Android
dimens.xml, strings.xml e colors.xml

SUMÁRIO

1. Informativo 3
2. Objetivo 4
3. Pré-Requisitos 5
4. Versão 6
   4.1. minSdkVersion 6
   4.2. targetSdkVersion 6
5. Novo Projeto 7
6. res 11
7. strings.xml 12
8. colors.xml 15
   8.1. Dica 17
9. dimens.xml 19
10. Aplicativo 21
11. Resumo 22
12. Referências 23

1. Informativo

Autor(a): Helena Strada
Data de Criação: jan/2018

2. Objetivo

O objetivo desta apostila é mostrar como utilizar os recursos do Android para manipular os valores de strings, variáveis e dimensões.

3. Pré-Requisitos

Java

- Orientação a Objetos;
- APIs;
- Bibliotecas.

Android

- Estrutura do Projeto;
- XML;
- Activity.

4. Versão

Android Studio 3.0.1

4.1. minSdkVersion

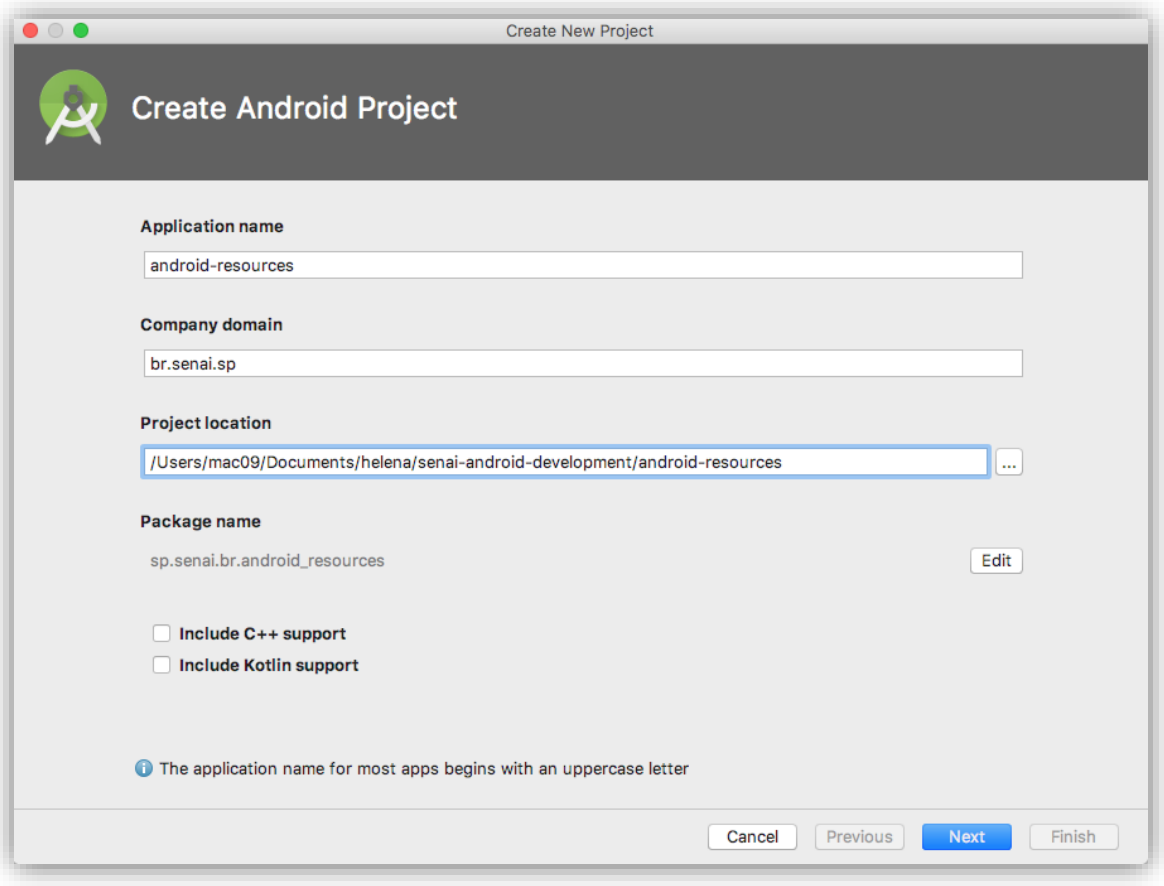
15

4.2. targetSdkVersion

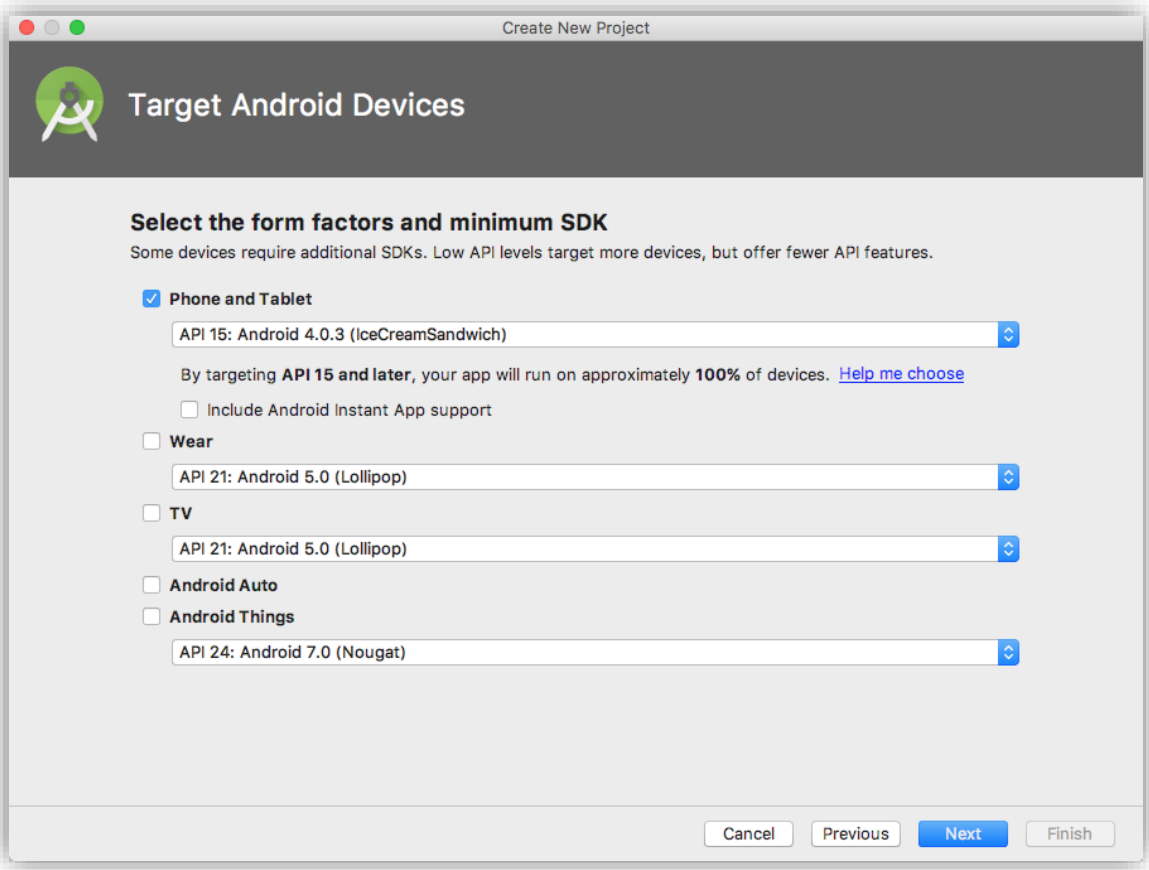
26

5. Novo Projeto

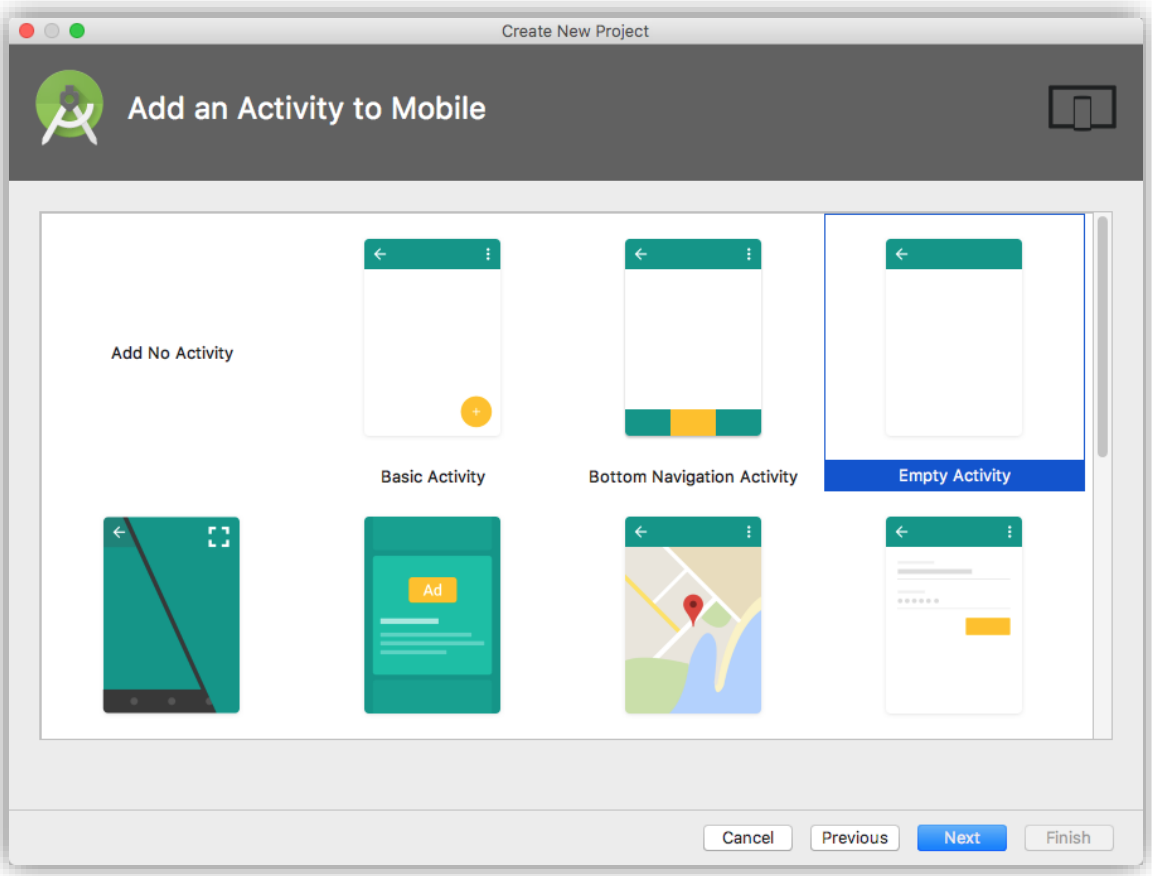
Vamos criar um novo projeto chamado android-resources.



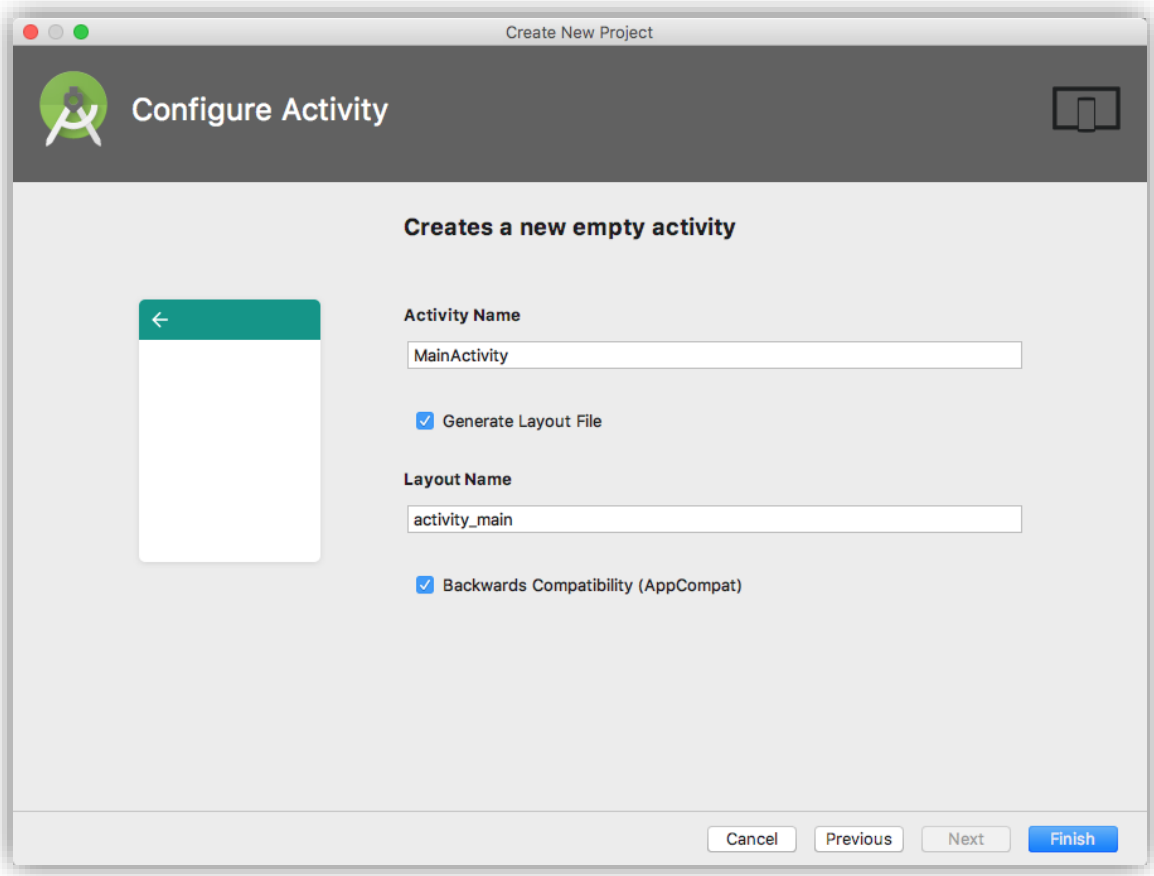
Clicar em ‘Next’.



Clicar em ‘Next’.



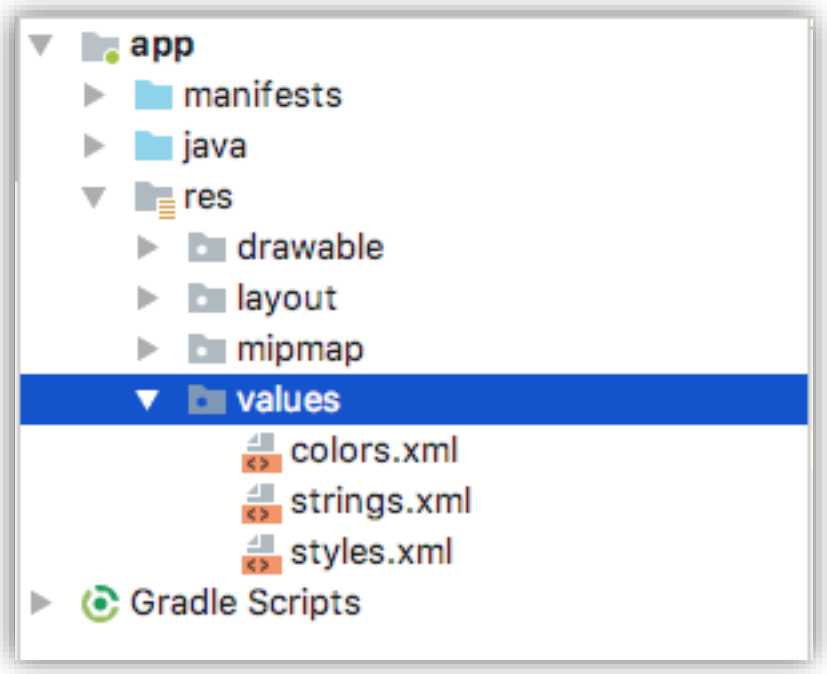
Clicar em ‘Next’.



Clicar em ‘Finish’.

6. res

Os arquivos que iremos alterar estão localizados dentro de: app -> res -> values.



Podemos perceber que os arquivos dimens.xml ainda não foi criado, porém, ao desejarmos adicionar um recurso como dimensão, ele será incluído automaticamente.

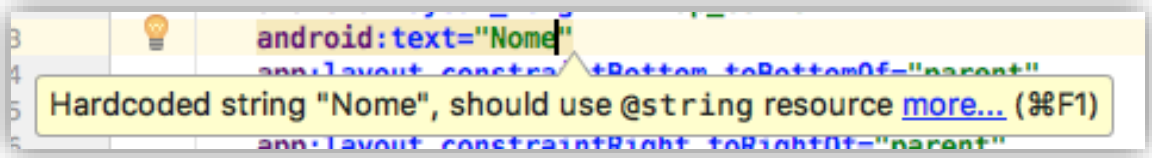
7. strings.xml

O primeiro arquivo que iremos trabalhar será o strings.xml.
Vamos criar dois campos do tipo TextView dentro do nosso CoordinatorLayout.

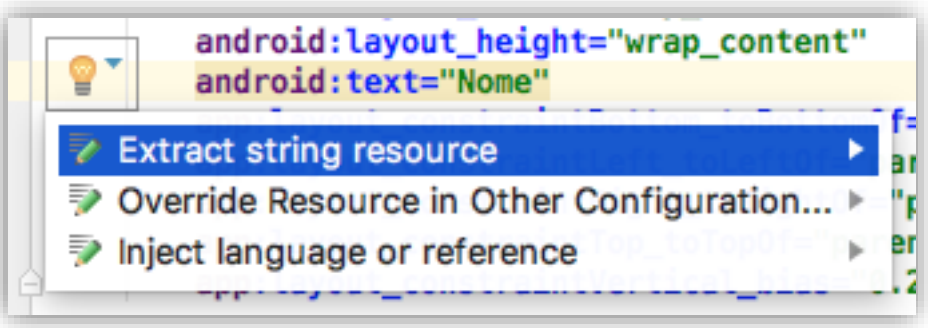
```
<TextView
    android:id="@+id/tvNome"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Nome"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.235" />

<TextView
    android:id="@+id/tvSobrenome"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sobrenome"
    app:layout_constraintStart_toStartOf="@+id/tvNome"
    app:layout_constraintTop_toBottomOf="@+id/tvNome" />
```

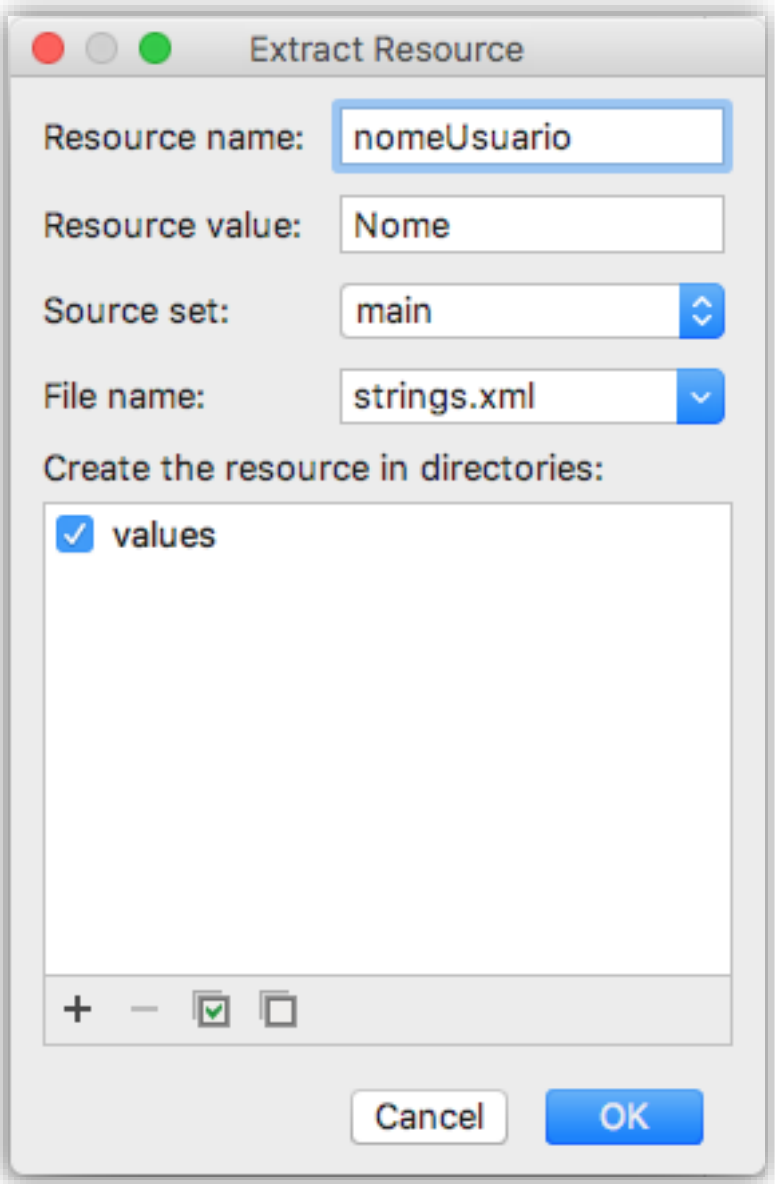
Ao olharmos do lado esquerdo da nossa tela (na lâmpada), ele nos dá como dica, criarmos uma string para não colocarmos direto na tela o que desejamos (Hardcoded).



Ao clicarmos na lâmpada, ele nos fornece como opção criarmos um recurso dentro de strings.xml para a String que desejamos.



Clique na primeira opção: ‘Extract string resource’.

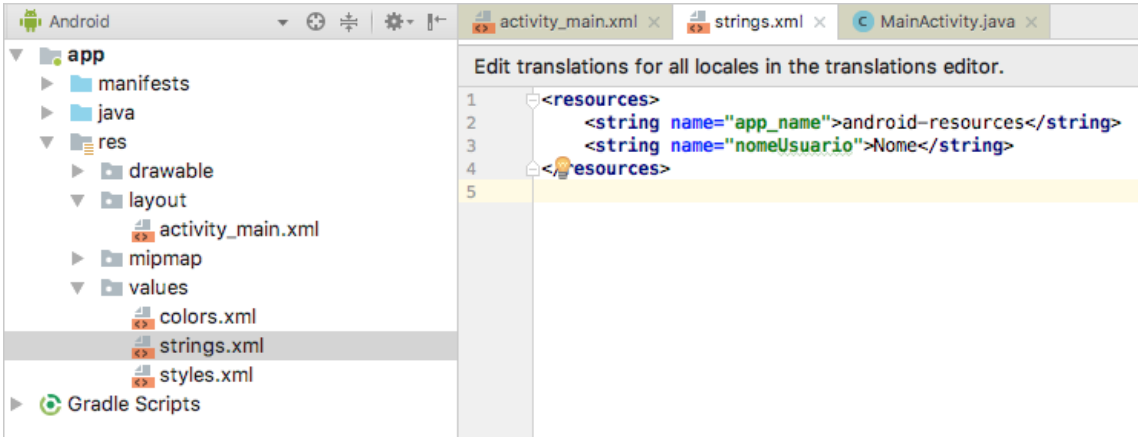


Selecione o nome do ‘Resource Name’ que deseja e clique em ‘Ok’. Veremos em outro exemplo, outra maneira de escrever o nome do recurso. Como observação, não podemos colocar traço (exemplo: nome-usuario).

Ao abrirmos o arquivos strings.xml podemos verificar que a string que criamos agora está dentro deste arquivo.

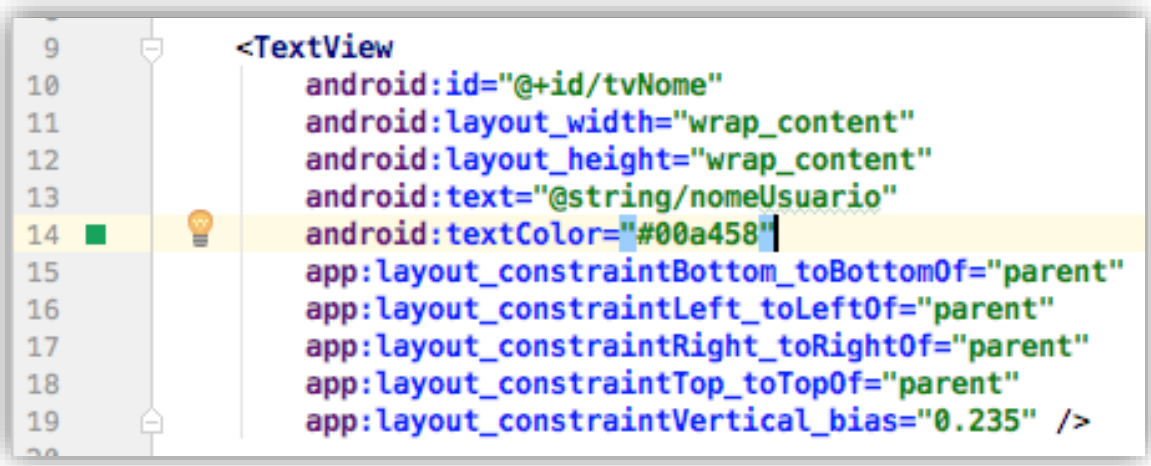
Qual a vantagem dessa abordagem em relação a codarmos diretamente na tela? Imagina que o título da aplicação esteja presente em mais de uma tela da nossa aplicação. Se precisarmos alterar o valor em todas as telas, quanto tempo perderíamos realizando essa alteração?

No caso de utilizarmos esses recursos centralizados, alteramos o valor que desejamos em apenas um local, fazendo assim com que todos os outros pontos da aplicação que façam referência para este campo, sejam também atualizados.

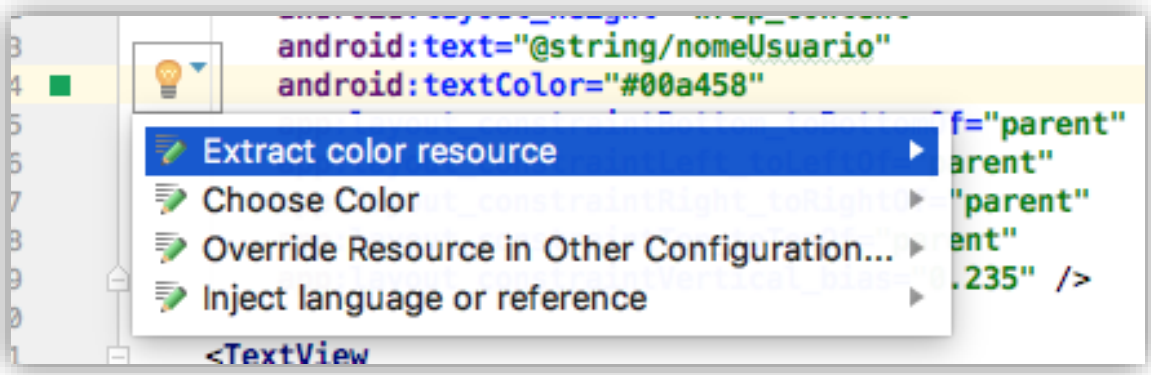


8. colors.xml

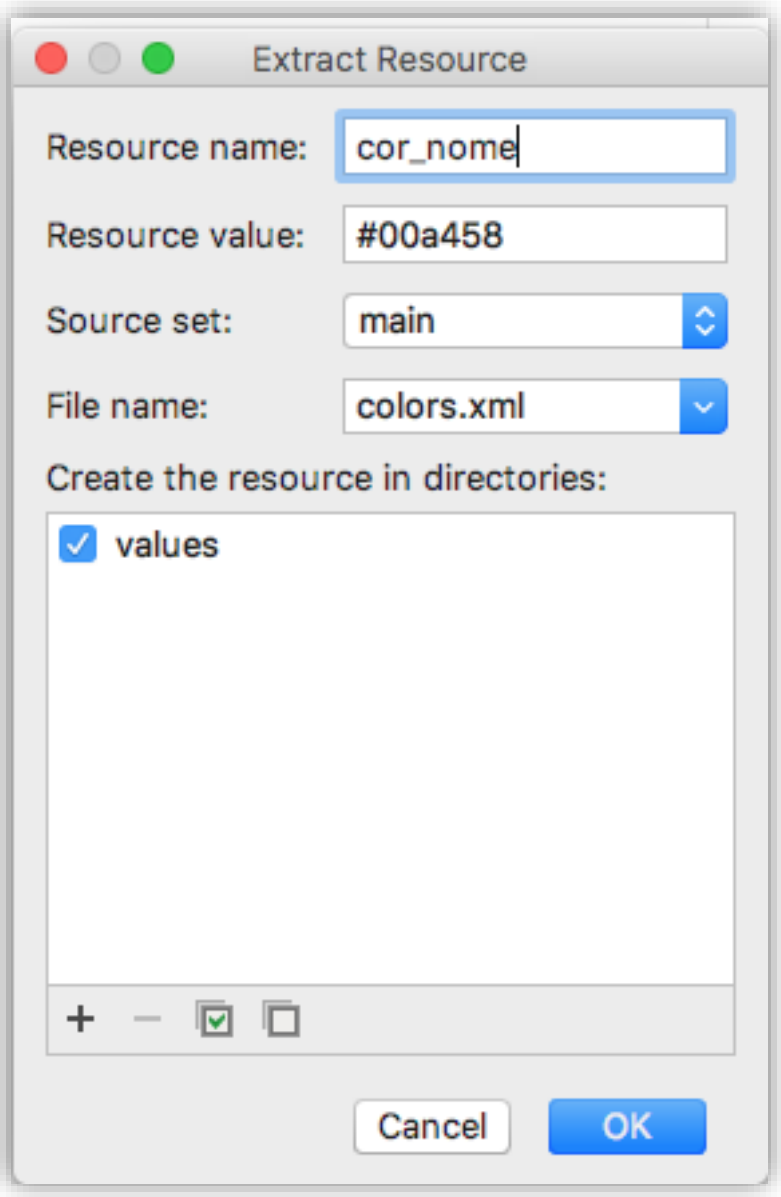
O arquivo colors.xml segue a mesma ideia do arquivo strings.xml, porém, ele é utilizado para determinar as cores do aplicativo e dos layouts que desejamos alterar e manipular as cores.



Sempre que colocarmos algo diretamente na tela, como vimos até agora (cores e strings), o Android Studio nos provê como dica e utilização, criarmos um valor referente e colocarmos em uma “variável”.



Neste exemplo, vamos escrever o nome de uma maneira diferente da maneira anterior criada em strings.xml.



Clique em ‘Ok’.

Se visualizarmos o arquivos colors.xml vemos a cor que criamos.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="colorPrimary">#3F51B5</color>
4      <color name="colorPrimaryDark">#303F9F</color>
5      <color name="colorAccent">#FF4081</color>
6      <color name="cor_nome">#00a458</color>
7  </resources>
8
```

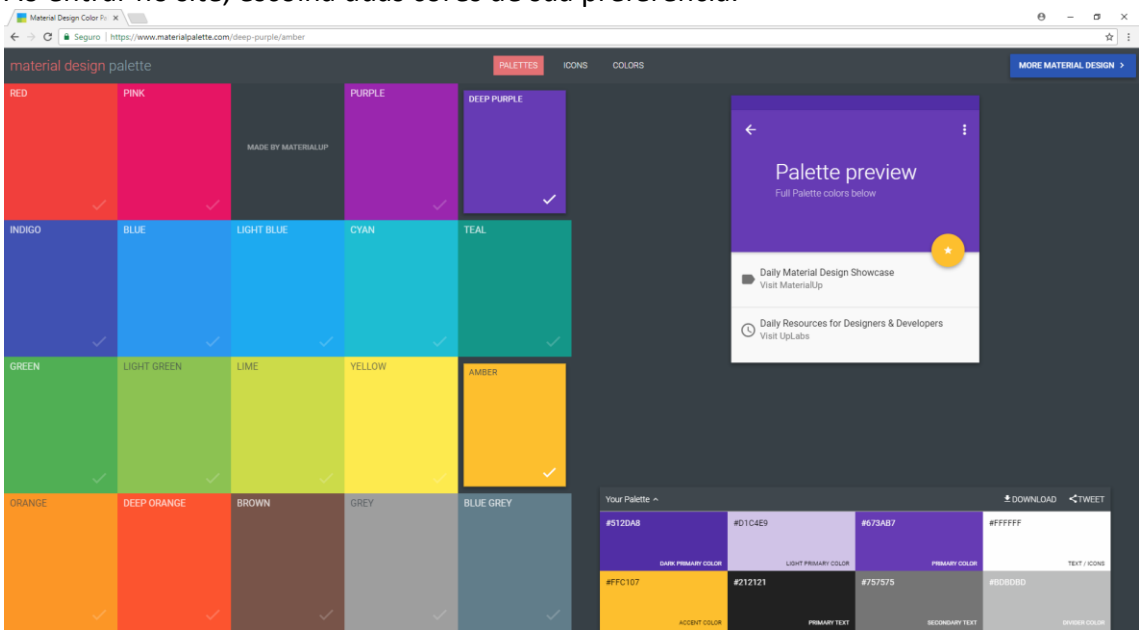
```
<TextView
    android:id="@+id/tvNome"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/nomeUsuario"
    android:textColor="@color/cor_nome"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.235" />
```

Quando quisermos fazer referência a um item que foi criado no strings.xml, utilizamos *@string/nomeDoItem*. E quando desejamos utilizar um recurso de cor: *@color/nomeDaCor*. Ao colocarmos @string ou @color, o próprio Android Studio nos provê uma lista dos itens que temos criado e que podemos utilizar.

8.1. Dica

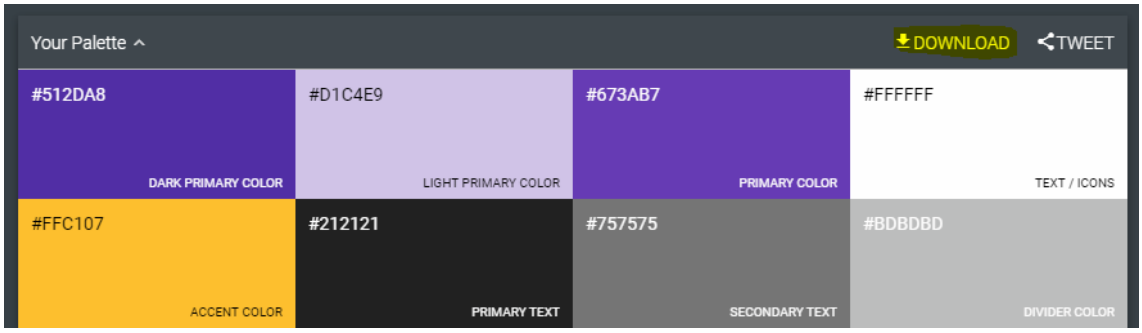
Há um site chamado Material Palette que nos auxilia com as cores primárias e secundárias do nosso aplicativo: https://www.materialpalette.com/

Ao entrar no site, escolha duas cores de sua preferência.

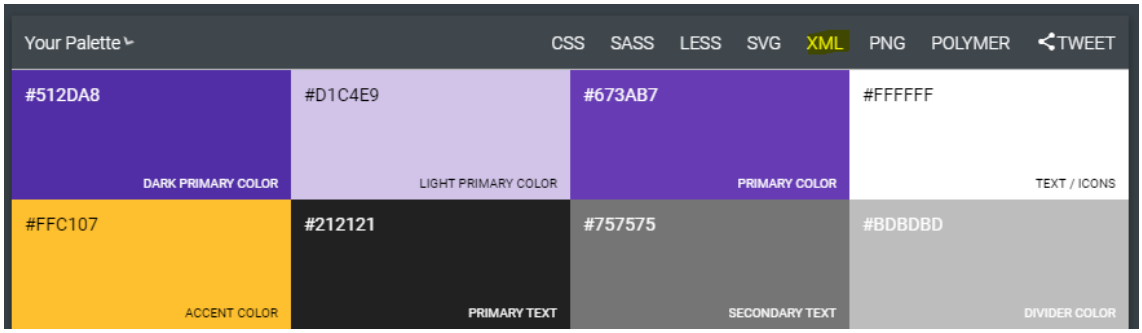


O site já irá mostrar um *preview* de como a sua aplicação irá ficar (design).

Após escolher as cores que você quer utilizar na sua aplicação, clique em Download localizado na parte inferior direito.



Após clicar, selecione a opção que deseja para realizar o download.



Como estamos trabalhando com Android, iremos realizar o download do XML.

Ao abrirmos o arquivo que acabamos de realizar o download com um editor de texto, iremos visualizar as informações que estão dentro deste arquivo.

colors_deep_purple_amber.xml

```xml
<!-- Palette generated by Material Palette - materialpalette.com/deep-purple/amber -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="primary">#673AB7</color>
  <color name="primary_dark">#512DA8</color>
  <color name="primary_light">#D1C4E9</color>
  <color name="accent">#FFC107</color>
  <color name="primary_text">#212121</color>
  <color name="secondary_text">#757575</color>
  <color name="icons">#FFFFFF</color>
  <color name="divider">#BDBDBD</color>
</resources>
```

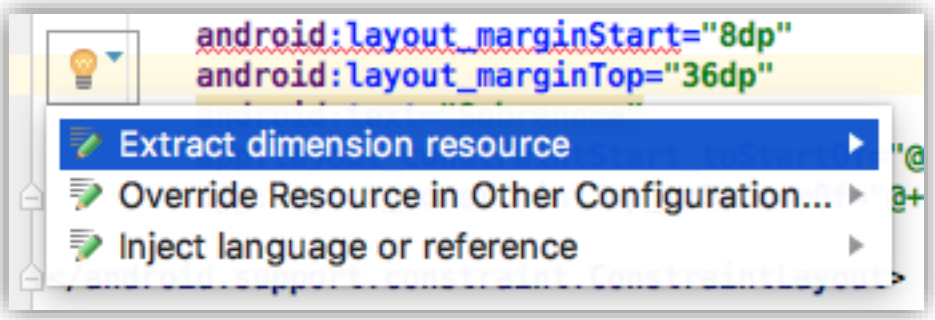
Basta apenas alteramos dentro do nosso colors.xml, as cores que desejamos.

9. dimens.xml

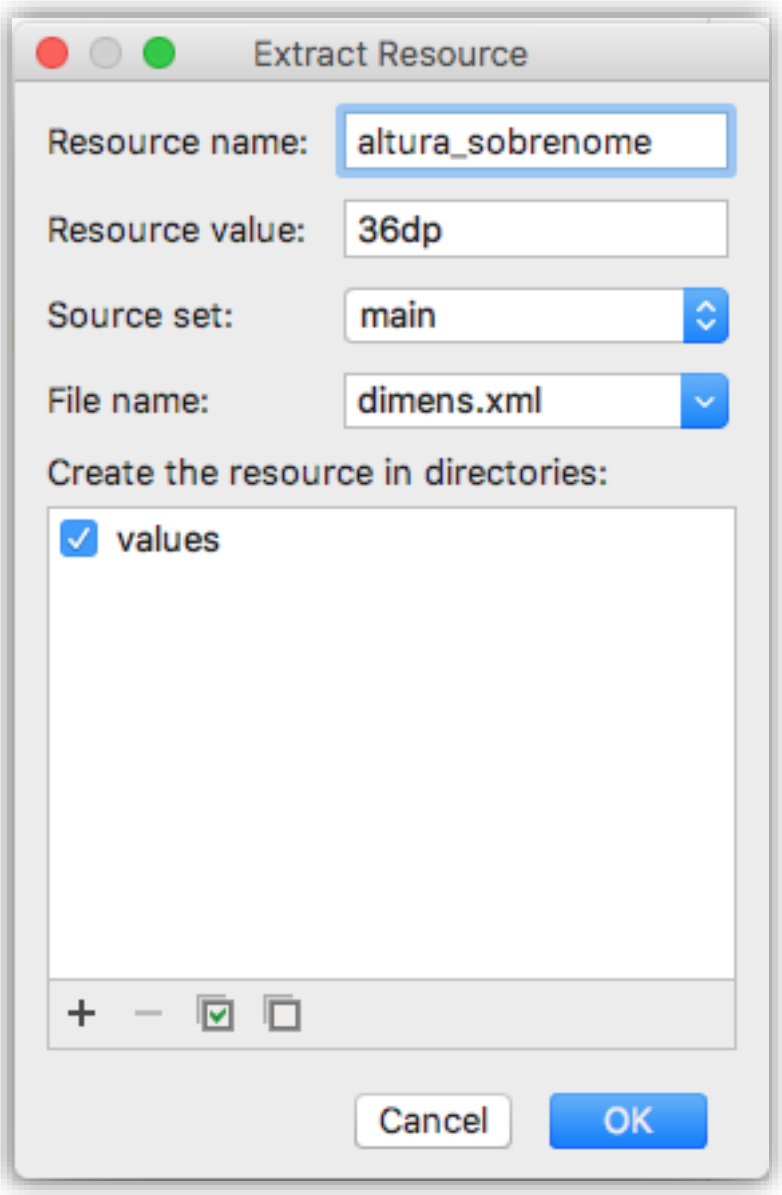
Vamos ao nosso último exemplo. No início do exercício criamos dois campos de texto do tipo TextView. No TextView que ficará abaixo do primeiro (nome), iremos colocar como sobrenome.

```
<TextView
    android:id="@+id/tvSobrenome"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="36dp"
    android:text="Sobrenome"
    app:layout_constraintStart_toStartOf="@+id/tvNome"
    app:layout_constraintTop_toBottomOf="@+id/tvNome" />
```

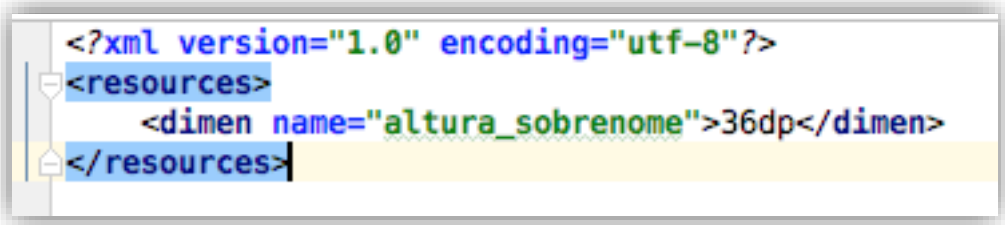
A ideia destes valores que criamos em arquivos, é podermos utilizar em mais de uma tela ou até mesmo ter um ponto de centralização para atualização do que desejamos. Não sendo necessário entrar em todas as telas do aplicativo para realizarmos uma alteração.



Para os arquivos de dimensionamento da aplicação, temos o dimens.xml (margem, espaçamento, altura, largura).

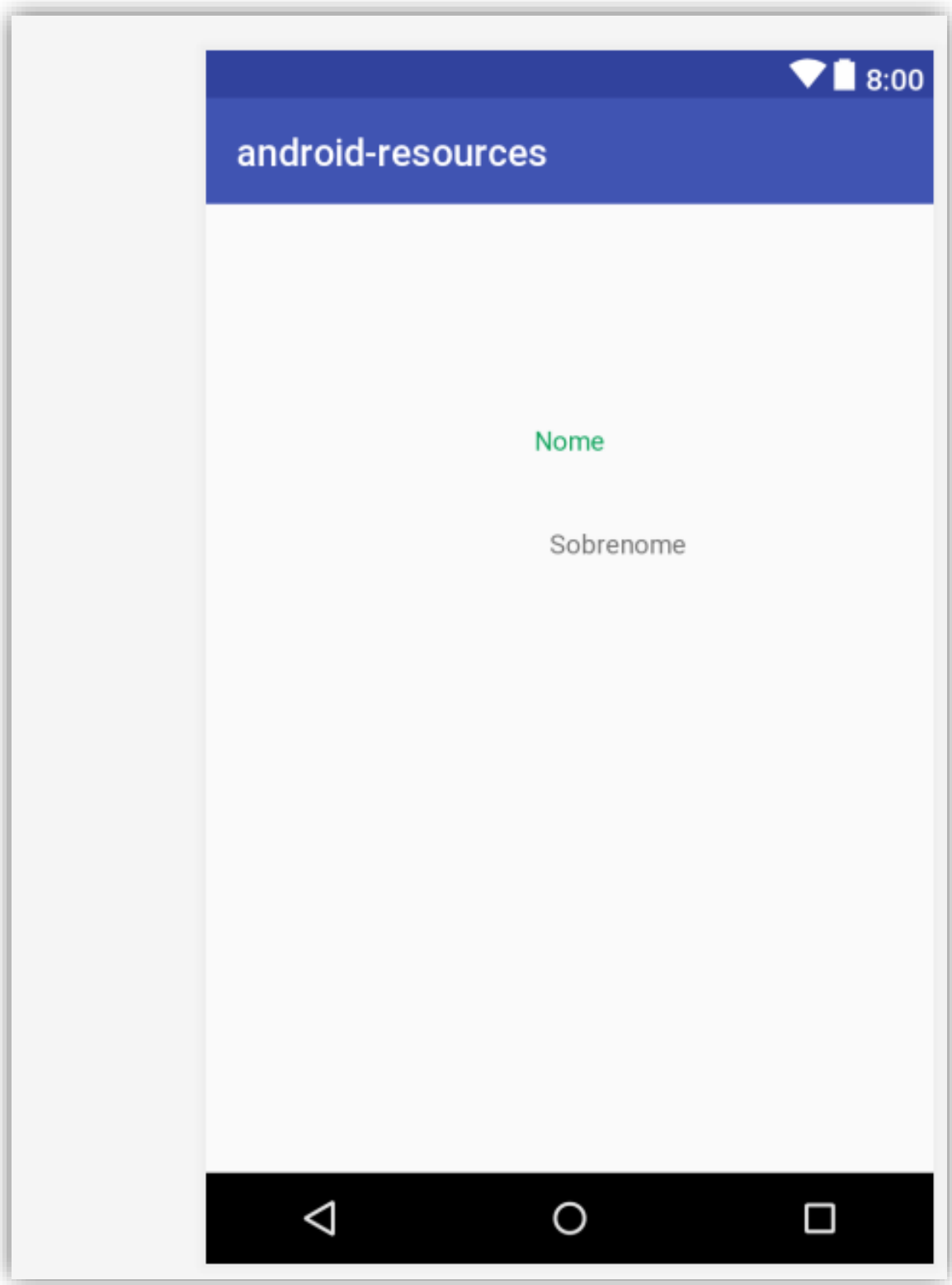

Extract Resource
Resource name:
altura_sobrenome
Resource value:
36dp
Source set:
main
File name:
dimens.xml
Create the resource in directories:
values
Cancel
OK

Clique em ‘Ok’.


<?xml version="1.0" encoding="utf-8"?>
<resources>
<dimen name="altura_sobrenome">36dp</dimen>
</resources>

10. Aplicativo

Como está o nosso aplicativo depois da alteração que fizemos?



11. Resumo

Nesta apostila, mostramos como não manipular os valores na mão e sim compartilhar o que desejamos para toda a aplicação. Refatoração.

12. Referências

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