

Sumário

1.	Informativo	2
2.	Pré-Requisitos	3
3.	Objetivo	4
4.	Versão	5
4.1.	minSdkVersion	5
4.2.	targetSdkVersion	5
5.	SplashScreen	6
6.	Novo Projeto	6
6.1.	styles.xml	9
6.2.	AndroidManifest.xml	10
6.3.	colors.xml	10
6.4.	activity_splash_screen.xml	11
6.5.	Animação	13
6.6.	Tela Secundária	17
7.	Resumo	20
8.	Referências	21

1. Informativo

Autor(a): Helena Strada

Data de criação: jan/2018

Escola Senai de Informática

CodeXP – Mobile

2. Pré-Requisitos

Java (Orientação a Objetos, APIs e Bibliotecas);

Básico de Android (Activites e Intents).

3. Objetivo

Entender o funcionamento de uma splashscreen (tela inicial), animações e como construí-la.

4. Versão

4.1. minSdkVersion

15

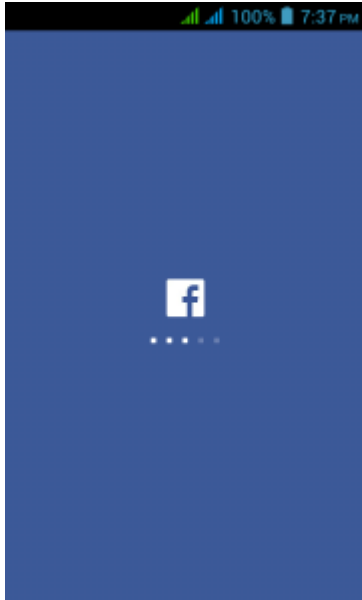
4.2. targetSdkVersion

26

5. SplashScreen

A splashscreen ou tela de apresentação é a tela inicial que aparece ao iniciarmos um aplicativo. Usualmente, as empresas utilizam a splashscreen para mostrar o logo da empresa ou o logo de quem desenvolveu o aplicativo.


Exemplo:



6. Novo Projeto

Vamos criar um novo projeto chamado SplashScreen.

Create New Project

 Create Android Project

Application name

SplashScreen

Company domain

br.senai.sp

Project location

C:\Users\helena.strada\Documents\XPMobileB-master\Projetos\SplashScreen

...

Package name

sp.senai.br.splashscreen

Edit

☐ Include C++ support

☐ Include Kotlin support

Previous

Next


Cancel

Finish

Clicar em “Next”.

Escolheremos a API mínima que queremos utilizar.

Create New Project

 Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ 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)

Previous

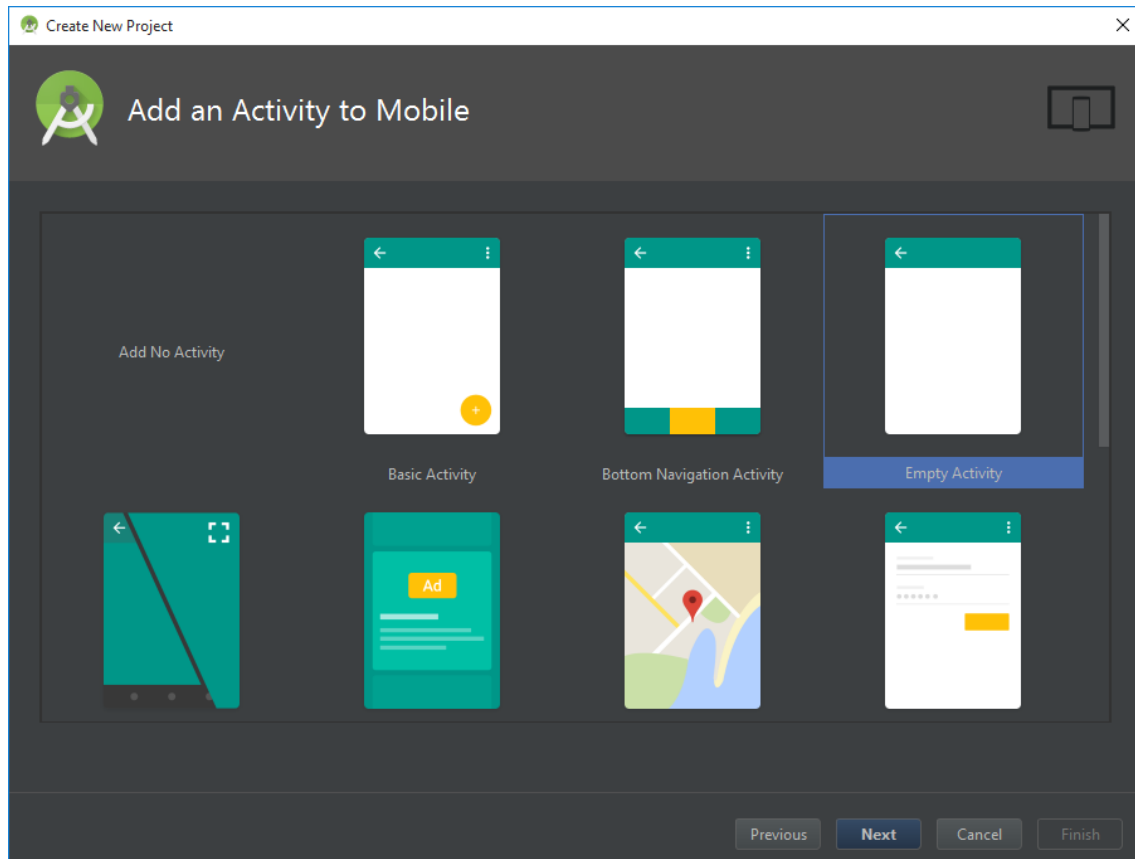
Next

Cancel

Finish

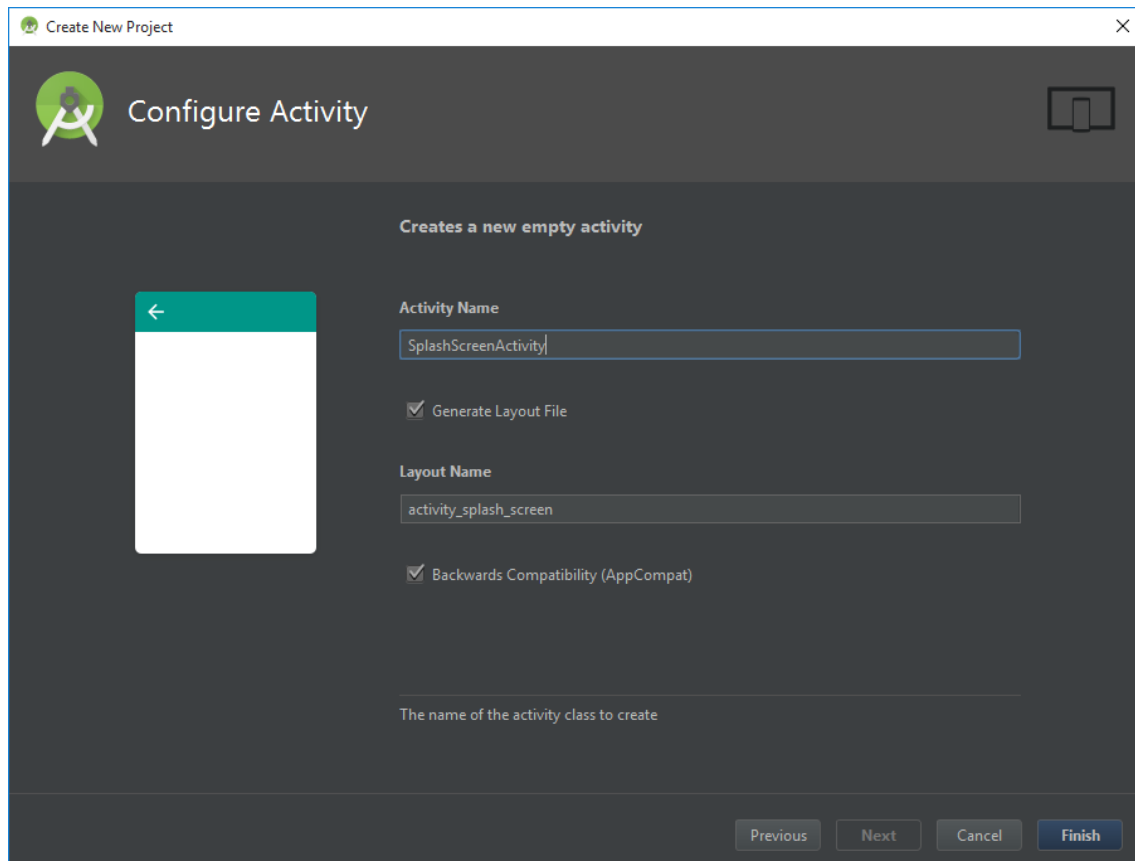
Clicar em “Next”.

Vamos manter como *Empty Activity*.



Clicar em “Next”.

Vamos alterar o nome da nossa activity principal para SplashScreenActivity.



Clicar em “Finish”.

6.1. styles.xml

Vamos abrir o nosso arquivo styles.xml e adicionar as nossas alterações:

```
<!-- FullScreen Theme -->
<style name="FullScreen" parent="AppTheme">
    <item name="windowActionBar">false</item>
    <item name="windowNoTitle">true</item>
    <item name="android:windowFullscreen">true</item>
</style>
```

O código completo deste arquivo:

```

<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <!-- FullScreen Theme -->
    <style name="FullScreen" parent="AppTheme">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
        <item name="android:windowFullscreen">true</item>
    </style>

</resources>

```

Vamos agora alterar o estilo da nossa tela principal, no nosso AndroidManifest.xml.

6.2. AndroidManifest.xml

Iremos alterar o estilo da nossa activity principal (SplashScreenActivity) para utilizar o tema atual que acabamos de adicionar no nosso arquivo *styles.xml*.

Além disso, iremos alterar a nossa cor de fundo da nossa tela principal.

```

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".SplashScreenActivity"
        android:theme="@style/FullScreen">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

```

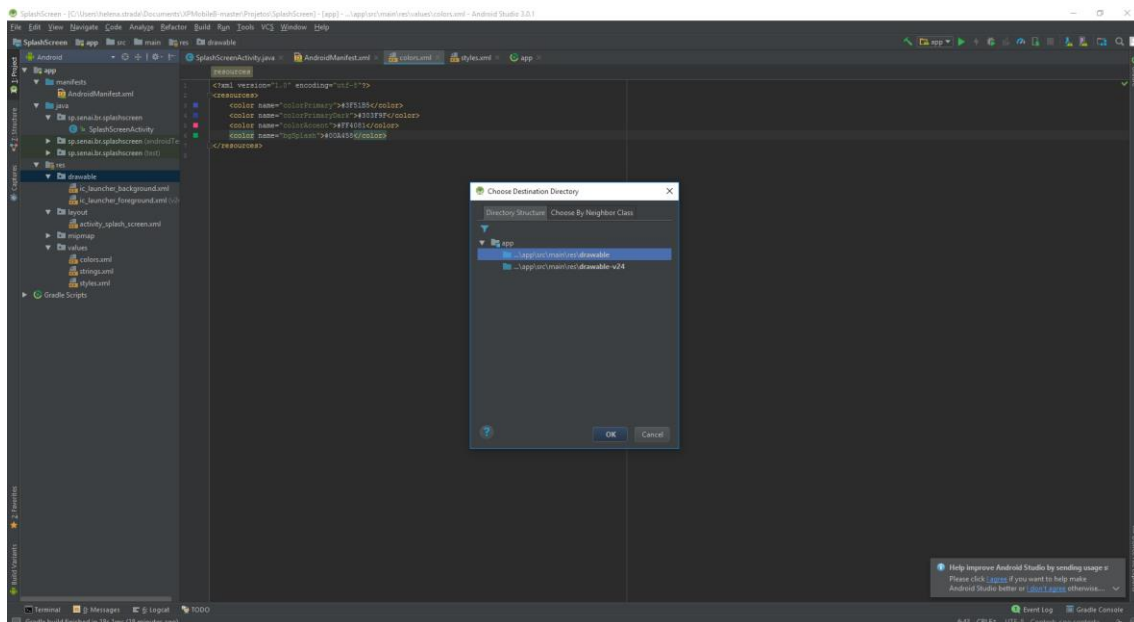
6.3. colors.xml

Iremos colocar uma cor apenas para teste. Neste caso, nosso colors.xml ficará assim:

```
resources
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="bgSplash">#00A458</color>
7  </resources>
8
```

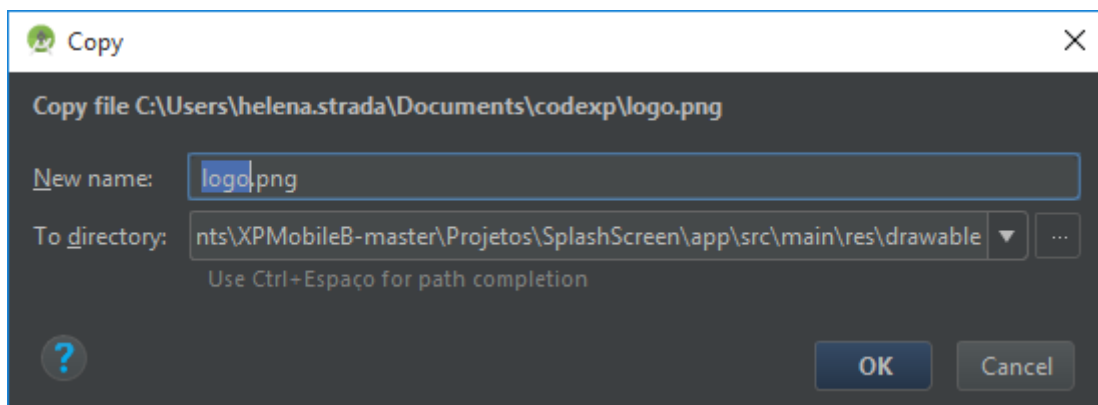
Iremos também, além do fundo, iremos escolher um logo para a nossa tela principal.

Escolha um logo e o inclua na pasta: res -> drawable -> arquivo.png



Clique em "Ok".

Escolha o nome do arquivo que deseja e clique em "Ok".



No nosso exemplo, iremos alterar o nome para "logo_splash.png".

6.4. activity_splash_screen.xml

Nós incluímos o logo na nossa aplicação, alteramos a cor, porém, ainda não fizemos a alteração na nossa activity.

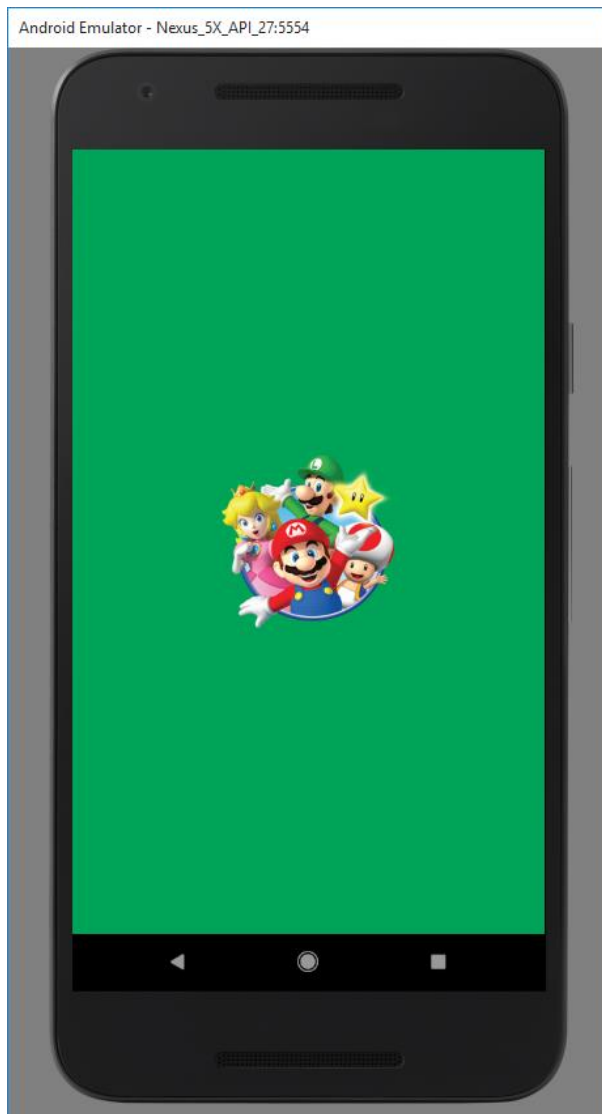
Nosso próximo passo será realizar as alterações que seguem:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="sp.genai.br.splashscreen.SplashScreenActivity"
    android:gravity="center"
    android:orientation="vertical"
    android:background="@color/bgSplash">

    <ImageView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:id="@+id/splash"
        android:background="@drawable/logo_splash" />

</LinearLayout>
```

Atualmente a nossa tela está assim:

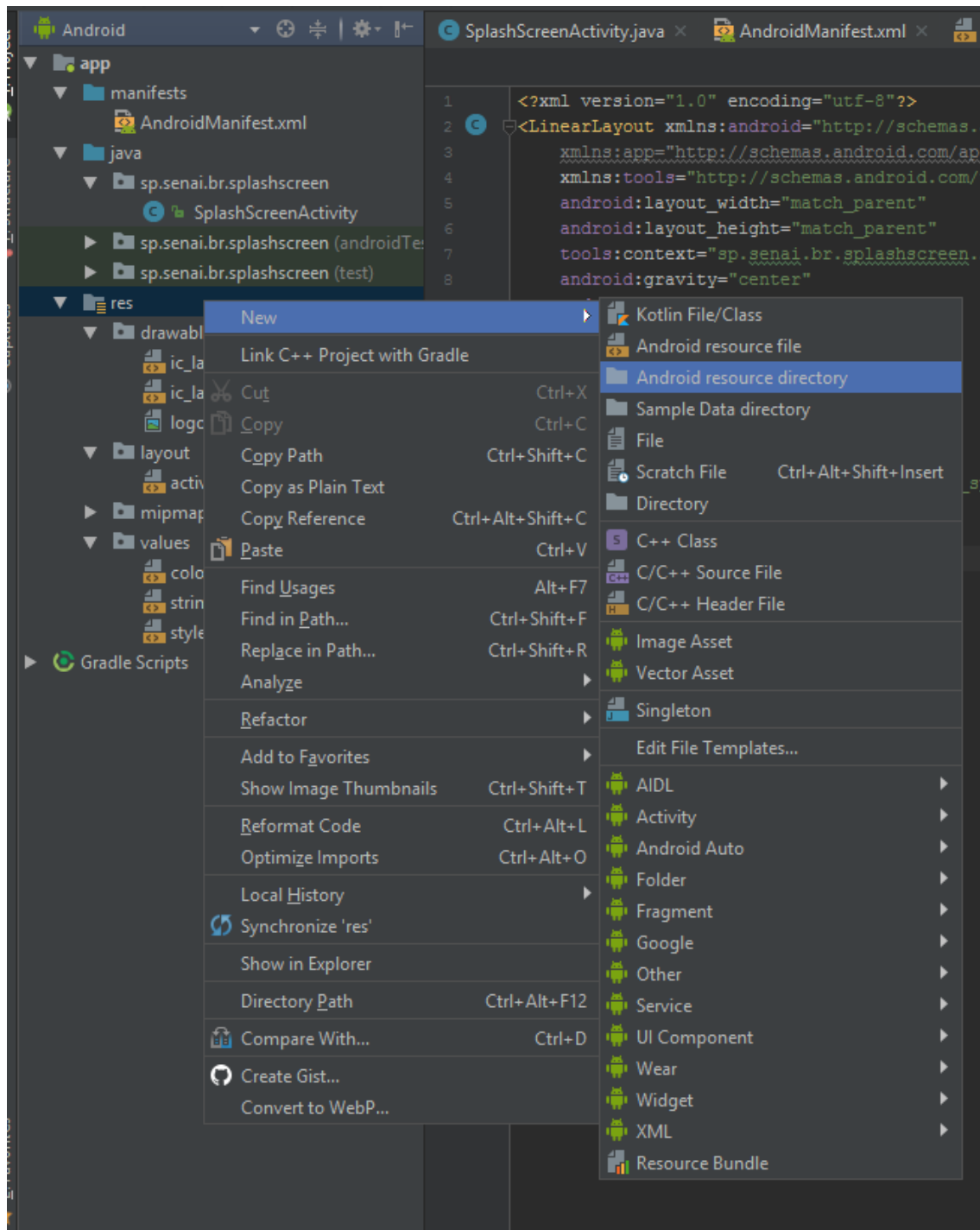


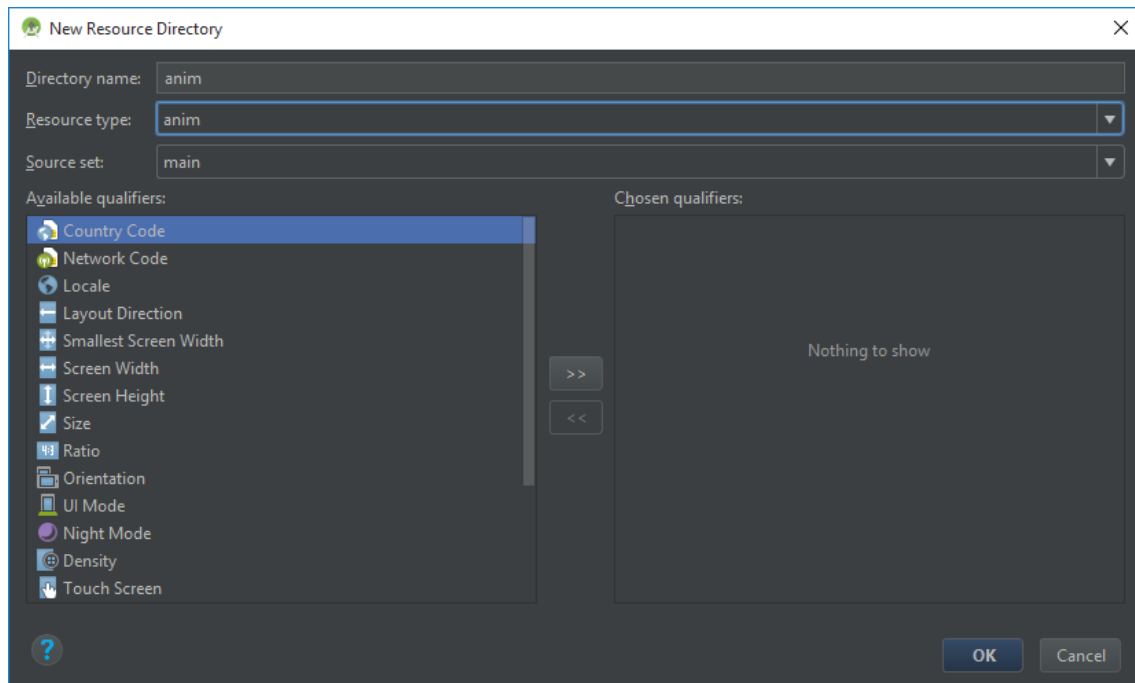
6.5. Animação

Já finalizamos a primeira parte do nosso projeto. Adicionamos o nosso logo e o nosso plano de fundo na tela principal. Porém, queremos que abra o nosso logo e logo em seguida, vá para uma outra tela de login, por exemplo.

Para isto, iremos criar uma animação na nossa tela principal e em seguida iremos redicioná-la para outra tela.

Dentro da pasta res, iremos criar uma pasta do tipo *"Android resource directory"*.

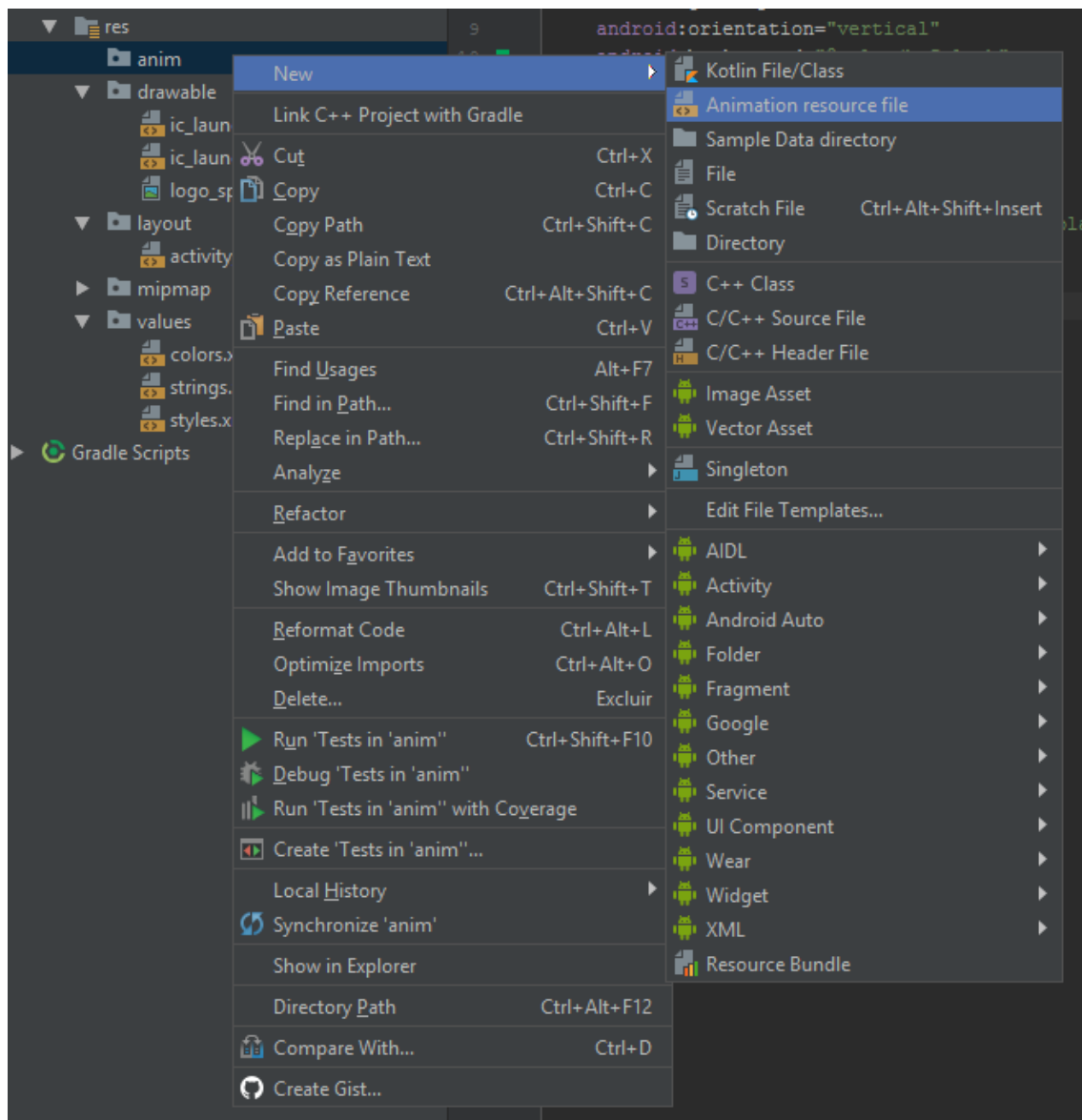




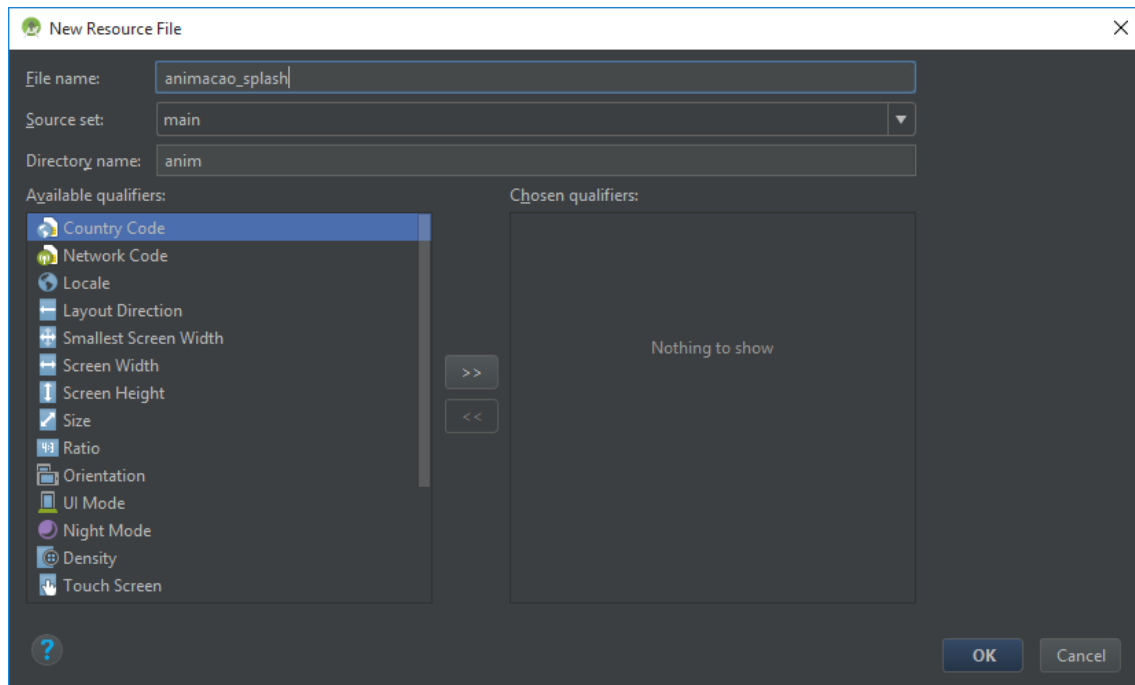
Escolheremos o nome do diretório de “*anim*” e o tipo de recurso que iremos utilizar também será *anim*.

Clicar em “*OK*” para finalizar a criação da nossa pasta.

Iremos criar agora um xml que conterà as informações que desejamos incluir em nossa animação.



Iremos colocar o nome de “*animacao_splash*”.



Clicar em “OK” para finalizar a criação do nosso arquivo.

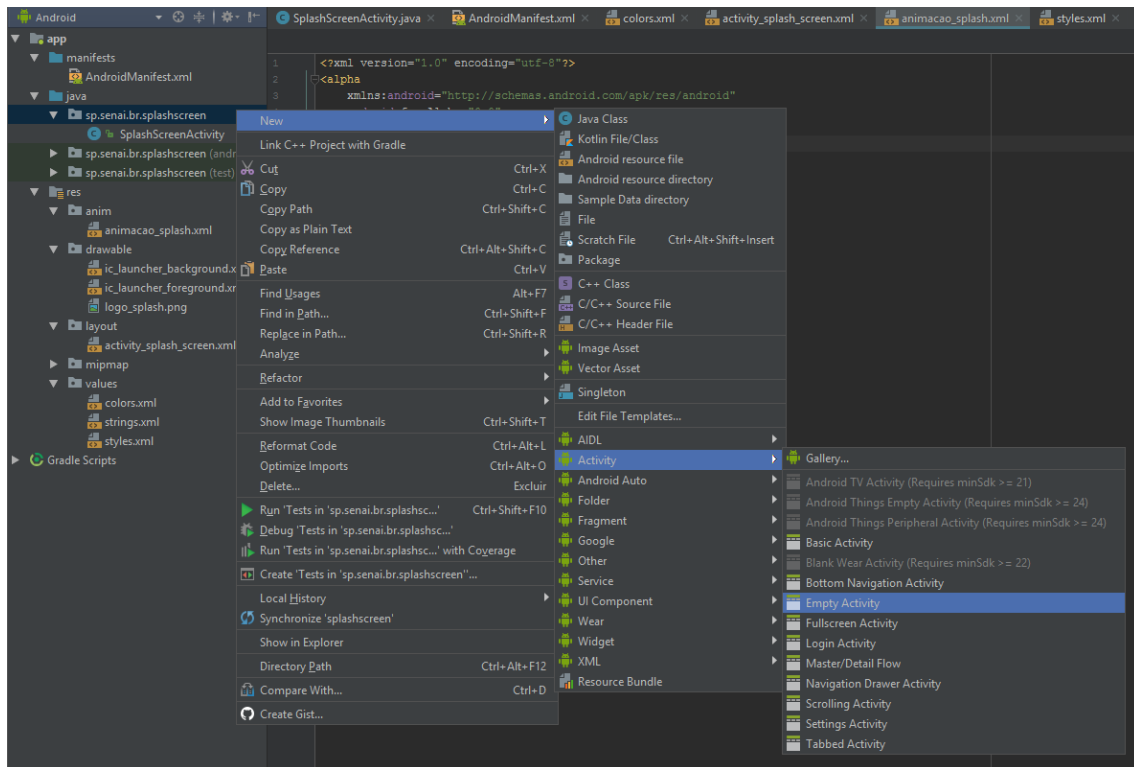
Iremos editar o arquivo que acabamos de criar (animacao_splash.xml) da seguinte maneira:

```
<?xml version="1.0" encoding="utf-8"?>
<alpha
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:fromAlpha="0.0"
  android:toAlpha="1.0"
  android:duration="3000" />
```

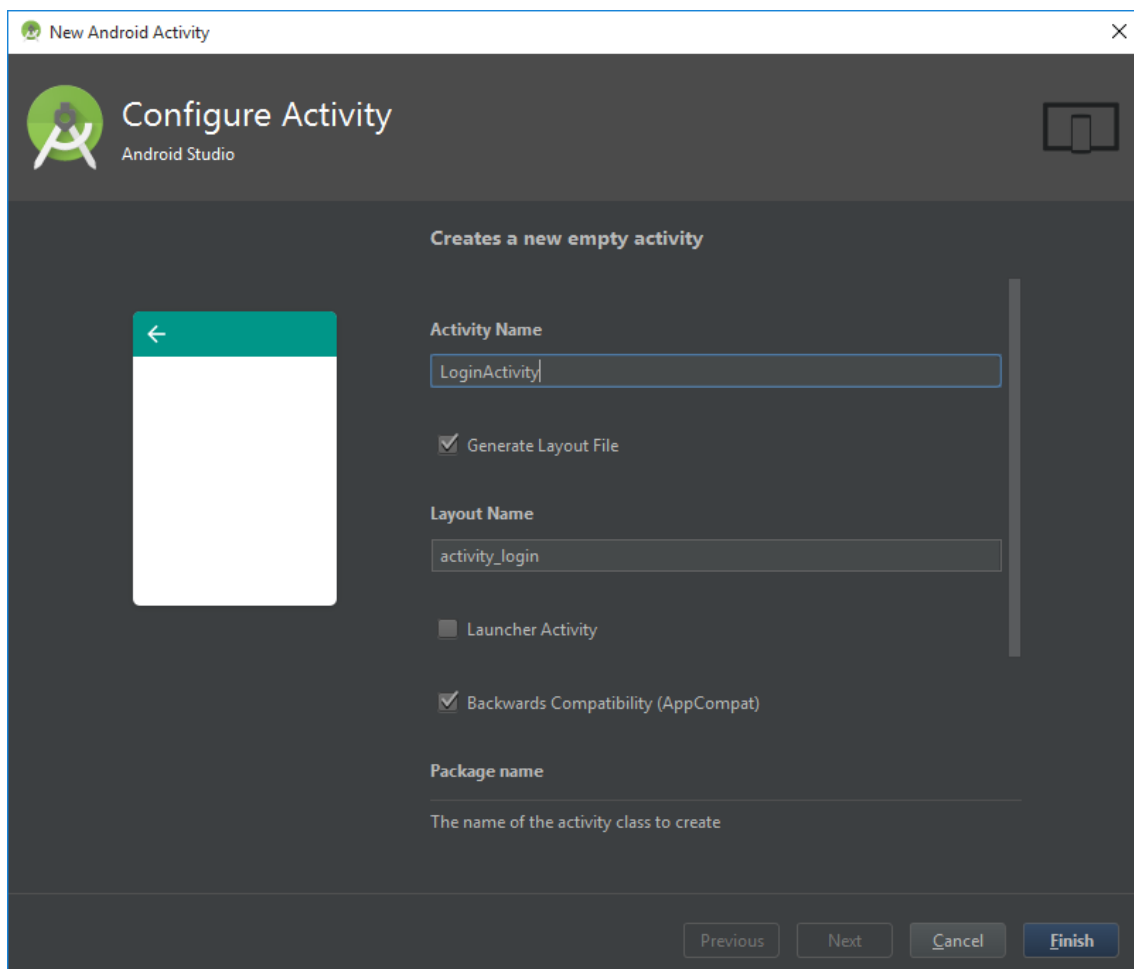
Com essas informações, iremos realizar um *fade-in* na nossa imagem.

6.6. Tela Secundária

Iremos agora adicionar uma segunda tela para o nosso aplicativo.



Iremos colocar o nome de *LoginActivity*.



Clicar em “Finish” para finalizar a criação da nossa segunda activity.

Criamos a nossa segunda tela, precisamos alterar agora o comportamento da nossa *SplashScreenActivity* para que, ao terminar de carregar o logo e tempo de carregamento do aplicativo, ele seja redirecionado para a tela de login ou qualquer outra tela correspondente.

```
package sp.senai.br.splashscreen;

import ...

public class SplashScreenActivity extends AppCompatActivity {

    // Tempo que a nossa SplashScreen ficará visível para o usuário
    private final int SPLASH_DISPLAY_LENGTH = 3500;

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

        // Iremos criar um método carregar para realizar a nossa ação
        carregar();
    }

    private void carregar() {
        Animation animation = AnimationUtils.loadAnimation(context: this, R.anim.animacao_splash);
        animation.reset();

        // Buscando o objeto que foi criado no layout (nossa imagem)
        ImageView imageView = findViewById(R.id.splash);
        if (imageView != null) {
            imageView.clearAnimation();
            imageView.startAnimation(animation);
        }

        new Handler().postDelayed(() - {
            // Após o tempo definido irá executar a próxima
            Intent intent = new Intent( packageContext: SplashScreenActivity.this, LoginActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
            // Como vimos anteriormente para irmos para uma nova Activity
            startActivity(intent);
            SplashScreenActivity.this.finish();
        }, SPLASH_DISPLAY_LENGTH);
    }
}
```

7. Resumo

Neste tutorial vimos como criar uma splashscreen e a importância para a nossa aplicação. Além de somente termos uma tela inicial, nós podemos carregar algo em segundo plano enquanto carregamos a tela inicial para o usuário.

8. Referências

<http://blog.alura.com.br/criando-uma-tela-de-abertura-no-android-splash-screen/>

<https://android.jlelse.eu/right-way-to-create-splash-screen-on-android-e7f1709ba154?gi=eecd127ab15f>