

Programação para dispositivos móveis com Android

Prof. Me. Eduardo Henrique Spies
Email: eduardo.spies@urisantiaago.br



O que iremos aprender?

- Visão geral do android
- Principais conceitos
 - Activities (Atividades)
 - Intents (Intenções)
 - Eventos
- Criação de um Player de música

O que é o Android?

- O Android é um **Sistema Operacional para dispositivos móveis** desenvolvido pela Google. Esse sistema possui um kernel (núcleo) baseado em Linux.



O que é o Android?

- Outra característica importante do Android é que ele é de **código aberto** e também oferece gratuitamente uma IDE de desenvolvimento chamada de **Android Studio**.



Principais componentes de um App Android

- São os blocos de construção de um aplicativo Android.

Cada componente é um ponto diferente pelo qual o sistema pode entrar em seu aplicativo. Cada componente é um bloco de construção exclusivo que ajuda a definir o comportamento geral do aplicativo.



Principais componentes de um App Android

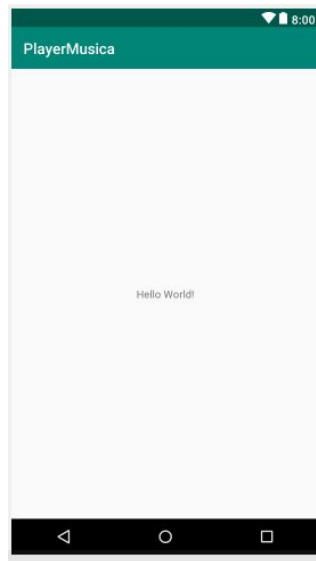
- Aplicativos Android tem quatro tipos principais de componentes:
 - **Atividades**
 - **Serviços**
 - **Provedores de conteúdo**
 - **Receptores de transmissão**



Principais componentes de um App Android

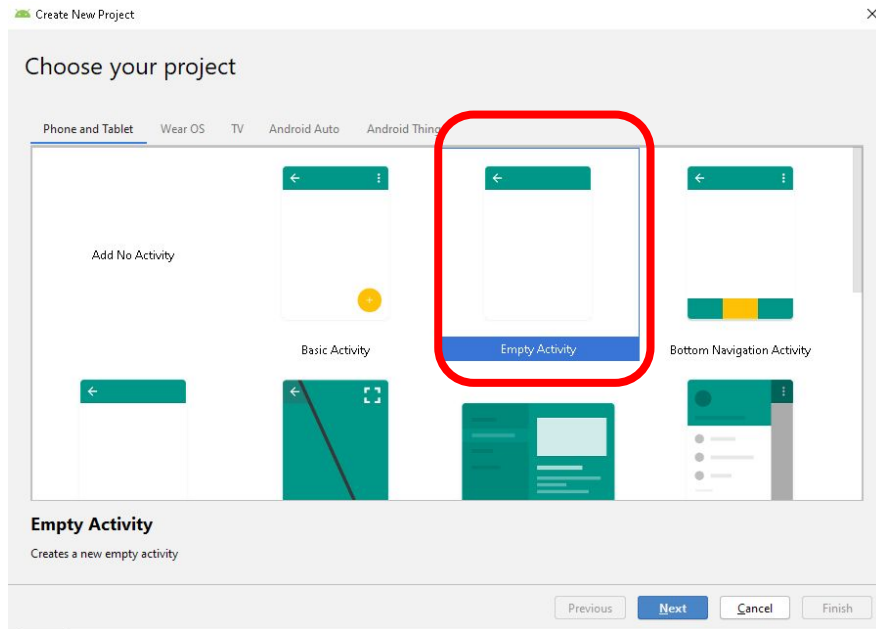
- **Atividades**

- Uma atividade representa uma **tela** única como uma interface de usuário. Por exemplo, um aplicativo de email pode ter uma atividade que mostra uma lista de novos e-mails, outra atividade que compõe um email e outra ainda que lê os e-mails.



Criando um novo projeto

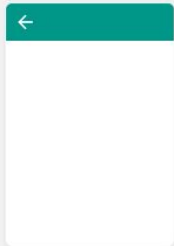
- No Android Studio Crie um novo projeto e escolha a opção **“Empty Activity”** (Atividade Vazia). e depois disso clique em **“Next”**.



Criando um novo projeto

- Após isso escolha o nome do projeto e selecione a linguagem de programação como Java e a versão do Android que você quer desenvolver.

Configure your project



Empty Activity

Creates a new empty activity

Name

Package name


Save location

Language

Java


Minimum API level

API 15: Android 4.0.3 (IceCreamSandwich)

 Your app will run on approximately 100% of devices.
[Help me choose](#)

☐ This project will support instant apps

☒ Use android:* artifacts

 Please enter an application name (shown in launcher), or a descriptive name for your library

Previous Next Cancel Finish

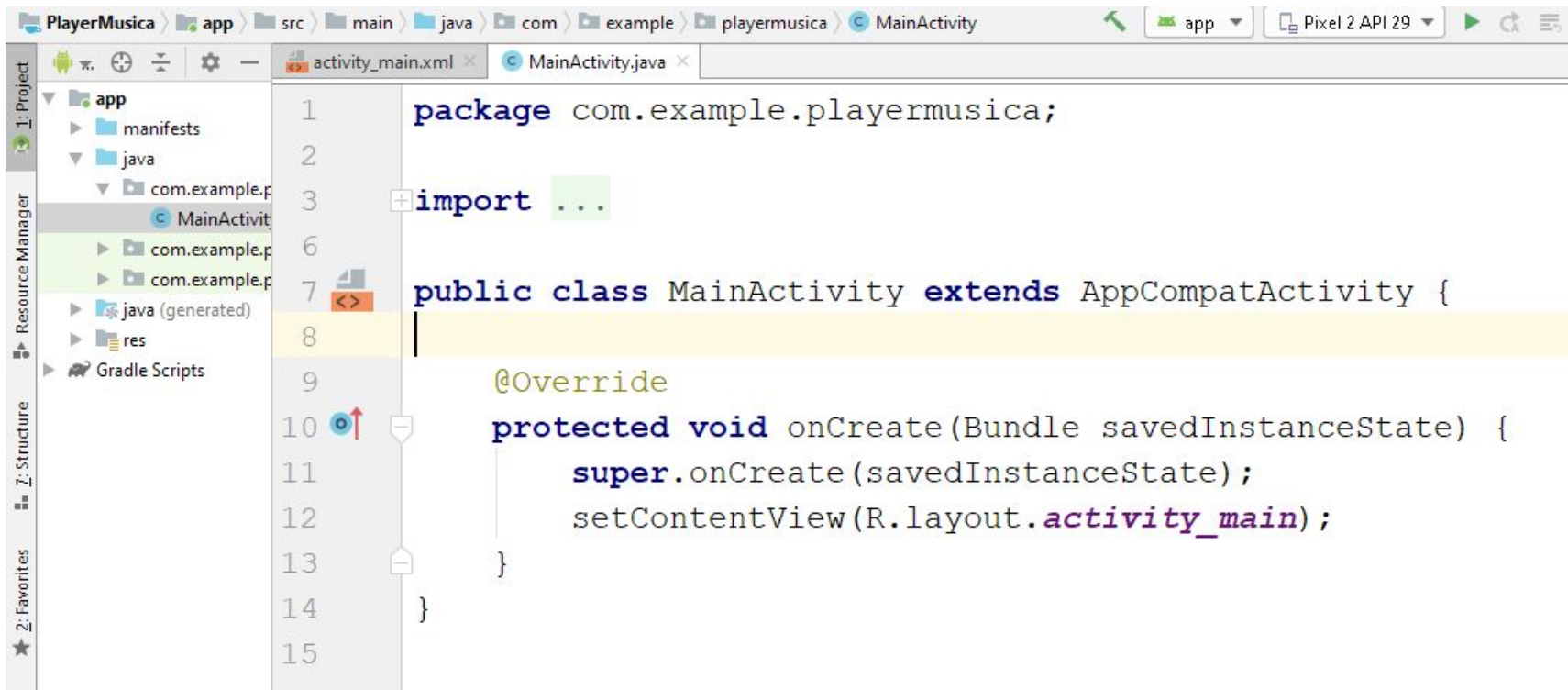


Manipulando uma atividade

- Cada atividade (*Activity*) representa uma tela de nosso programa Android. Manipulamos o comportamento dessa tela através de dois diferentes arquivos.
 - **Arquivo Java** - Representa a programação do comportamento do nosso app, é nesse arquivo que iremos realizar a programação.
 - **Arquivo de layout XML** - Arquivo utilizado para manipular a aparência da nossa tela.



Manipulando uma atividade



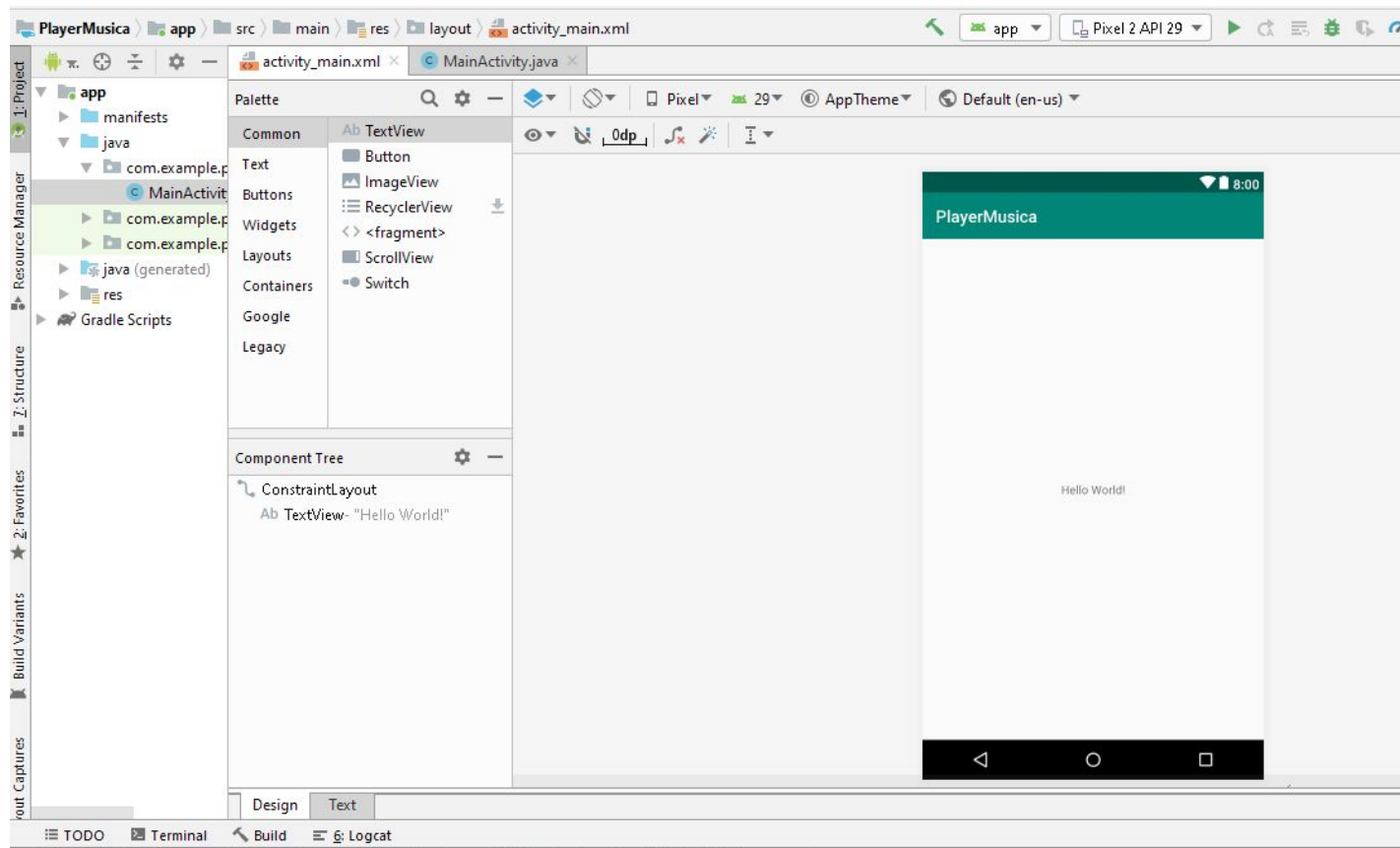
The screenshot shows the Android Studio IDE with the following components:

- Top Bar:** Displays the project name "PlayerMusica", the current file path "app > src > main > java > com > example > playermusica > MainActivity", and the target device "Pixel 2 API 29".
- Left Panel:** Contains the "Project" view showing the file structure of the "app" module, including "manifests", "java", and "res" folders. The "Resource Manager" and "Structure" tabs are also visible.
- Main Editor:** Displays the code for "MainActivity.java". The code is as follows:

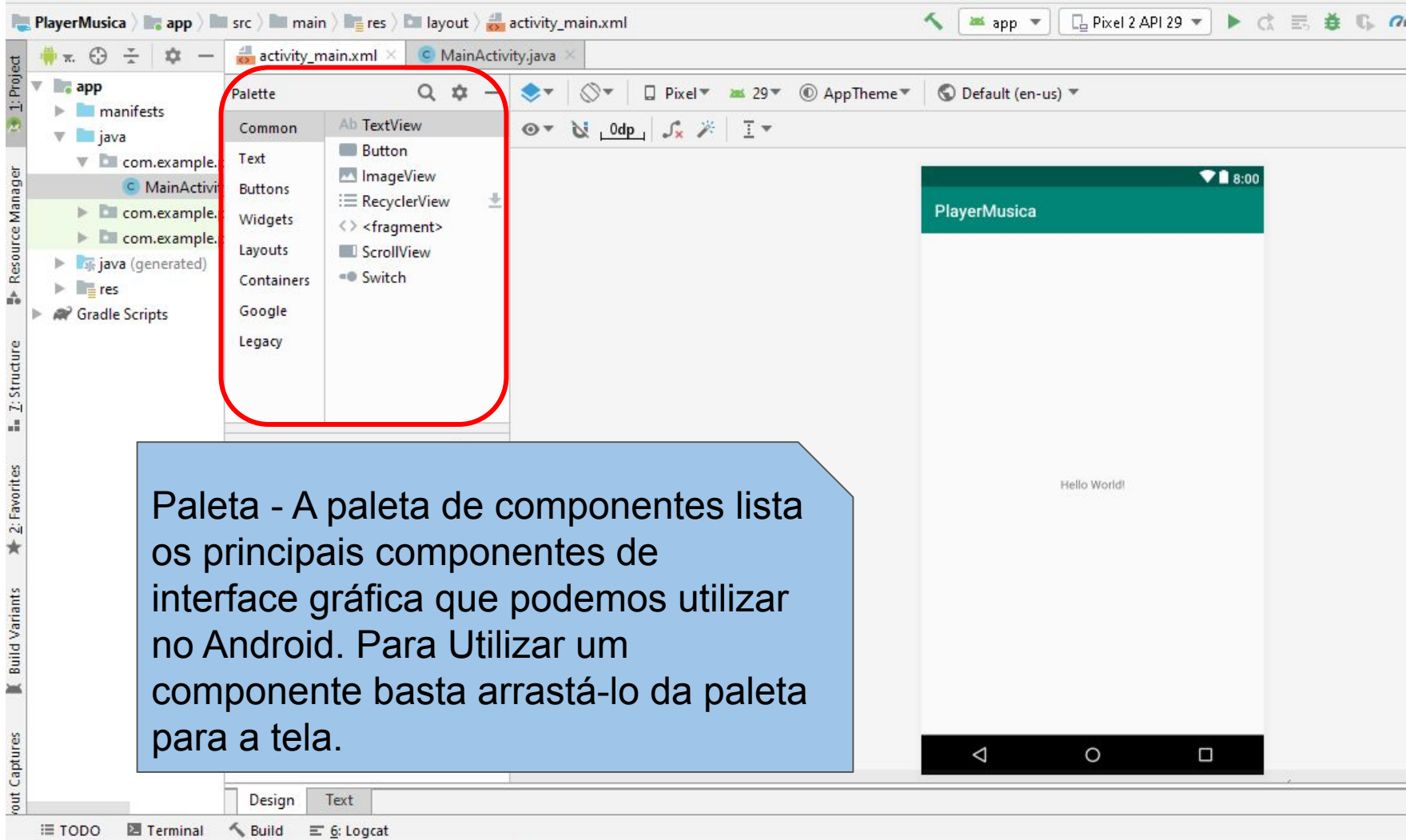
```
1 package com.example.playermusica;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
15
```



Manipulando uma atividade



ANDROID

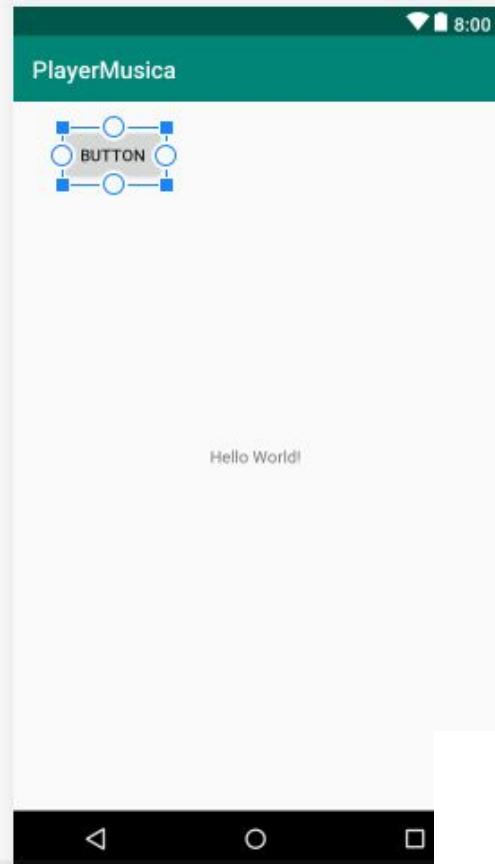


Paleta - A paleta de componentes lista os principais componentes de interface gráfica que podemos utilizar no Android. Para Utilizar um componente basta arrastá-lo da paleta para a tela.

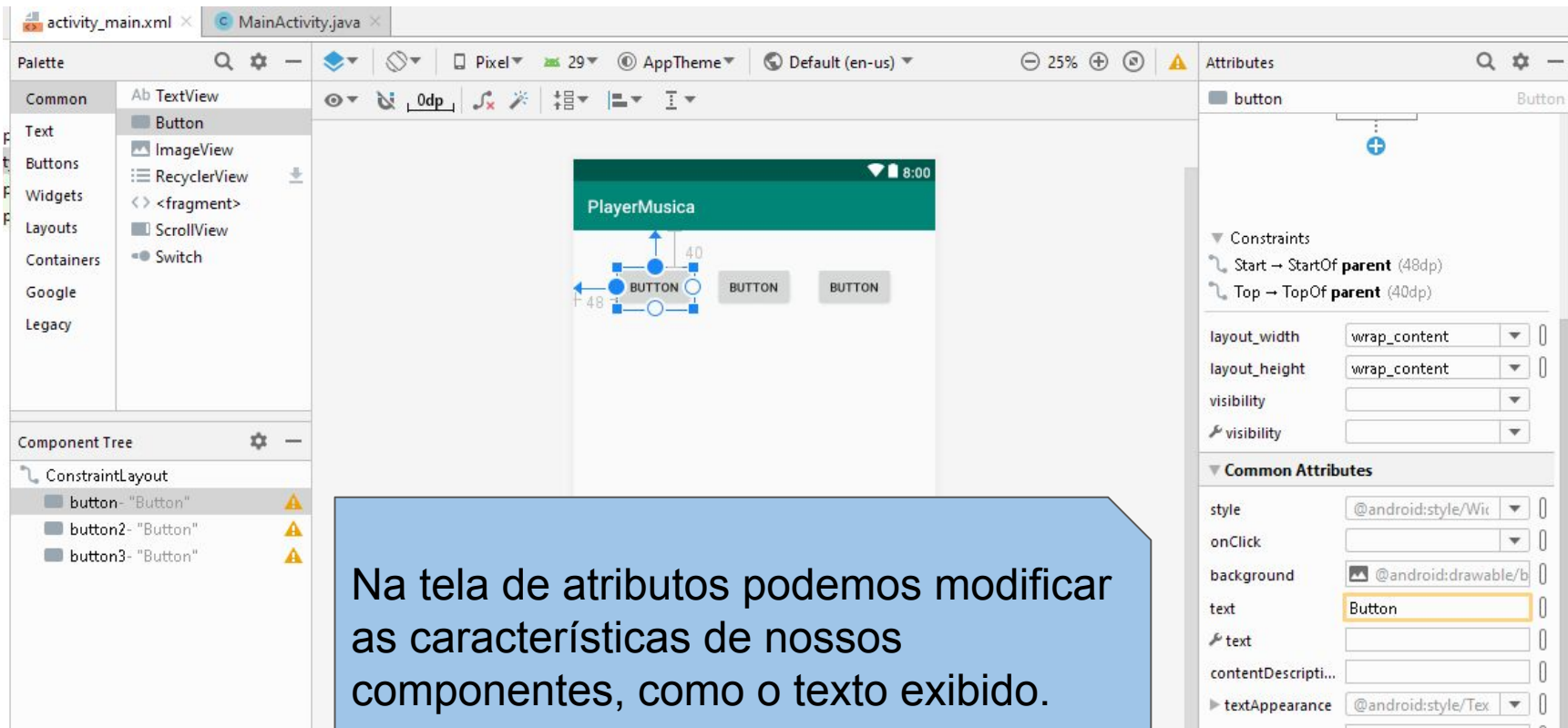
Componentes arrastado da paleta para a tela precisam ser “ancorados” ou seja, deve se estabelecer a sua posição relativa.

Isso é realizado conectando os pontos exibidos no componente com outros componentes ou com as bordas da tela.

Para funcionar, cada componente na tela precisa estar ancorado em pelo menos um local na horizontal e em um na vertical.



Alterando propriedades de componentes



Na tela de atributos podemos modificar as características de nossos componentes, como o texto exibido.

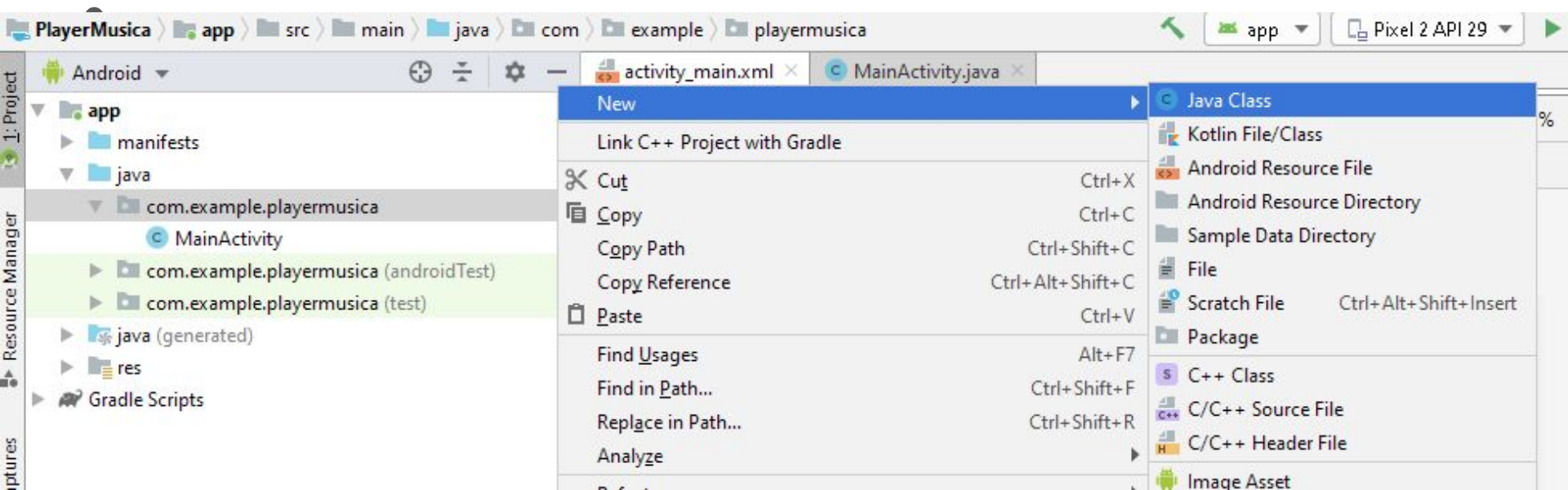


Partindo para a programação

- **Passo 1** - Criar uma classe Musica com informações de de nome e de artista.
 - Para fazer isso, selecione o projeto no canto esquerdo, dentro dele selecione a pasta java. Após isso selecione a pasta onde está a atividade recentemente criada e clique com o botão direito nele e selecione a opção “new->Java Class”



Partindo para a programação



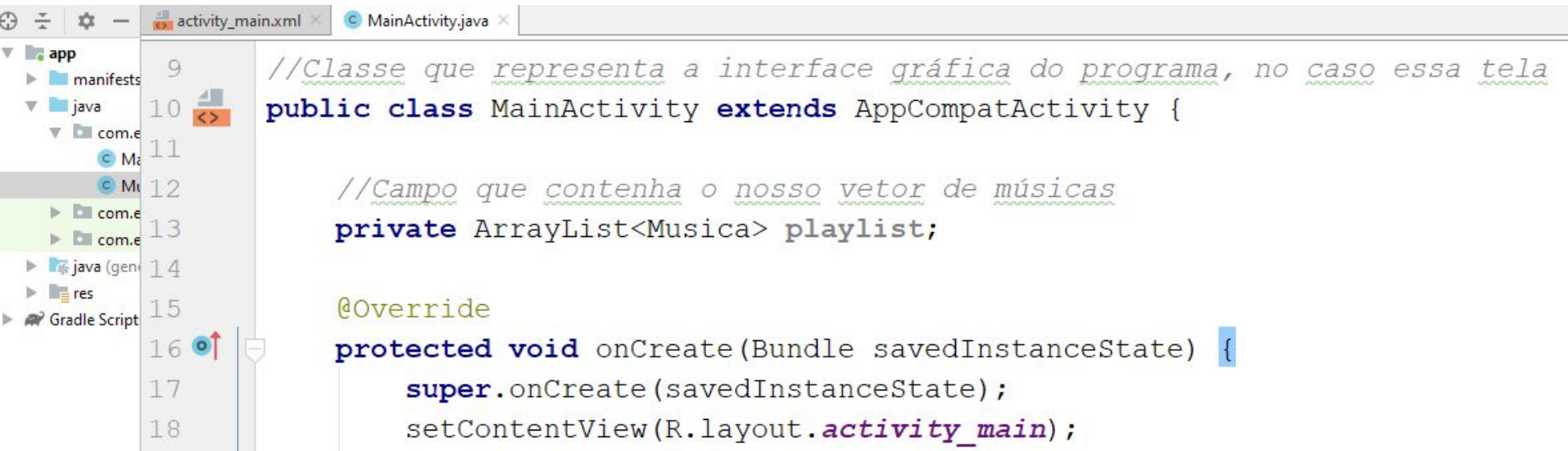
Partindo para a programação

```
2 //classe que representa a música que iremos mostrar na tela;
3 public class Musica {
4
5
6     private String nome;
7     private String artista;
8     private String nomeArquivo;
```

Para adicionar um construtor e os métodos get e set basta selecionar uma linha abaixo e usar o atalho “alt+insert” e selecionar a opção “constructor” marcando os dois campos depois disso a opção “getter e setter” também selecionando os campos.

Partindo para a programação

- **Passo 2** - Criar um campo do tipo ArrayList para representar nossa playlist de músicas.



```
9 //Classe que representa a interface gráfica do programa, no caso essa tela
10 public class MainActivity extends AppCompatActivity {
11
12     //Campo que contenha o nosso vetor de músicas
13     private ArrayList<Musica> playlist;
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main);
19     }
20 }
```

Partindo para a programação

- **Passo 3** - Inicializar o ArrayList dentro do método “onCreate”.
- **Passo 4** - Criar um método “inicializaPlaylist” que é chamado dentro do método “onCreate”.



Partindo para a programação

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    //Inicializa o ArrayList  
    playlist = new ArrayList<Musica>();  
  
    //método que faz a adição das músicas na playlist.  
    inicializaPlaylist();  
}
```



Partindo para a programação

- **Passo 5** - Criar o método “inicializaPlaylist” e adicionar as músicas dentro dele.



```
//Método que inicializa a nossa playlist de músicas.  
public void inicializaPlaylist() {  
    System.out.println("entrei na inicialização");  
    //Criando um novo objeto do tipo Musica com as informações necessárias.  
    Musica musica = new Musica( nome: "Hit Em Up (Dirty)",  
        artista: "2 Pac", nomeArquivo: "musica1");  
    //Usamos o método "add" da classe ArrayList para adicionar na playlist.  
    playlist.add(musica);  
  
    //adicionando a segunda música  
    musica = new Musica( nome: "Who shot Ya",  
        artista: "Notorious Big", nomeArquivo: "musica2");  
    playlist.add(musica);  
  
    //Adicionando a terceira música ...
```



Partindo para a programação

- **Passo 6** - Criar um atributo “musicaAtual” para controle de que música está tocando. E inicializar ele como “0” dentro do método “onCreate”



Partindo para a programação

```
activity_main.xml x MainActivity.java x
9 //Classe que representa a interface gráfica do programa, no caso essa tela
10 public class MainActivity extends AppCompatActivity {
11
12     //Campo que contenha o nosso vetor de músicas
13     private ArrayList<Musica> playlist;
14     private int musicaAtual;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.activity_main);
20         //Inicializa a musicaAtual como zero
21         musicaAtual = 0;
```



Partindo para a programação

- **Passo 7** - Implementar o método “tocarMusica”
- **Passo 8** - Criar um método “play” do botão e associá-lo ao mesmo.

```
//Método que manda tocar uma música ao pressionar "Play"  
public void play(View view) {  
    tocarMusica();  
}
```



Partindo para a programação

- **Passo 9** - Dentro do método tocar música, pegar os componentes de interface sobre os quais iremos mostrar as músicas.
- **Passo 10** - Criar uma música temporária no índice da música atual e atualizar a nossa interface gráfica com esses dados.



Partindo para a programação

```
//Método que manda tocar música e atualiza as informações da interface  
public void tocarMusica()  
//Pega os componentes textView que mostram o nome da música e artista  
TextView tv_mostraNome = findViewById(R.id.tv_nomeMusica);  
TextView tv_mostraArtista = findViewById(R.id.tv_mostraArtista);  
  
//Cria uma música temporária que representa a música que está sendo tocada  
Musica estaMusica = playlist.get(musicaAtual);  
//Coloca o texto na interface da música que está tocando  
tv_mostraArtista.setText(estaMusica.getArtista());  
tv_mostraNome.setText(estaMusica.getNome());  
}
```



Partindo para a programação

- **Passo 11** - Criar um método “passarFrente” e um “passarTras” e associar esses métodos aos seus respectivos botões.



Partindo para a programação

```
//Método que passa para frente  
public void passarFrente(View view) {  
    if (musicaAtual==playlist.size()-1) {  
        musicaAtual=0;  
        tocarMusica();  
    } else {  
        musicaAtual++;  
        tocarMusica();  
    }  
}
```



Partindo para a programação

```
//Método que passa para trás
public void passarTras(View view) {
    if (musicaAtual==0) {
        musicaAtual=4;
        tocarMusica();
    } else {
        musicaAtual--;
        tocarMusica();
    }
}
```



Rodando o app

- Para rodar o app Android temos duas principais alternativas:
 - Visualizar uma simulação do programa rodando através do emulador do Android Studio.
 - Instalando em um celular que possua Android



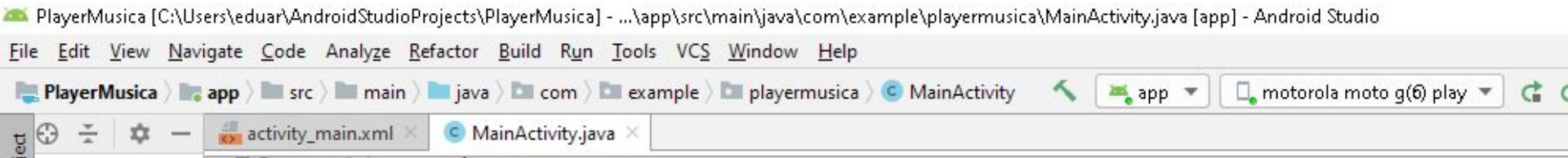
Instalando o app em um celular Android

- O primeiro passo para instalar o programa em um celular é habilitar o **modo de desenvolvedor**.
 - Vá em configurações do celular>Sistema>Sobre o dispositivo>Numero da versão e clique 8 vezes em cima dele. E uma mensagem aparecerá sobre ter habilitado o modo de desenvolvedor do celular.



Instalando o app em um celular Android

- Uma nova área de configuração do celular é habilitada chamada “opções do desenvolvedor”. Vá nela e habilite a opção depuração USB.
- Conecte o celular com o PC através de um cabo USB e no Android Studio verifique se ele foi encontrado.



Instalando o app em um celular Android

- Uma vez encontrado basta clicar no botão “play” verde ao lado para mandar instalar o aplicativo no seu celular.



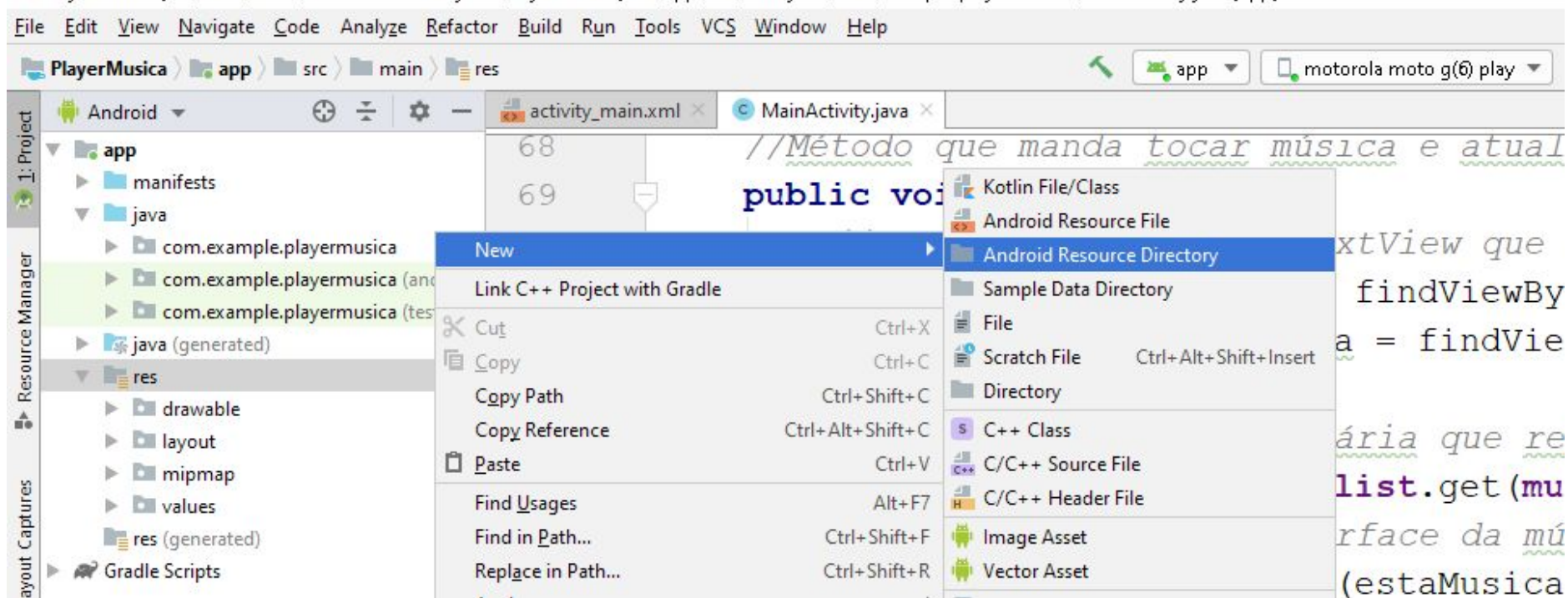
Continuando a programação

- **Passo 12** - Adicionar as nossas músicas ao programa, com as devidas nomenclaturas declaradas no código.
 - Criar uma pasta “raw” na pasta de recursos do Android.
 - Adicionar os arquivos corretamente nomeados nessa Pasta.

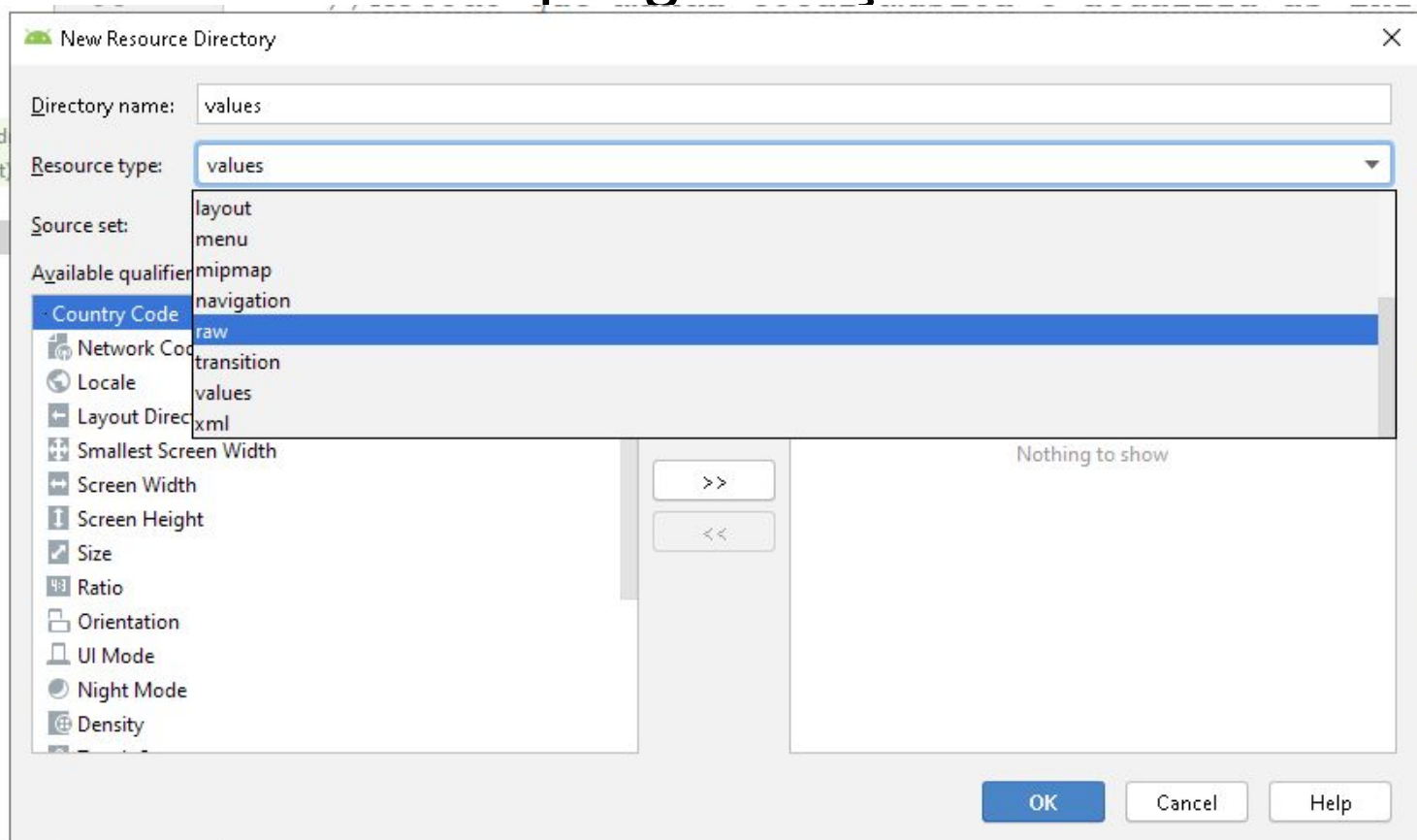


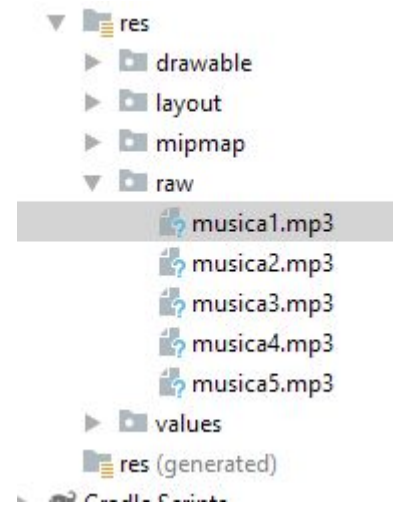
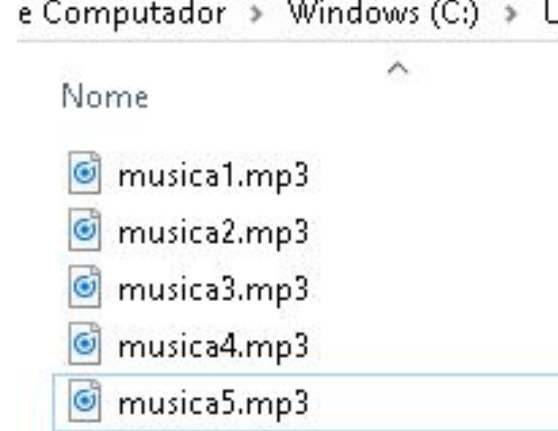
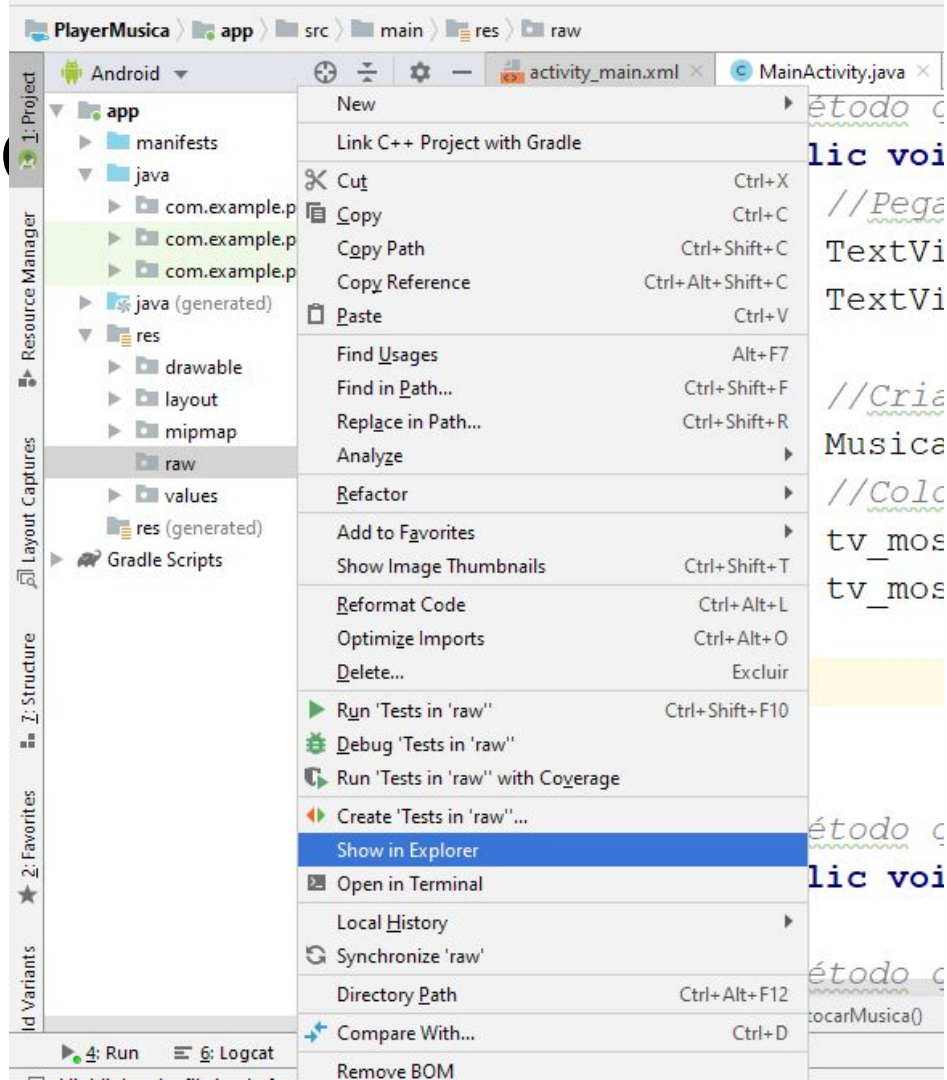
Continuando a programação

PlayerMusica [C:\Users\eduar\AndroidStudioProjects\PlayerMusica] - ...app\src\main\java\com\example\playermusica\MainActivity.java [app] - Android Studio



Continuando a programação





Continuando a programação

- **Passo 13** - Criar um campo do tipo MediaPlayer responsável pela execução de mídias.
- **Passo 14** - Inicialização do campo mediaPlayer no método “onCreate” da atividade.



Continuando a programação

```
//Classe que representa a interface gráfica do programa, no caso essa tela  
public class MainActivity extends AppCompatActivity {
```

```
//Campos que representam o nosso vetor de músicas, a música atual  
//e o MediaPlayer
```

```
private ArrayList<Musica> playlist;
```

```
private int musicaAtual;
```

```
private MediaPlayer mediaPlayer;
```

```
//Isso aqui já está dentro do onCreate
```

```
//Inicializa a variável mediaPlayer
```

```
mediaPlayer = MediaPlayer.create(context: this, R.raw.musical1);
```



Continuando a programação

- **Passo 15** - Atualizar o método “tocarMusica” para mudar a música do mediaPlayer para a música presente no índice “músicaAtual” da nossa “playlist”.
 - Para isso é necessário criar uma variável idArquivoMusica que será obtido através do nome do arquivo no objeto da música atual.
 - Instanciar o campo mediaPlayer utilizando o id da variável criada.



Continuando a programação

```
//AQUI IREMOS COMEÇAR COM AS AÇÕES DE MANIPULAÇÃO DE MÍDIA

//Criar uma variável local para armazenar o id do recurso mpr
int idArquivoMusica = getResources()
    .getIdentifier(estaMusica.getNomeArquivo(),
        defType: "raw", getPackageName());

//Criar uma variável do tipo MediaPlayer responsável por tocar o mp3
//Essa variável recebe como parâmetro o idArquivoMusica que acabamos
//de gerar com o nome do arquivo da música.
MediaPlayer mediaPlayer = MediaPlayer
    .create(context: this, idArquivoMusica);
```

Continuando a programação

- **Passo 16** - Mandar tocar a música no método “tocarMusica” através do método “start()” do objeto mediaPlayer.
- **Passo 17** - Mandar a música parar sempre que mandarmos trocar de música (senão as músicas entrarão uma em cima da outra)



Continuando a programação

```
//Agora basta mandar tocar  
mediaPlayer.start();
```

```
//Método que passa para frente  
public void passarFrente(View view) {  
    //parar a música antes de tocar a próxima  
    mediaPlayer.stop();  
    if(musicaAtual==playlist.size()-1) {
```



Resumo

- O que é o Android
- Principais componentes do Android
 - Em especial a Activity (Tela)
- Manipulação de Activities
 - Código Java com funcionalidade
 - Código XML com interfaces
- Criação de métodos para resposta de eventos
- Utilização da classe MediaPlayer para execução de áudio



Muito Obrigado!

- Prof. quero usar esse código fonte!
 - Disponível no github na url:
 - <https://github.com/Spiesssss/PlayerMusica.git>

