



Pós-Graduação Lato Sensu
Curso de Especialização em Tecnologia Java

APOSTILA – IMPLEMENTANDO CLASSES ENTIDADES

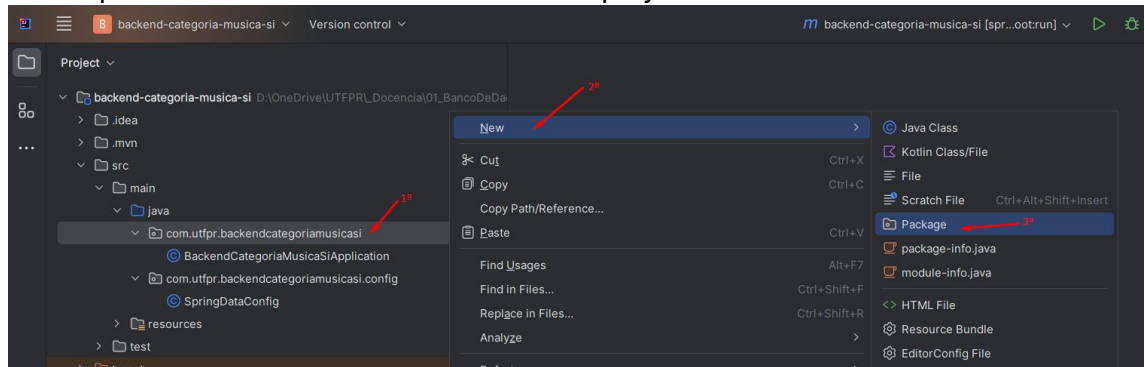
Professor Hugo Baker Goveia
hbakergoveia@hotmail.com



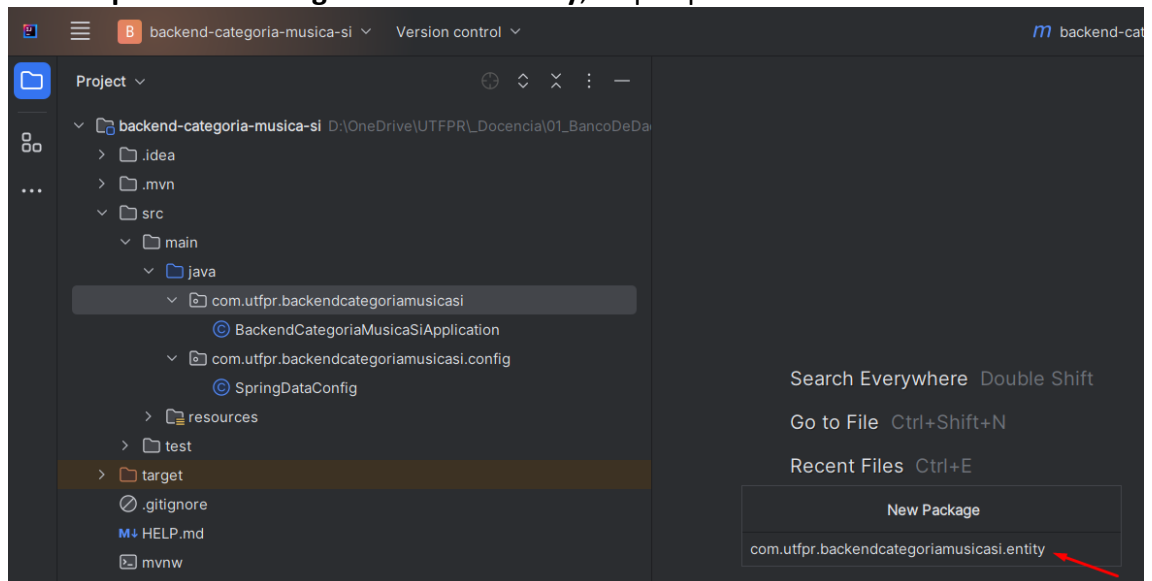
1. Implementando Classe Entidade

Basicamente, cada tabela em nosso banco de dados será refletida em uma classe entidade implementada em nossa aplicação e cada coluna que temos em nossa tabela no banco de dados será um atributo de nossa classe entidade.

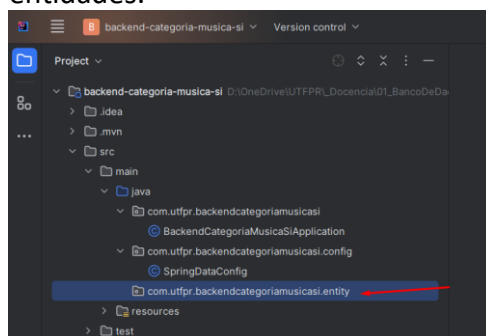
1. Clicar com o botão direito no **pacote raiz do seu projeto** -> **New** -> **Package** para criar o pacote das classes entidades de nosso projeto.



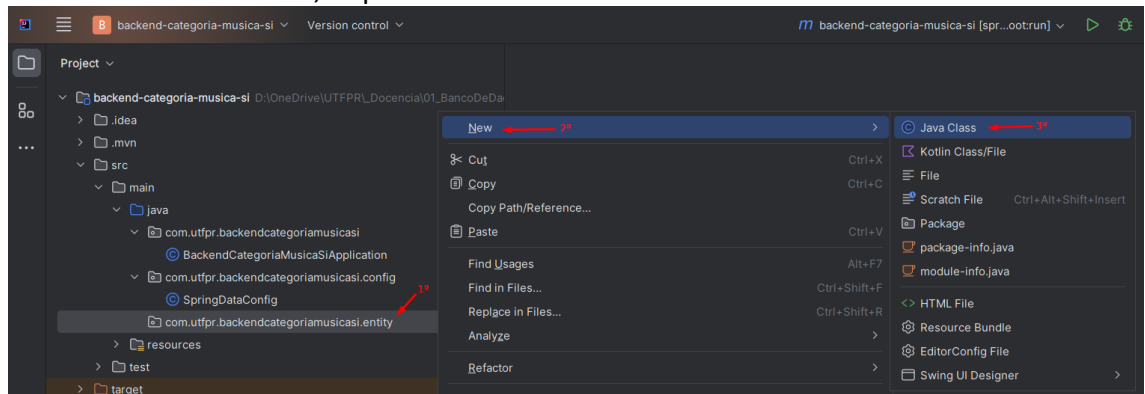
2. Na modal que abrir, informar o nome do pacote que será **com.utfpr.backendcategoriamicasasi.entity**, e após pressionar "ENTER".



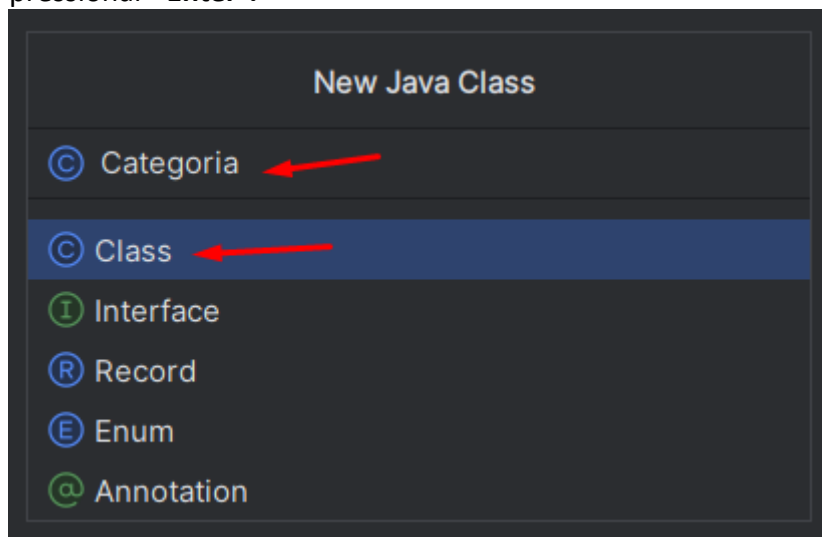
3. Após essa ação o novo pacote é criado e será nele que iremos criar nossas classes entidades.



4. Para criar a classe **Categoria**, você deve clicar no pacote (**com.utfpr.backendcategoriamicasi.entity**) que acabamos de criar com o botão direito do mouse, depois **"New->Java Class"**.



5. Manter o **"Class"** selecionado, informar o nome da classe **"Categoria"** e pressionar **"Enter"**.



6. Agora devemos inserir as anotações acima da assinatura da classe entidade que irão informar para o Spring que essa classe é uma classe entidade e deverá representar uma tabela do banco de dados. Também devemos fazer as devidas importações.

- **"@Entity"** esta anotação marca a classe como uma entidade gerenciada pelo framework ORM;

- **@Table(name = "categoria")** é usada para informar o nome da tabela que está no banco de dados, nesse exemplo temos a tabela categoria no banco de dados. Podem existir casos em que no banco de dados temos um nome de tabela diferente do nome que resolvemos dar para nossa classe entidade, por isso essa anotação é muito importante, pois é nela que fazemos essa relação.

```

1 package com.utfpr.backendcategoriamicasi.entity;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.Table;
5
6 no usages
7 @Entity
8 @Table(name = "categoria")
9 public class Categoria {
10
11 }

```

7. Agora devemos implementar os atributos da classe, que devem referenciar as colunas que temos na tabela “categoria” do banco de dados.

@Id é usada para indicar o atributo que representa a chave primária da tabela correspondente no banco de dados.

@GeneratedValue(strategy = GenerationType.IDENTITY) é usada em conjunto com a anotação **@Id** para especificar como o valor da chave primária deve ser gerado durante a persistência.

Quando uma entidade é persistida no banco de dados, é necessário atribuir um valor à sua chave primária. A estratégia de geração de valor especificada com **@GeneratedValue** define como esse valor será gerado.

No caso específico da estratégia **GenerationType.IDENTITY**, ela indica que a geração do valor da chave primária será delegada ao banco de dados. Isso significa que o banco de dados será responsável por gerar e atribuir um valor único à chave primária no momento da inserção do registro.

Essa estratégia é comumente usada quando o banco de dados suporta colunas de autoincremento. Ao usar a estratégia **GenerationType.IDENTITY**, o JPA se integra com a funcionalidade de autoincremento do banco de dados para gerar valores sequenciais para a chave primária automaticamente.

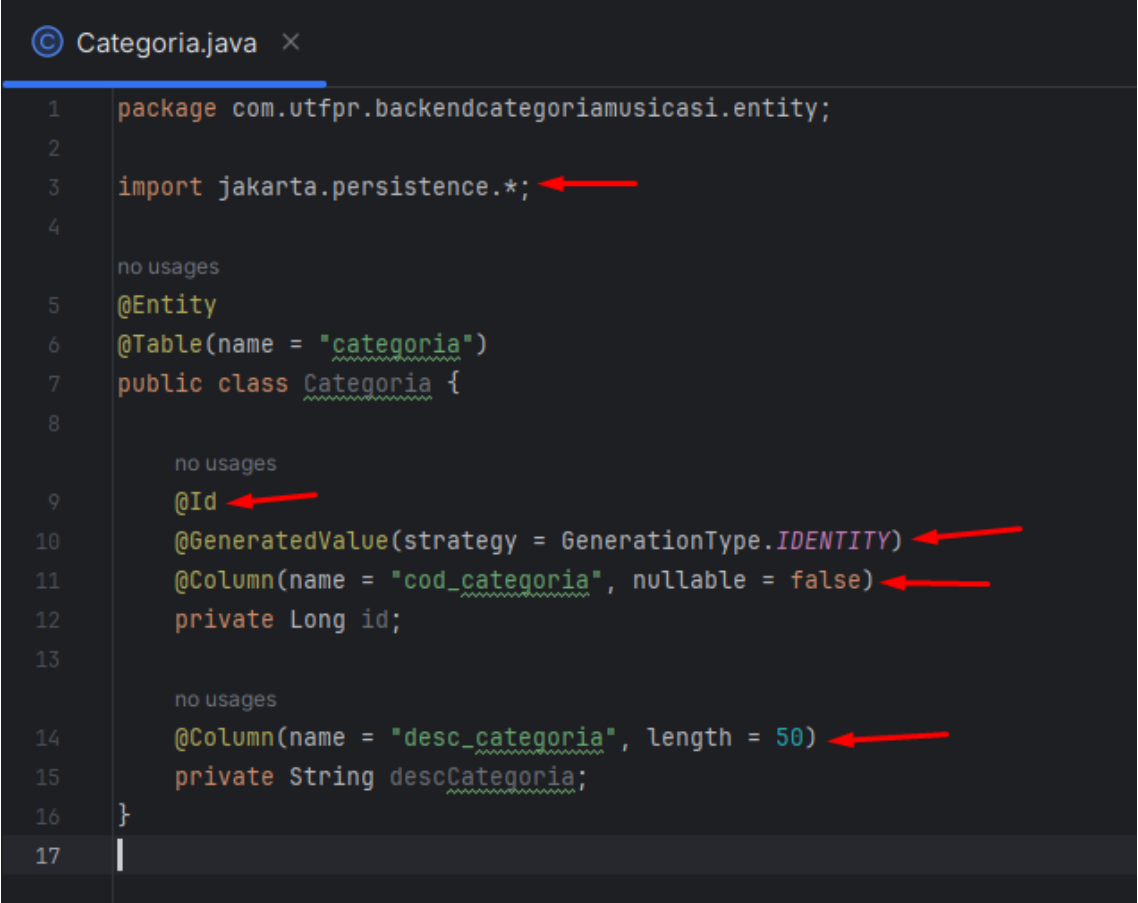
@Column(name = "cod_categoria", nullable = false) é usada para mapear um atributo da classe de entidade a uma coluna específica do banco de dados, permitindo personalizar detalhes da coluna.

A propriedade **name** especifica o nome da coluna no banco de dados.

Além disso, a propriedade **nullable** está definida como **false**, indicando que a coluna “cod_categoria” não pode conter valores nulos. Isso implica que sempre

que um registro for inserido ou atualizado nessa tabela, o valor para a coluna "cod_categoria" deverá ser fornecido.

@Column(name = "desc_categoria", length = 50) Nessa anotação temos uma particularidade que é o tamanho do campo (length) que nesse exemplo ficou como 50 caracteres.



```
1 package com.utfpr.backendcategoriamicasi.entity;
2
3 import jakarta.persistence.*;
4
5 no usages
6 @Entity
7 @Table(name = "categoria")
8 public class Categoria {
9
10     no usages
11     @Id
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     @Column(name = "cod_categoria", nullable = false)
14     private Long id;
15
16     no usages
17     @Column(name = "desc_categoria", length = 50)
18     private String descCategoria;
19 }
20
```

É importante que você faça a devida importação de tudo que for solicitado pela IDE. No caso do IntelliJ, ao perceber que vamos usar várias importações do “**jakarta.persistence**” ele automaticamente muda a importação para “**jakarta.persistence.***” visando contemplar todas as anotações que estamos usando nessa classe.

8. Para finalizar, devemos incluir a anotação **@Data** na assinatura da classe entidade para que o **Lombok** seja utilizado nessa classe e assim temos automaticamente em nossa classe a implementação dos getters e setters, de acordo com a imagem abaixo.

```
© Categoria.java x
1 package com.utfpr.backendcategoriamicas.entity;
2
3 import jakarta.persistence.*;
4 import lombok.Data;
5
6 no usages
7 @Entity
8 @Table(name = "categoria")
9 @Data
10 public class Categoria {
11
12     no usages
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     @Column(name = "cod_categoria", nullable = false)
16     private Long id;
17
18     no usages
19     @Column(name = "desc_categoria", length = 50)
20     private String descCategoria;
21 }
22
```

9. Repita o processo para implementar a classe entidade **Música** conforme imagem abaixo. É importante ressaltar que temos nela uma diferença no atributo Categoria que referencia a chave estrangeira do relacionamento das duas tabelas no banco de dados que será implementado aqui também. Nesse exemplo usamos o tipo de relacionamento **ManyToOne** de forma que Muitas músicas podem ter uma categoria. E para isso usamos a anotação **@ManyToOne**

No **@JoinColumn(name = "cod_categoria", nullable = false)** nós informamos que o nome da coluna no banco de dados que possui nossa chave estrangeira é **cod_categoria**.

Abaixo segue imagem que demonstra a implementação da classe entidade Música e o relacionamento ManyToOne.

