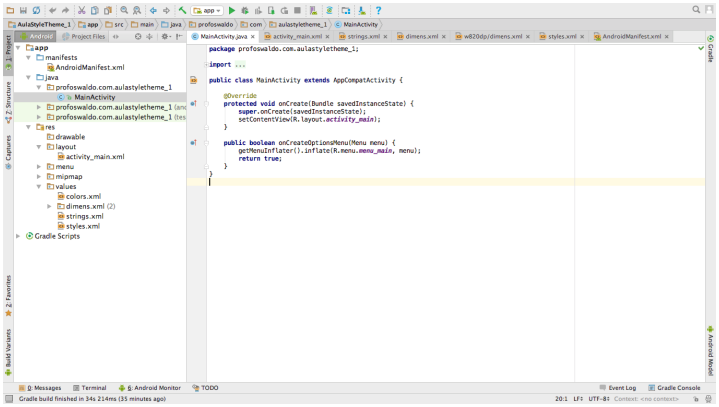
Vamos também definir novas constantes em nosso arquivo **strings.xml:**



Agora estamos implementando o arquivo de layout **activity_main.xml**, que fará uso dos layouts definidos.



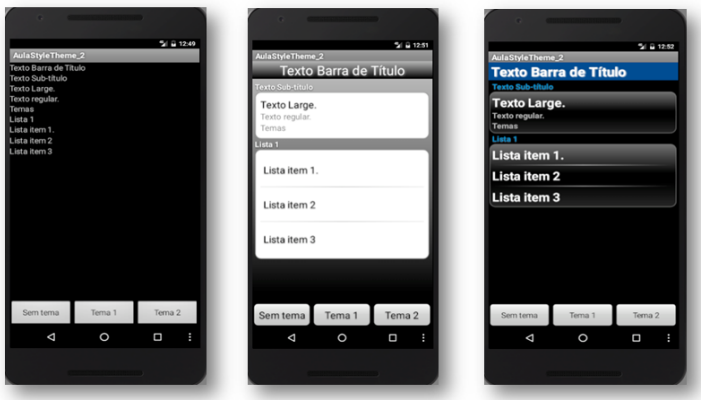
Como nosso objetivo é estilo e temas, nossa **MainActivity.java** é bastante simples:



Execute o aplicativo para ver a tela abaixo:

ATIVIDADE

Observe as imagens:



Desenvolva um pequeno aplicativo que, como demonstrado nas telas acima, ao clicarmos no botão Sem tema, exiba a primeira tela, Tema 1, a segunda, Tema 2, e a terceira. É compulsório o uso de estilos.

Resposta Correta

Glossário