

Desenvolvimento Mobile

Aula 8

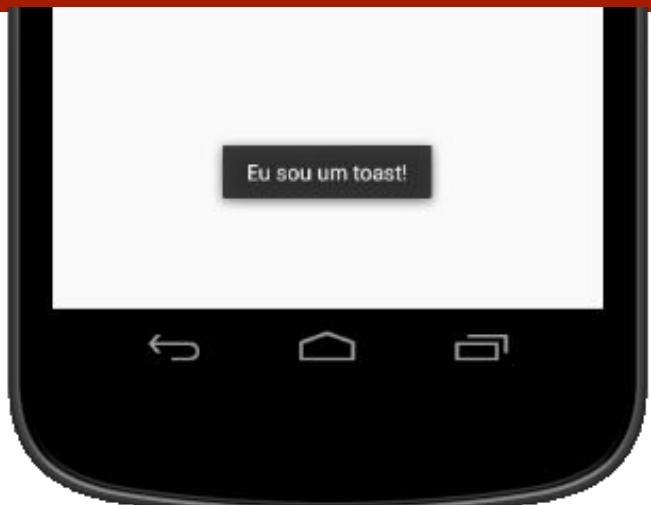
Toast

- ✗ Não pode fechar por *swiping*
- ✓ Pode ser exibido sem nenhuma Activity
- ✗ Não aceita ações do usuário

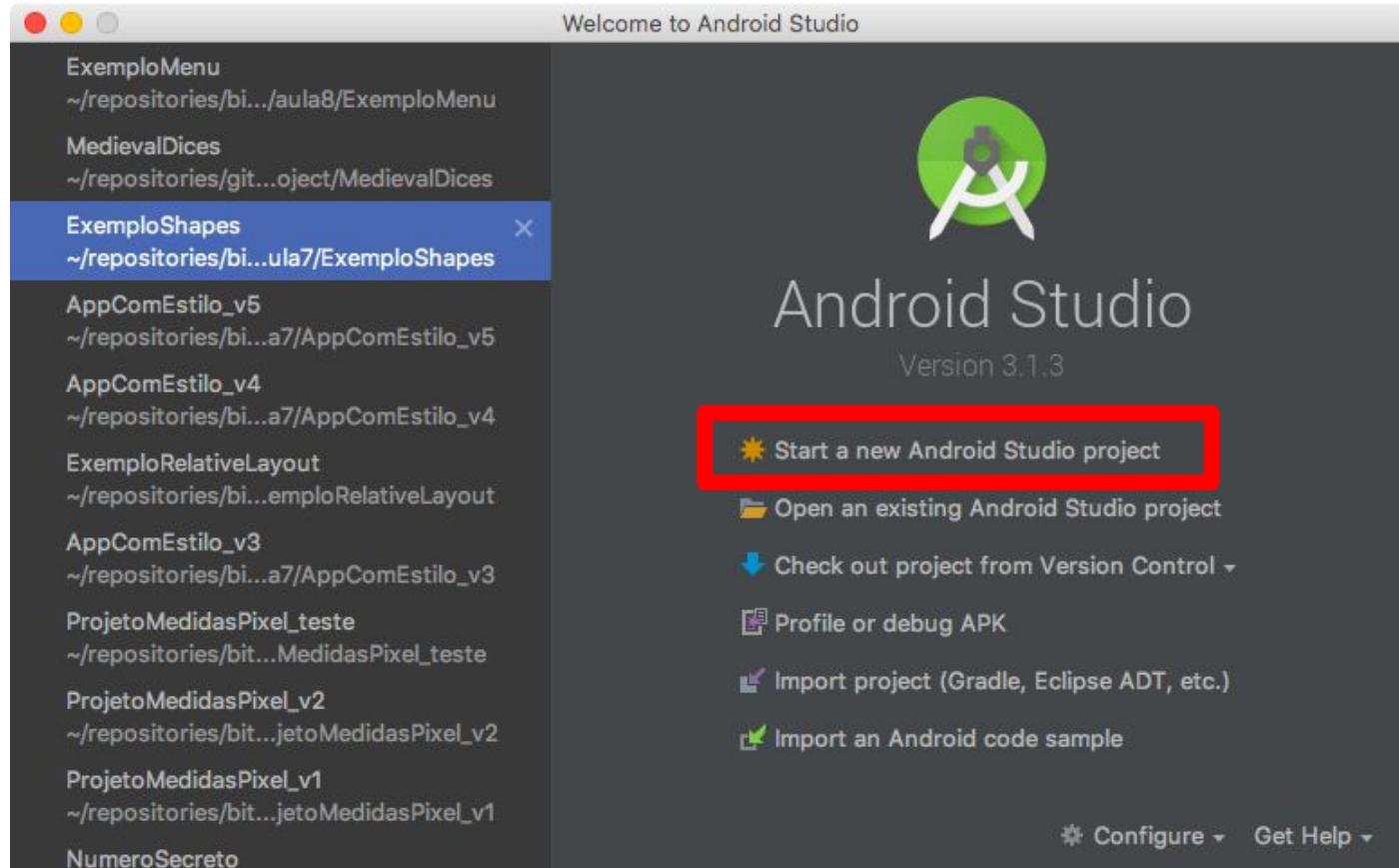
VS

SnackBar

- ✓ Pode fechar por *swiping*
- ✗ Não pode ser exibido fora da Activity
- ✓ Aceita ações do usuário



Trabalhando com Toast (*Exibindo Mensagem*)



Trabalhando com Toast (*Exibindo Mensagem*)

Application name

Exibindo Mensagem

Company domain

lgapontes.com

Project location

/Users/lgapontes/repositories/bitbucket/aulas/desenvolvimento-mobile/aula8/ExibindoMensagem

...

Package name

com.lgapontes.exibindomensagens_v1

Done

Trabalhando com Toast (*Exibindo Mensagem*)

Phone and Tablet

API 15: Android 4.0.3 (IceCreamSandwich)



By targeting **API 15 and later**, your app will run on approximately **100%** of devices. [Help me choose](#)

Include Android Instant App support

Wear

API 21: Android 5.0 (Lollipop)



TV

API 21: Android 5.0 (Lollipop)



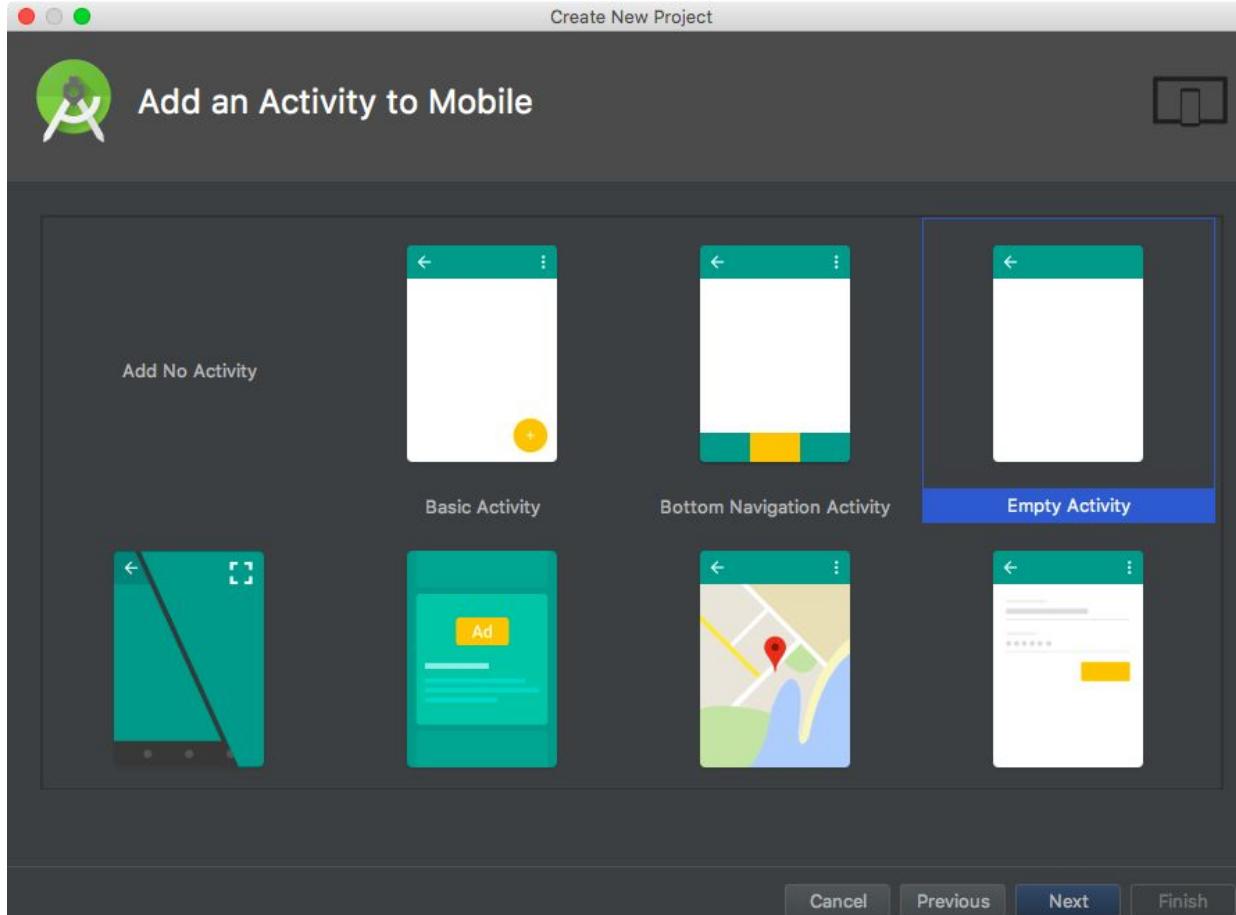
Android Auto

Android Things

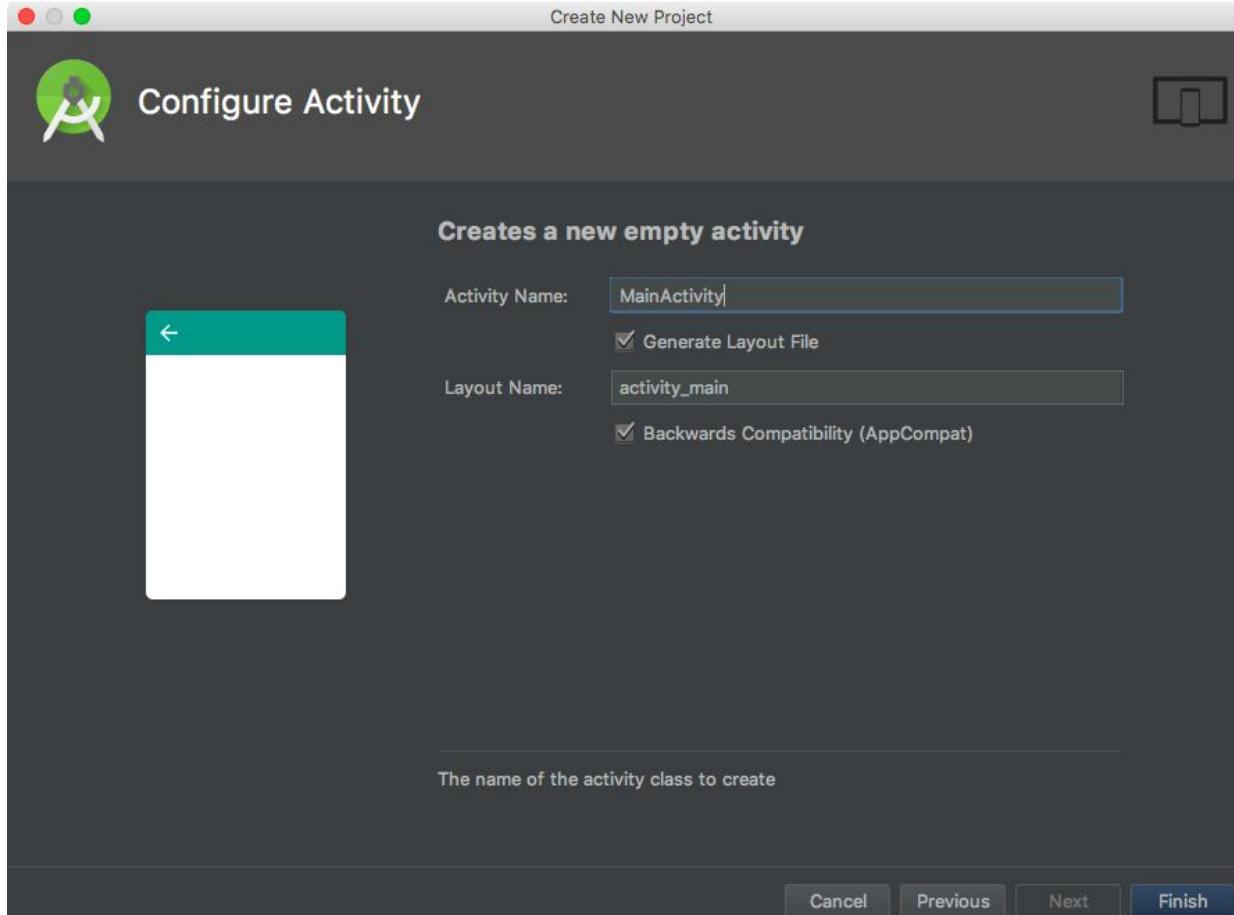
API 24: Android 7.0 (Nougat)



Trabalhando com Toast (*Exibindo Mensagem*)



Trabalhando com Toast (*Exibindo Mensagem*)



Trabalhando com Toast (*Exibindo Mensagem*)

```
Toast.makeText( context: this, text: "Eu sou um toast!", Toast.LENGTH_SHORT).show();
```

O Android fornece uma classe chamada *Toast* para exibirmos um toast no dispositivo (através do método *makeText*). O primeiro parâmetro do método é o contexto (referente à Activity).

Tempo de exibição:
LONG_DELAY \Rightarrow 3.5s
SHORT_DELAY \Rightarrow 2s

Não se esqueça de chamar o método *show()*

Texto da mensagem. Segundo as boas práticas do Material Design, o Toast deve ter no máximo 2 linhas.

Trabalhando com Toast (*Exibindo Mensagem*)

The screenshot shows the Android Studio interface with the code editor open. The tab bar at the top has 'activity_main.xml' and 'MainActivity.java' selected. The code in 'MainActivity.java' is as follows:

```
1 package com.lgapontes.exibindomensagens_v1;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.widget.Toast;
7
8 public class MainActivity extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_main);
14    }
15
16    public void exibirToast(View view) {
17        Toast.makeText(context: this, text: "Eu sou um toast!", Toast.LENGTH_SHORT).show();
18    }
19}
```

The code demonstrates how to use the `Toast` class to display a message. It includes imports for `AppCompatActivity`, `Bundle`, `View`, and `Toast`. The `onCreate` method sets the content view to `activity_main`. The `exibirToast` method takes a `View` parameter and uses `Toast.makeText` to show a short toast message containing the text "Eu sou um toast!".

Trabalhando com Toast (*Exibindo Mensagem*)

The screenshot shows the Android Studio interface with two tabs open: `activity_main.xml` and `MainActivity.java`.

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">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Exibir Toast"
        android:onClick="exibirToast" />

</LinearLayout>
```

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {

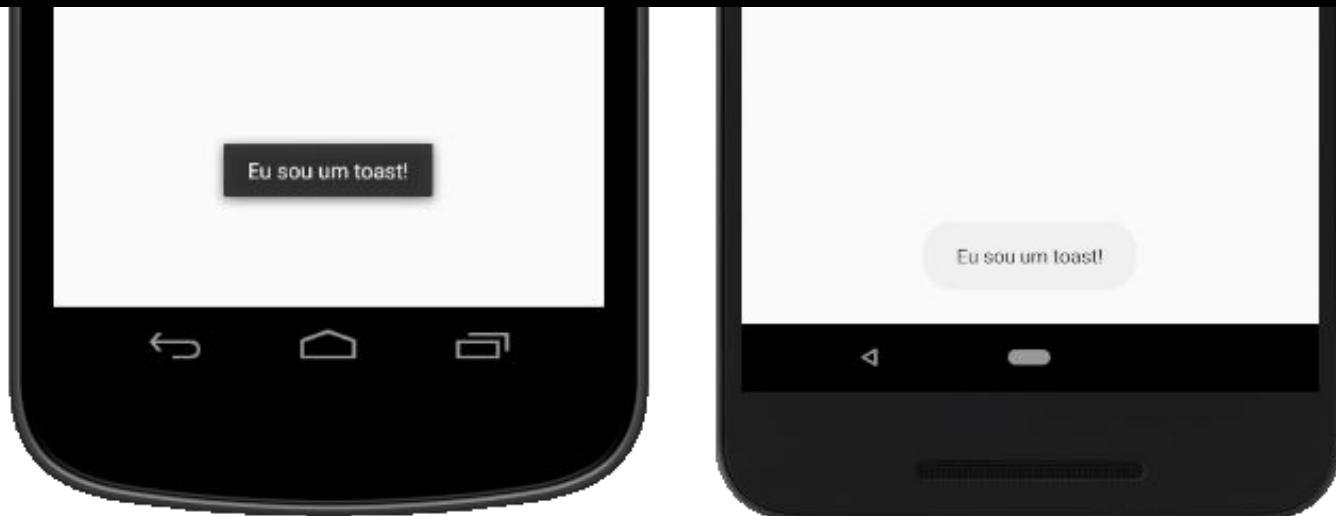
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void exibirToast(View view) {
        // Implementação para exibir uma mensagem de toast
    }
}
```

Trabalhando com Toast (*Exibindo Mensagem*)

O Toast pode ter um visual particular nas diferentes versões do Android SDK. Apesar da recomendação do Material Design (que também é da Google) indicar cantos arredondados, em SDK's mais antigos às vezes eles não tem esse detalhe.

É possível, entretanto, criar um Toast customizado!



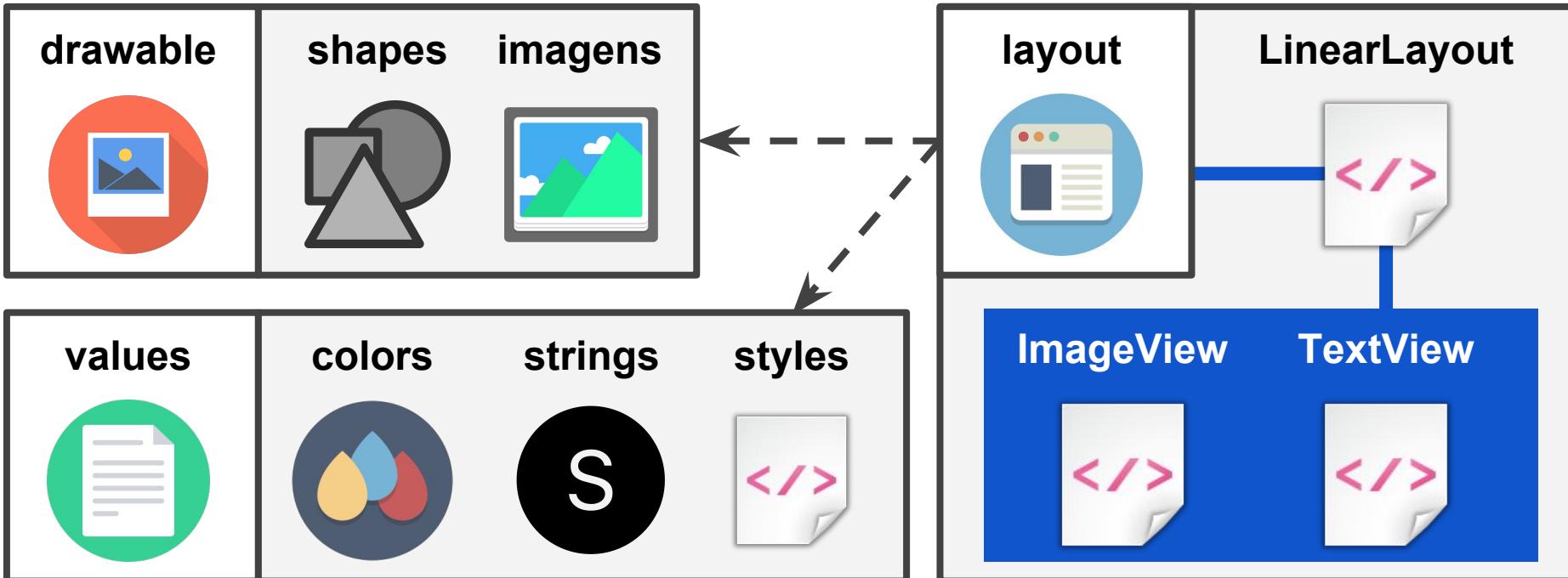
Trabalhando com Toast Customizado (*Exibindo Mensagem*)

Vamos criar o seguinte Toast...



Trabalhando com Toast Customizado (*Exibindo Mensagem*)

Um Toast customizado nada mais é do que um objeto *Toast* inflando um Layout criado na pasta *layout*.



O termo **inflar** é utilizado quando criamos um **Layout** a partir de um arquivo XML



1- Importar imagem SVG

2- Criar cores

3- Criar um shape

4- Criar o layout

5- Inflar o layout para criar o Toast

MainActivity



5

drawable



shapes



imagens



3

1

layout



LinearLayout



4

values



colors



strings



styles



2

ImageView



TextView



Trabalhando com Toast Customizado (*Exibindo Mensagem*)

1

Primeiramente veremos como acrescentar ícones escaláveis



Também podemos utilizar imagens comuns (PNG, JPG, etc). Veremos, entretanto, que imagens vetoriais são muito poderosas!

Material Design Tools

<https://material.io/tools/>

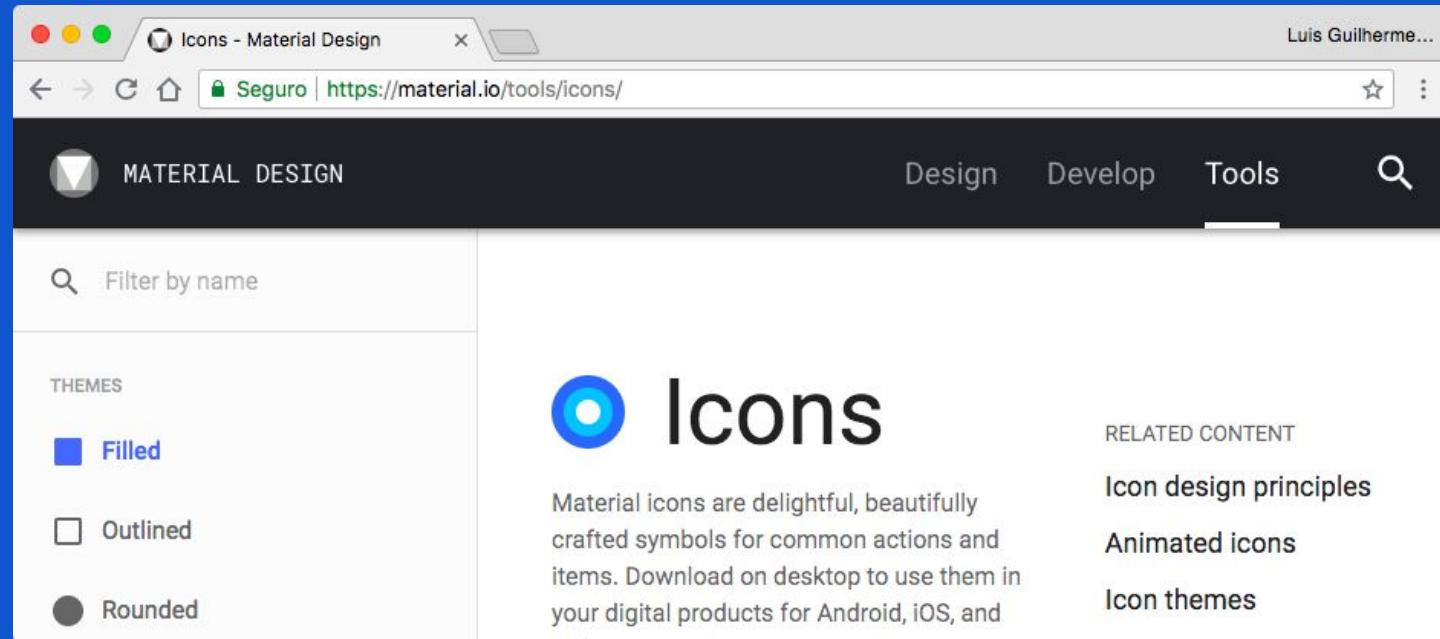
The screenshot shows a web browser window titled "Tools - Material Design". The URL in the address bar is <https://material.io/tools/>. The page has a dark theme with a navigation bar at the top featuring a Material Design icon, "Design", "Develop", "Tools", and a search icon. Below the navigation bar, the text "Material tools to simplify your workflow." is displayed. The main content area contains several tool components:

- Material Theme Editor:** A grid-based interface for designing UI components. It shows a "NEXT" button and a card with a heart icon.
- Shape:** A tool for picking corner styles. It includes a preview of a rounded square and a slider for corner radius.
- Corner style:** A list of five options: Baseline, Round, Sharp, Outline, and Two-tone.
- Icons:** A grid of icons including a person in a wheelchair, a speaker, a car, and an "SVG" download icon.

At the bottom left, the text "Material Theme Editor" is visible.

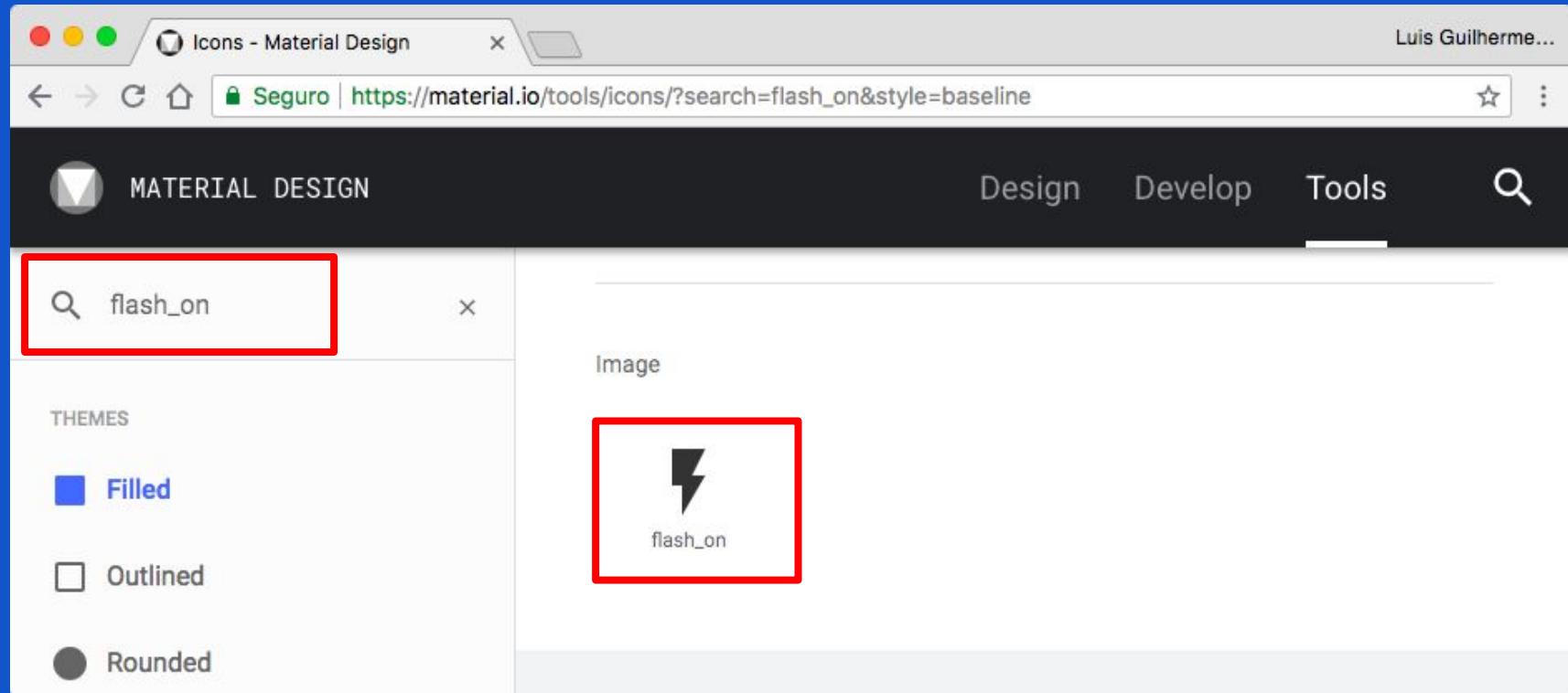
Trabalhando com Scalable Vector Graphics (SVG)

Imagens SVG são automaticamente ajustadas pelo Android de acordo com a densidade de tela do aparelho. O próprio Google oferece uma infinidade de ícones para utilização em sua App: <https://material.io/tools/icons/>



Trabalhando com Scalable Vector Graphics (SVG)

Procure pelo ícone *flash_on* no campo de busca.



Icons - Material Design Luis Guilherme...

Seguro | https://material.io/tools/icons/?search=fla... :

Design Develop Tools

flash_on

THEMES

Filled

Outlined

Rounded

Clique no ícone desejado

Image

1

flash_on

Clique em ^ (seta para cima) para abrir as opções de download do ícone

CATEGORIES

All

Selected Icon

2

adaptable system of guidelines, components, and tools that support the best

SVG 24

This screenshot shows the Material Design Icons tool interface. A red callout with the text 'Clique no ícone desejado' (Click the desired icon) has a red circle with the number '1' over it, pointing to the 'flash_on' icon in the main preview area. Below the preview, another red callout with the text 'Clique em ^ (seta para cima) para abrir as opções de download do ícone' (Click the up arrow (up arrow) to open the download options for the icon) has a red circle with the number '2' over it, pointing to the '^' symbol in the 'Selected Icon' dropdown menu.

Icons - Material Design Luis Guilherme...

Seguro | https://material.io/tools/icons/?search=fla... :

Design Develop Tools

Selected Icon

Available under Apache license version 2.0.

Image

flash_on

18dp

Assets

3

Icon font

Faça o download no formato SVG

Salve o arquivo com o nome bolt.svg

SVG

PNG

Follow the guidelines in your style guide using CS

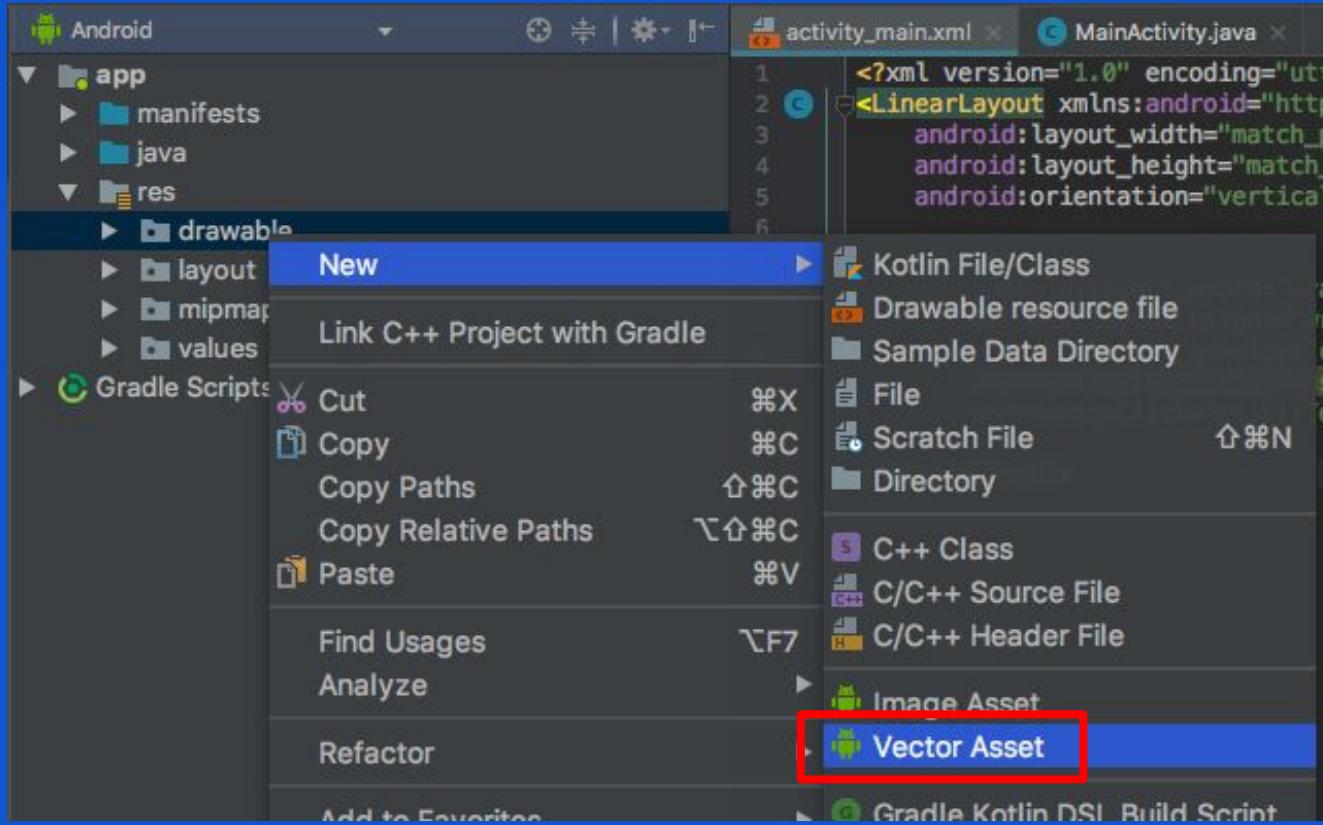
Usage:

<!-- For example -->

<i class="material-icons">flash_on</i>

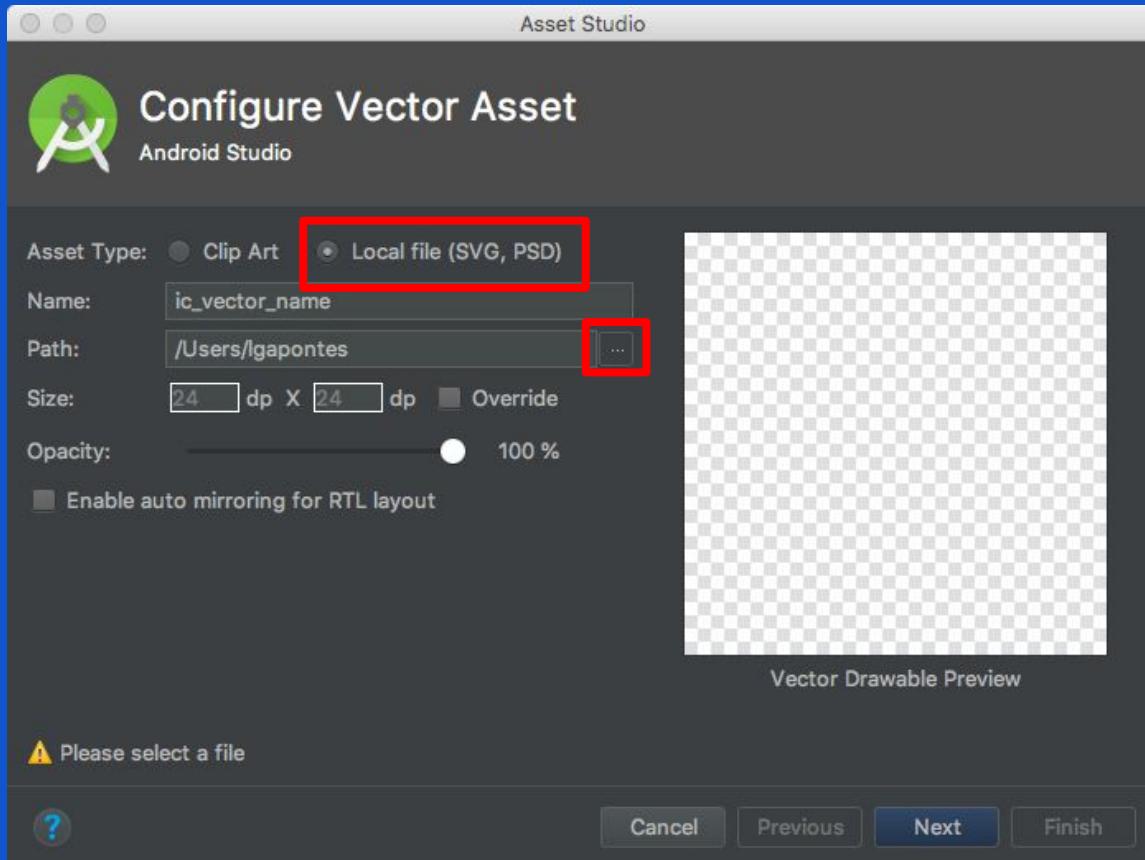
This screenshot shows the 'Selected Icon' dropdown menu open, displaying the 'flash_on' icon. The 'Assets' section below it shows download buttons for 'SVG' and 'PNG'. A red callout with the text 'Faça o download no formato SVG' (Download in SVG format) has a red circle with the number '3' over it, pointing to the 'SVG' download button. Another red callout with the text 'Salve o arquivo com o nome bolt.svg' (Save the file with the name bolt.svg) is partially visible at the bottom.

Trabalhando com Scalable Vector Graphics (SVG)



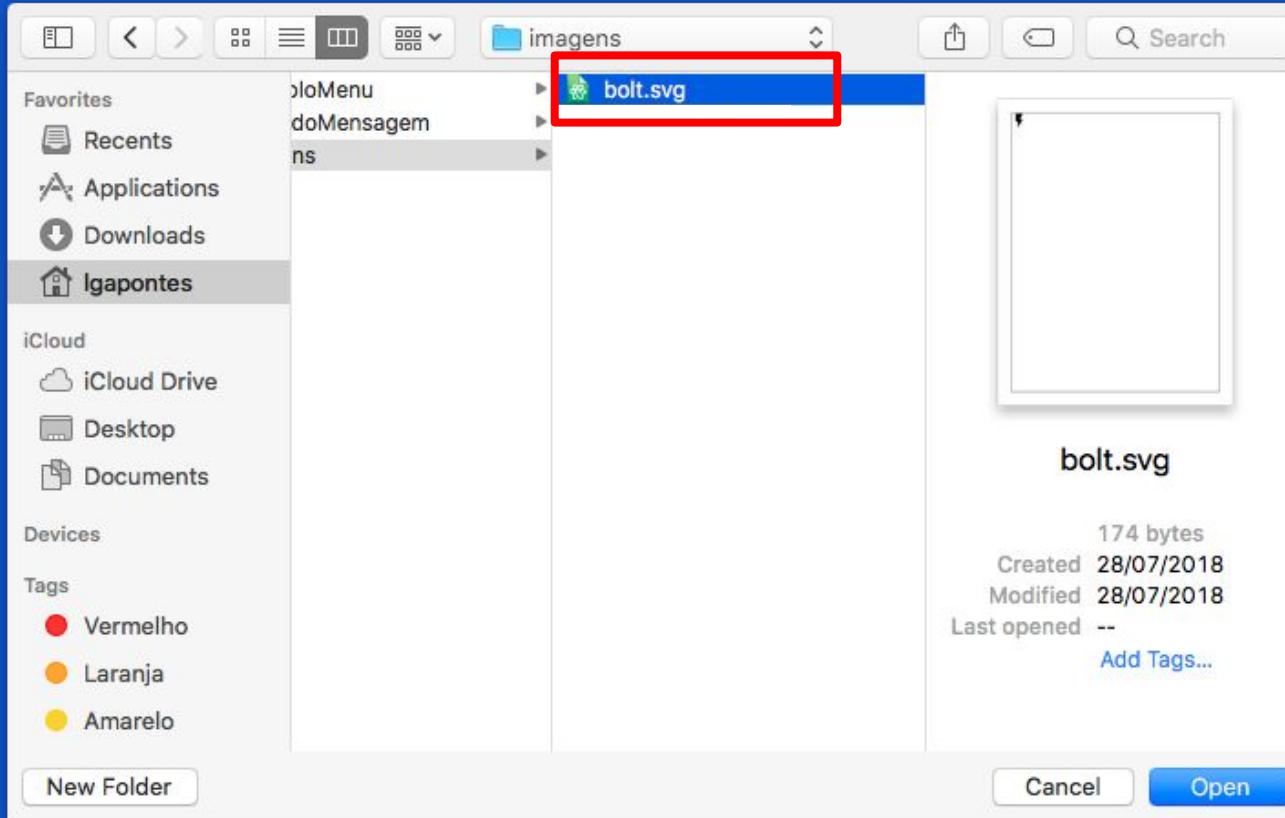
Clique com o botão direito na pasta *drawable*, selecione o menu *new*, submenu *Vector Asset*

Trabalhando com Scalable Vector Graphics (SVG)



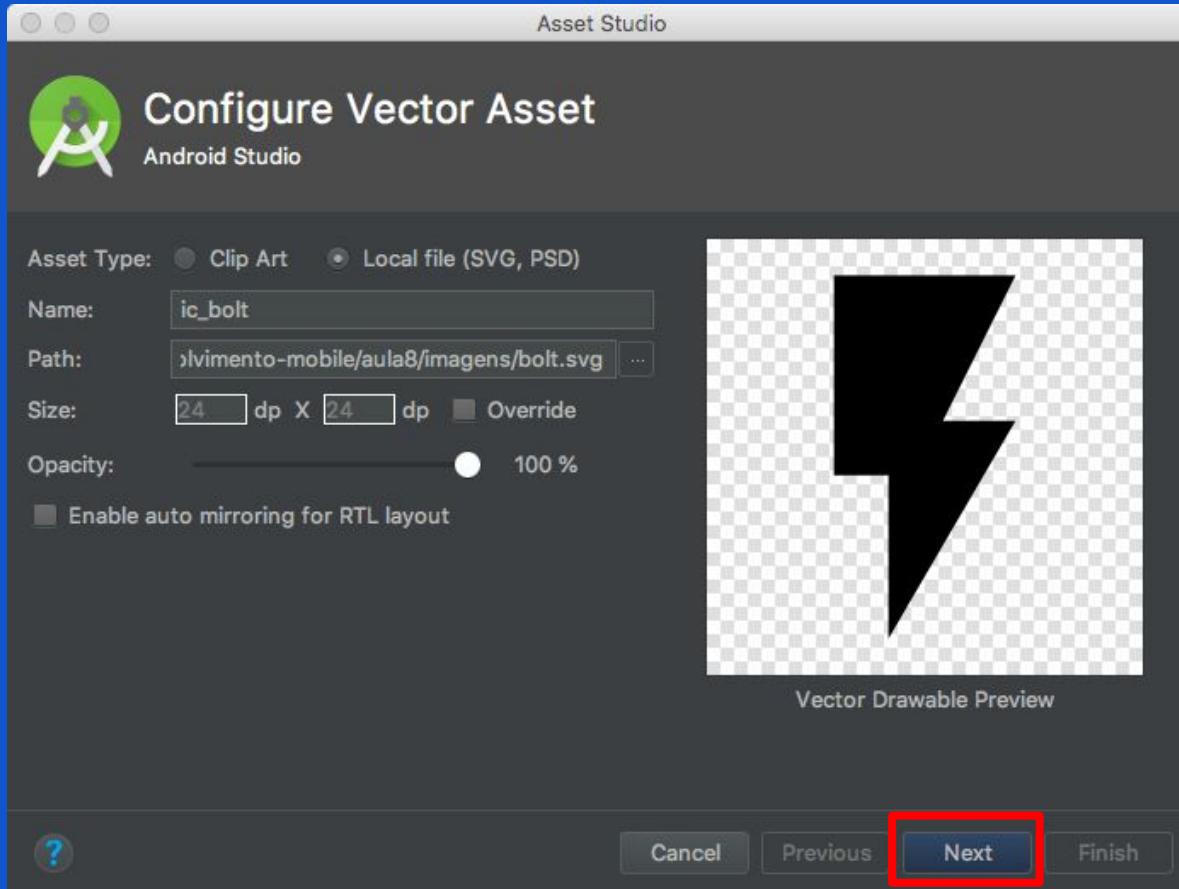
Clique na radiobutton
Local file e em seguida,
clique no botão ... para
selecionar o arquivo
baixado.

Trabalhando com Scalable Vector Graphics (SVG)

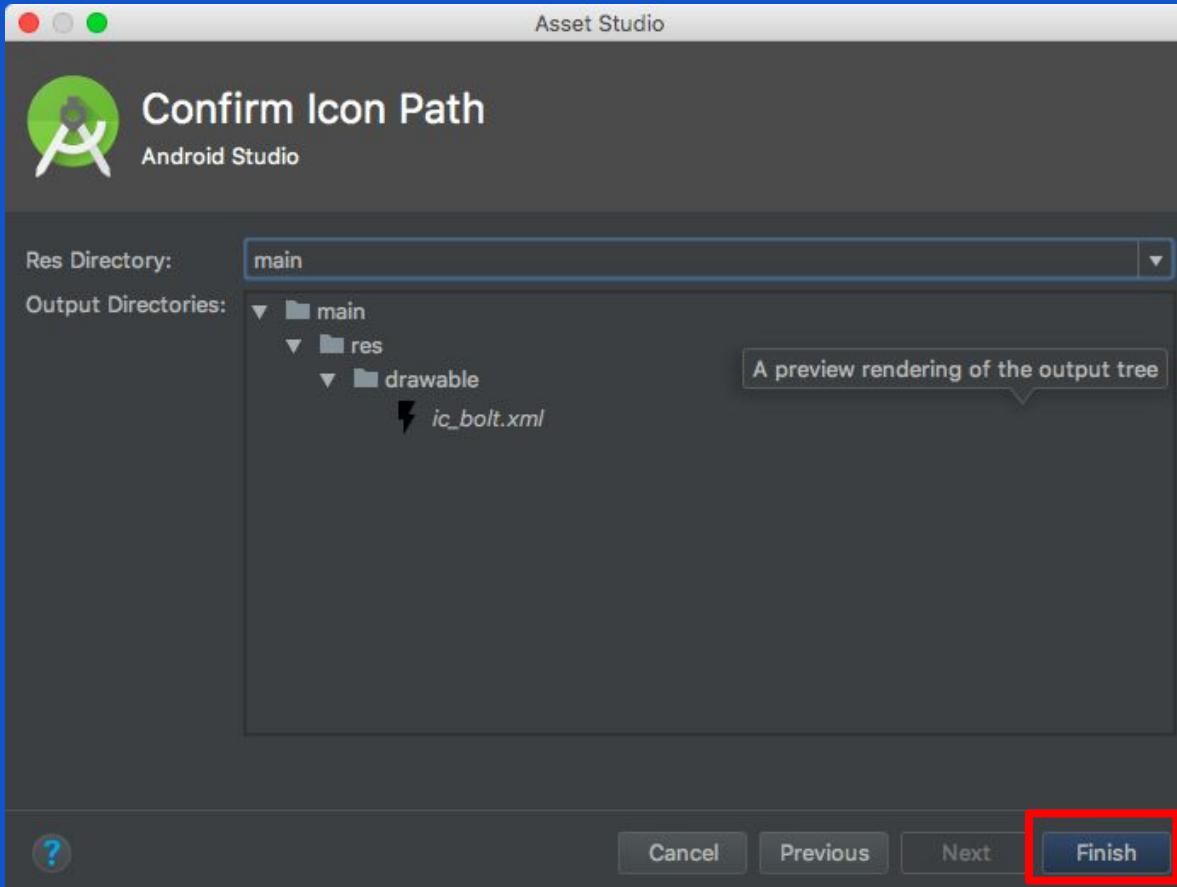


Selecione o
arquivo *bolt.svg*

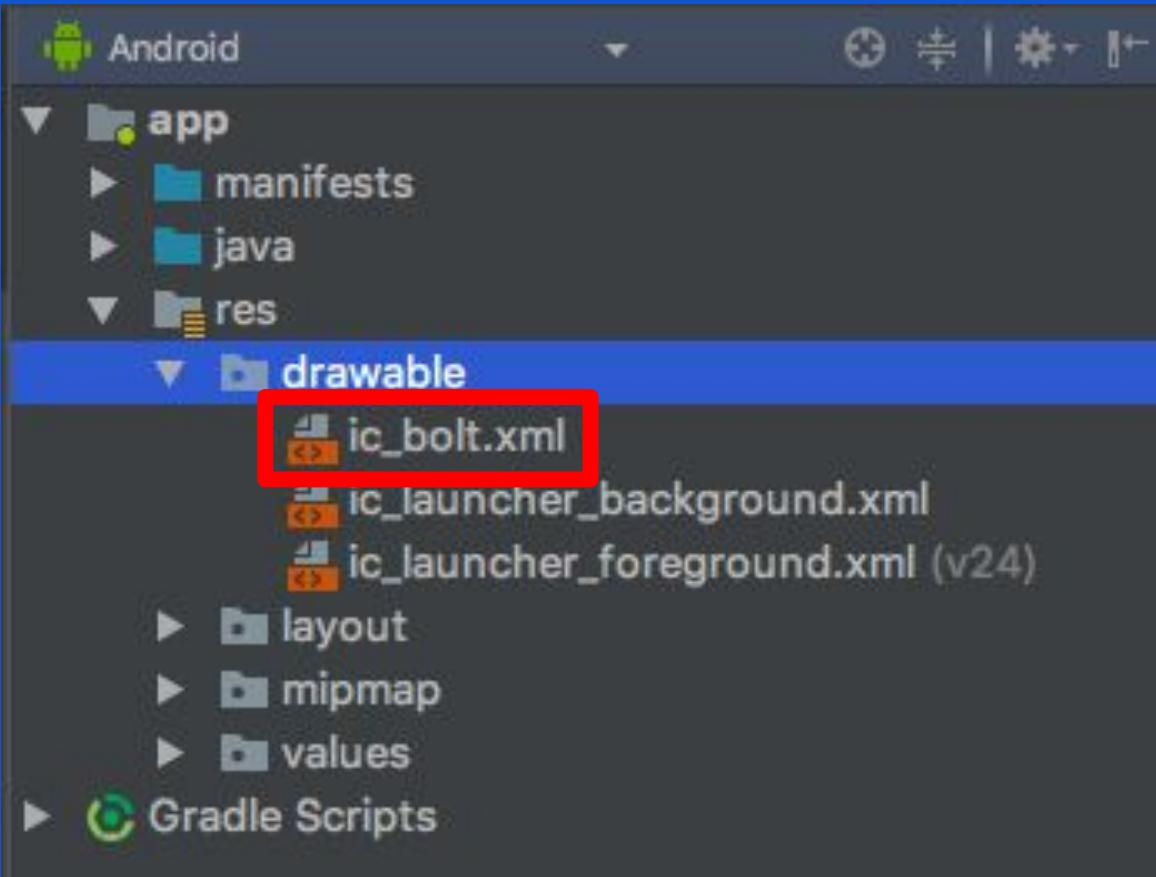
Trabalhando com Scalable Vector Graphics (SVG)



Trabalhando com Scalable Vector Graphics (SVG)



Trabalhando com Scalable Vector Graphics (SVG)



O Android Studio vai criar um arquivo XML com os dados do SVG e colocar um prefixo *ic_* (ícone).

O guia do Android orienta colocarmos prefixos *ic_* para todos os ícones e prefixos *bg_* para imagens de fundo.

Trabalhando com Toast Customizado (*Exibindo Mensagem*)

2

Vamos definir as cores utilizadas no Toast no arquivo *colors.xml*



The screenshot shows the Android Studio code editor with the file "colors.xml" open. The code defines several color resources:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>

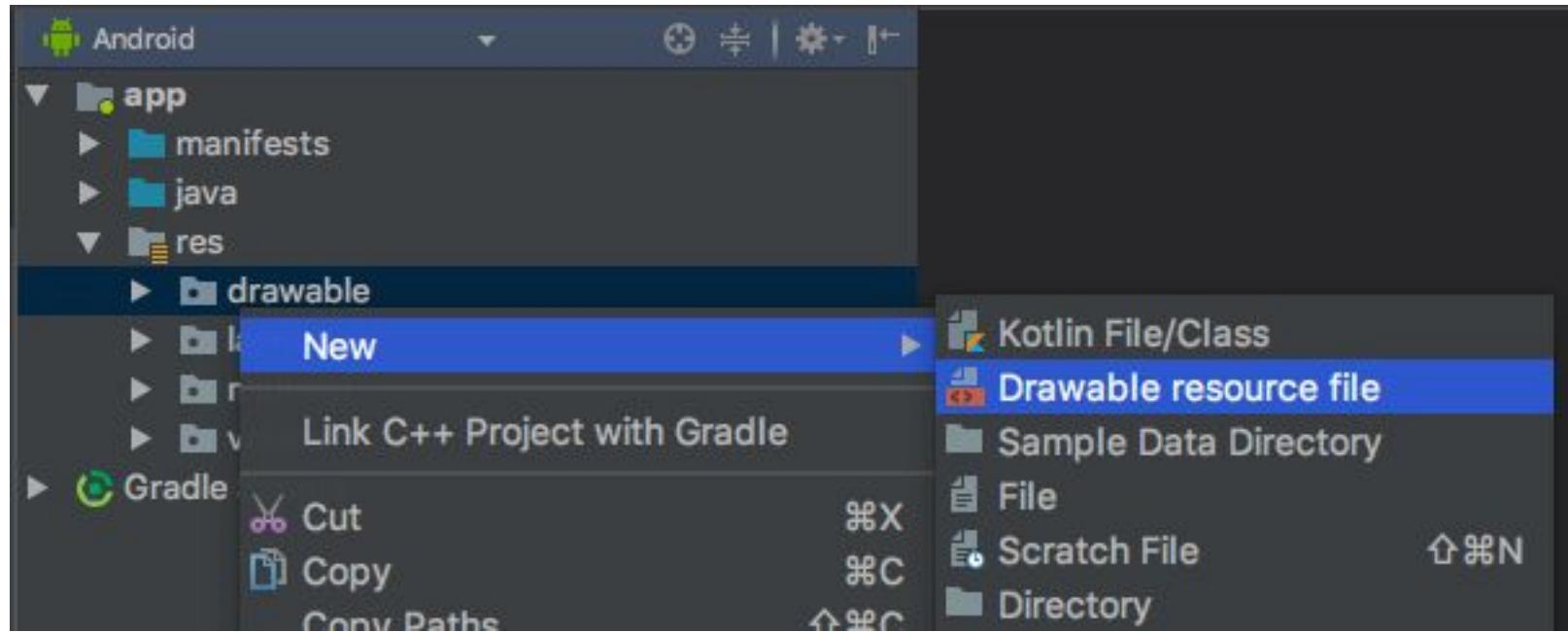
    <color name="amarelo">#ffc518</color>
    <color name="preto">#0d0d0d</color>
    <color name="branco">#ffffff</color>
</resources>
```

On the left side of the editor, there is a vertical color palette with six colored squares corresponding to the defined colors: blue, dark blue, orange, yellow, black, and white.

Trabalhando com Toast Customizado (*Exibindo Mensagem*)

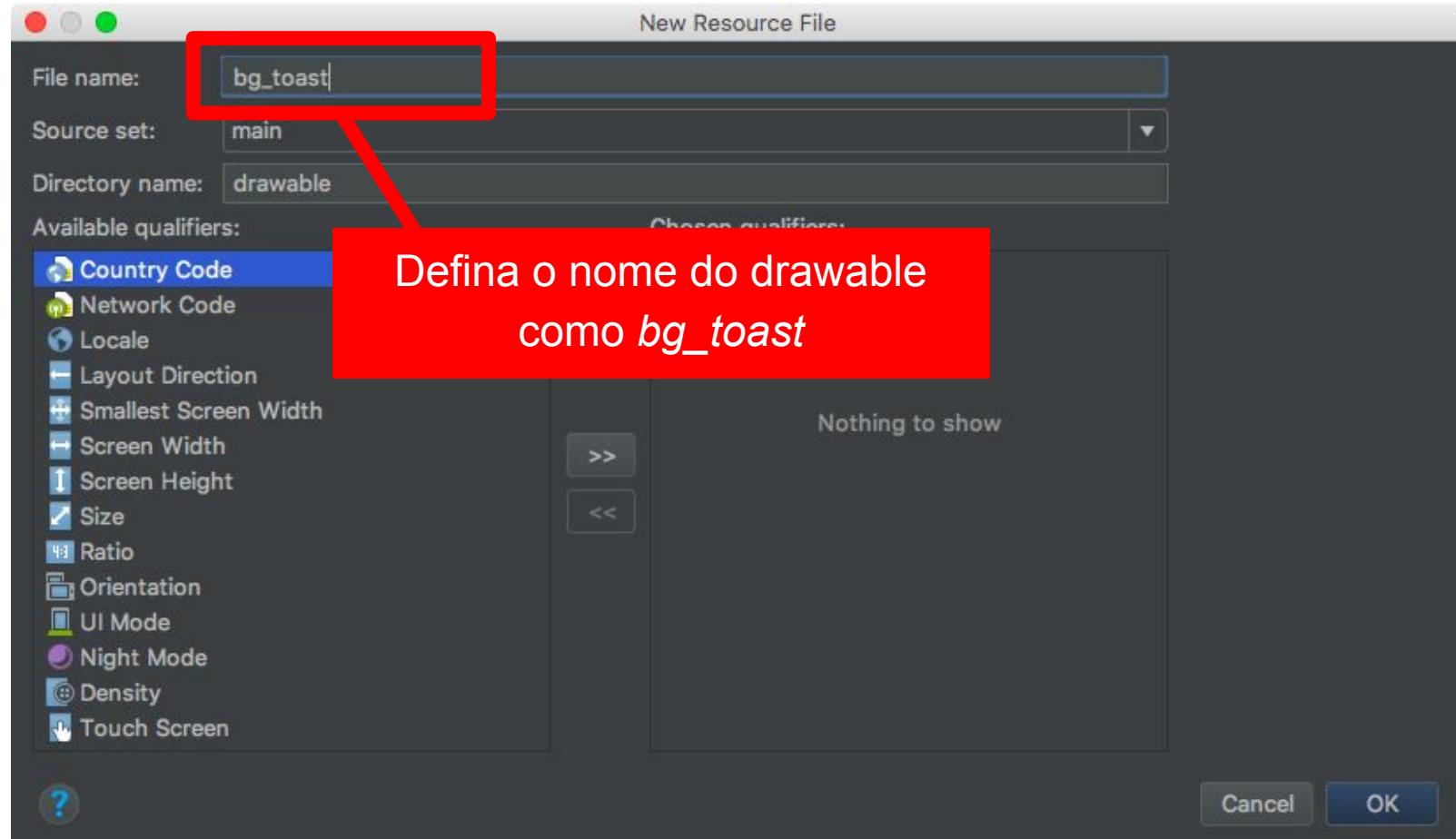
3

Vamos criar um shape para arredondar os cantos do Toast



Trabalhando com Toast Customizado (*Exibindo Mensagem*)

3



Defina o nome do drawable
como *bg_toast*

Trabalhando com Toast Customizado (*Exibindo Mensagem*)

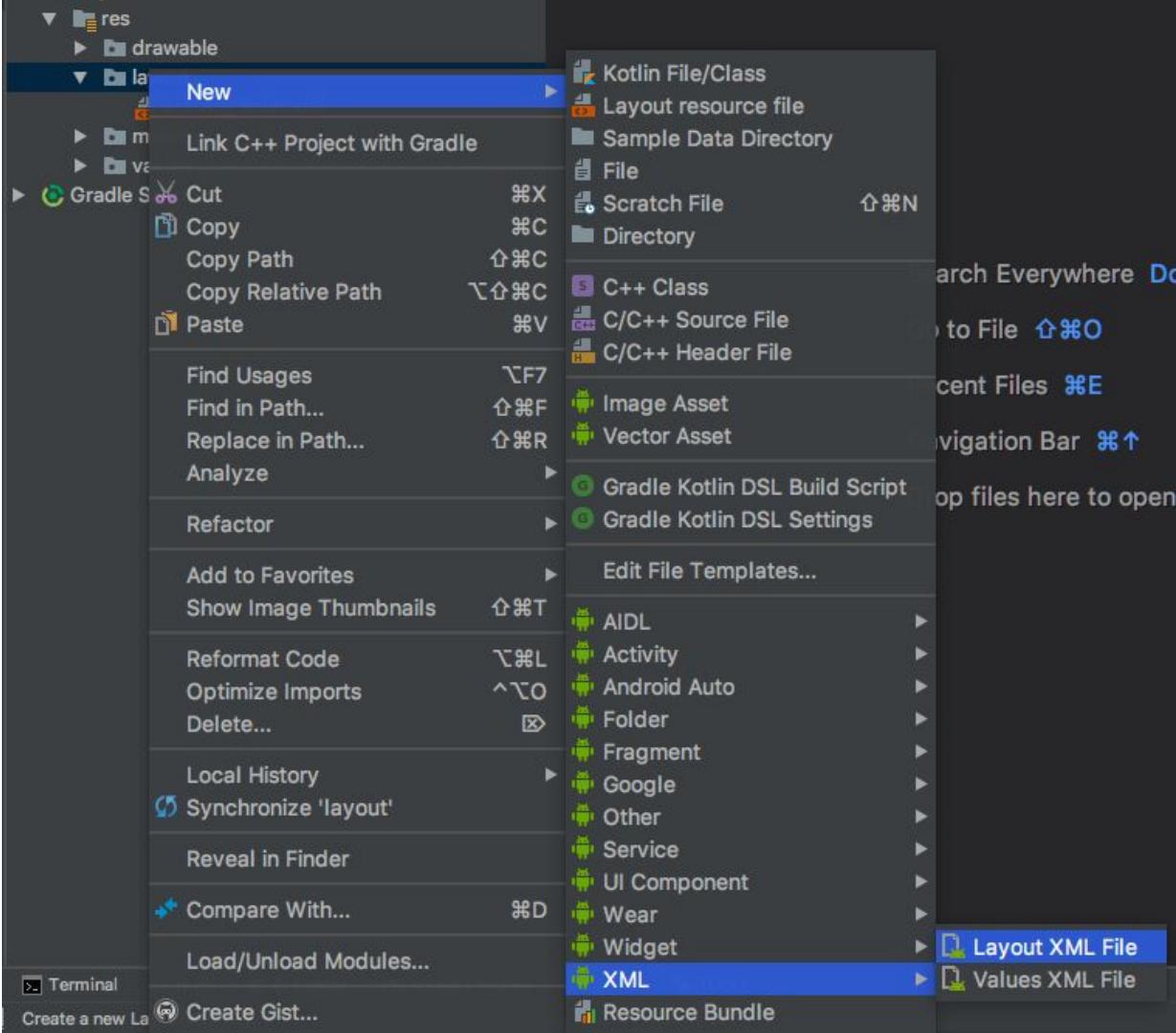
3

O shape deve ser um *rectangle* com cor preta e *radius* de *5dp*

```
gb_toast.xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="@color/preto"/>
    <corners android:radius="5dp" />
</shape>
```

4

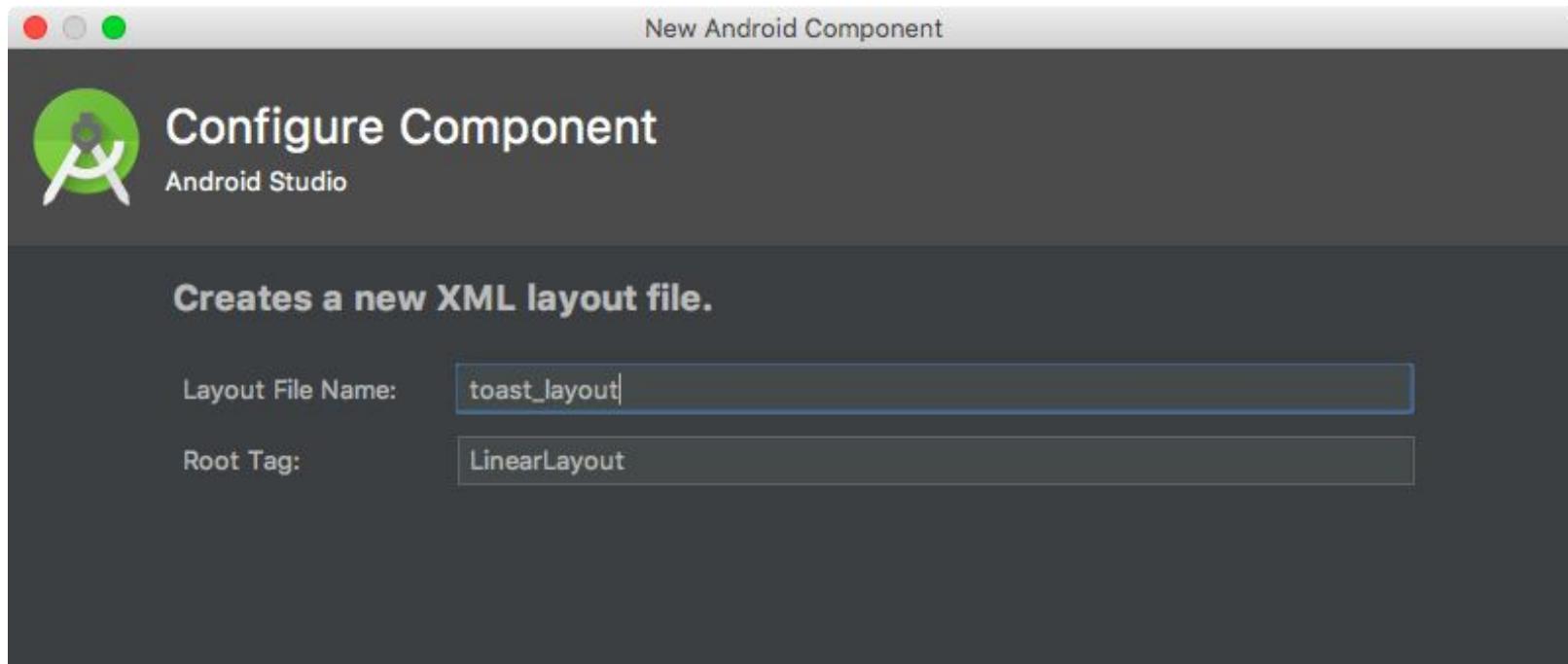
Crie um novo arquivo de layout XML dentro da pasta *layout*



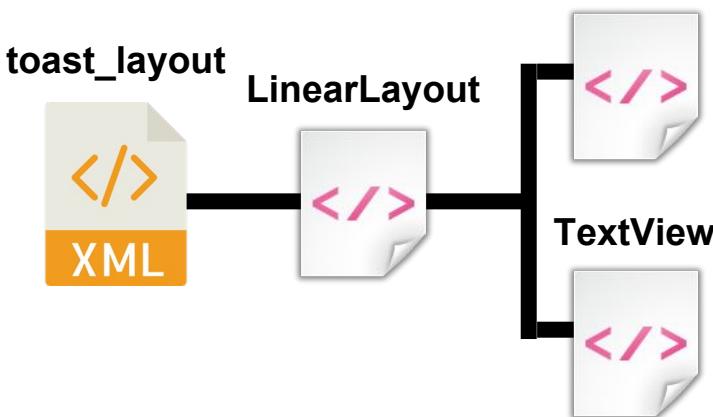
Trabalhando com Toast Customizado (*Exibindo Mensagem*)

4

Nomeie o layout de *toast_layout* e mantenha como *LinearLayout*



A estrutura do Toast será composta pelos seguintes elementos:



```
_layout.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="horizontal"
    android:background="@drawable/gb_toast"
    android:id="@+id/toast_container">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="10dp"
        android:layout_marginBottom="10dp"
        android:src="@drawable/ic_bolt"
        android:tint="@color/amarelo" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:textSize="18sp"
        android:textColor="@color/branco"
        android:text="toast"
        android:id="@+id/toast_message" />
</LinearLayout>
```

Linear Layout:

- Background apontando para *bg_toast*
- Defina um ID

Image View:

- Aponte para *ic_bolt*
- Altere a cor para amarelo através do atributo *tint*

TextView:

- Defina um ID
- Defina a cor do texto como branca

```
<?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="horizontal"
    android:background="@drawable/gb_toast"
    android:id="@+id/toast_container" >
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="10dp"
        android:layout_marginBottom="10dp"
        android:src="@drawable/ic_bolt"
        android:tint="@color/amarelo" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:textSize="18sp"
        android:textColor="@color/branco"
        android:text="toast"
        android:id="@+id/toast_message" />
</LinearLayout>
```

5

Crie um método *exibirToastCustomizado()* para exibir o Toast

```
public void exibirToastCustomizado(View view) {  
    Toast toast = new Toast(context: this);  
    toast.setGravity(Gravity.CENTER, xOffset: 0, yOffset: 0);  
    toast.setDuration(Toast.LENGTH_LONG);  
  
    LayoutInflater inflater = getLayoutInflater();  
    View layout = inflater.inflate(  
        R.layout.toast_layout,  
        (LinearLayout) findViewById(R.id.toast_container)  
    );  
  
    TextView textView = (TextView) layout.findViewById(R.id.toast_message);  
    textView.setText("Eu sou um toast customizado!");  
  
    toast.setView(layout);  
    toast.show();  
}
```

5

Crie um método `exibirToastCustomizado()` para exibir o Toast

```
public void exibirToastCustomizado(View view) {  
    Toast toast = new Toast(context: this);  
    toast.setGravity(Gravity.CENTER, xOffset: 0, yOffset: 0);  
    toast.setDuration(Toast.LENGTH_LONG);
```

```
    LayoutInflater inflater = getLayoutInflater();  
    View layout = inflater.inflate(  
        R.layout.toast_layout,  
        (LinearLayout) findViewById(R.id.  
    );
```

```
    TextView textView = (TextView) layout.findViewById(R.id.textView);  
    textView.setText("Eu sou um toast customizado");  
  
    toast.setView(layout);  
    toast.show();  
}
```

1

- Crie um Toast
- Centralize-o através da constante `Gravity.CENTER`. Os valores `xOffset` e `yOffset` afastam horizontal e verticalmente da esquerda (medindo em pixels).
- Defina a duração

5

Crie um método *exibirToastCustomizado()* para exibir o Toast

```
public void exibirToastCustomizado(View view) {  
    Toast toast = new Toast( context: this );  
    toast.setGravity( Gravity.CENTER, xOffset: 0, yOffset: 0 );  
    toast.setDuration( Toast.LENGTH_LONG );  
  
    LayoutInflater inflater = getLayoutInflater();  
    View layout = inflater.inflate(  
        R.layout.toast_layout,  
        (LinearLayout) findViewById(R.id.toast_container)  
    );  
}
```

2

LayoutInflater: o conceito de *inflater* é utilizado para indicar a "criação" de um novo layout a partir de um arquivo XML. Neste caso, estamos "inflando" o layout que será exibido no Toast. O método *inflate()* recebe dois parâmetros: o arquivo XML a partir do qual o *inflater* criará a nova estrutura e o elemento que vai receber essa estrutura (neste caso, o próprio *LinearLayout* exposto no XML).

5

Crie um método *exibirToastCustomizado()* para exibir o Toast

- Obtenha o *TextView* para alterar o texto que será exibido na tela.
Atenção: note que neste caso o método *findViewById()* não é invocado a partir da Activity. Ele é invocado a partir do layout inflado.
- Defina o view do Toast e invoque o método *show()*

```
    View layout = inflater.inflate(  
        R.layout.toast_layout,  
        (LinearLayout) findViewById(R.id.toast_container)  
);
```

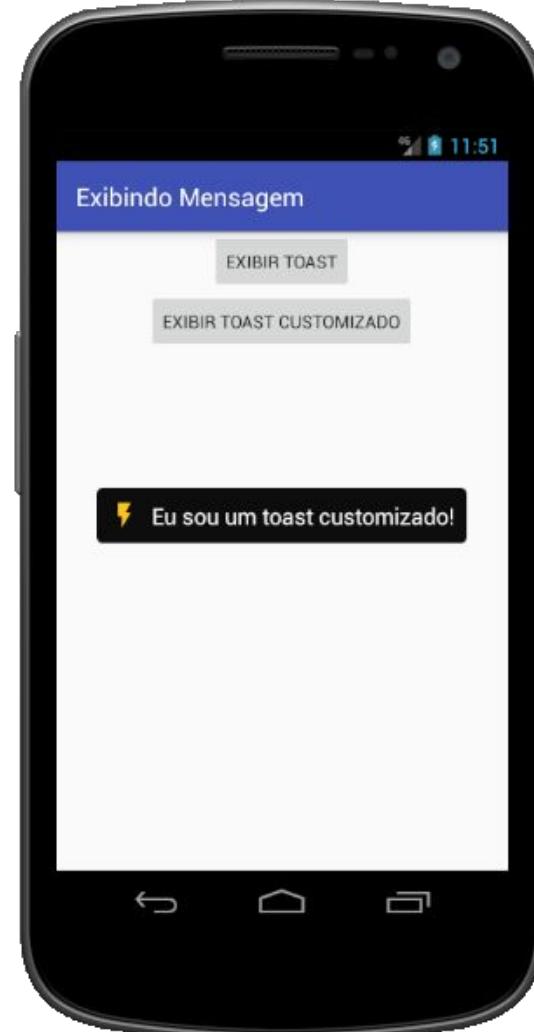
3

```
    TextView textView = (TextView) layout.findViewById(R.id.toast_message);  
    textView.setText("Eu sou um toast customizado!");  
  
    toast.setView(layout);  
    toast.show();  
}
```

Trabalhando com Toast Customizado (Exibindo Mensagem)

Por fim, acrescente um novo *Button* no layout principal do app e execute-o para ver o resultado!

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center_horizontal"  
    android:text="Exibir Toast Customizado"  
    android:onClick="exibirToastCustomizado" />
```



Trabalhar com toast (Exibindo Mensagem)

Por fim,
principal

Eu posso colocar um Button
dentro do Toast customizado?

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center_horizontal"  
    android:text="Exibir Toast Customizado"  
    android:onClick="exibirToastCustomizado" />
```

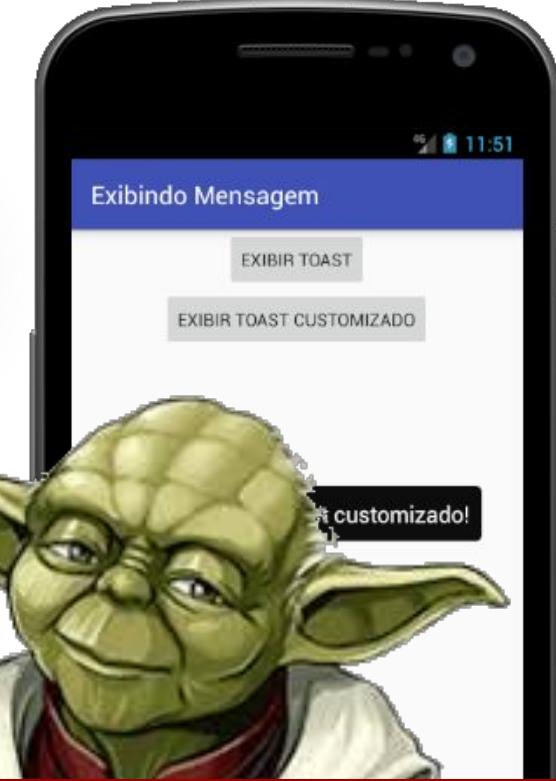


Trabalhar com toast (Exibindo)

Por fim,
principal

*Eu posso colocar um Button
dentro do Toast customizado?*

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center_horizontal"  
    android:text="Exibir Toast Customizado"  
    android:onClick="exibirToastCustomizado" />
```



Tecnicamente sim, porém caso haja necessidade do usuário interagir com a mensagem, a recomendação do Material Design é utilizar outro componente: **SnackBar**

Trabalhando com Snackbar (*Exibindo Mensagem*)

Vamos agora criar um Snackbar...



Curiosidade

Reza a lenda que o componente *SnackBar* recebeu este nome porque permite ao usuário realizar uma ação a partir de uma mensagem rápida.



Design Support Library

<https://developer.android.com/topic/libraries/support-library/packages?hl=pt-br#design>

O Android possui uma biblioteca chamada *Design Support Library* que oferece vários recursos apresentados pela especificação do Material Design que as bibliotecas padrões de *Support Libraries* não oferecem por default.

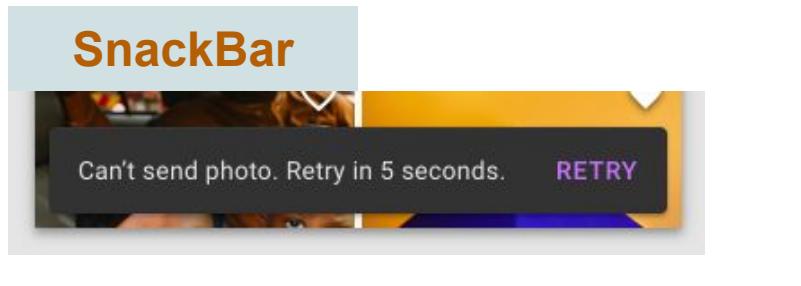


Design Support
Library

The image displays four examples of the Design Support Library:

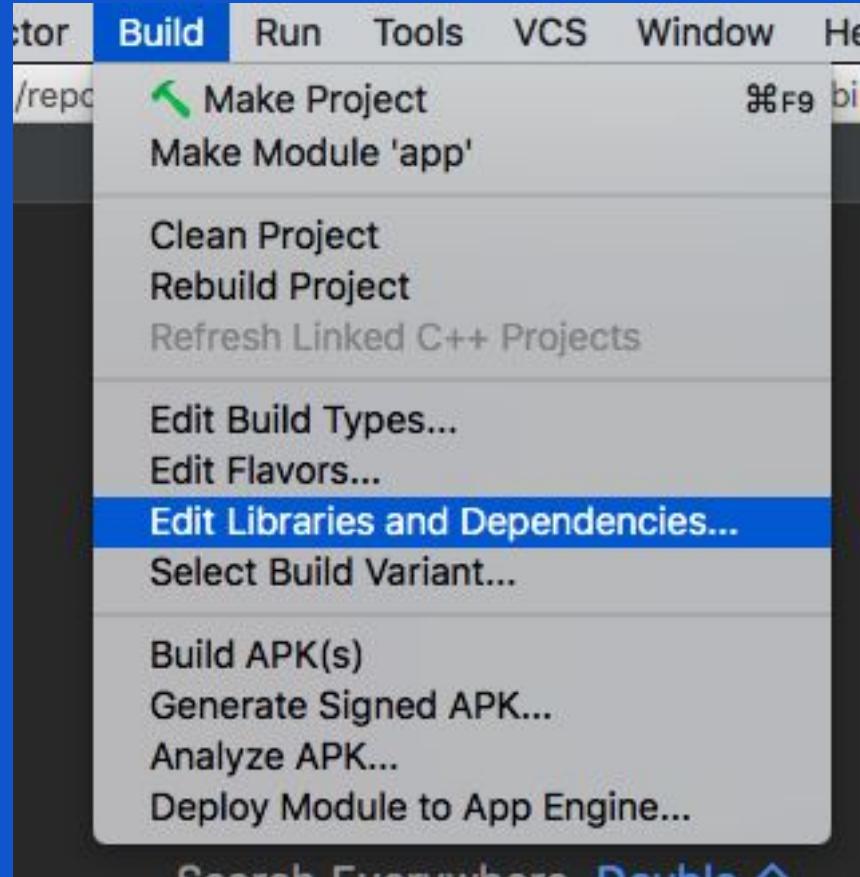
- FAB**: A floating action button (FAB) is shown in a white card interface, with a plus sign inside a black circle.
- Snackbar**: A horizontal message bar at the bottom of a screen with a "RETRY" button.
- Tabs**: A purple tab bar with three tabs labeled "DOGS", "CATS", and "BIRDS".
- Navigation Drawer**: A navigation drawer open on the right side of a screen, showing a profile picture and name "Sandra Adams".

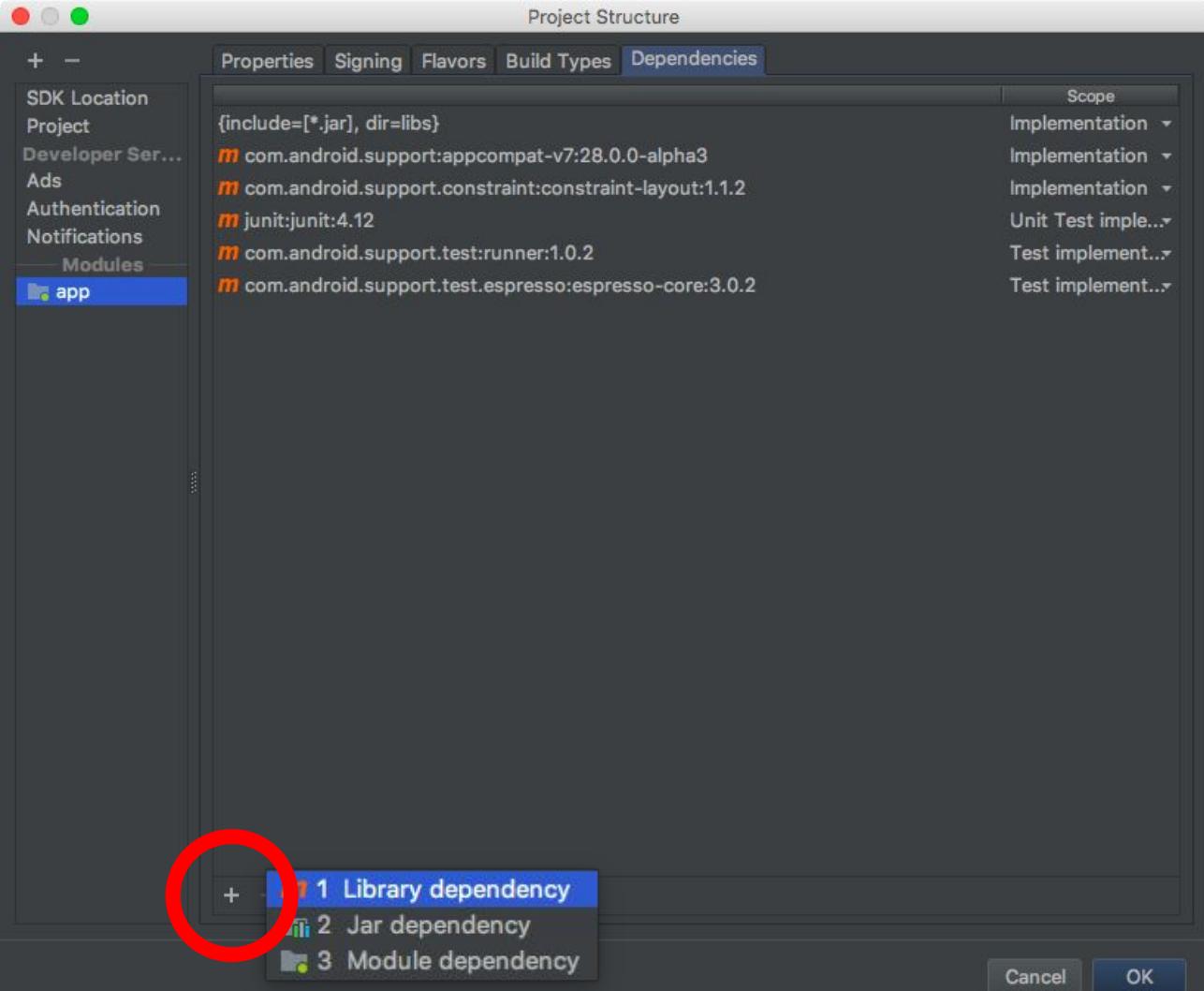
Utilizando o Snackbar



Design Support Library

Vamos importar a biblioteca
Design Support Library através
do menu *Edit Libraries and
Dependencies...*





OU



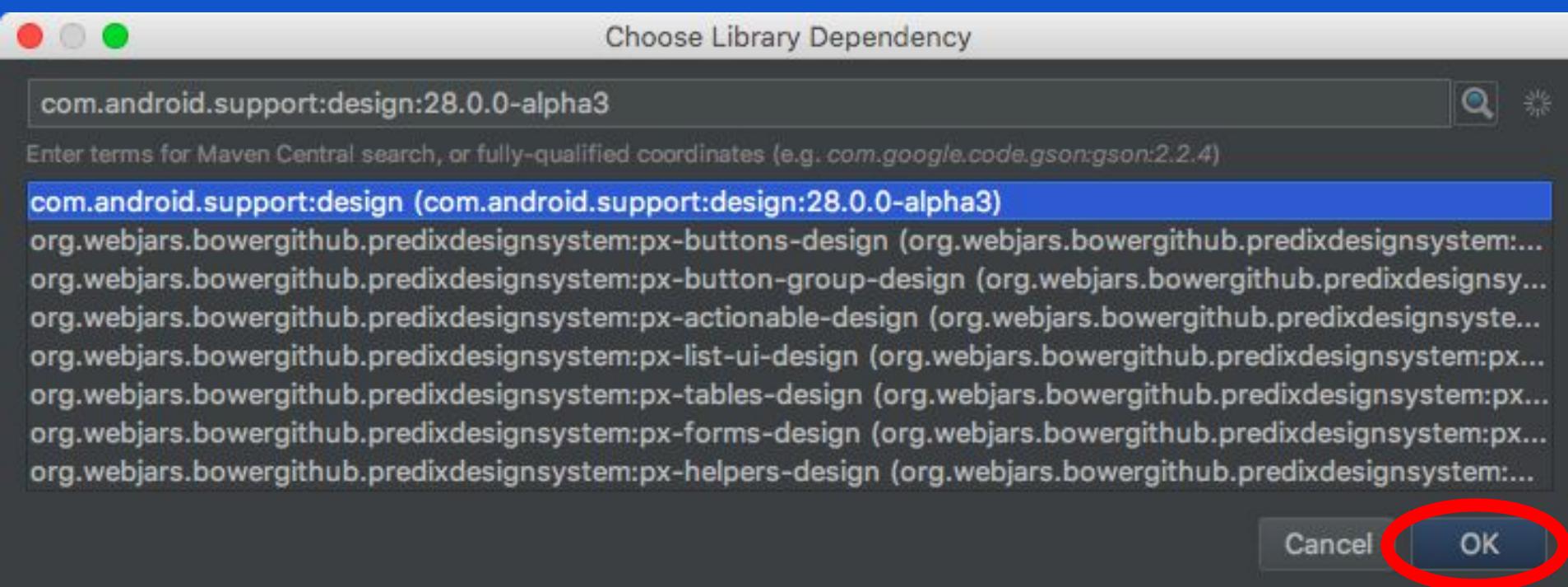
No **Windows** ou **Linux**,
o botão de importar nova
biblioteca fica no canto
superior à direita.



No **Mac** fica na parte
inferior (vide imagem)

Escolha a opção *Library dependency*

Digite "design" no campo de busca e selecione a biblioteca *com.android.support:design*. A versão exibida neste print certamente será diferente da exibida em seu Android Studio. Não se preocupe com isso, o Gradle vai resolver as dependências corretamente!



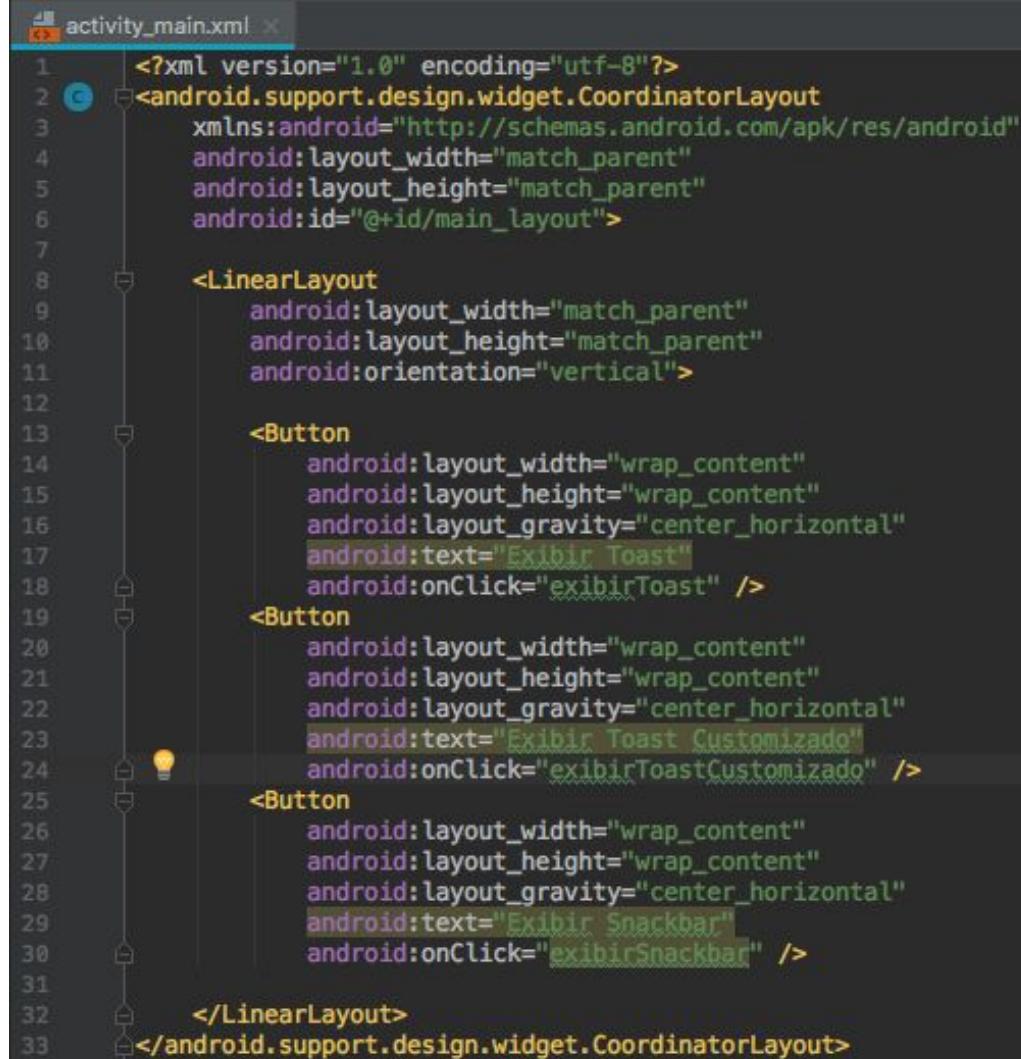
Trabalhando com Snackbar (Exibindo Mensagem)

Primeiramente vamos ajustar o layout para permitir o efeito de *swiping* que será aplicado ao Snackbar.

Para isso, basta envolver o layout que já temos com um *CoordinatorLayout*.

Aumente um ID a ele.

Aproveite para aumentar um *Button* chamado o método *exibirSnackbar()*



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/main_layout">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="Exibir Toast"
            android:onClick="exibirToast" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="Exibir Toast Customizado"
            android:onClick="exibirToastCustomizado" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="Exibir Snackbar"
            android:onClick="exibirSnackbar" />

    </LinearLayout>
</android.support.design.widget.CoordinatorLayout>
```

Trabalhando com Snackbar (*Exibindo Mensagem*)

Vamos começar criando um simples Snackbar (sem ação ou tratamento do evento de *swiping*).

```
public void exibirSnackbar(View view) {  
    CoordinatorLayout main = (CoordinatorLayout) findViewById(R.id.main_layout);  
    Snackbar snackbar = Snackbar.make(main, text: "Eu sou um Snackbar", Snackbar.LENGTH_LONG);  
    snackbar.show();  
}
```



Neste exemplo, o *swiping* já funciona, porém não há Listener escutando a ação.

Trabalhando com Snackbar (*Exibindo Mensagem*)

Vamos agora acrescentar uma ação "DESFAZER".

```
public void exibirSnackbar(View view) {
    CoordinatorLayout main = (CoordinatorLayout) findViewById(R.id.main_layout);
    Snackbar snackbar = Snackbar.make(main, text: "Eu sou um Snackbar", Snackbar.LENGTH_LONG);
    snackbar.setAction( text: "DESFAZER", new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText( context: MainActivity.this, text: "Parabéns, ação desfeita!", Toast.LENGTH_SHORT).show();
        }
    });
    snackbar.show();
}
```



Trabalhando com Snackbar (*Exibindo Mensagem*)

Vamos agora escutar os eventos do Snackbar, em especial o que é comumente chamado de **swipe-to-dismiss**.

```
snackbar.addCallback(new Snackbar.Callback() {  
    @Override  
    public void onDismissed(Snackbar snackbar, int event) {}  
    @Override  
    public void onShown(Snackbar snackbar) {}  
});
```

Primeiramente vamos adicionar um objeto de callback no Snackbar através do método `addCallback()`. A classe anônima `Snackbar.Callback` deve implementar dois métodos:

- `onDismissed()`: invocado após o fechamento do Snackbar
- `onShown()`: invocado após a exibição do Snackbar

Trabalhando com Snackbar (*Exibindo Mensagem*)

```
snackbar.addCallback(new Snackbar.Callback() {  
    @Override  
    public void onDismissed(Snackbar snackbar, int event) {}  
    @Override  
    public void onShown(Snack...});
```

Entretanto, existem várias ações que podem fechar o Snackbar!

<https://developer.android.com/reference/android/support/design/widget/Snackbar.Callback>

Vamos usar:

DISMISS_EVENT_SWIPE

DISMISS_EVENT_ACTION

Indicates that the Snackbar was dismissed via an action click.

DISMISS_EVENT_CONSECUTIVE

Indicates that the Snackbar was dismissed from a new Snackbar being shown.

DISMISS_EVENT_MANUAL

Indicates that the Snackbar was dismissed via a call to `dismiss()`.

DISMISS_EVENT_SWIPE

Indicates that the Snackbar was dismissed via a swipe.

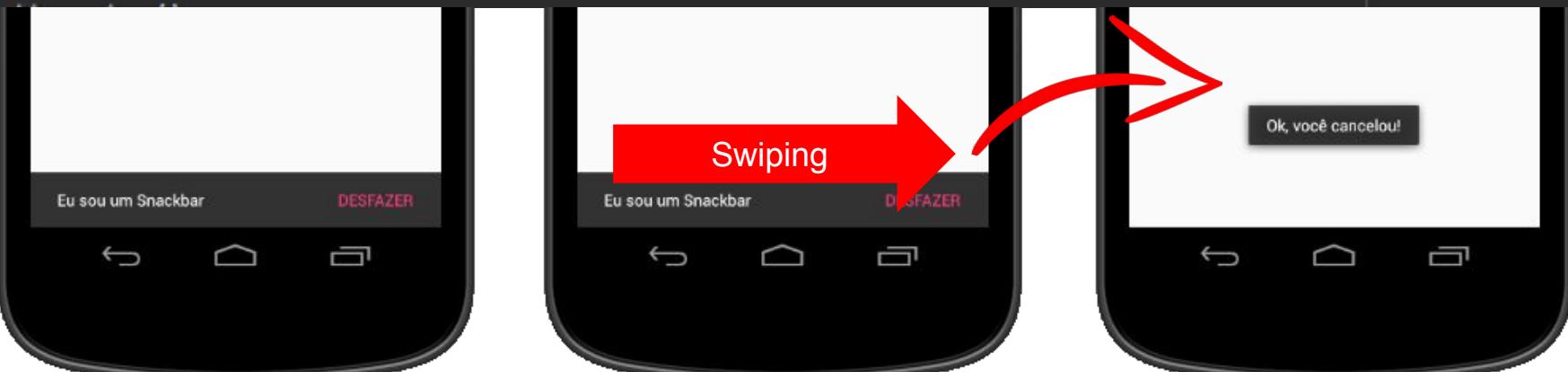
DISMISS_EVENT_TIMEOUT

Indicates that the Snackbar was dismissed via a timeout.

Trabalhando com Snackbar (*Exibindo Mensagem*)

No método `onDismissed()`, verifique se o código do evento é `DISMISS_EVENT_SWIPE` e, se for o caso, exiba uma mensagem.

```
snackbar.addCallback(new Snackbar.Callback() {  
    @Override  
    public void onDismissed(Snackbar snackbar, int event) {  
        if (event == Snackbar.Callback.DISMISS_EVENT_SWIPE) {  
            Toast.makeText(context: MainActivity.this, text: "Ok, você cancelou!", Toast.LENGTH_SHORT).show();  
        }  
    }  
    @Override  
    public void onShown(Snackbar snackbar) {}  
});
```



Exercício em Sala

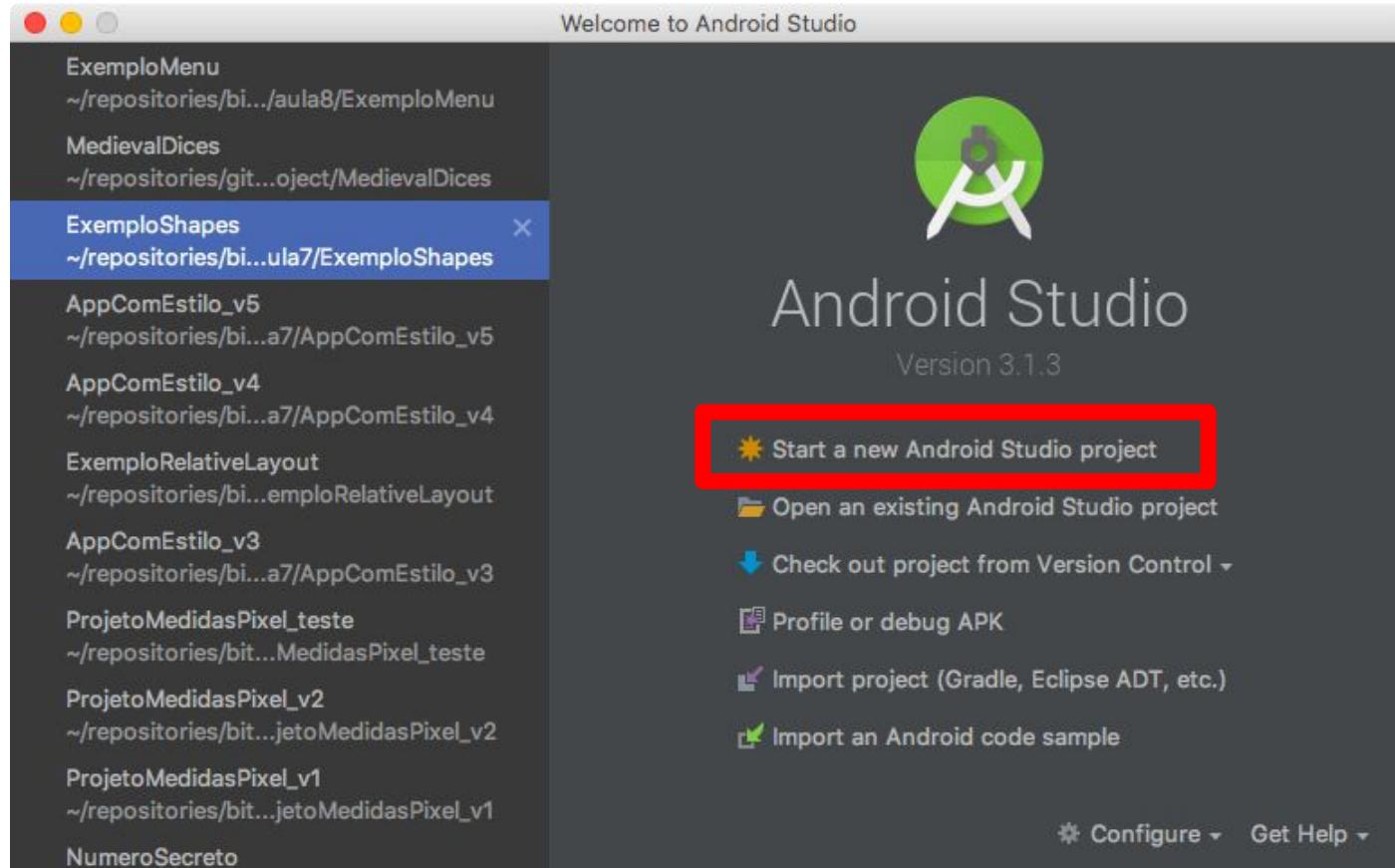
Crie um novo projeto chamado *Notepad* (*Empty Activity, API 15*). Ao clicar no botão "Limpar", o app deve exibir um *SnackBar* perguntando se o usuário realmente deseja apagar o texto. Ao clicar em APAGAR, deve-se limpar o texto.

Veja ao lado como criar um *EditText* com muitas linhas!

```
<EditText  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:padding="5dp"  
    android:background="@color/cinza"  
    android:textSize="20sp"  
    android:fontFamily="monospace"  
    android:gravity="top|left"  
    android:scrollbars="vertical" />
```



Resolvendo o Exercício



Resolvendo o Exercício

Application name

Notepad

Company domain

lgapontes.com

Project location

/Users/lgapontes/repositories/bitbucket/aulas/desenvolvimento-mobile/aula8/Notepad_v1

...

Package name

com.lgapontes.notepad_v1

Done

Include C++ support

Include Kotlin support

Resolvendo o Exercício

Phone and Tablet

API 15: Android 4.0.3 (IceCreamSandwich)



By targeting **API 15 and later**, your app will run on approximately **100%** of devices. [Help me choose](#)

Include Android Instant App support

Wear

API 21: Android 5.0 (Lollipop)



TV

API 21: Android 5.0 (Lollipop)



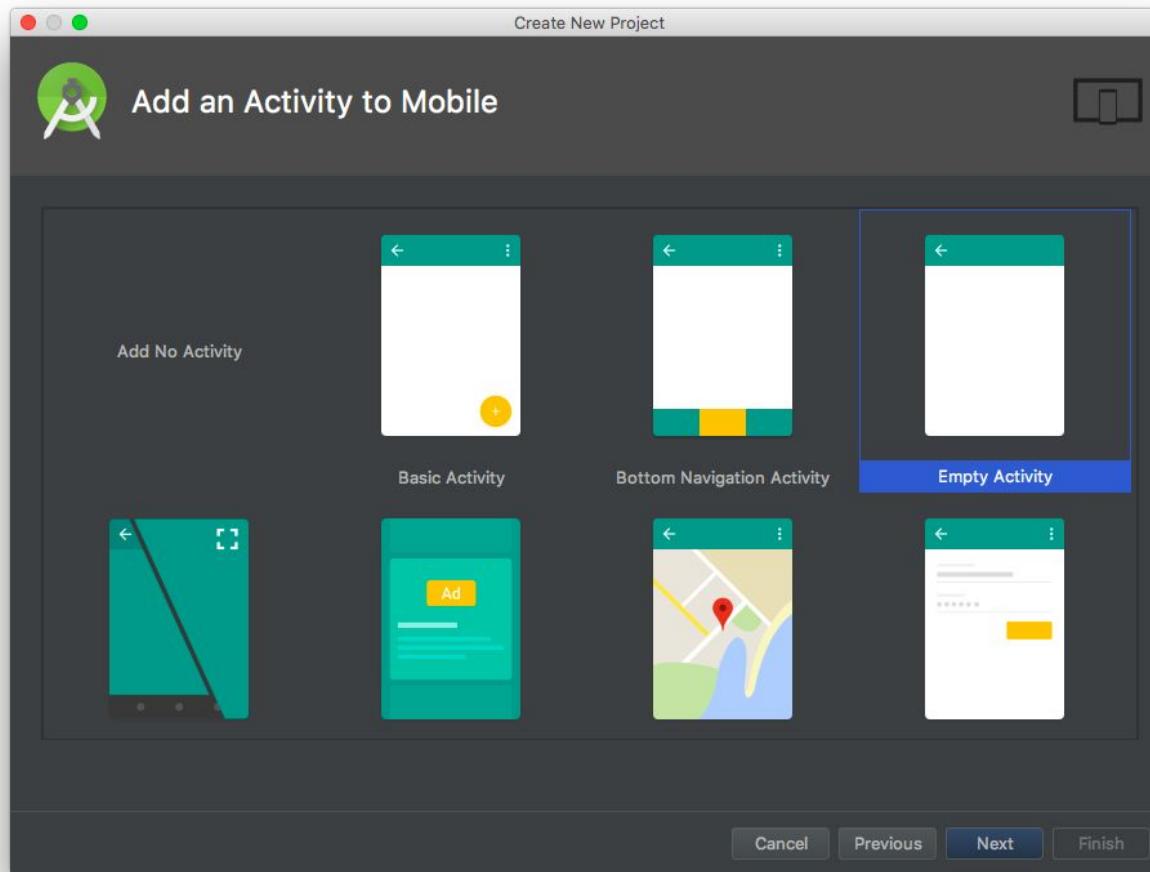
Android Auto

Android Things

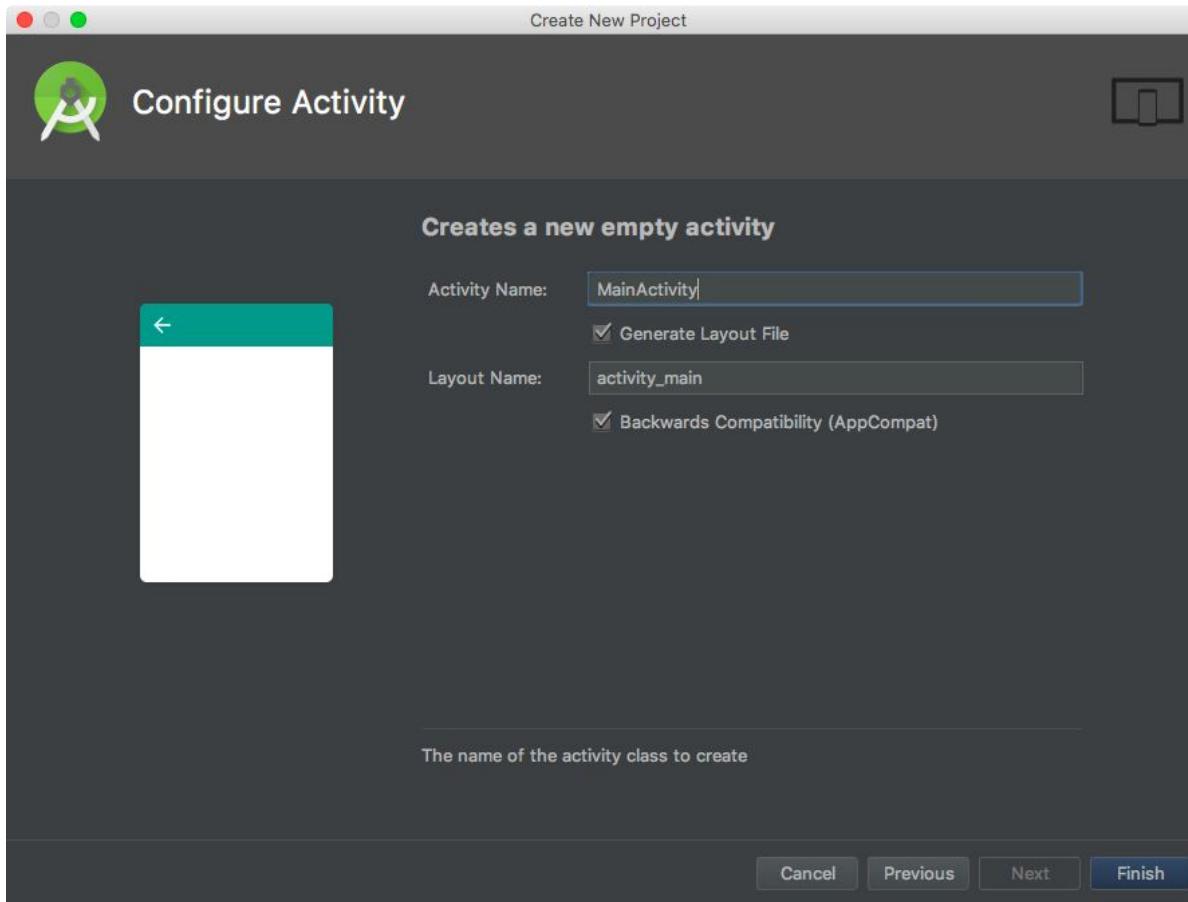
API 24: Android 7.0 (Nougat)



Resolvendo o Exercício



Resolvendo o Exercício

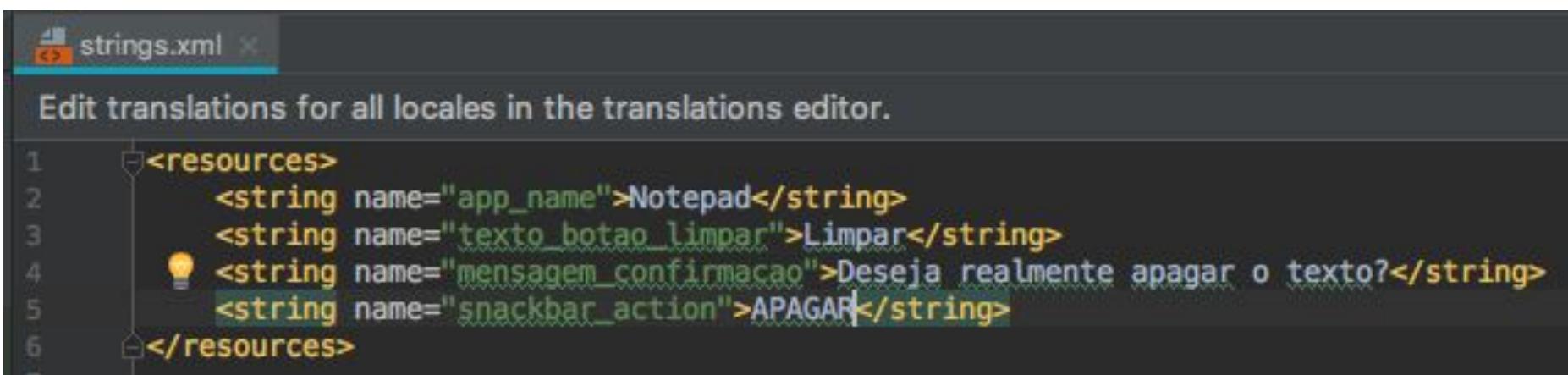


Resolvendo o Exercício



The screenshot shows the Android Studio interface with the colors.xml file open. The code defines four color resources:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="cinza">#efefef</color>
</resources>
```



The screenshot shows the Android Studio interface with the strings.xml file open. The code defines several string resources, including one with a warning icon indicating a translation is missing:

```
1 <resources>
2     <string name="app_name">Notepad</string>
3     <string name="texto_botao_limpar">Limpar</string>
4     <string name="mensagem_confirmacao">Deseja realmente apagar o texto?</string>
5     <string name="snackbar_action">APAGAR</string>
6 </resources>
```

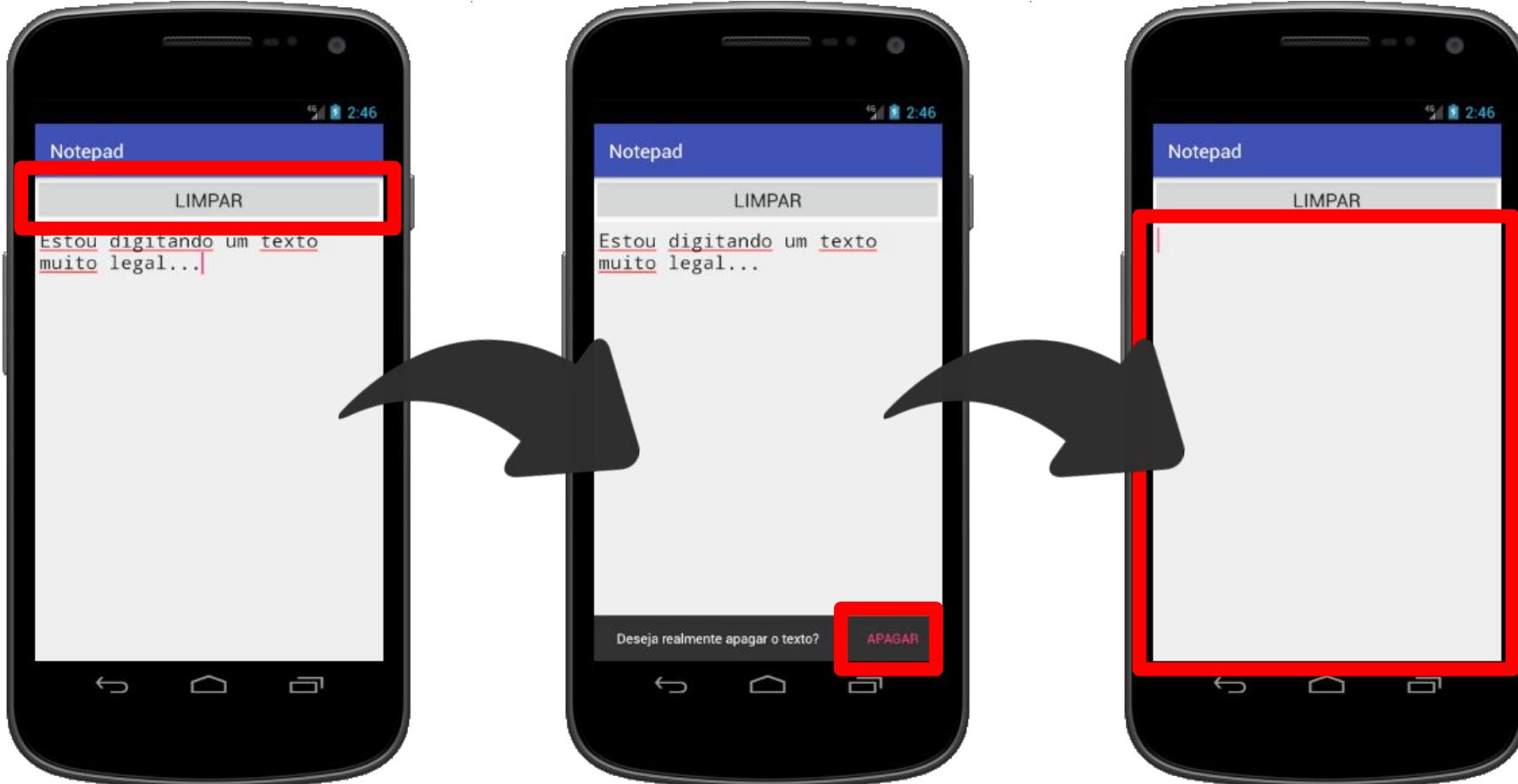
Resolvendo o Exercício

```
activity_main.xml <?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/coordinator_layout">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/texto_botao_limpar"
            android:textSize="20sp"
            android:id="@+id/botao_limpar"/>
        <EditText
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:padding="5dp"
            android:background="@color/cinza"
            android:textSize="20sp"
            android:fontFamily="monospace"
            android:gravity="top|left"
            android:scrollbars="vertical"
            android:id="@+id/edit_text" />
    </LinearLayout>
</android.support.design.widget.CoordinatorLayout>
```

Resolvendo o Exercício

```
public class MainActivity extends AppCompatActivity {  
    private CoordinatorLayout layout;  
    private Button botaoLimpar;  
    private EditText editText;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        layout = (CoordinatorLayout) findViewById(R.id.coordinator_layout);  
        editText = (EditText) findViewById(R.id.edit_text);  
  
        botaoLimpar = (Button) findViewById(R.id.botao_limpar);  
        botaoLimpar.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                Snackbar snackbar = Snackbar.make(layout, "Deseja realmente apagar o texto?", Snackbar.LENGTH_LONG);  
                snackbar.setAction("APAGAR", new View.OnClickListener() {  
                    @Override  
                    public void onClick(View view) {  
                        editText.setText("");  
                    }  
                });  
                snackbar.show();  
            }  
        });  
    }  
}
```

Resolvendo o Exercício



Mas esse botão "Limpar"
em layout mobile é uma
boa prática?



Notepad

LIMPAR

Estou digitando um texto
muito legal...

*Mas esse botão "Limpar"
em layout mobile é uma
boa prática?*

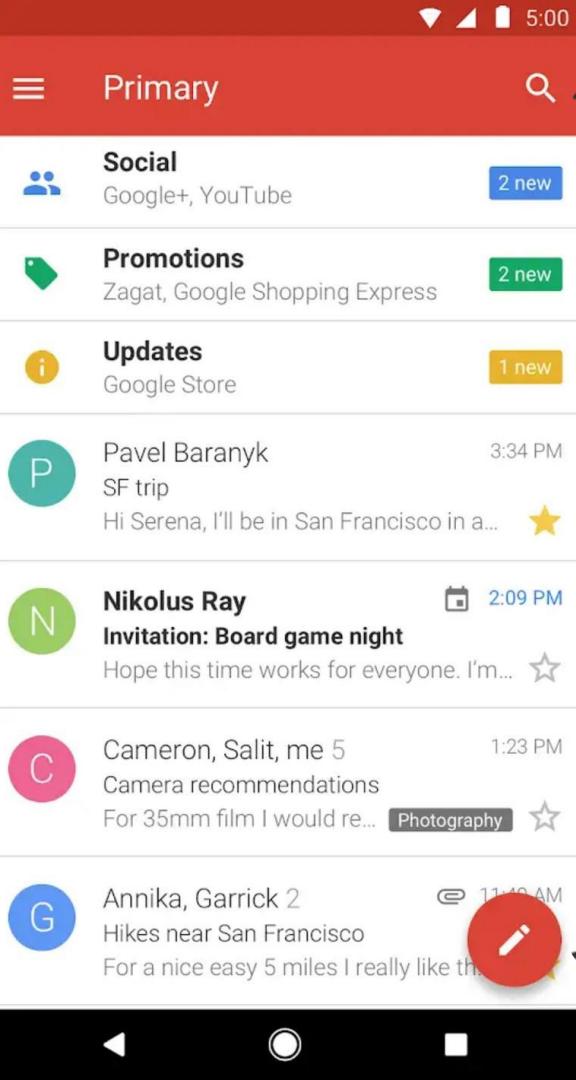


Discussão em sala:

Abram algum aplicativo mobile em seu celular Android e veja onde ficam posicionados os botões de ação...

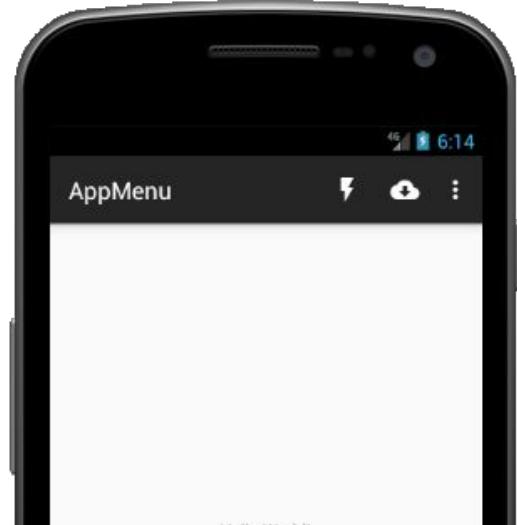
Navigation Drawer

Este componente
será estudado no
futuro pois antes
precisamos abordar
os conceitos de
Fragment



ActionBar Menu

Floating Action Button

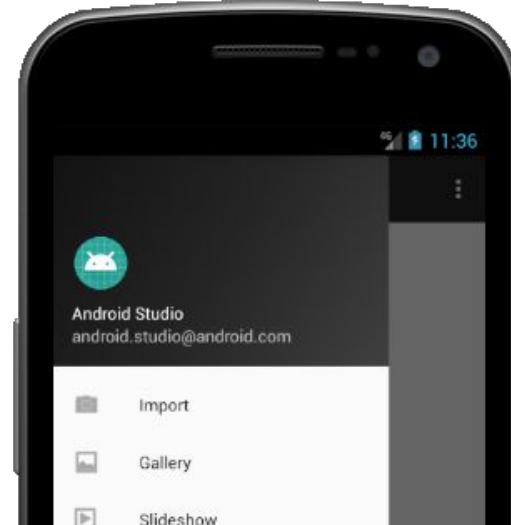


ActionBar Menu

Recomendado para casos mais simples, com poucas Views, onde as principais ações podem ficar expostas diretamente sem poluir o ActionBar Menu.

VS Navigation Drawer

Recomendado para Apps mais complexas, com muitas Views, cuja navegação seria facilitada quando houver uma Navigation Drawer.



ActionBar Menu

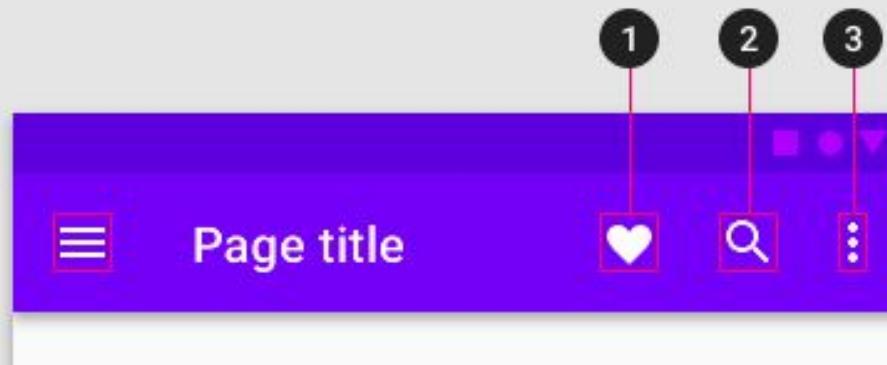
Material Design:

<https://material.io/design/components/app-bars-top.html#anatomy>

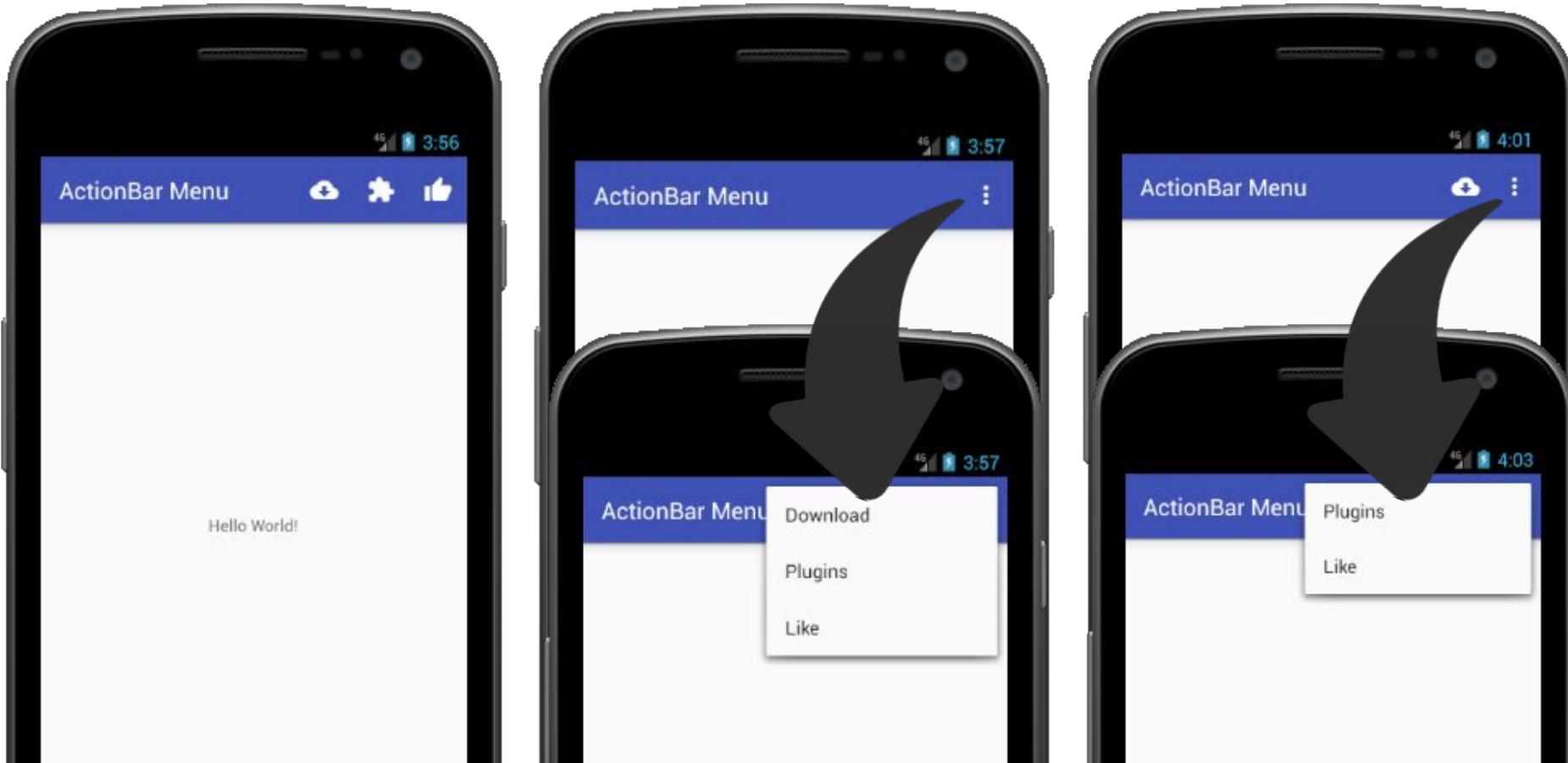
Icon placement

Place most-used actions on the left, progressing towards the least-used actions on the far right. Any remaining actions that don't fit on the app bar can go into an overflow menu.

Actions move into and out of the overflow menu as the app bar width changes, such as if a device is rotated from landscape to portrait orientation. More guidance on this is available in **top app bar behavior**.

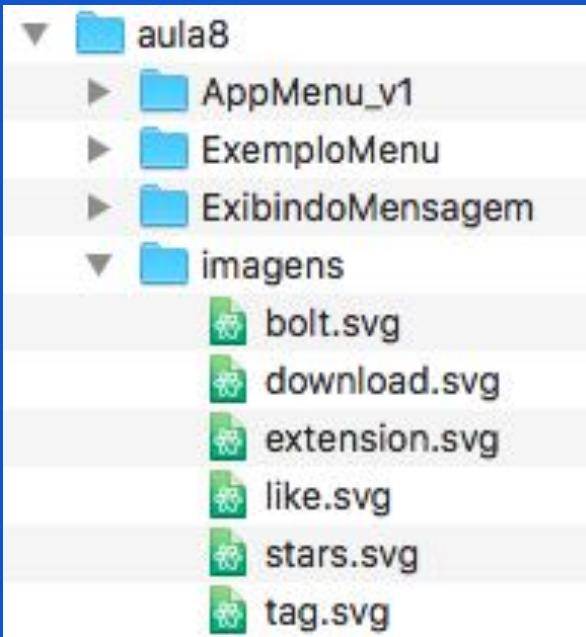


ActionBar Menu



Imagens para os menus (em formato SVG)

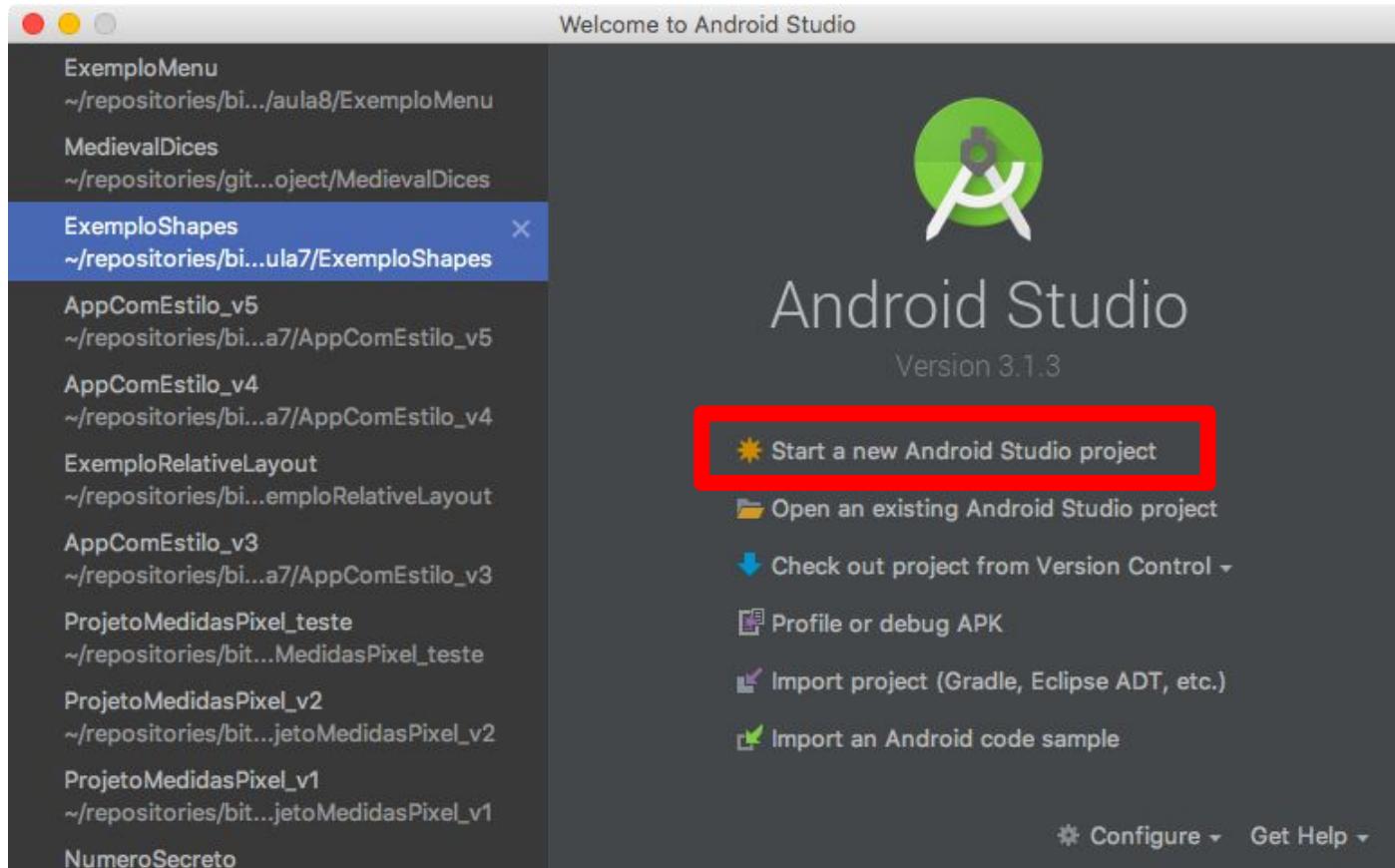
Baixe 6 imagens no Material Design Icons (<https://material.io/tools/icons/>), com formato SVG. Caso prefira, na pasta *imagens* da Aula 8 do EAD já existem 6 imagens.



As imagens devem se chamar:

- bolt.svg
- download.svg
- extension.svg
- like.svg
- stars.svg
- tag.svg

Trabalhando com ActionBar Menu (AppMenu)



Trabalhando com ActionBar Menu (AppMenu)

Ap The name that will be shown in the Android launcher for this application
AppMenu

Company domain
lgapontes.com

Project location
/Users/lgapontes/repositories/bitbucket/aulas/desenvolvimento-mobile/aula8/AppMenu_v1 ...

Package name
com.lgapontes.appmenu_v1 Done

Include C++ support

Include Kotlin support

Trabalhando com ActionBar Menu (AppMenu)

Phone and Tablet

API 15: Android 4.0.3 (IceCreamSandwich) ▾

By targeting **API 15 and later**, your app will run on approximately **100%** of devices. [Help me choose](#)

Include Android Instant App support

Wear

API 21: Android 5.0 (Lollipop) ▾

TV

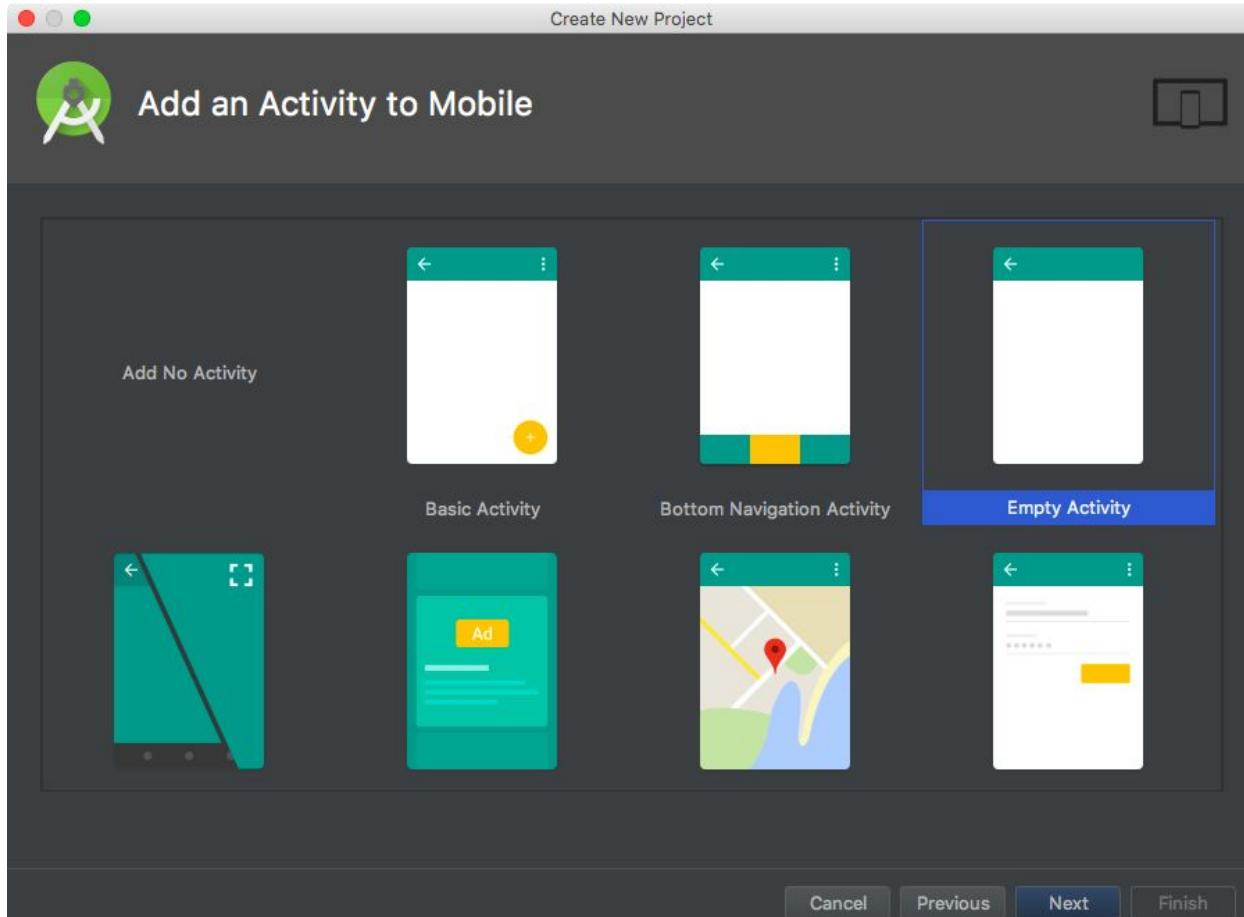
API 21: Android 5.0 (Lollipop) ▾

Android Auto

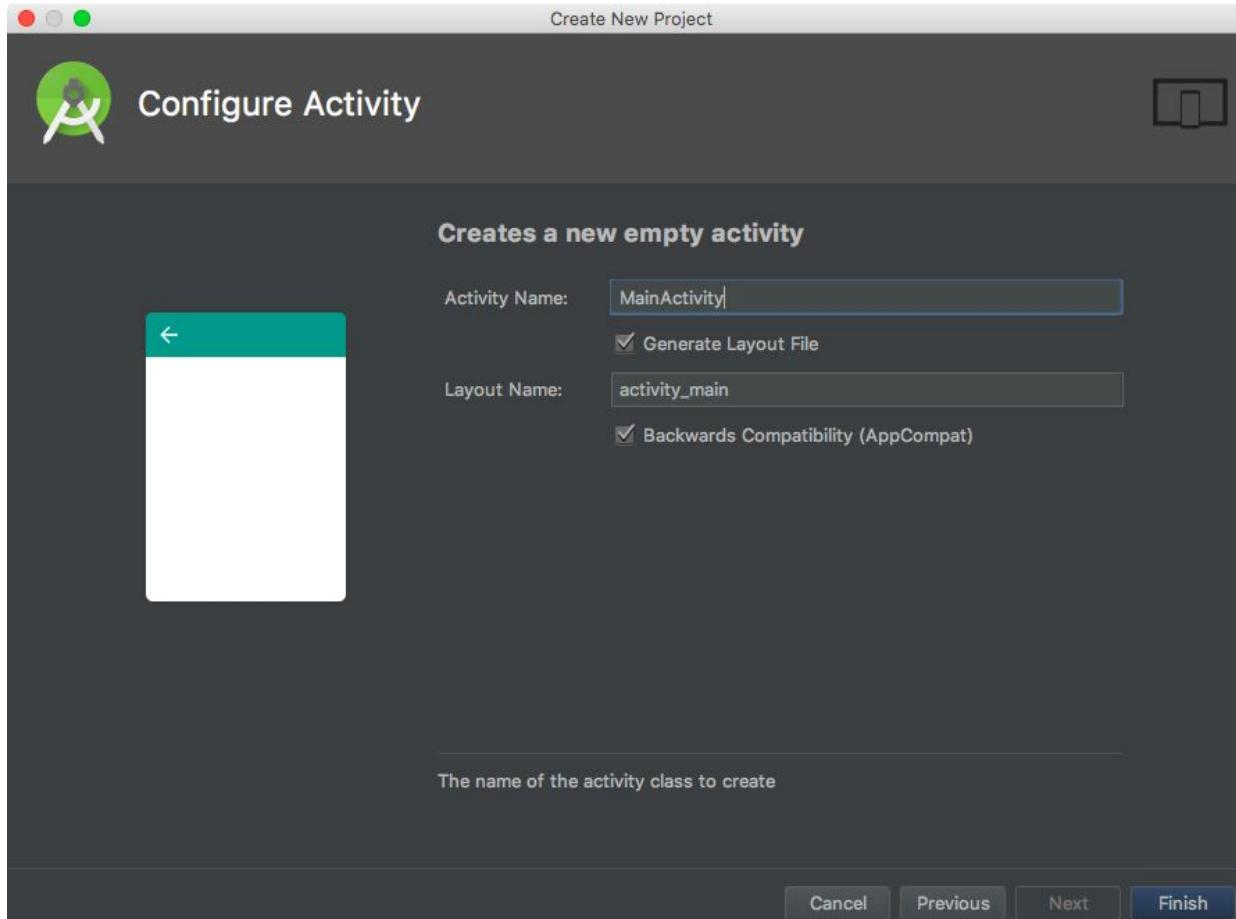
Android Things

API 24: Android 7.0 (Nougat) ▾

Trabalhando com ActionBar Menu (AppMenu)

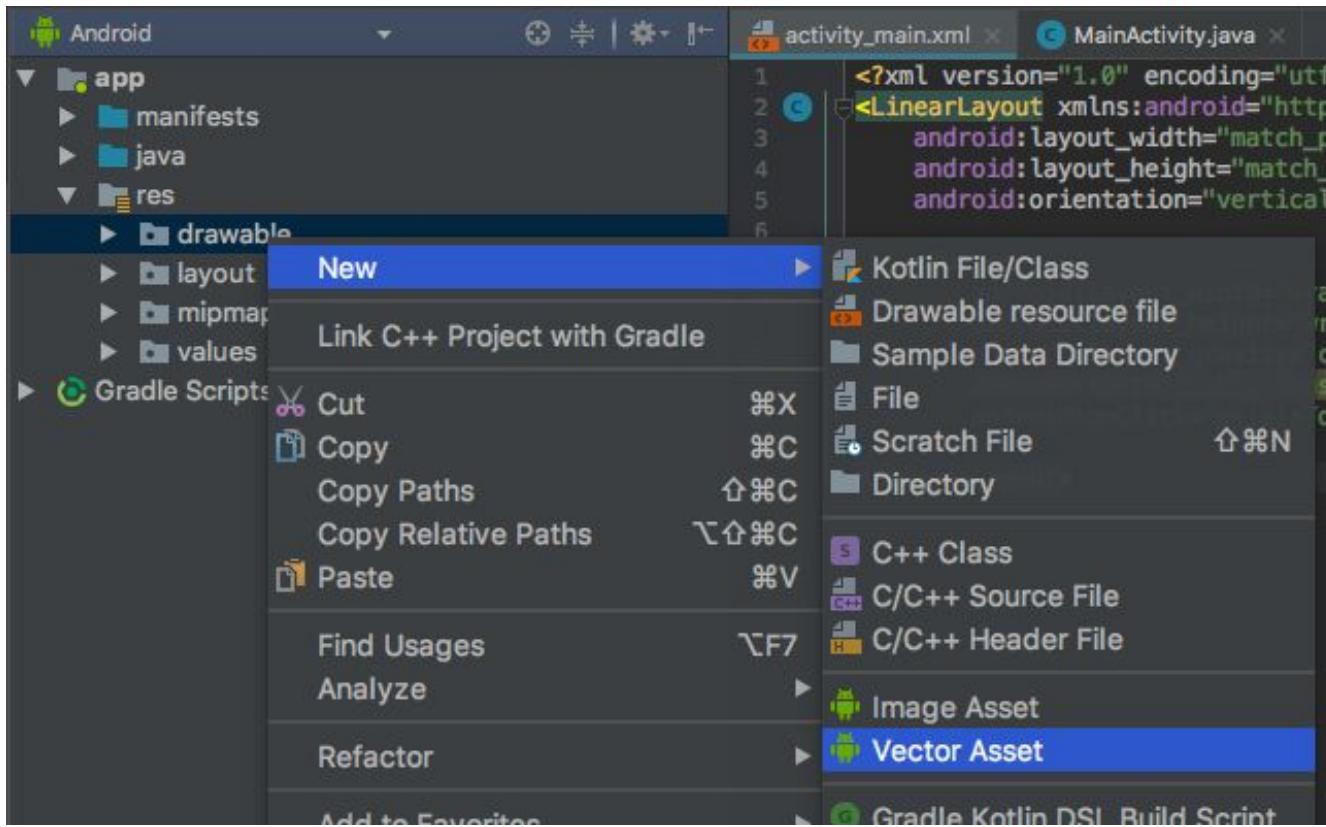


Trabalhando com ActionBar Menu (AppMenu)



Trabalhando com ActionBar Menu (AppMenu)

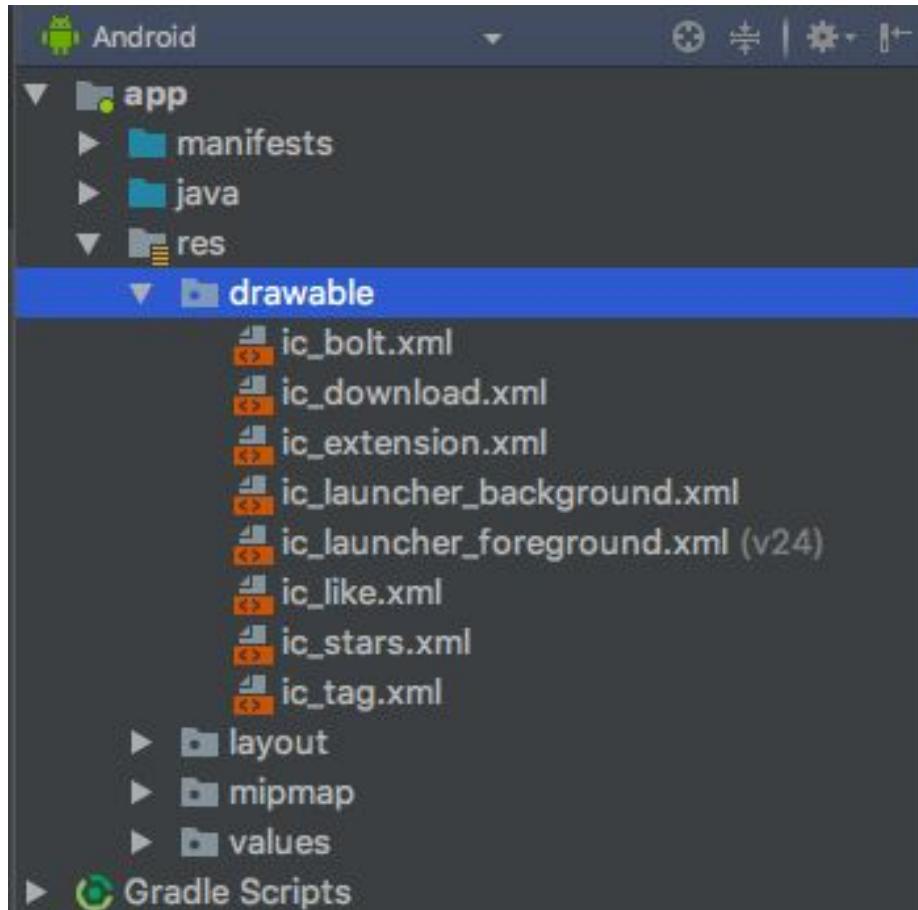
Importe as imagens como Vector Asset para o projeto AppMenu.



Trabalhando com ActionBar Menu (AppMenu)

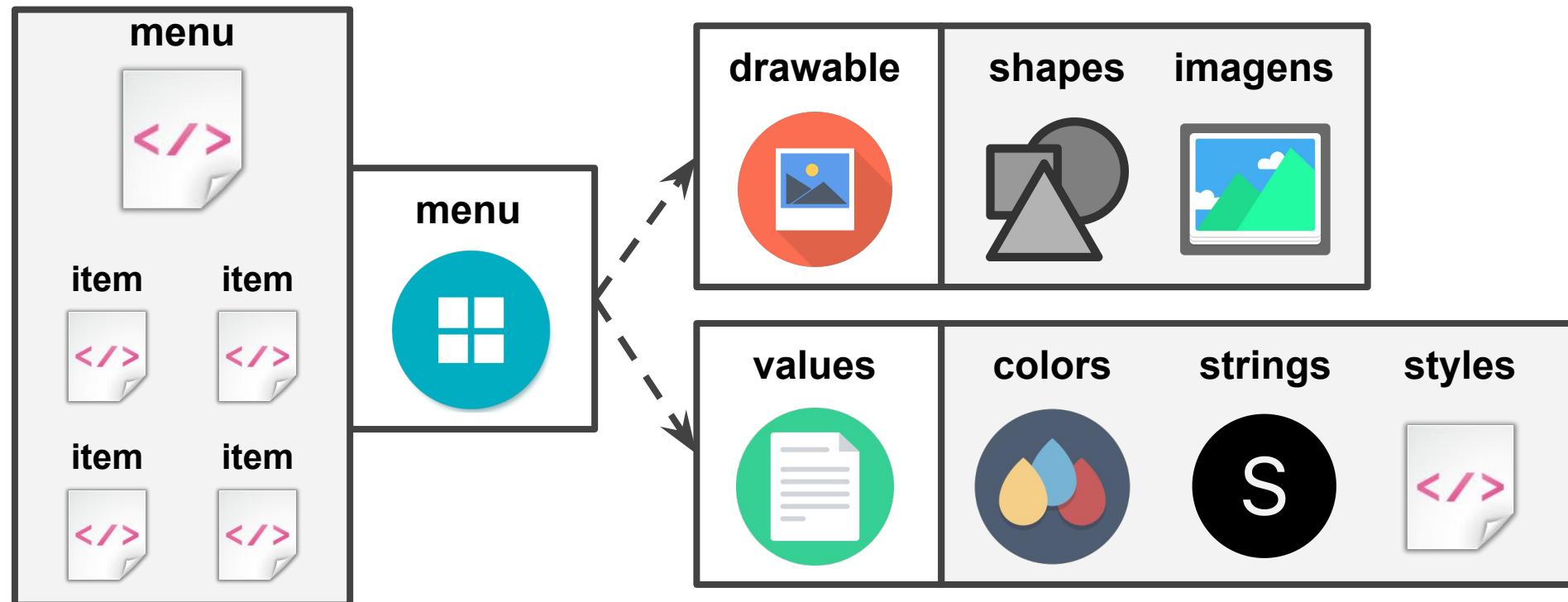
Ao final do processo, você deve ter os seguintes drawables:

- ic_bolt.xml
- ic_download.xml
- ic_extension.xml
- ic_like.xml
- ic_starts.xml
- ic_tag.xml



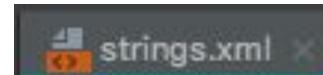
Trabalhando com ActionBar Menu (AppMenu)

Um menu é um arquivo de Layout disposto na pasta `res/menu/` que contém umas série de tags `<item>`



Trabalhando com ActionBar Menu (AppMenu)

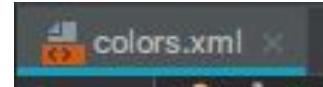
Vamos começar definindo a string dos menus. Menus exibidos como ícones na ActionBar não exibem as labels. Por outro lado, menus exibidos no *Overflow Menu* não exibem as imagens. Vamos também definir as cores.



The screenshot shows the Android Studio interface with the strings.xml file open. The title bar says "strings.xml". The code editor displays the following XML:

```
<resources>
    <string name="app_name">AppMenu</string>

    <string name="bolt">Flash</string>
    <string name="download">Download</string>
    <string name="plugin">Plugin</string>
    <string name="like">Like</string>
    <string name="stars">Importante</string>
    <string name="tag">Marcador</string>
</resources>
```



The screenshot shows the Android Studio interface with the colors.xml file open. The title bar says "colors.xml". The code editor displays the following XML:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#252525</color>
    <color name="colorPrimaryDark">#121212</color>
    <color name="colorAccent">#4967ff</color>
    <color name="branco">#FFFFFF</color>
</resources>
```

O Android que controla as cores do *ActionBar*. Para sobreescriver este comportamento, devemos criar um estilo e aplicá-lo no tema geral.



```
1 <resources>
2     <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
3         <item name="colorPrimary">@color/colorPrimary</item>
4         <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
5         <item name="colorAccent">@color/colorAccent</item>
6
7         <item name=" actionBarTheme">@style/ActionBarTheme</item>
8     </style>
9     <style name="ActionBarTheme">
10        <item name="tint">@color/branco</item>
11        <item name="iconTint">@color/branco</item>
12        <item name="titleTextColor">@color/branco</item>
13        <item name="actionMenuTextColor">@color/branco</item>
14    </style>
15 </resources>
```

The code defines a new style 'AppTheme' that inherits from 'Theme.AppCompat.Light.DarkActionBar'. It then overrides three items: 'colorPrimary', 'colorPrimaryDark', and 'colorAccent', each pointing to a color resource named 'colorPrimary', 'colorPrimaryDark', and 'colorAccent' respectively. It also overrides the 'actionBarTheme' item to point to a new style named 'ActionBarTheme'. This new style 'ActionBarTheme' overrides four items: 'tint', 'iconTint', 'titleTextColor', and 'actionMenuTextColor', all pointing to a color resource named 'branco' (white).



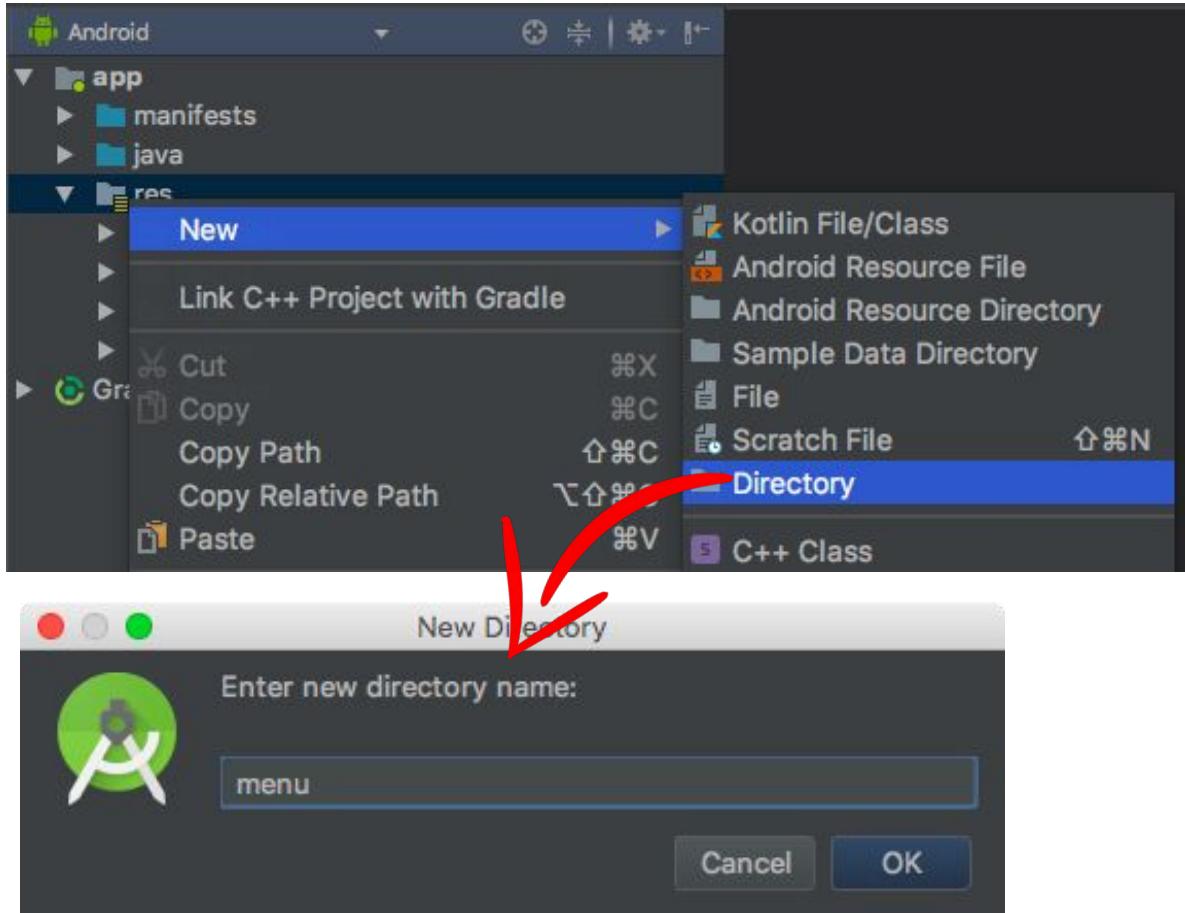
- tint ⇒ cor do ícone (três pontos) do *Overflow Menu*
- iconTint ⇒ cor do ícone que é exibido no ActionBar
- titleTextColor ⇒ cor do título da ActionBar
- actionMenuItemTextColor ⇒ para exibição do app em orientação *landscape* e configuração para exibição do título e do ícone (ao mesmo tempo), define a cor do nome do item que aparece sobre a ActionBar

```
10 ■  
11 ■  
12 ■  
13 ■  
14 </style>  
15 </resources>
```

```
<item name="tint">@color/branco</item>  
<item name="iconTint">@color/branco</item>  
<item name="titleTextColor">@color/branco</item>  
<item name="actionMenuItemTextColor">@color/branco</item>
```

↓

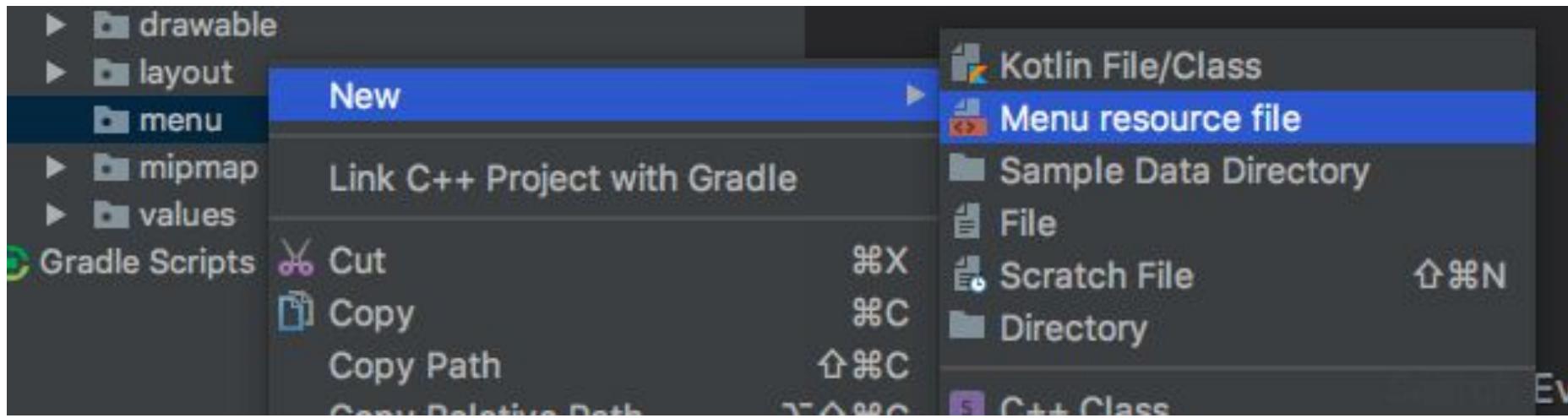
Trabalhando com ActionBar Menu (AppMenu)



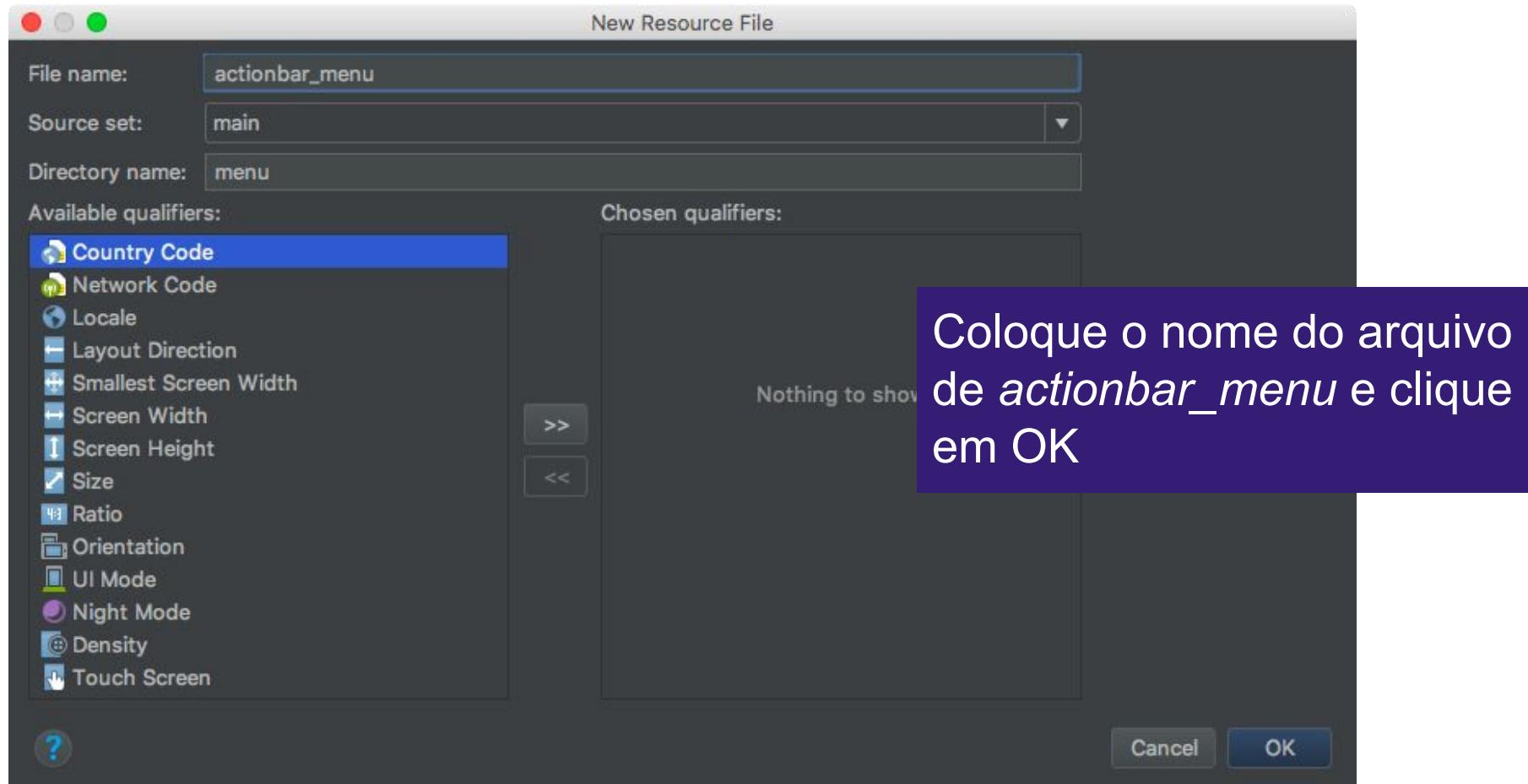
Algumas versões do Android Studio já criam o projeto com a pasta `res/menu/` por padrão. Caso esta pasta não exista, crie-a clicando com o botão direito na pasta `res`, menu `New, Directory`

Trabalhando com ActionBar Menu (AppMenu)

Agora clique com o botão direito na pasta `res/menu/`, selecione a opção `New`, submenu *Menu resource file*

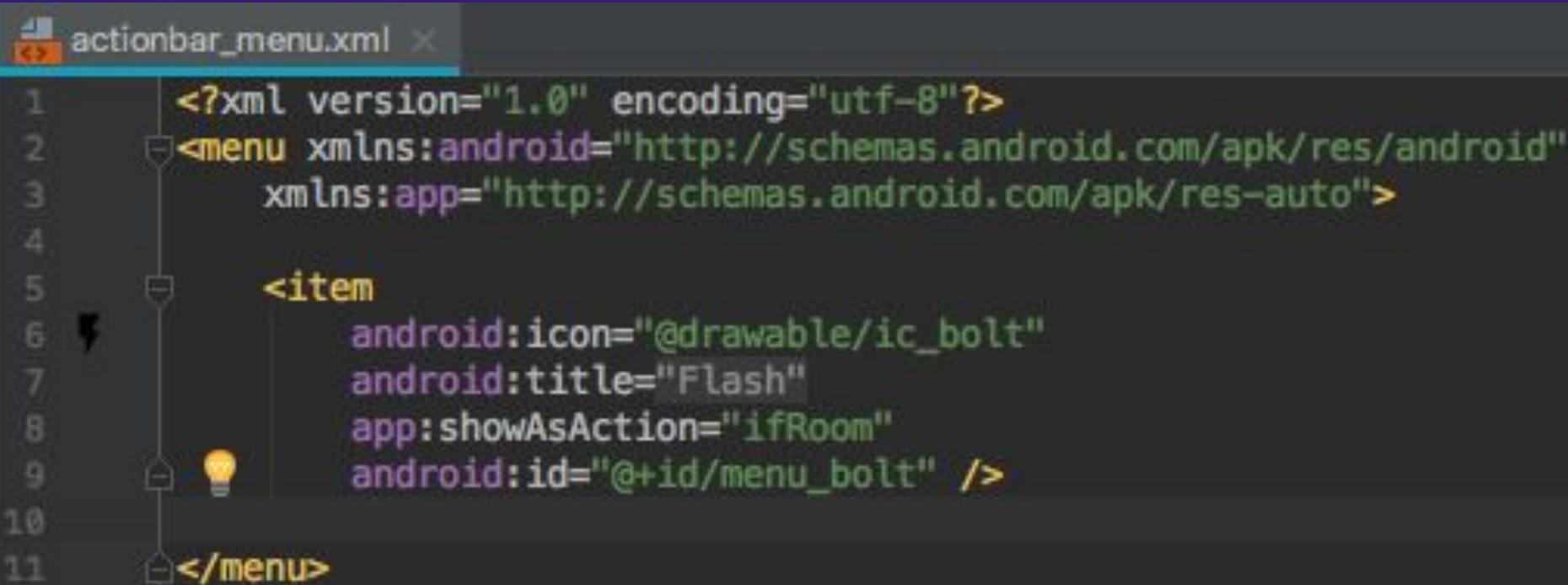


Trabalhando com ActionBar Menu (AppMenu)



Trabalhando com ActionBar Menu (AppMenu)

Vejamos um código com um único item de menu



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto">
4
5   <item
6     android:icon="@drawable/ic_bolt"
7     android:title="Flash"
8     app:showAsAction="ifRoom"
9     android:id="@+id/menu_bolt" />
10
11 </menu>
```

Trabalhando com ActionBar Menu (AppMenu)

- android:icon ⇒ Aponta para o drawable
- android:title ⇒ Define o nome do menu (para o caso de exibição dentro do *Overflow Menu*)
- android:id ⇒ ID do item. Será necessário para verificar qual menu foi clicado.

```
3   <menu>
4
5     <item
6       android:icon="@drawable/ic_bolt"
7       android:title="Flash"
8       app:showAsAction="ifRoom"
9       android:id="@+id/menu_bolt" />
10
11 </menu>
```

Trabalhando com ActionBar Menu (AppMenu)

- ifRoom ⇒ exibe sobre a *ActionBar* se houver espaço
- withText ⇒ exibe o ícone e o texto sobre a *ActionBar* se houver espaço. Como o texto aparece ao lado do ícone, geralmente só é utilizado em orientação *landscape*. Pode acumular com outro parâmetro através do pipe | (exemplo: always |withText)
- never ⇒ sempre será exibido no *Overflow Menu*
- always ⇒ sempre será exibido sobre a *ActionBar*

```
    android:icon="@drawable/ic_bolt"
    android:title="Flash"
    app:showAsAction="ifRoom"
    android:id="@+id/menu_bolt" />
```

app:showAsAction="ifRoom"

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto">
4      <item
5          android:icon="@drawable/ic_bolt" android:title="@string/bolt"
6          app:showAsAction="ifRoom" android:id="@+id/menu_bolt" />
7      <item
8          android:icon="@drawable/ic_download" android:title="@string/download"
9          app:showAsAction="ifRoom" android:id="@+id/menu_download" />
10     <item
11         android:icon="@drawable/ic_extension" android:title="@string/plugin"
12         app:showAsAction="ifRoom" android:id="@+id/menu_plugin" />
13     <item
14         android:icon="@drawable/ic_like" android:title="@string/like"
15         app:showAsAction="ifRoom" android:id="@+id/menu_like" />
16     <item
17         android:icon="@drawable/ic_stars" android:title="@string/stars"
18         app:showAsAction="ifRoom" android:id="@+id/menu_stars" />
19     <item
20         android:icon="@drawable/ic_tag" android:title="@string/tag"
21         app:showAsAction="ifRoom" android:id="@+id/menu_tag" />
22  </menu>
```

Trabalhando com ActionBar Menu (AppMenu)

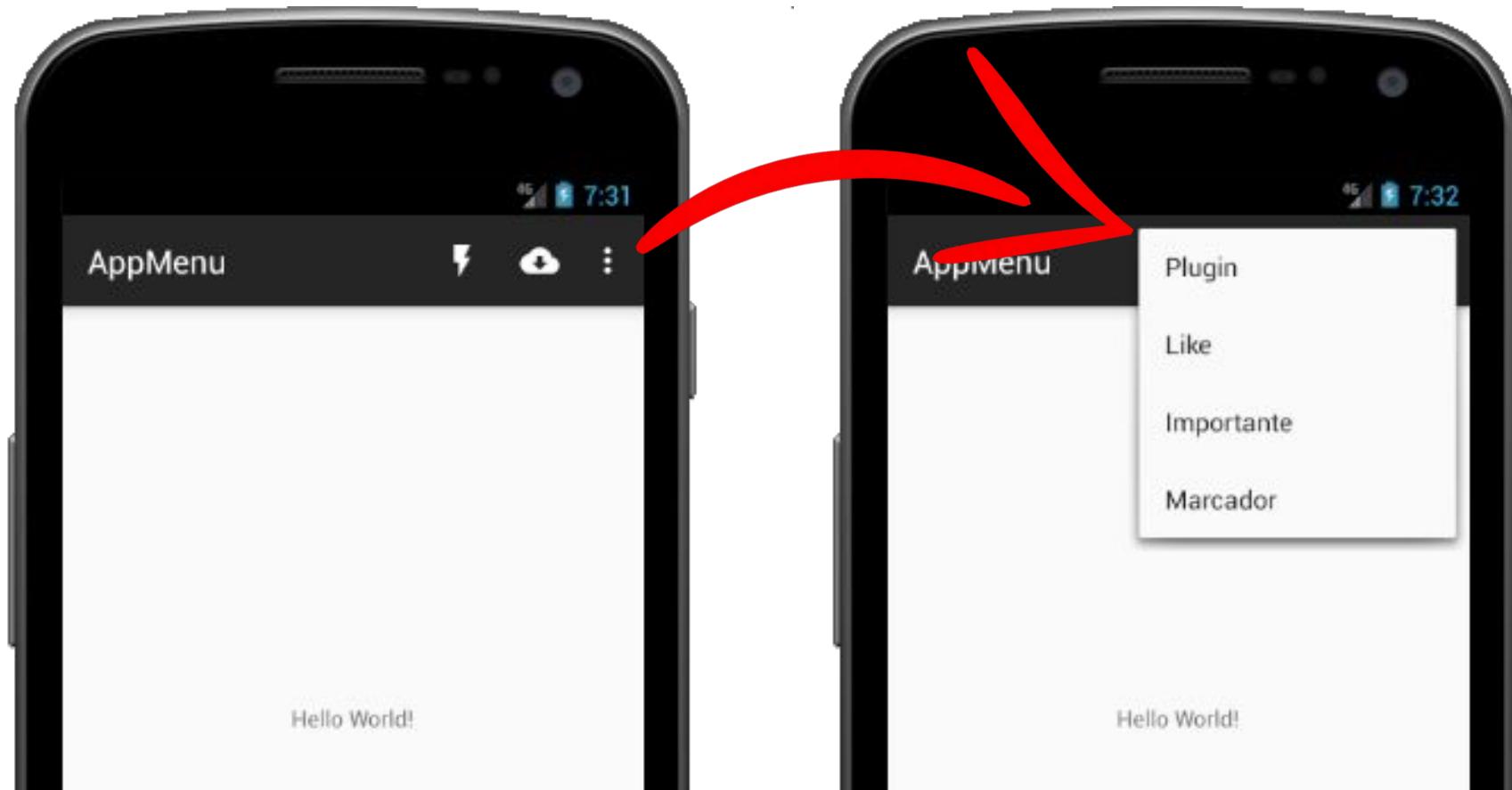
Agora que já temos o menu pronto, vamos inflá-lo na *Activity* através do método *onCreateOptionsMenu()*

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.actionbar_menu,menu);
        return super.onCreateOptionsMenu(menu);
    }
}
```

Trabalhando com ActionBar Menu (AppMenu)



Trabalhando com ActionBar Menu (AppMenu)

Para escutar o click nos itens de menu, precisamos apenas implementar o método `onOptionsItemSelected()`. Como este método recebe o *item* clicado, podemos simplesmente comparar o ID do item com o ID das tags `<item>` do XML e executar a ação desejada.

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.menu_bolt:  
            Toast.makeText(context: this, text: "Flash clicado!", Toast.LENGTH_SHORT).show();  
            break;  
    }  
    return super.onOptionsItemSelected(item);  
}
```

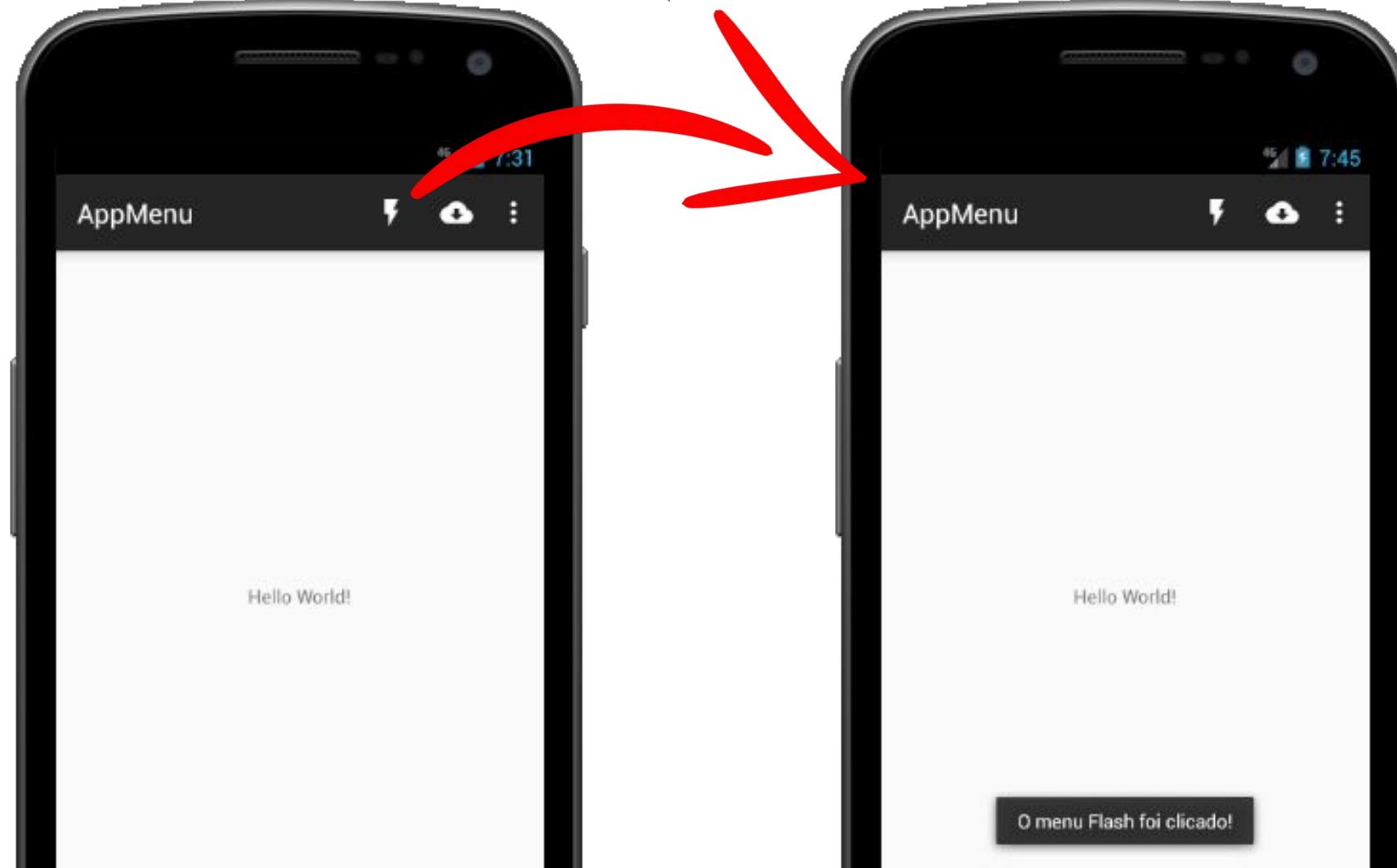
```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.actionbar_menu,menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_bolt: exibirToast( menu: "Flash"); break;
            case R.id.menu_download: exibirToast( menu: "Download"); break;
            case R.id.menu_plugin: exibirToast( menu: "Plugin"); break;
            case R.id.menu_like: exibirToast( menu: "Like"); break;
            case R.id.menu_stars: exibirToast( menu: "Importante"); break;
            case R.id.menu_tag: exibirToast( menu: "Marcador"); break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void exibirToast(String menu) {
        Toast.makeText( context: this, text: "O menu " + menu + " foi clicado!", Toast.LENGTH_SHORT).show();
    }
}
```



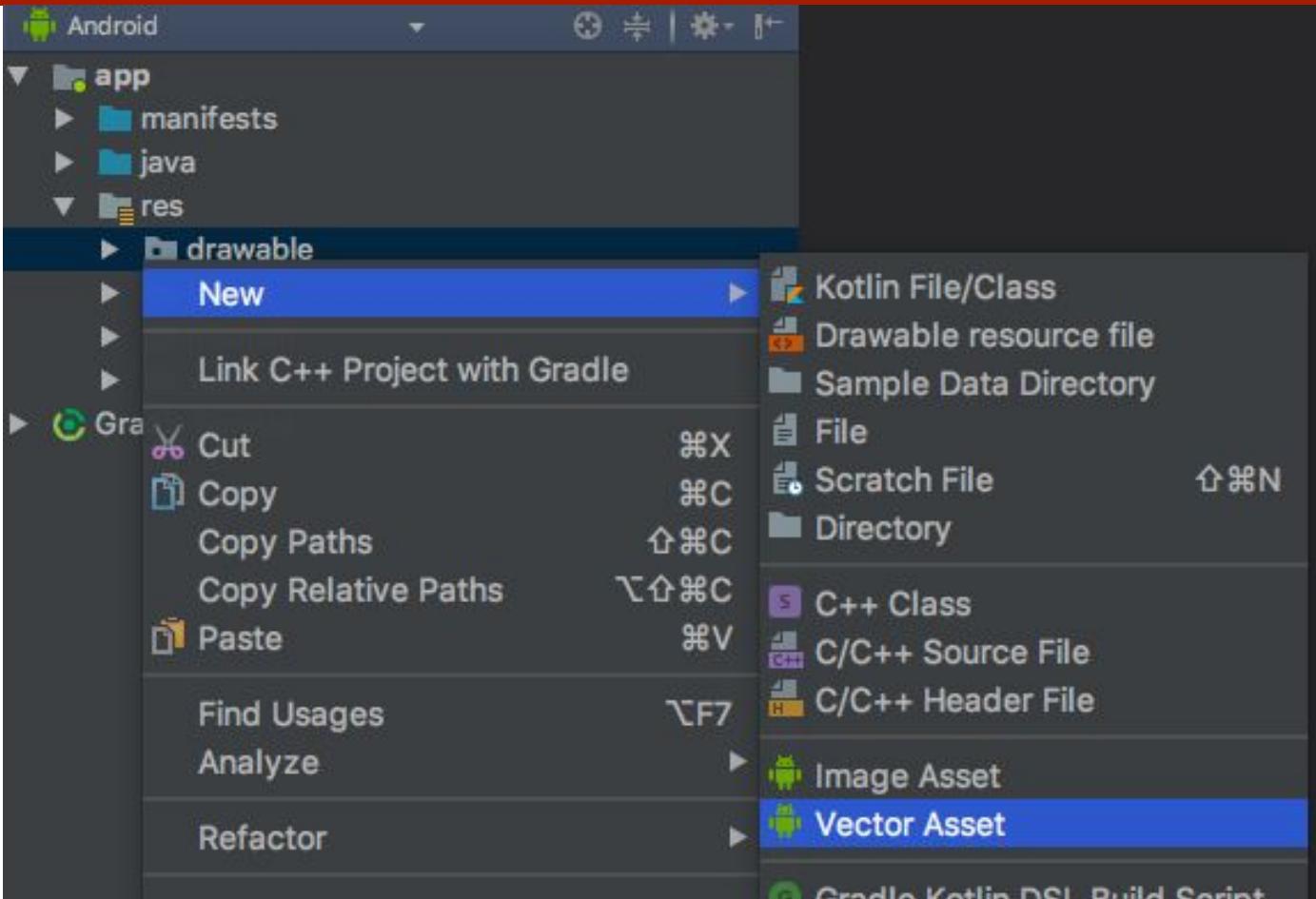
Exercício em Sala

Altere o app *Notepad* observando os seguintes detalhes:

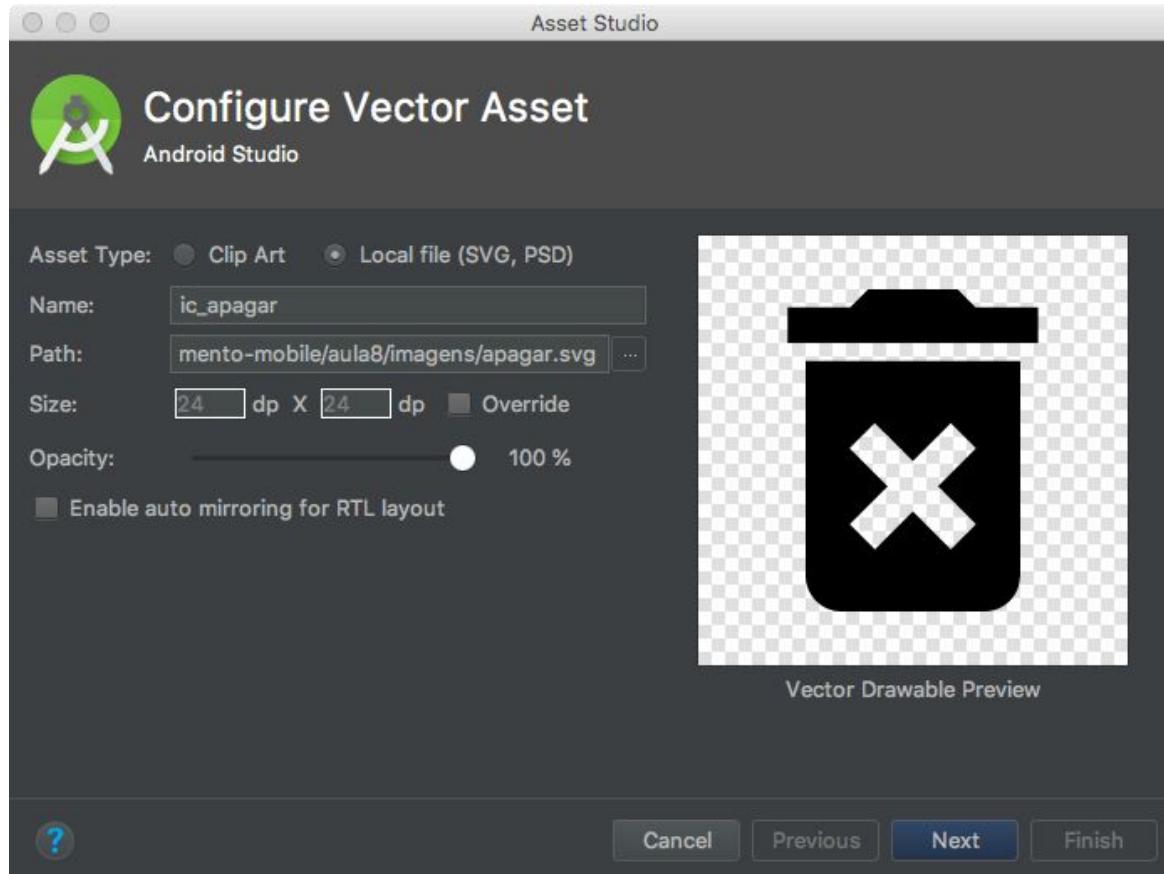
- Remova o botão "LIMPAR"
- Adicione um *ActionBar Menu* com um ícone semelhante ao da figura.
- Quando o usuário clicar neste ícone, o app deve exibir o *SnackBar* para confirmar a exclusão do texto.



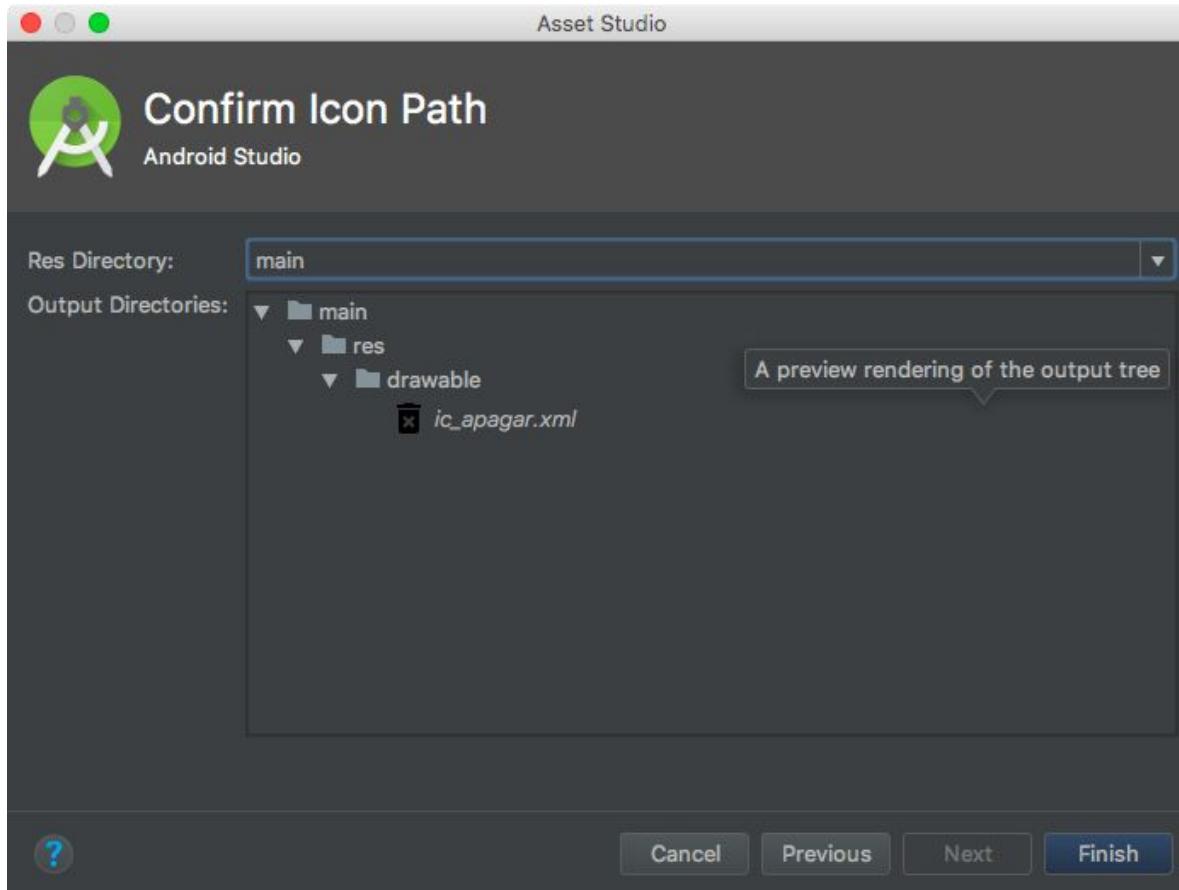
Resolvendo o Exercício



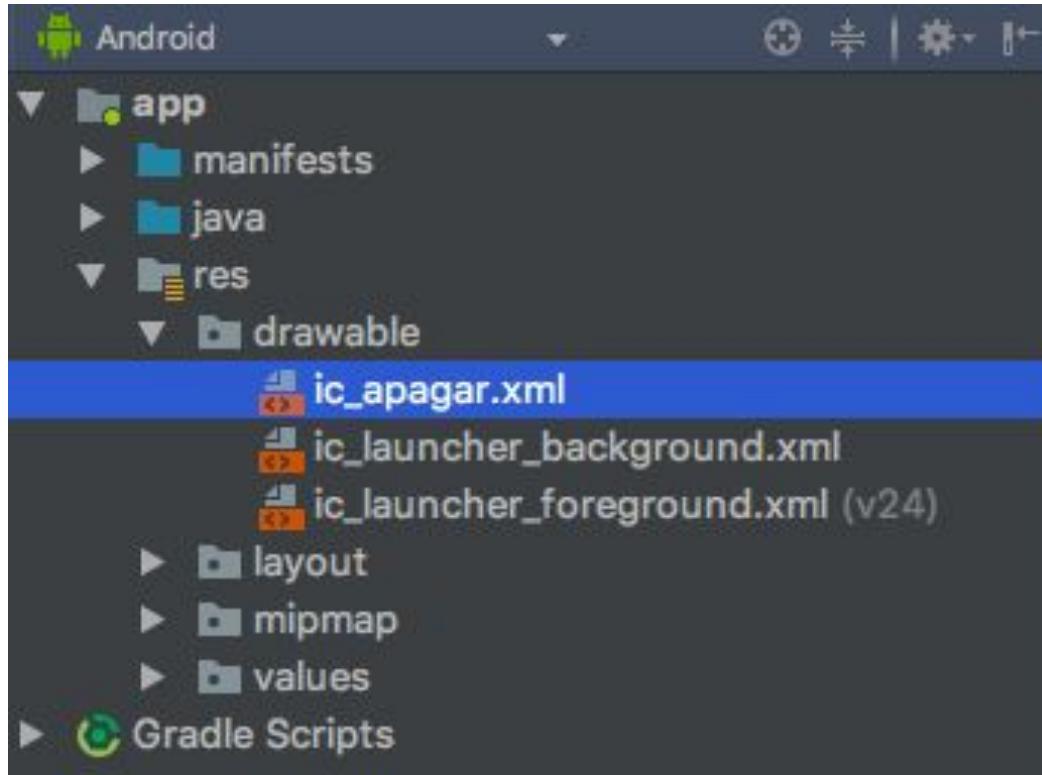
Resolvendo o Exercício



Resolvendo o Exercício



Resolvendo o Exercício



Resolvendo o Exercício



The screenshot shows the Android Studio code editor with the file "colors.xml" open. The code defines several color resources:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="cinza">#efefef</color>
    <color name="branco">#ffffff</color>
</resources>
```

Color swatches are provided for each color definition:

- colorPrimary: Blue (#3F51B5)
- colorPrimaryDark: Dark Blue (#303F9F)
- colorAccent: Red/Pink (#FF4081)
- cinza: Light Gray (#efefef)
- branco: White (#ffffff)

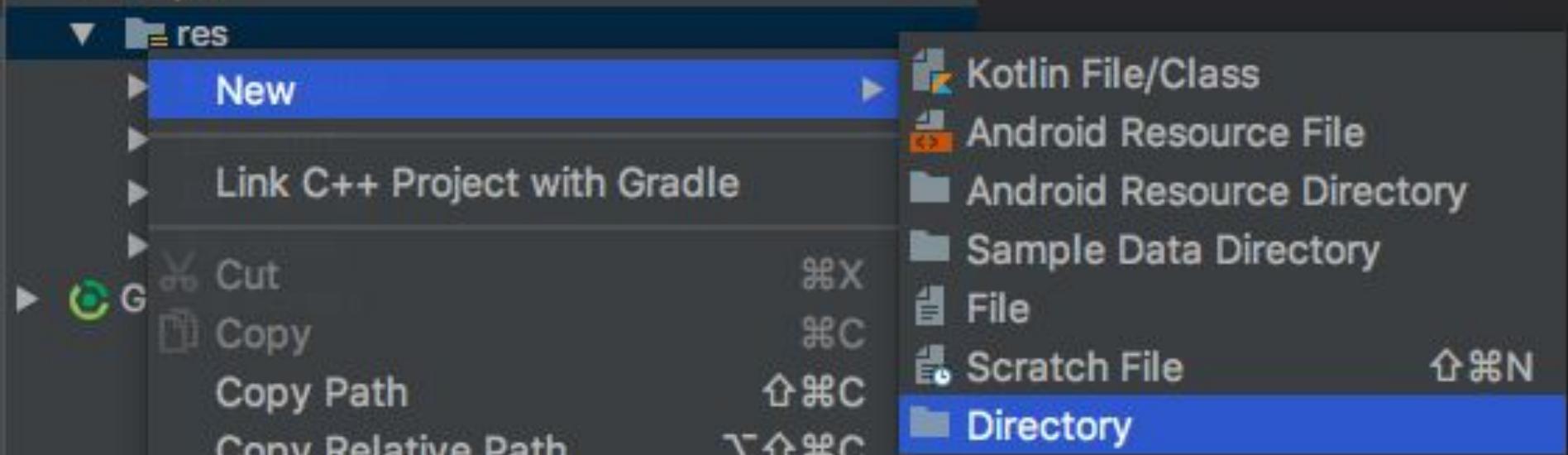
Resolvendo o Exercício

The screenshot shows the Android Studio interface with the 'styles.xml' file open in the editor. The title bar says 'Edit all themes in the project in the theme editor.' The code in the editor is as follows:

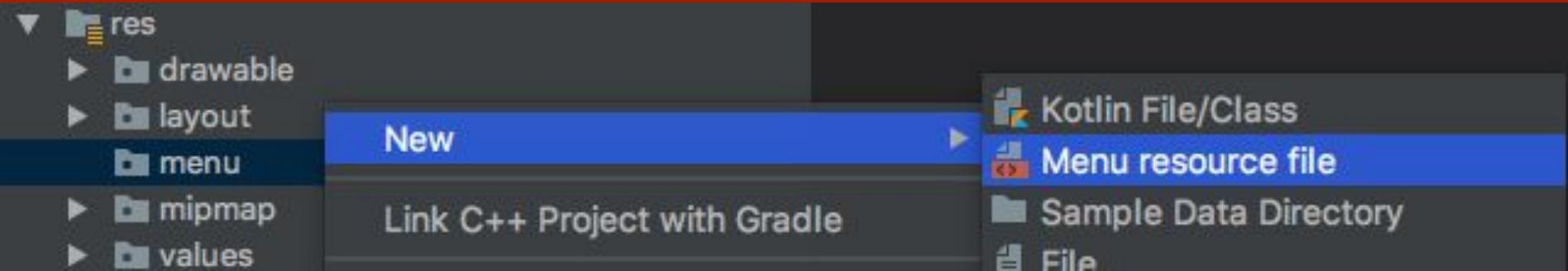
```
1 <resources>
2     <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
3         <item name="colorPrimary">@color/colorPrimary</item>
4         <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
5         <item name="colorAccent">@color/colorAccent</item>
6         <item name=" actionBarTheme">@style/cores_menu</item>
7     </style>
8     <style name="cores_menu">
9         <item name="tint">#ffffff</item>
10        <item name="iconTint">#ffffff</item>
11        <item name="titleTextColor">#ffffff</item>
12        <item name="actionMenuTextColor">#ffffff</item>
13    </style>
14 </resources>
```

The code defines two styles: 'AppTheme' and 'cores_menu'. The 'AppTheme' style inherits from 'Theme.AppCompat.Light.DarkActionBar' and sets four items: colorPrimary, colorPrimaryDark, colorAccent, and actionBarTheme to 'cores_menu'. The 'cores_menu' style contains four items: tint, iconTint, titleTextColor, and actionMenuTextColor, all set to the color '#ffffff'.

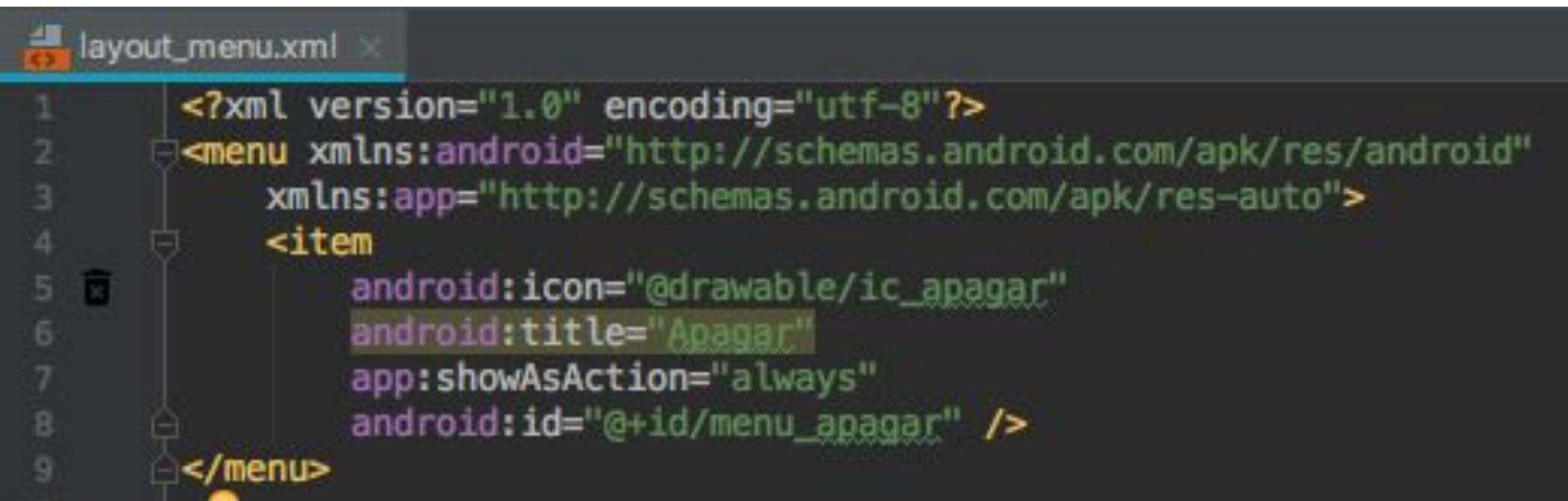
Criando o diretório menu



Criando o arquivo de menu



Resolvendo o Exercício



The screenshot shows an Android XML editor window with the file `layout_menu.xml` open. The code defines a menu item for a power-off action:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:icon="@drawable/ic_apagar"
        android:title="Apagar"
        app:showAsAction="always"
        android:id="@+id/menu_apagar" />
</menu>
```

The code uses standard Android XML syntax for defining a menu. It includes declarations for namespaces (`xmlns:android` and `xmlns:app`), a single menu item with an icon, title, and action settings, and a closing tag for the menu.

Ajustando a *MainActivity*

Podemos criar um método *apagarTexto()* para centralizar a chamada ao *SnackBar*. O método *onCreate()* servirá apenas para obter os *Views* do XML.

```
private CoordinatorLayout layout;
private EditText editText;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    layout = (CoordinatorLayout) findViewById(R.id.coordinator_layout);
    editText = (EditText) findViewById(R.id.edit_text);
}

private void apagarTexto() {
    Snackbar snackbar = Snackbar.make(layout, R.string.mensagem_confirmacao, Snackbar.LENGTH_LONG);
    snackbar.setAction(R.string.snackbar_action, new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            editText.setText("");
        }
    });
    snackbar.show();
}
```

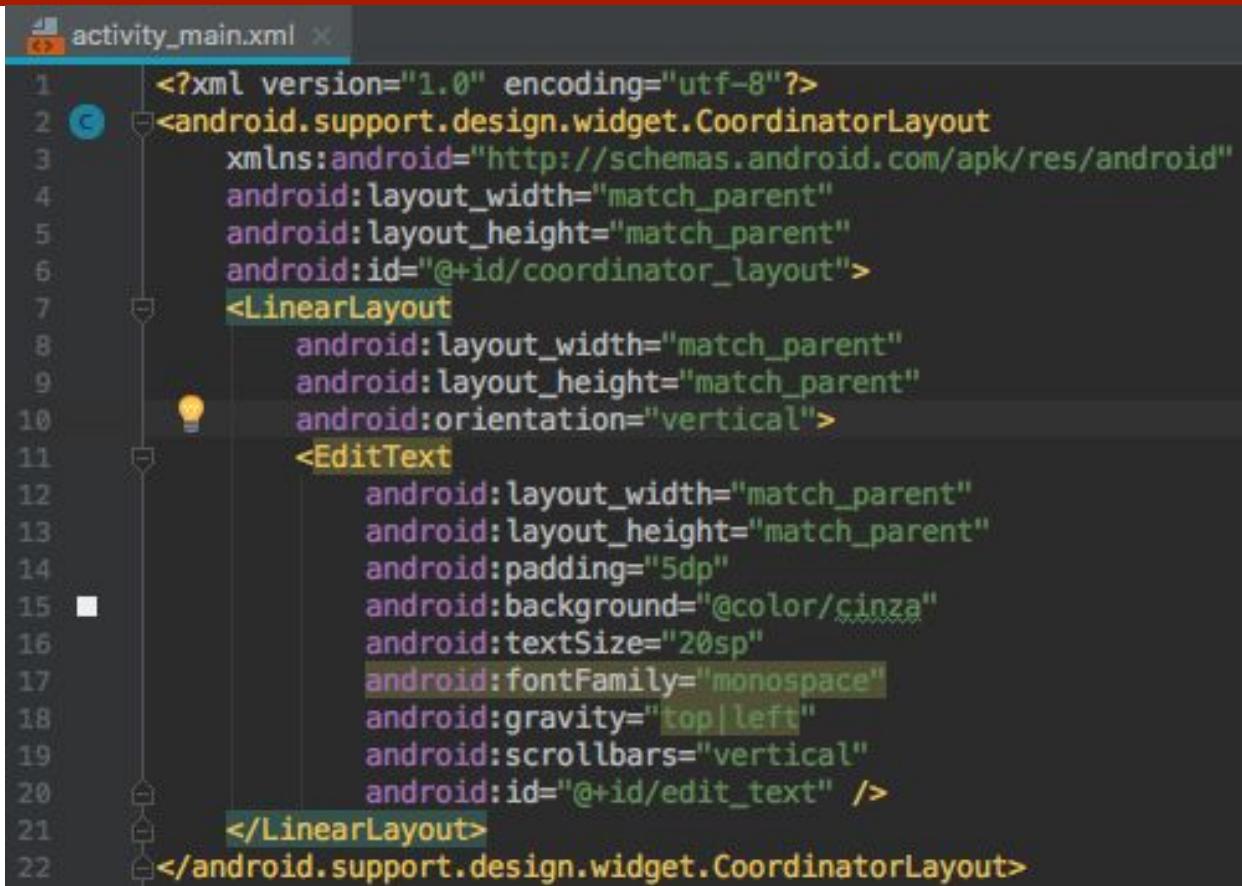
Ajustando a *MainActivity*

Devemos também implementar o método *onCreateOptionsMenu()* para inflar o XML do menu, e o método *onOptionsItemSelected()* para escutar o clique do usuário no item de menu.

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.menu_apagar:  
            apagarTexto();  
            break;  
    }  
    return super.onOptionsItemSelected(item);  
  
}  
  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.layout_menu,menu);  
    return super.onCreateOptionsMenu(menu);  
}
```

Ajustando a *activity_main.xml*

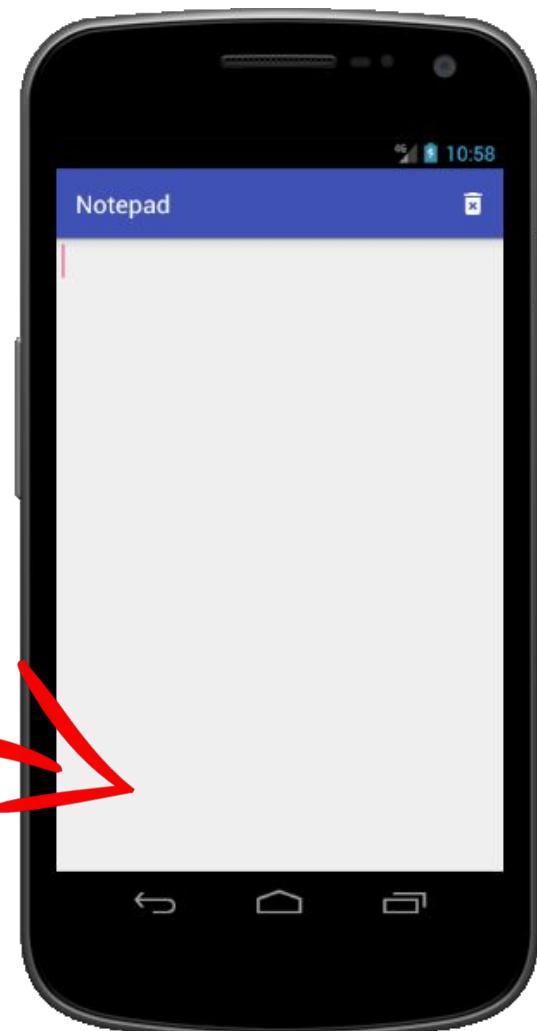
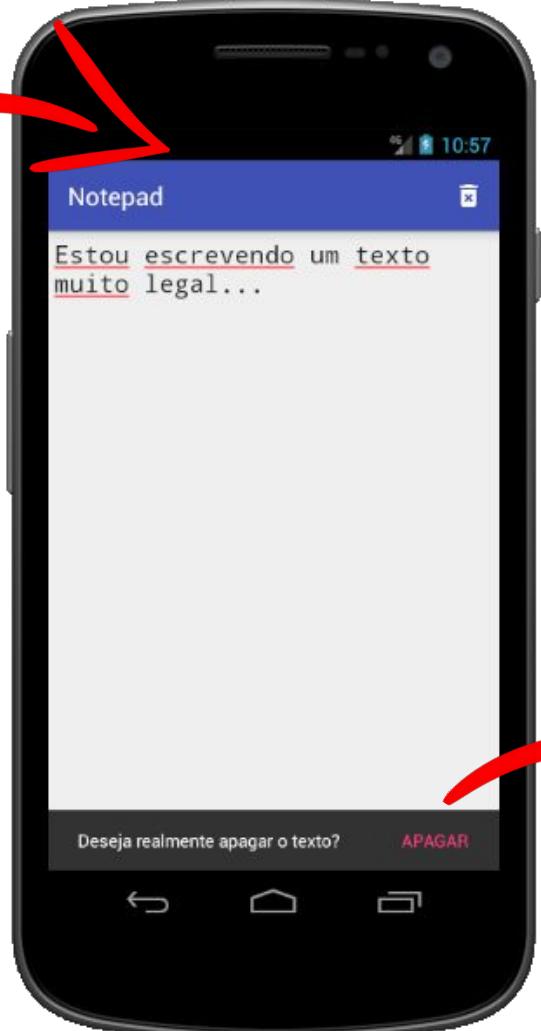
Por fim, vamos remover o botão do XML principal.



The screenshot shows the Android Studio interface with the XML file 'activity_main.xml' open. The code is displayed in a syntax-highlighted editor. The XML structure is as follows:

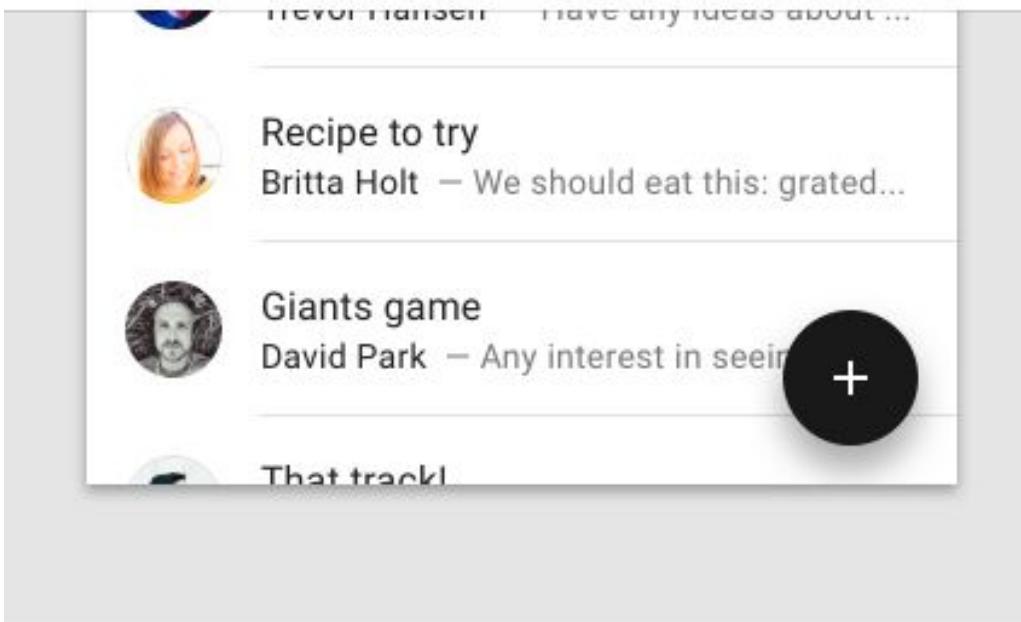
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/coordinator_layout">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <EditText
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:padding="5dp"
            android:background="@color/cinza"
            android:textSize="20sp"
            android:fontFamily="monospace"
            android:gravity="top|left"
            android:scrollbars="vertical"
            android:id="@+id/edit_text" />
    </LinearLayout>
</android.support.design.widget.CoordinatorLayout>
```

The code editor highlights several attributes in green and purple, such as 'monospace' and 'top|left'. A yellow lightbulb icon is visible near the 11th line, indicating a potential issue or suggestion. The file tab at the top shows 'activity_main.xml'.



Floating Action Button

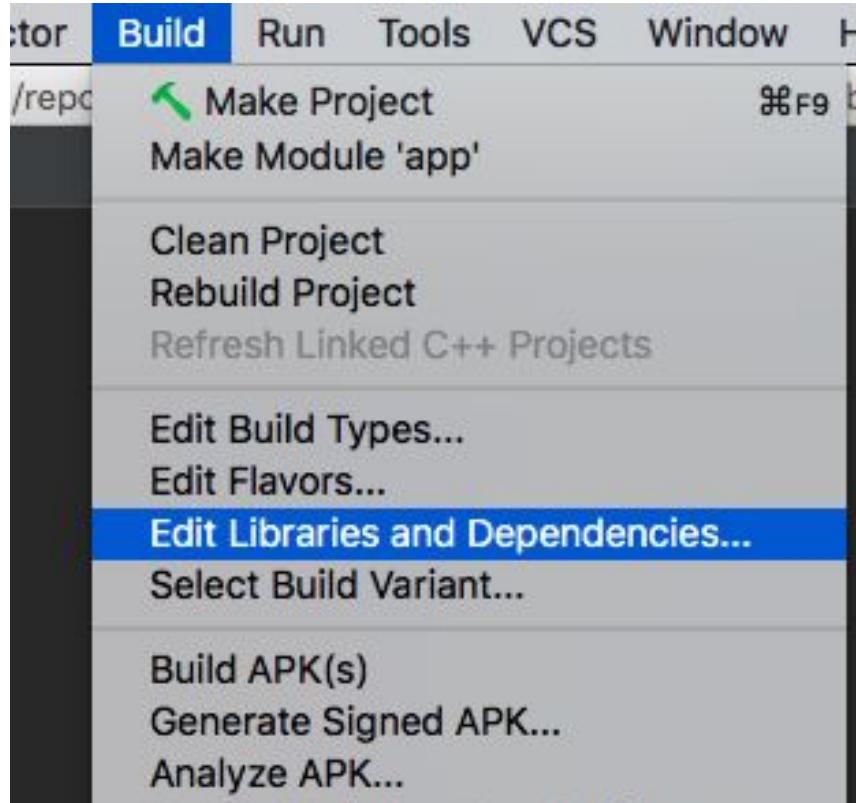
Vamos agora criar o Floating Action Button (FAB). Basicamente, um FAB é um botão totalmente arredondado, geralmente posicionado no canto inferior à direita, e que fica sobre todos os demais componentes da *Activity* (com exceção do *Toast* e *SnackBar*).
<https://material.io/design/components/buttons-floating-action-button.html>



Antigamente (antes de 2015), para criar um FAB precisávamos criar um *shape* arredondado para um botão com elevation de 6dp. Ele provocava este mesmo efeito (sombreado) que hoje está disponível pela ***Design Support Library***.

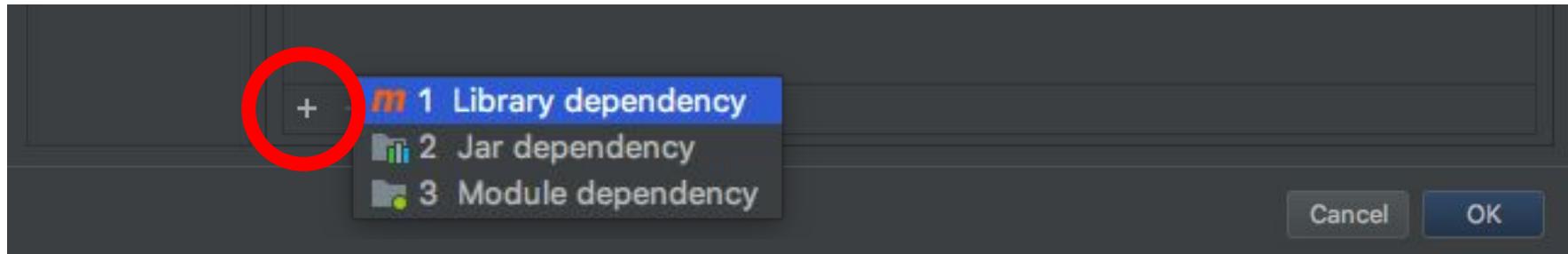
Floating Action Button

Ainda no projeto *AppMenu*, vamos importar a *Design Support Library*.



Floating Action Button

Clique na opção *Library dependency*

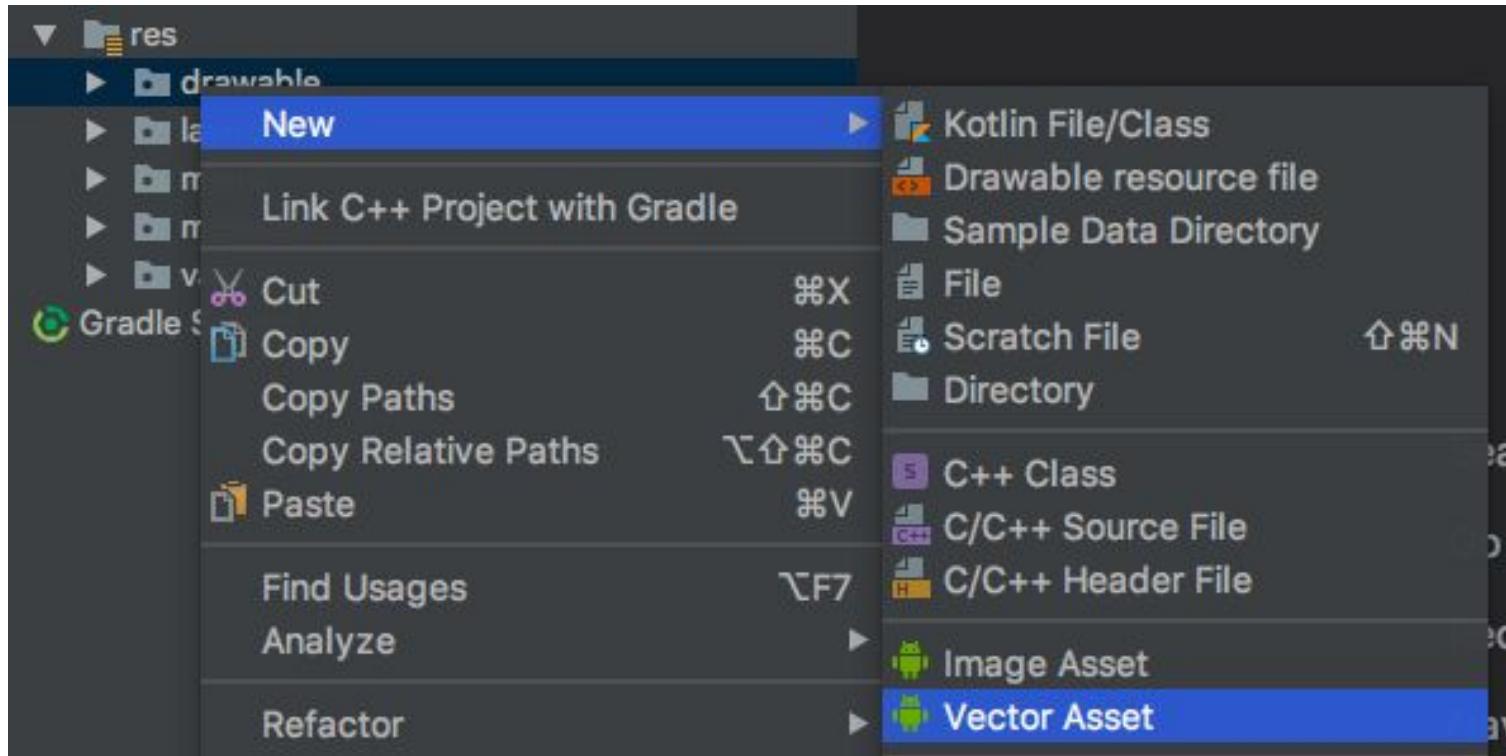


Faça uma busca pela palavra *Design* e selecione o item a seguir:

This screenshot shows the search results for 'com.android.support:design' in the 'Choose Library Dependency' dialog. The search bar at the top contains the text 'com.android.support:design:28.0.0-alpha3'. Below the search bar, the results are listed. The first result, 'com.android.support:design (com.android.support:design:28.0.0-alpha3)', is highlighted with a blue background, matching the highlighted text in the previous step. Other results include 'org.webjars.bowergithub.predixdesignsystem:px-buttons-design' and 'org.webjars.bowergithub.predixdesignsystem:px-button-group-design'.

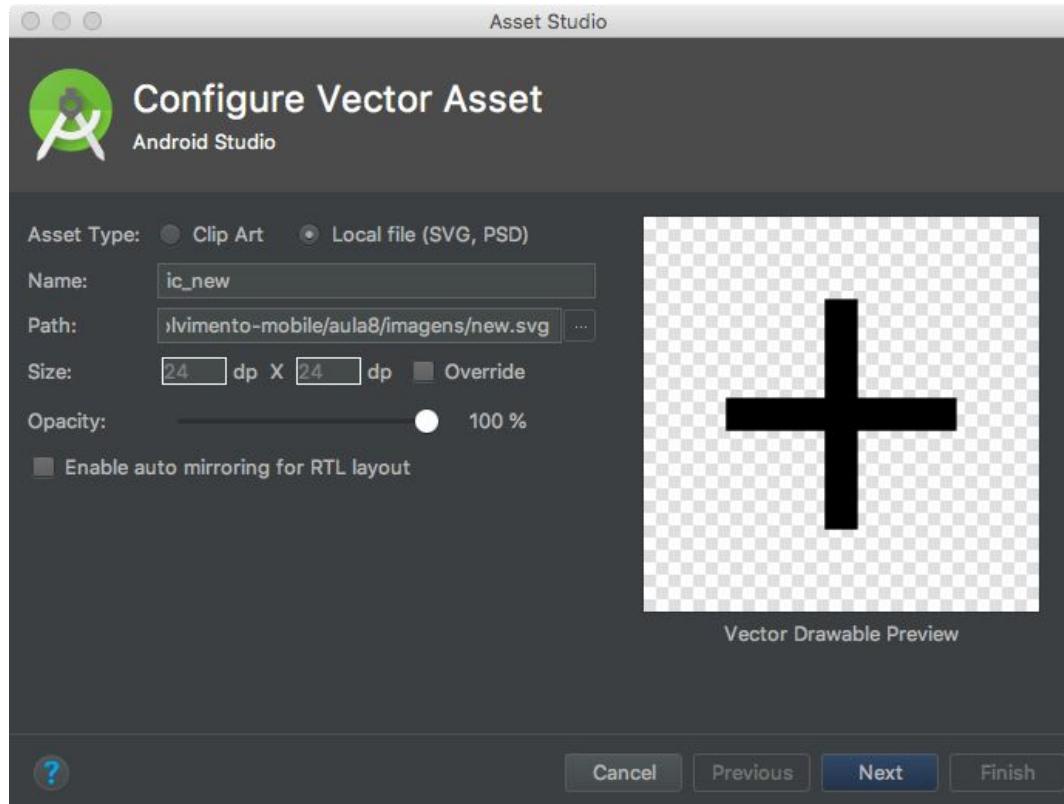
Floating Action Button

Vamos agora importar um ícone para colocar dentro do FAB



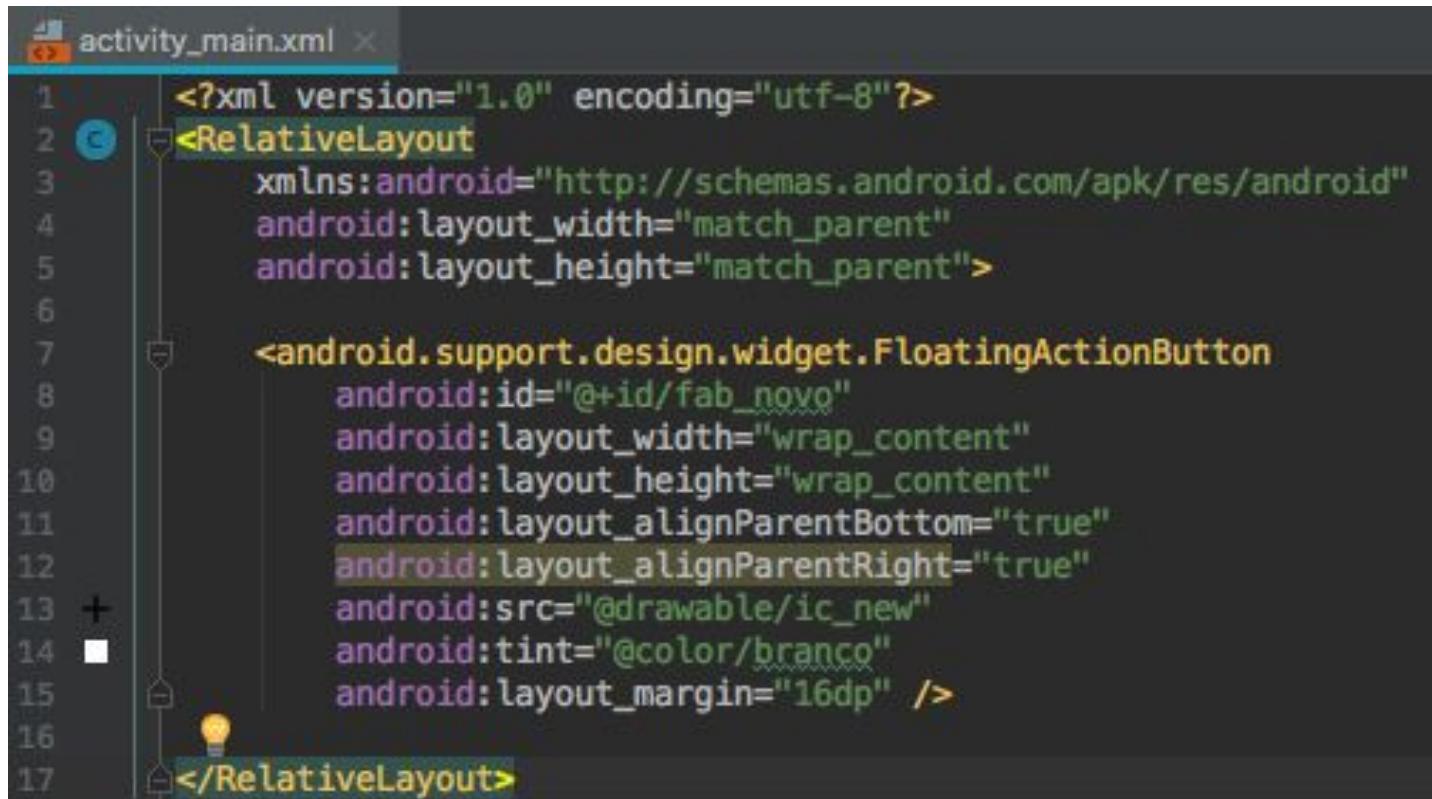
Floating Action Button

Vamos agora importar um ícone para colocar dentro do FAB



Floating Action Button

A inclusão do FAB no Layout é muito simples! Altere o layout para *RelativeLayout* para posicionar o FAB corretamente.



The screenshot shows the Android Studio code editor with the file `activity_main.xml` open. The code defines a `RelativeLayout` containing a `FloatingActionButton`. The FloatingActionButton is positioned at the bottom right of the screen. The code is as follows:

```
<?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">

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab_novo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:src="@drawable/ic_new"
        android:tint="@color/branco"
        android:layout_margin="16dp" />

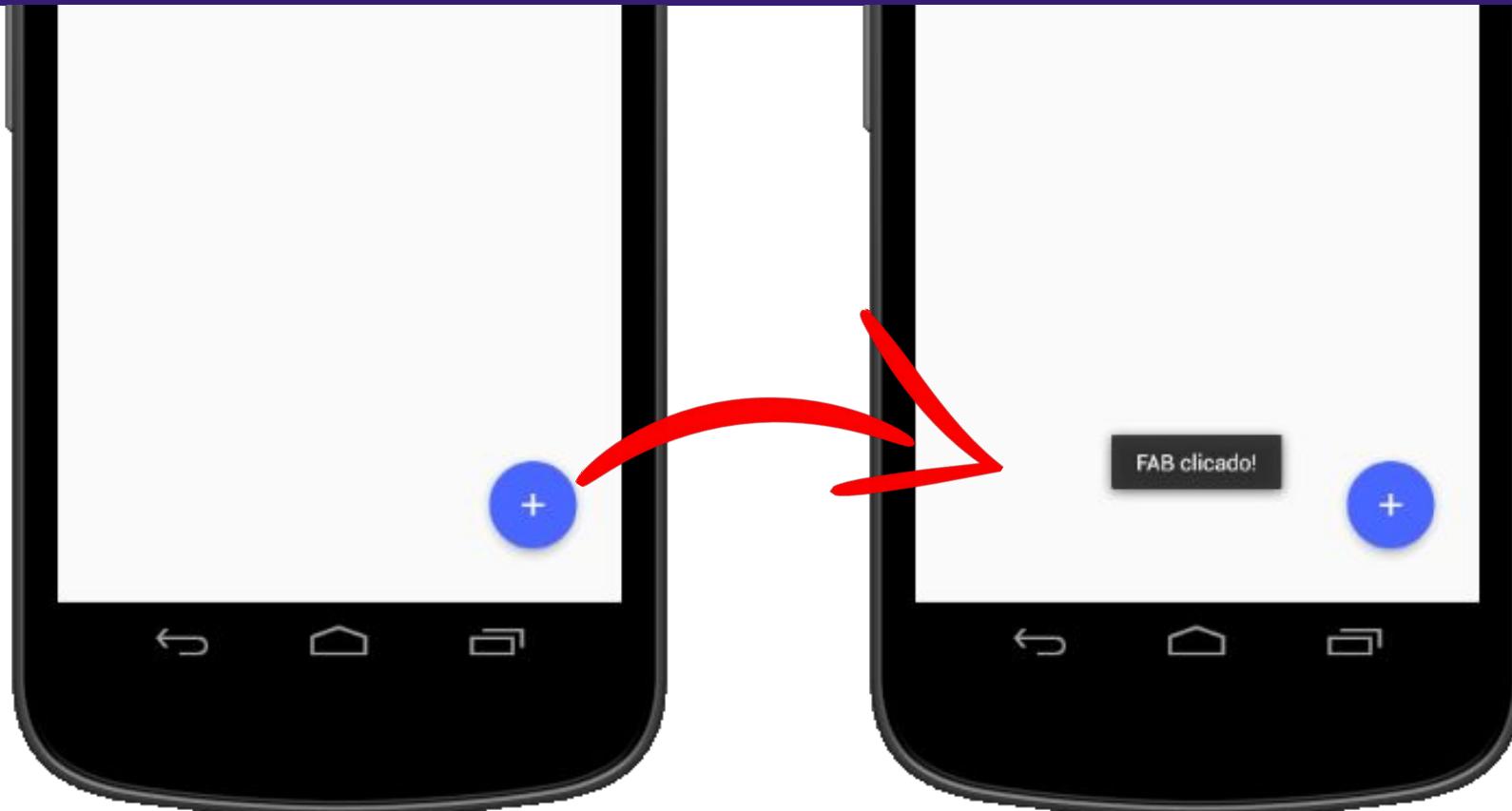
</RelativeLayout>
```

Floating Action Button

Por fim, no método `onCreate()` podemos obtê-lo através do `findViewById()` e adicionar um `OnClickListener` apropriado.

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    FloatingActionButton fab = findViewById(R.id.fab_novo);  
    fab.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View view) {  
            Toast.makeText(context: MainActivity.this, text: "FAB clicado!", Toast.LENGTH_LONG).show();  
        }  
    });  
}
```

Floating Action Button



Icon Launcher



Orientação oficial (Android):

- 512 x 512 pixels. Na prática, eles serão menores no Android, porém ao publicar no Google Play, você precisará de um ícone deste tamanho.
- Separar entre *background* e *foreground*. Isso é útil para auxiliar os efeitos dos diferentes dispositivos.

Dicas Extras:

- 1024 x 1024 pixels, para aproveitar uma possível publicação na web
- Mantenha um ícone completo (*background* e *foreground*) para facilitar a publicação no Google Play
- Evite ícones arredondados ou com formatos diferenciados. Isso pode dificultar a adaptação do ícone em vários dispositivos. **O ideal é usar ícones quadrados.**

Icon Launcher

Vamos começar excluindo os ícones que já existem no projeto. Na prática, os ícones são salvos nas pastas `mipmap`, organizadas por densidade.



`mipmap-mdpi`



`mipmap-hdpi`



`mipmap-xhdpi`



`mipmap-xxhdpi`



`mipmap-xxxhdpi`

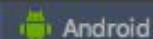
Além da densidade, os ícones também são organizados por ícones normais e arredondados (para do Android 7.1).



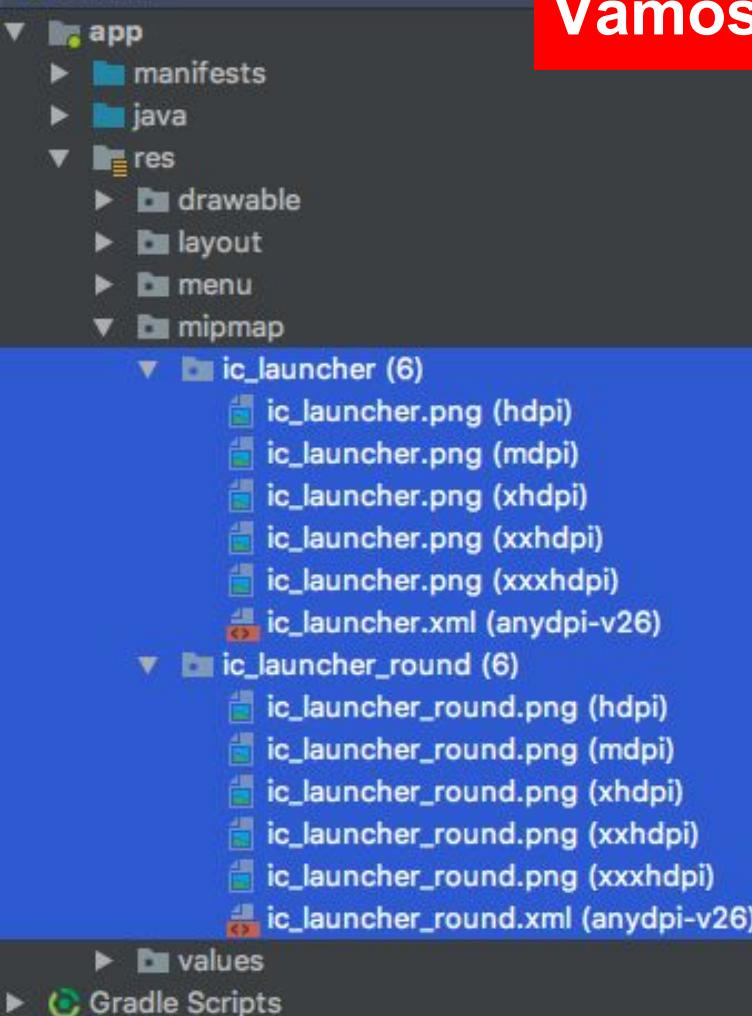
`ic_launcher`



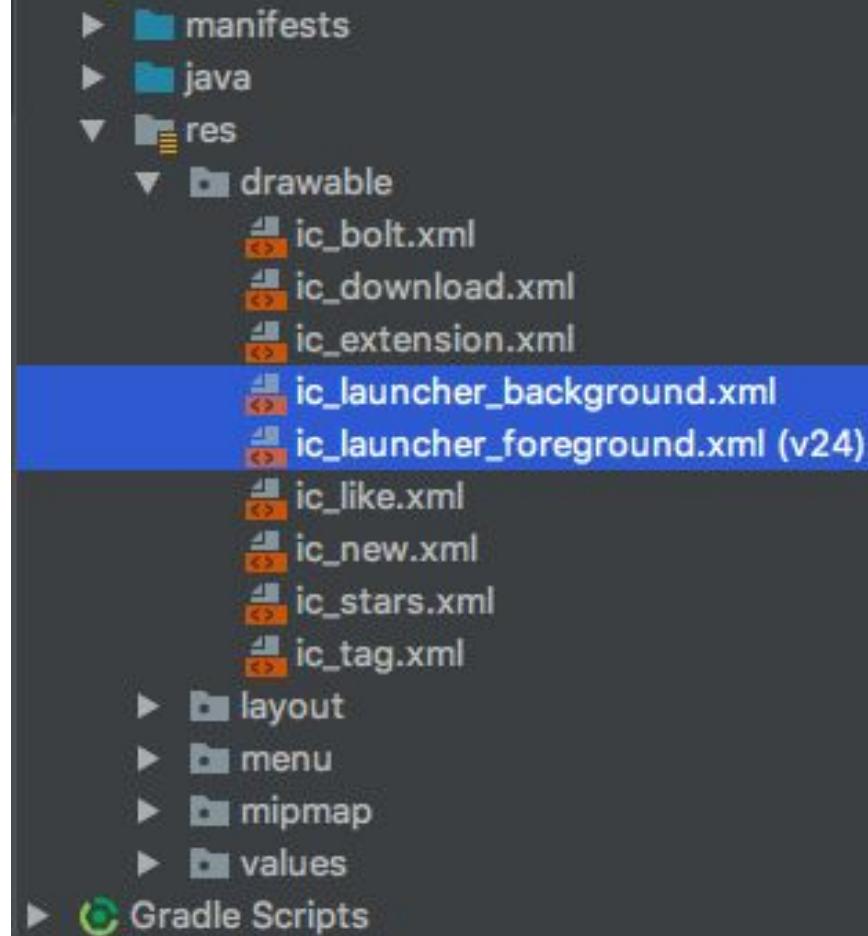
`ic_launcher_round`

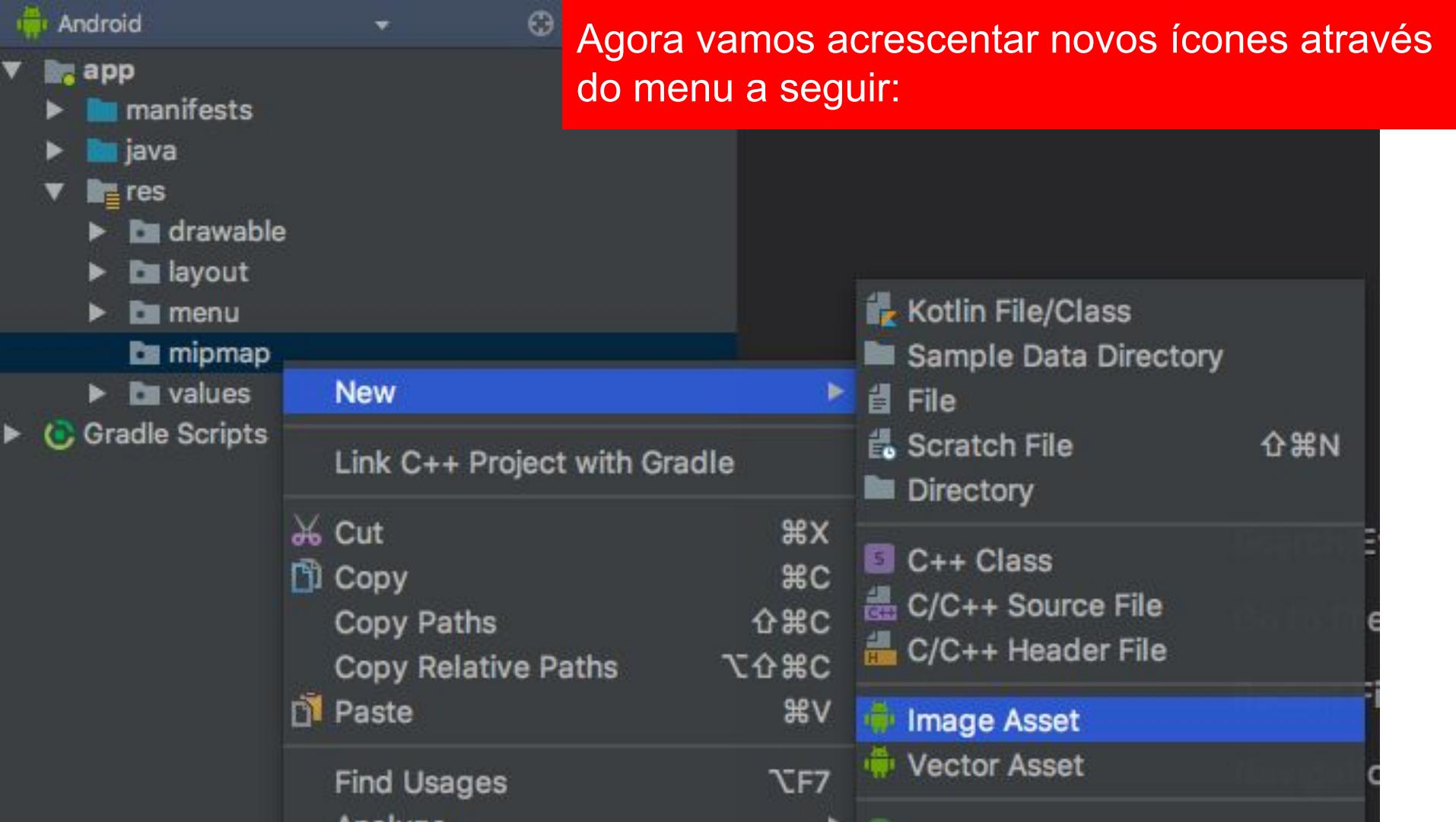


Android



Vamos apagar os seguintes arquivos:

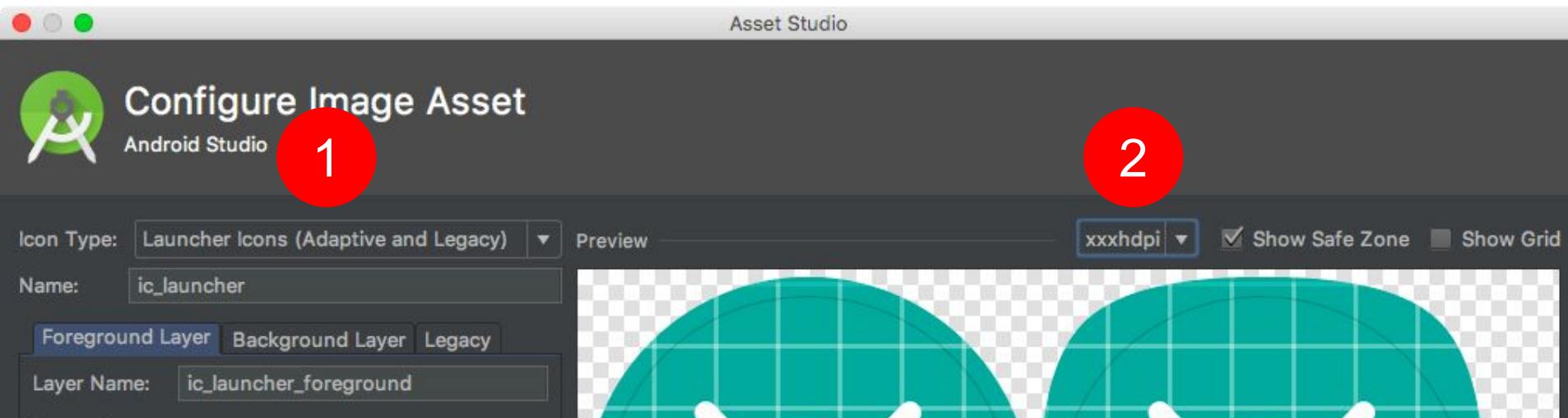




Selecione as seguintes opções:

1- **Icon Type:** Launcher Icons (Adaptive and Legacy)

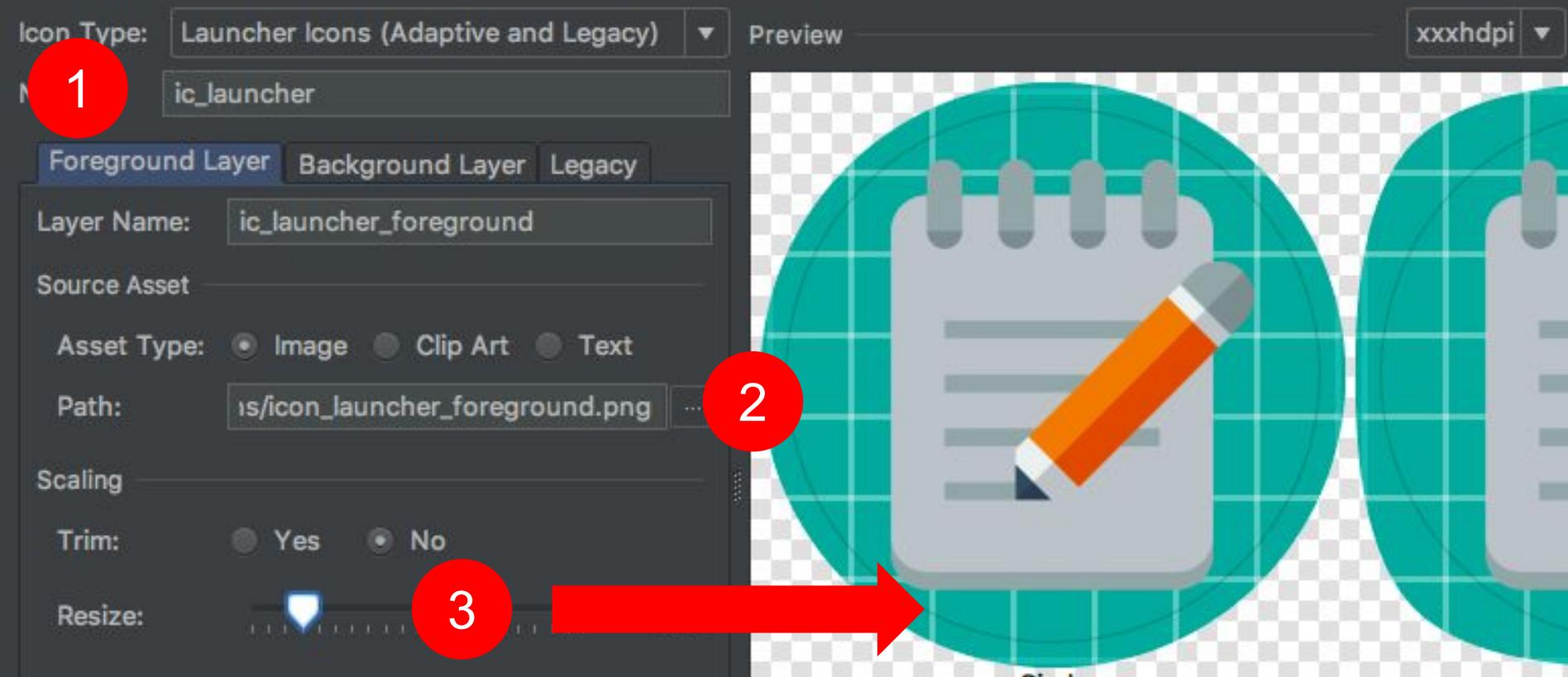
2- **Densidade:** xxxhdpi



1- Aba Foreground Layer

2- Arquivo icon_launcher_foreground.png disponível na pasta imagem do EAD

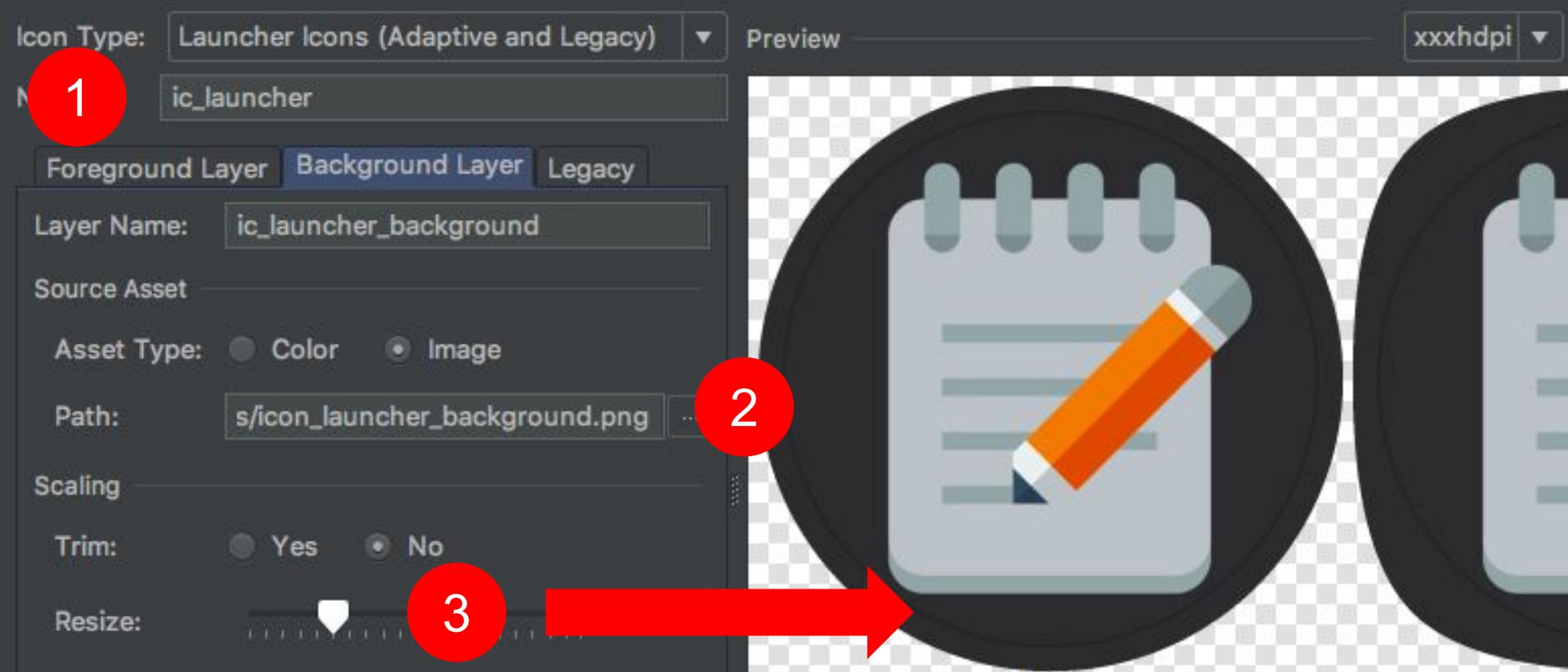
3- Ajuste para encaixar corretamente em todos os layouts apresentados



1- Aba Background Layer

2- Arquivo icon_launcher_background.png disponível na pasta imagem do EAD

3- Se necessário, ajuste para encaixar corretamente nos layouts



Na aba Legacy, verifique apenas se todas as opções de geração dos ícones legados encontram-se marcados com Yes

Icon Type: Launcher Icons (Adaptive and Legacy) ▾ Preview xxxhdpi ▾ Sh

Name: ic_launcher

Foreground Layer Background Layer **Legacy**

Legacy Icon (API ≤ 25)

Generate: Yes No

Shape: Square ▾

Round Icon (API = 25)

Generate: Yes No

Google Play Store Icon

Generate: Yes No

Shape: Square ▾

The screenshot shows the Android Asset Studio's icon generation interface. The 'Legacy' tab is selected. Under 'Legacy Icon (API ≤ 25)', the 'Generate' option is set to 'Yes'. The 'Shape' dropdown is set to 'Square'. Below it, under 'Round Icon (API = 25)', the 'Generate' option is also set to 'Yes'. At the bottom, for the 'Google Play Store Icon', both 'Generate' and 'Shape' are set to 'Yes'. On the right, there's a 'Preview' section showing two circular icons. The first is labeled 'Circle' and has a black border. The second is labeled 'Square' and has a white border. Both icons feature a stylized notepad with horizontal lines and a large orange pencil.



Configure Image Asset

Android Studio

Icon Type: Launcher Icons (Adaptive and Legacy) ▾

Name: ic_launcher

 Foreground Layer Background Layer Legacy

Layer Name: ic_launcher_foreground

Source Asset

Asset Type: Image Clip Art Text

Path: /res/icon_launcher_foreground.png ...

Scaling

Trim: Yes NoResize: 61 %

Cancel

Previous

Next

Finish



Confirm Icon Path

Android Studio

Res Directory:

main

Output Directories:

- ▼ main
 - ▼ res
 - ▼ mipmap-anydpi-v26
 - ic_launcher.xml
 - ic_launcher_round.xml
 - ▼ mipmap-mdpi
 - ic_launcher.png
 - ic_launcher_background.png
 - ic_launcher_foreground.png
 - ic_launcher_round.png
 - ▼ mipmap-hdpi
 - ic_launcher.png
 - ic_launcher_background.png
 - ic_launcher_foreground.png
 - ic_launcher_round.png
 - ▼ mipmap-xhdpi
 - ic_launcher.png
 - ic_launcher_background.png
 - ic_launcher_foreground.png
 - ic_launcher_round.png
 - ▼ mipmap-xxhdpi
 - ic_launcher.png

Output File

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@mipmap/ic_launcher_background"/>
    <foreground android:drawable="@mipmap/ic_launcher_foreground"/>
</adaptive-icon>
```



Cancel

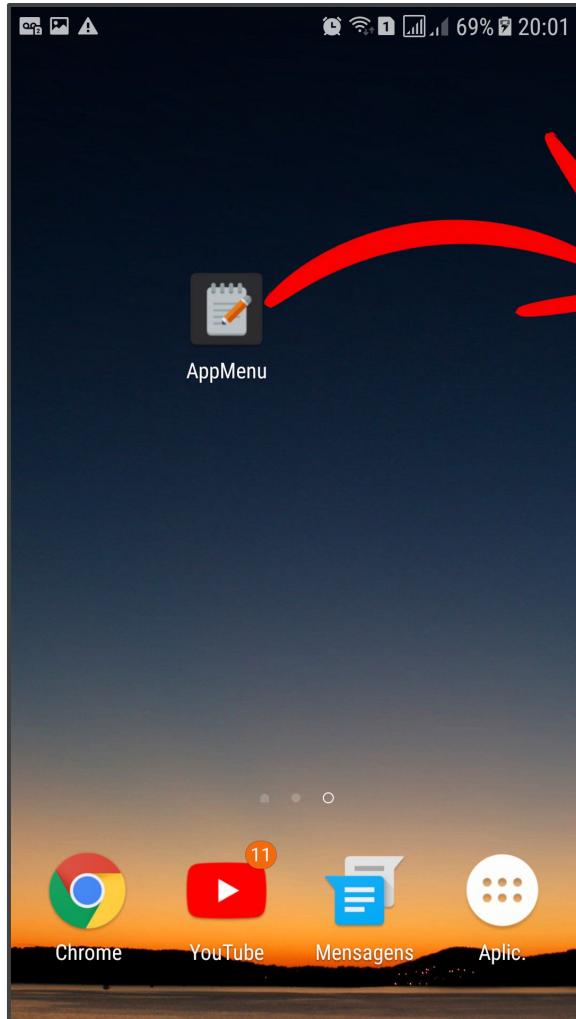
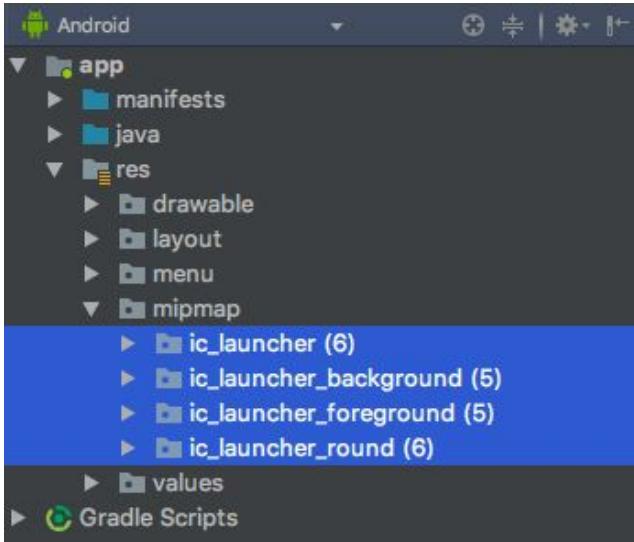
Previous

Next

Finish

Icon Launcher

Após este procedimento os ícones serão criados e publicados junto com seu aplicativo!



Exercício em Sala

Altere o app *Notepad* observando os seguintes detalhes:

- Altere as cores do App para:
 - colorPrimary: #2C2C2E
 - colorPrimaryDark #000000
 - colorAccent: #4967ff

- Altere o *Icon Launcher* para os arquivos

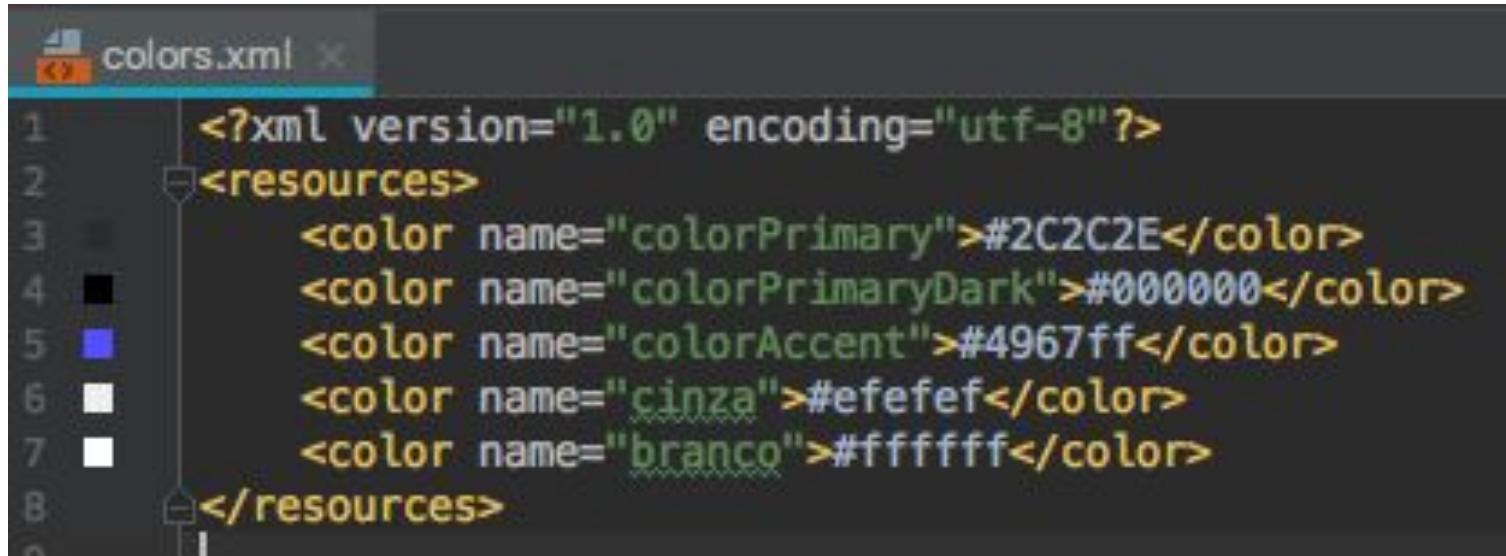
icon_launcher_background.png e

icon_launcher_foreground.png

disponíveis na pasta imagens do EAD



Resolvendo o Exercício



The screenshot shows the Android Studio code editor with the file "colors.xml" open. The code defines five color resources:

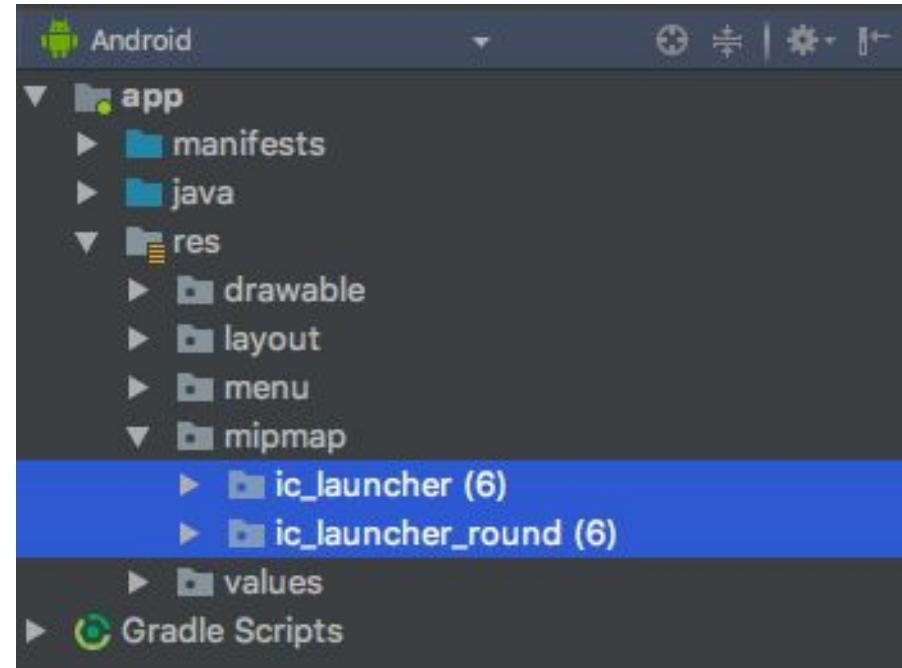
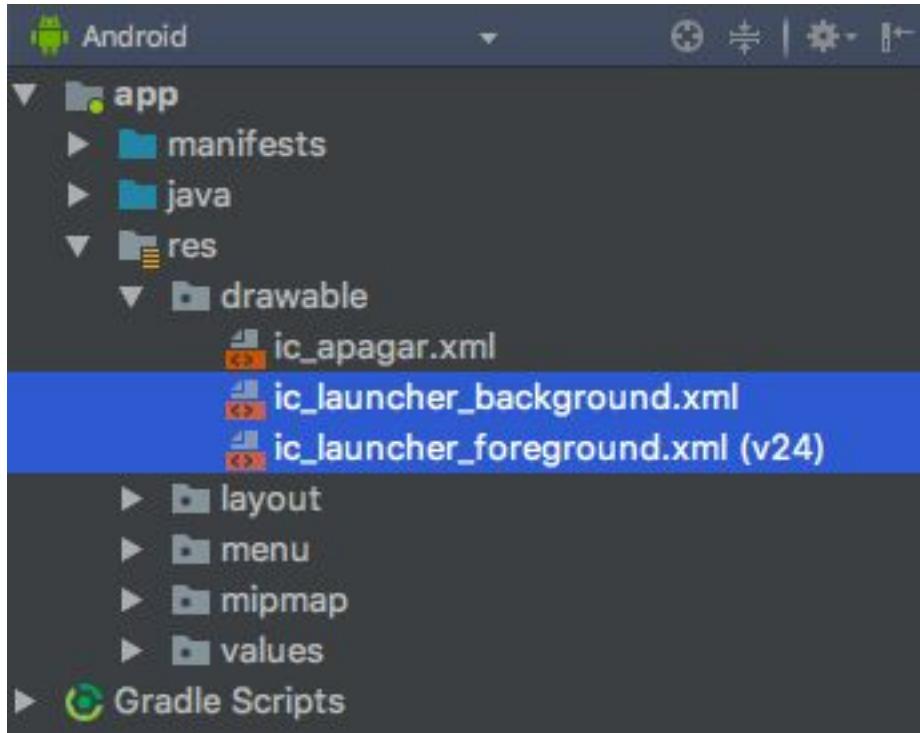
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#2C2C2E</color>
    <color name="colorPrimaryDark">#000000</color>
    <color name="colorAccent">#4967ff</color>
    <color name="cinza">#efefef</color>
    <color name="branco">#ffffff</color>
</resources>
```

Color swatches are provided for each color definition:

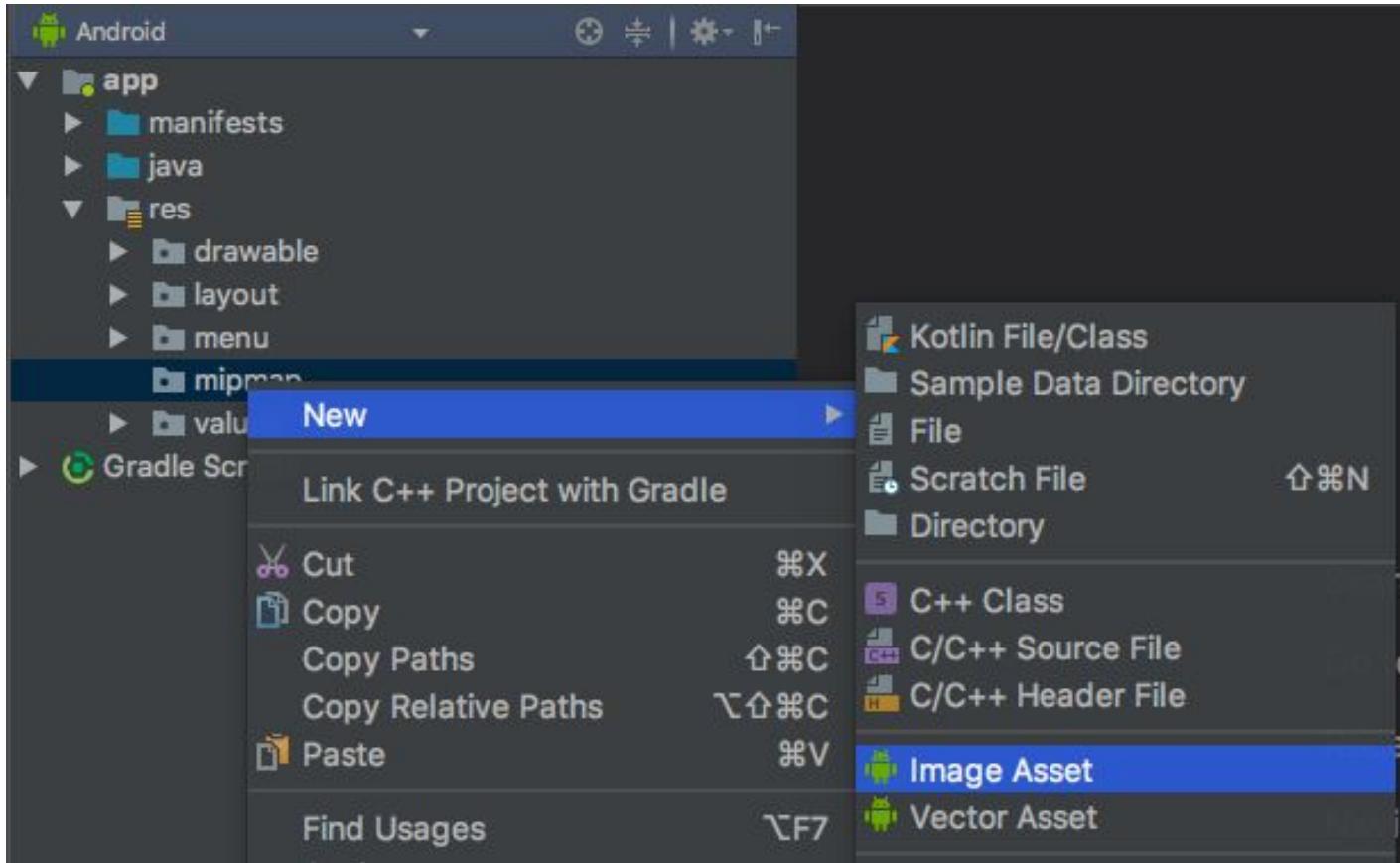
- colorPrimary: #2C2C2E (dark grey)
- colorPrimaryDark: #000000 (black)
- colorAccent: #4967ff (blue)
- cinza: #efefef (light grey)
- branco: #ffffff (white)

Resolvendo o Exercício

Apague os seguintes arquivos...



Resolvendo o Exercício





Configure Image Asset

Android Studio

Icon Type: Launcher Icons (Adaptive and Legacy) ▾

Name: ic_launcher

 Foreground Layer Background Layer Legacy

Layer Name: ic_launcher_foreground

Source Asset

Asset Type: Image Clip Art Text

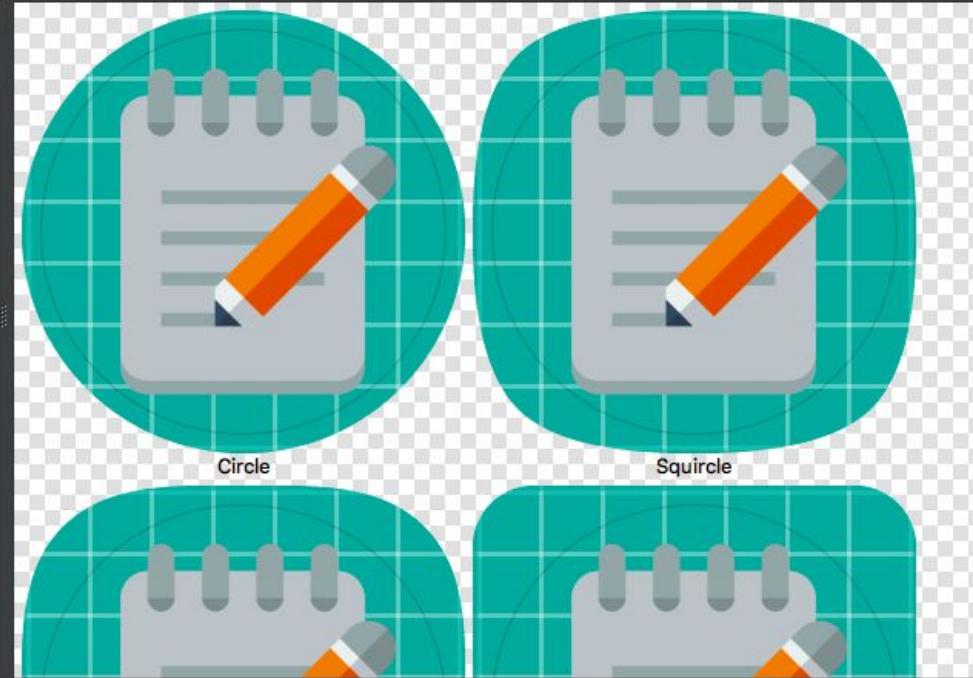
Path: /is/icon_launcher_foreground.png ...

Scaling

Trim: Yes NoResize: 61 %

Preview

xxxhdpi ▾

 Show Safe Zone Show Grid

Cancel

Previous

Next

Finish



Configure Image Asset

Android Studio

Icon Type: Launcher Icons (Adaptive and Legacy) ▾

Name: ic_launcher

 Foreground Layer Background Layer Legacy

Layer Name: ic_launcher_background

Source Asset

Asset Type: Color Image

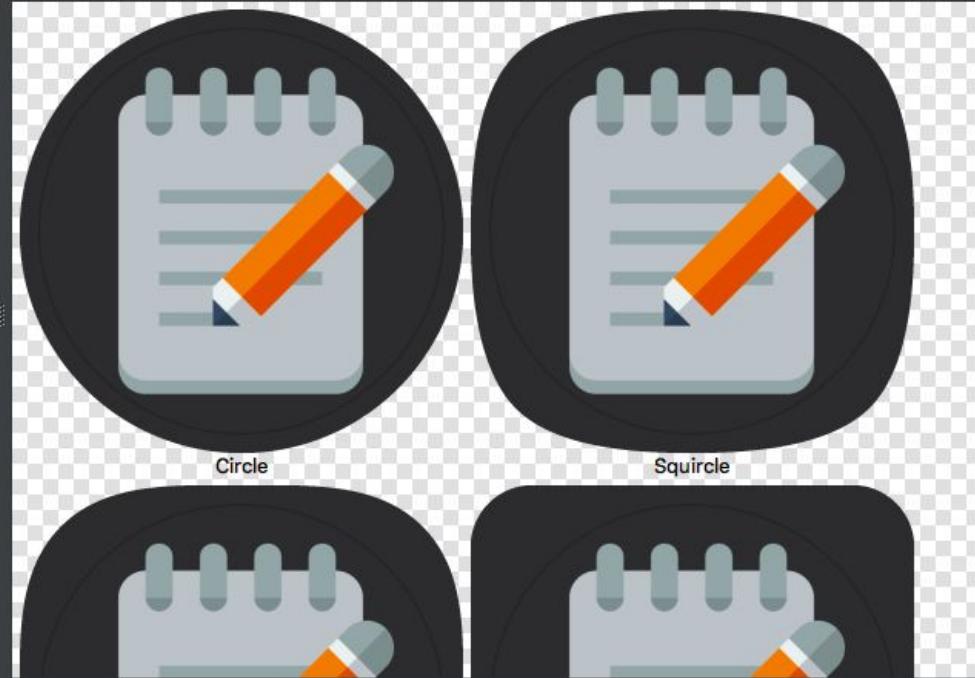
Path: s/icon_launcher_background.png ...

Scaling

Trim: Yes NoResize: 100 %

Preview

xxxhdpi ▾

 Show Safe Zone Show Grid

Cancel

Previous

Next

Finish



Configure Image Asset

Android Studio

Icon Type: Launcher Icons (Adaptive and Legacy) ▾

Name: ic_launcher

Foreground Layer Background Layer Legacy

Legacy Icon (API ≤ 25)

Generate: Yes No

Shape: Square ▾

Round Icon (API = 25)

Generate: Yes No

Google Play Store Icon

Generate: Yes No

Shape: Square ▾

Preview

xxxhdpi ▾

 Show Safe Zone Show Grid

Cancel

Previous

Next

Finish



Confirm Icon Path

Android Studio

Res Directory:

main

Output Directories:

- ▼ main
 - ▼ res
 - ▼ mipmap-anydpi-v26
 - ic_launcher.xml
 - ic_launcher_round.xml
 - ▼ mipmap-mdpi
 - ic_launcher.png
 - ic_launcher_background.png
 - ic_launcher_foreground.png
 - ic_launcher_round.png
 - ▼ mipmap-hdpi
 - ic_launcher.png
 - ic_launcher_background.png
 - ic_launcher_foreground.png
 - ic_launcher_round.png
 - ▼ mipmap-xhdpi
 - ic_launcher.png
 - ic_launcher_background.png
 - ic_launcher_foreground.png
 - ic_launcher_round.png
 - ▼ mipmap-xxhdpi
 - ic_launcher.png

Output File

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@mipmap/ic_launcher_background"/>
    <foreground android:drawable="@mipmap/ic_launcher_foreground"/>
</adaptive-icon>
```



Cancel

Previous

Next

Finish



Obrigado!