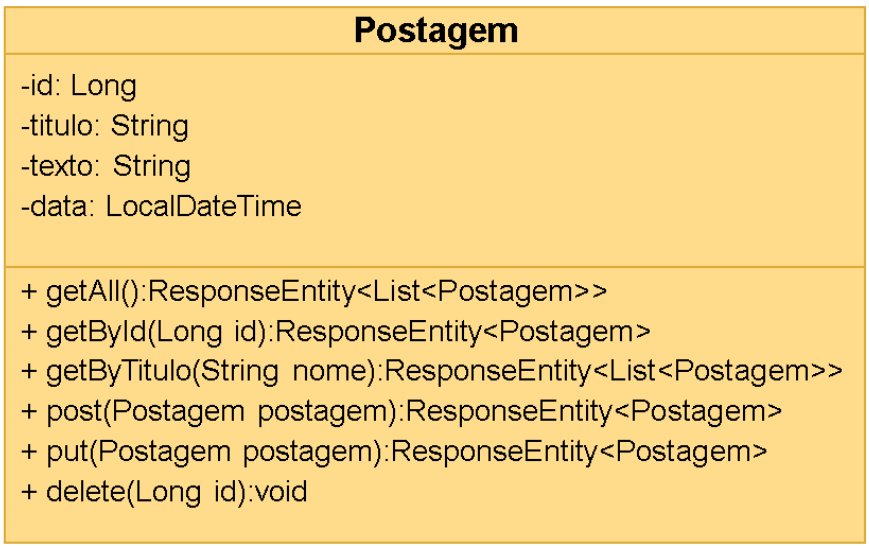
Projeto 02 - Blog Pessoal - CRUD 03

O que veremos por aqui:

1. Criar a Interface PostagemRepository

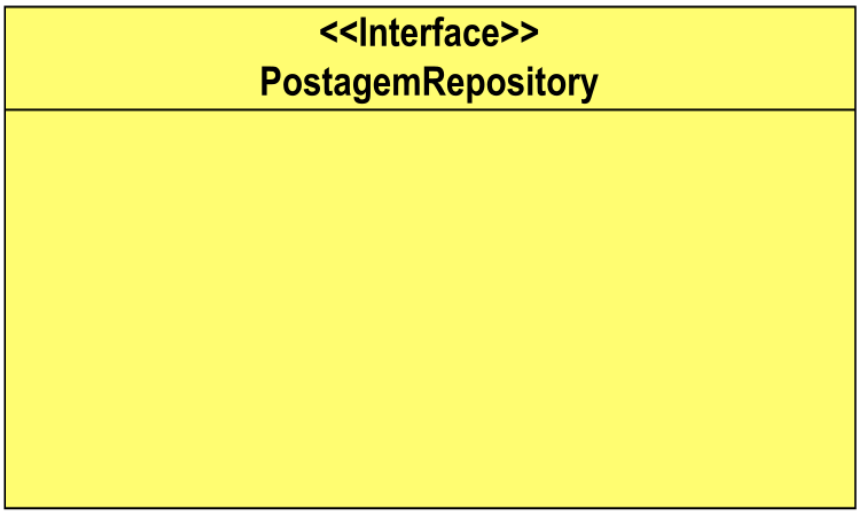
1. O Recurso Postagem

Na etapa anterior, começamos a construir o Recurso Postagem, a partir da Classe Model Postagem, onde implementamos todos os atributos do recurso. Veja o Diagrama de Classes abaixo:



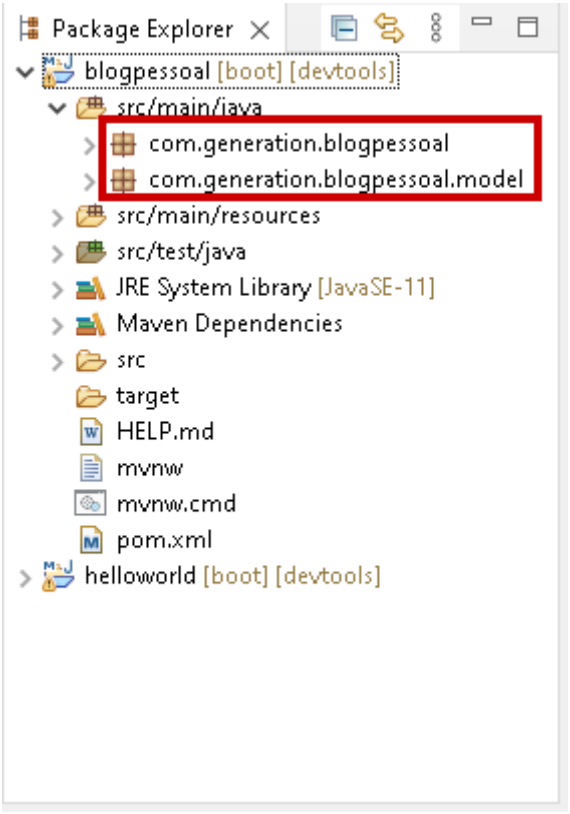
Nesta etapa, vamos construir a **Interface PostagemRepository**, que irá nos auxiliar na interação com o Banco de dados. Esta Interface contém diversos métodos pré-implementados para a manipulação dos dados de uma entidade, como métodos para salvar, deletar, listar e recuperar dados dos objetos persistidos, além de criar consultas personalizadas chamadas **Query Methods**, que veremos mais a frente. Para utilizar todos os métodos da Interface Repository, vamos criar dentro da Interface PostagemRepository uma **Herança** (Extends) com a Interface **JpaRepository**. Os métodos descritos no Diagrama de Classes serão implementados na próxima etapa, na Classe **PostagemController**.

O Diagrama de Classes da nossa Interface será igual a imagem abaixo:



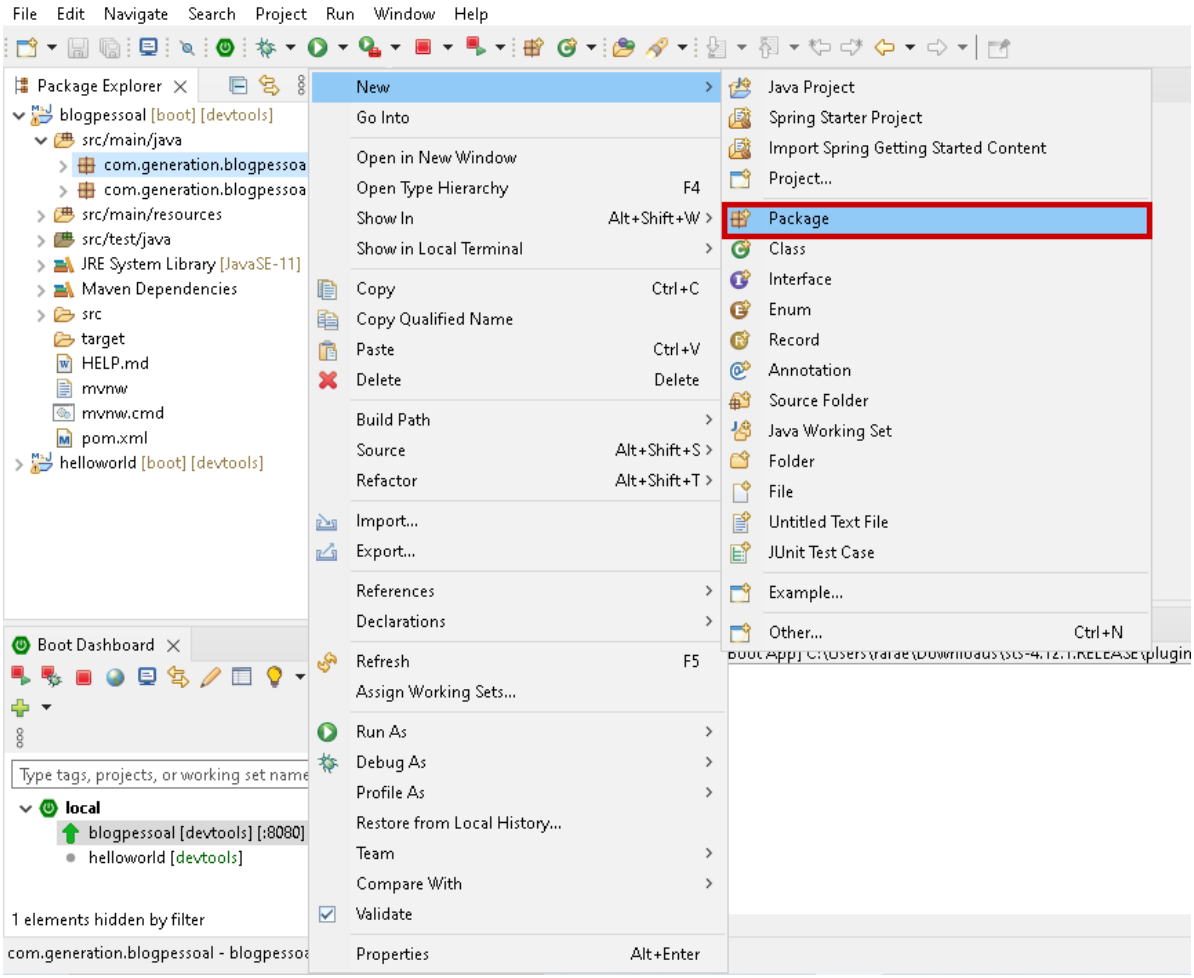
👣 Passo 01 - Criar o Pacote Repository

Na Source Folder Principal (**src/main/java**), observe que já foi criado o pacote Principal da nossa aplicação (**com.generation.blogpessoal**) e o pacote Model (**com.generation.blogpessoal.model**). Na figura abaixo, podemos visualizar os 2 pacotes:

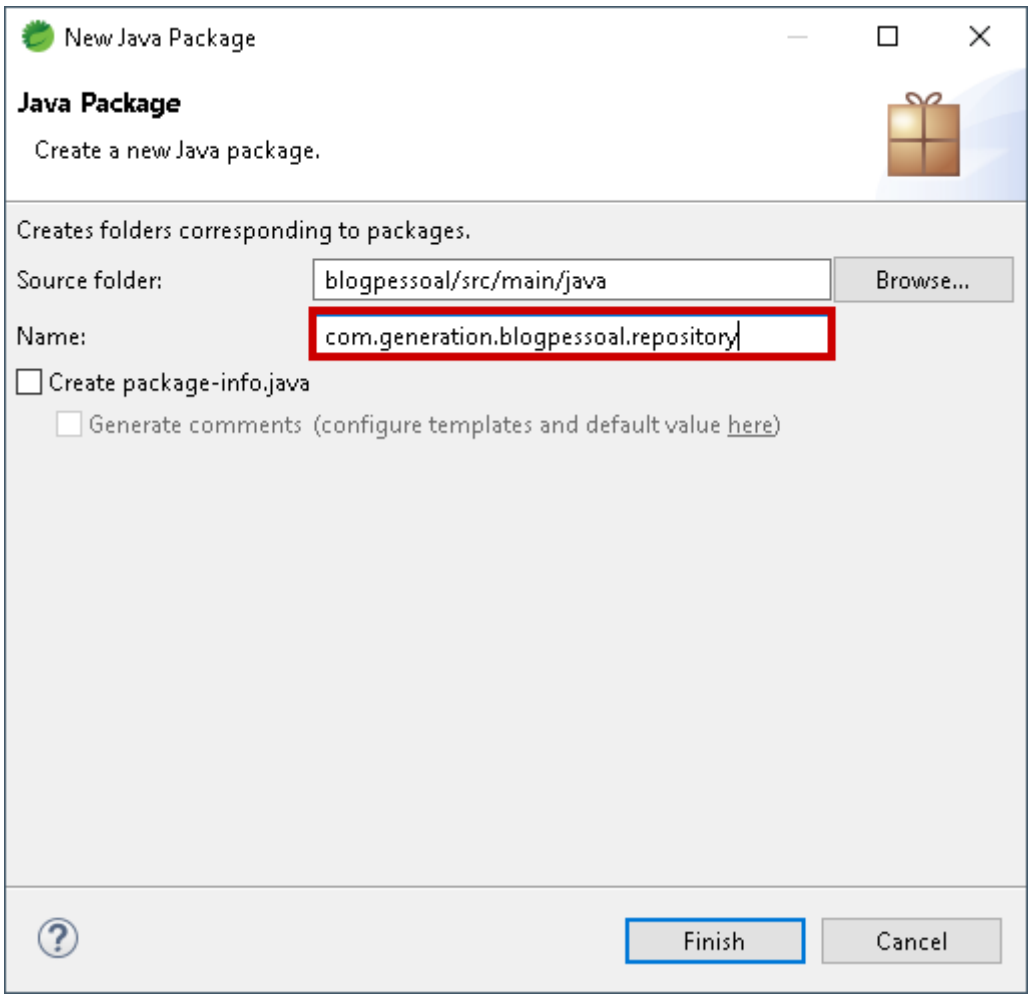


Nesta etapa, vamos criar a **Camada Repository**:

1. No lado esquerdo superior, na Guia **Package explorer**, clique com o botão direito do mouse sobre a Package **com.generation.blogpessoal**, na Source Folder **src/main/java** e clique na opção **New → Package**.

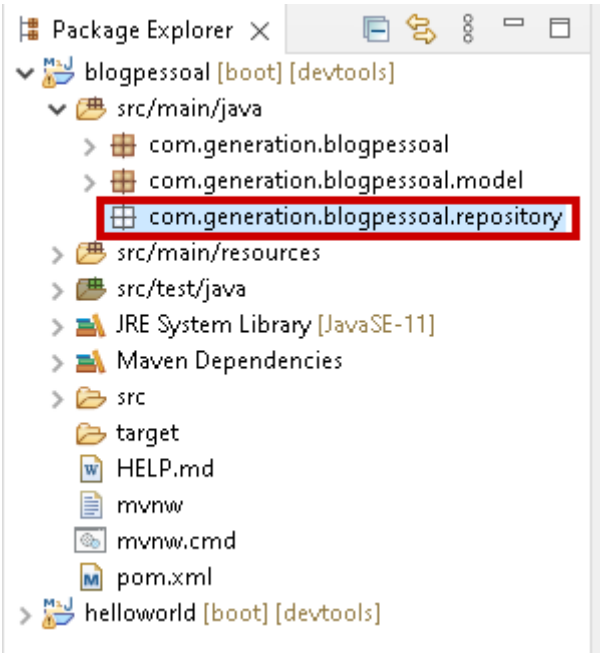


2. Na janela **New Java Package**, no item **Name**, acrescente no final do nome da Package **.repository**, como mostra a figura abaixo:



3. Clique no botão **Finish** para concluir.

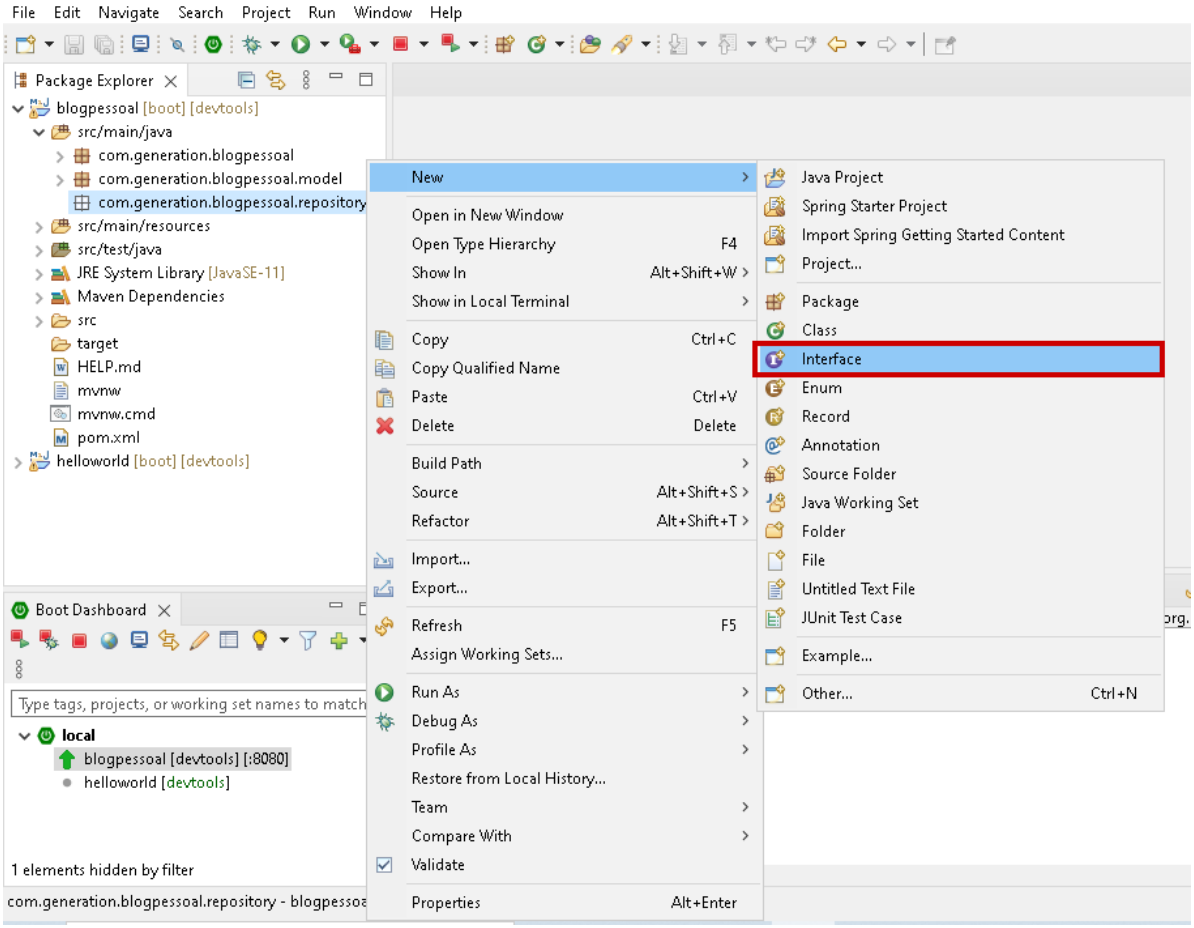
Quando você terminar de criar a **Camada Repository**, a sua estrutura de pacotes ficará igual a figura abaixo:



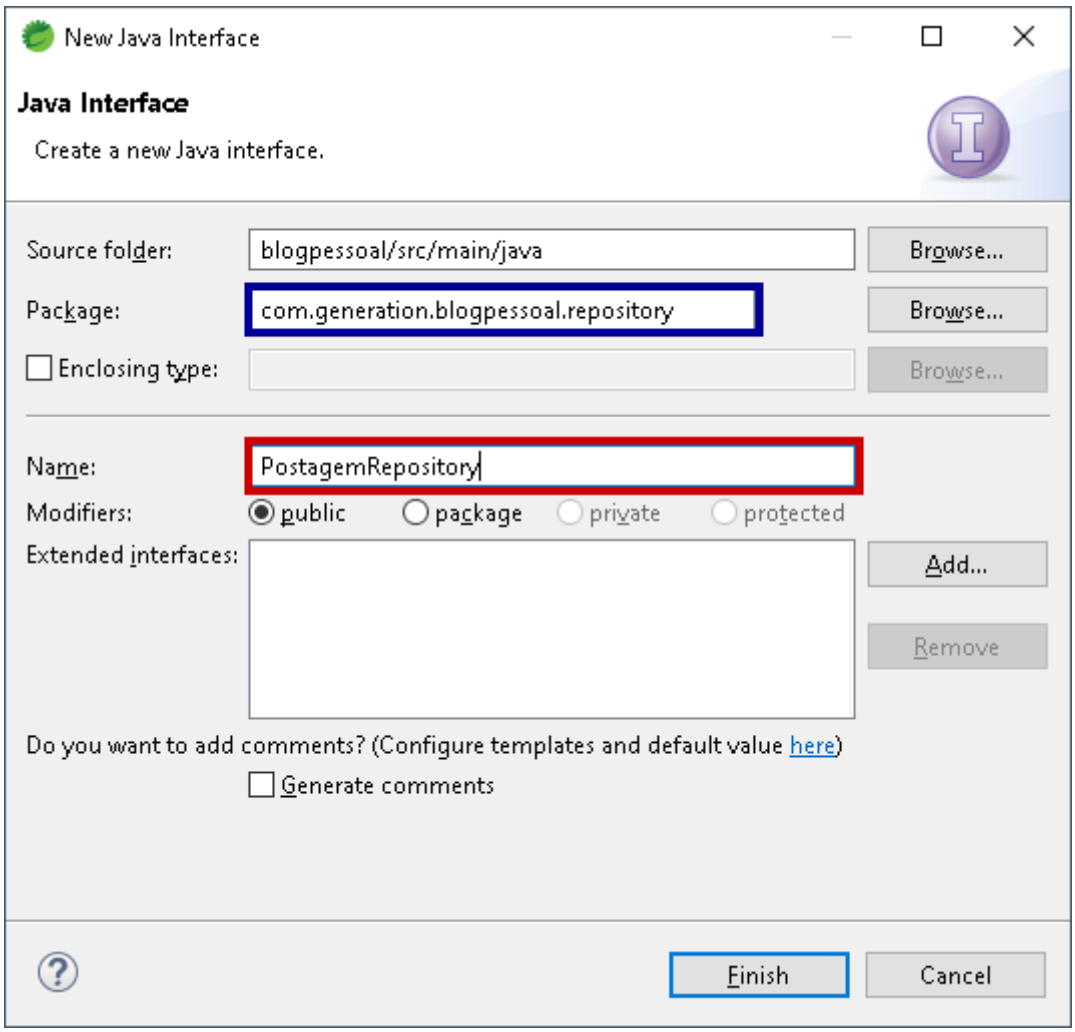
🦶 Passo 02 - Criar a Interface PostagemRepository na Camada Repository

Agora vamos criar a Interface Repository que chamaremos de **PostagemRepository**.

1. Clique com o botão direito do mouse sobre o **Pacote Repository** (**com.generation.blogpessoal.repository**), na Source Folder Principal (**src/main/java**), como mostra a figura abaixo:
2. Na sequência, clique na opção **New → Interface**



3. Na janela **New Java Interface**, no item **Name**, digite o nome da Interface (**PostagemRespository**), como mostra a figura abaixo:



4. Clique no botão **Finish** para concluir.

Agora vamos criar o código da **Interface Repository PostagemRepository**, como mostra a figura abaixo:

```
2
3import org.springframework.data.jpa.repository.JpaRepository;
4import org.springframework.stereotype.Repository;
5
6import com.generation.blogpessoal.model.Postagem;
7
8@Repository
9public interface PostagemRepository extends JpaRepository<Postagem, Long> {
10
11}
12
```

Vamos analisar o código:

Antes de continuar, vamos relembrar **o que é uma Interface?**

Uma **interface em Java** é uma **Classe Abstrata** (uma classe que serve de modelo para outras classes), composta somente por métodos abstratos. E como tal, obviamente não pode ser instanciada, ou seja, ela só contém as declarações dos métodos e constantes, nenhuma implementação, apenas as assinaturas dos métodos, que serão implementados em uma Classe.

Linha 08: A Anotação (Annotation) **@Repository** indica que a Interface é do tipo repositório, ou seja, ela é responsável pela interação com o Banco de dados através dos métodos padrão (**Herdados da Interface JPA Repository**) e das **Query Methods**, que são métodos personalizados que geram consultas (Instruções SQL do tipo Select), através da combinação de palavras chave, que representam os comandos da linguagem SQL.

Linha 09: Observe que na declaração da Interface foi adicionada a Herança através da palavra reservada **extends** com a Interface JpaRepository, que recebe 2 parâmetros:

1. A **Classe Postagem**, que é a Entidade que será mapeada em nosso Banco de dados (Lembre-se que a Classe Postagem foi quem gerou a nossa tabela tb_postagens)
2. O **Long** representa a nossa Chave Primária (Primary Key), que é o atributo que recebeu a anotação **@Id** na nossa Classe Postagem (o atributo também se chama id em nossa Classe Postagem).

Estes 2 parâmetros são do tipo **Java Generics** (podem receber qualquer tipo de Objeto <T, T>). Dentro contexto do JPA são o mínimo necessário para executar os Métodos padrão da Interface Repository, que serão implementados na próxima etapa na Classe **PostagemController**. Estes métodos básicos já ficam automaticamente disponíveis no Recurso Postagem a partir do momento que a Interface PostagemRepository herda a Interface JpaRepository.

Documentação: *@Repository*

Documentação: *Java Interface*

Documentação: *Classes Abstratas*

Documentação: *Java Generics*

Métodos Padrão da Interface JpaRepository

`save(Objeto objeto)`	Cria ou Atualiza um objeto no Banco de Dados.
`findById(Long id)`	Retorna (exibe) um Objeto persistido de acordo com o id informado.
`existsById(Long id)`	Retorna True se um Objeto identificado pelo id estiver persistido no Banco de dados.
`findAll()`	Retorna (exibe) todos os Objetos persistidos.
`deleteById(Long id)`	Localiza um Objeto persistido pelo id e deleta caso ele seja encontrado. Não é possível desfazer esta operação.
`deleteAll()`	Deleta todos os Objetos persistidos.Não é possível desfazer esta operação.

Documentação: Métodos Padrão

Código fonte do Projeto