

Projeto 02 - Blog Pessoal - Classe PostagemController - Método Consultar Postagens por Título

O que veremos por aqui:

1. Criar o Método `getByTitulo(String titulo)` para listar uma Postagem específica

1. O Recurso Postagem

Nas etapas anteriores, começamos a construir a Classe **PostagemController** e implementamos os Métodos:

- **getAll()** → Retorna todos os Objetos da Classe Postagem persistidos no Banco de dados.
- **getById(Long id)** → Retorna um Objeto específico da Classe Postagem persistidos no Banco de dados. A Postagem é identificada pelo Atributo id.

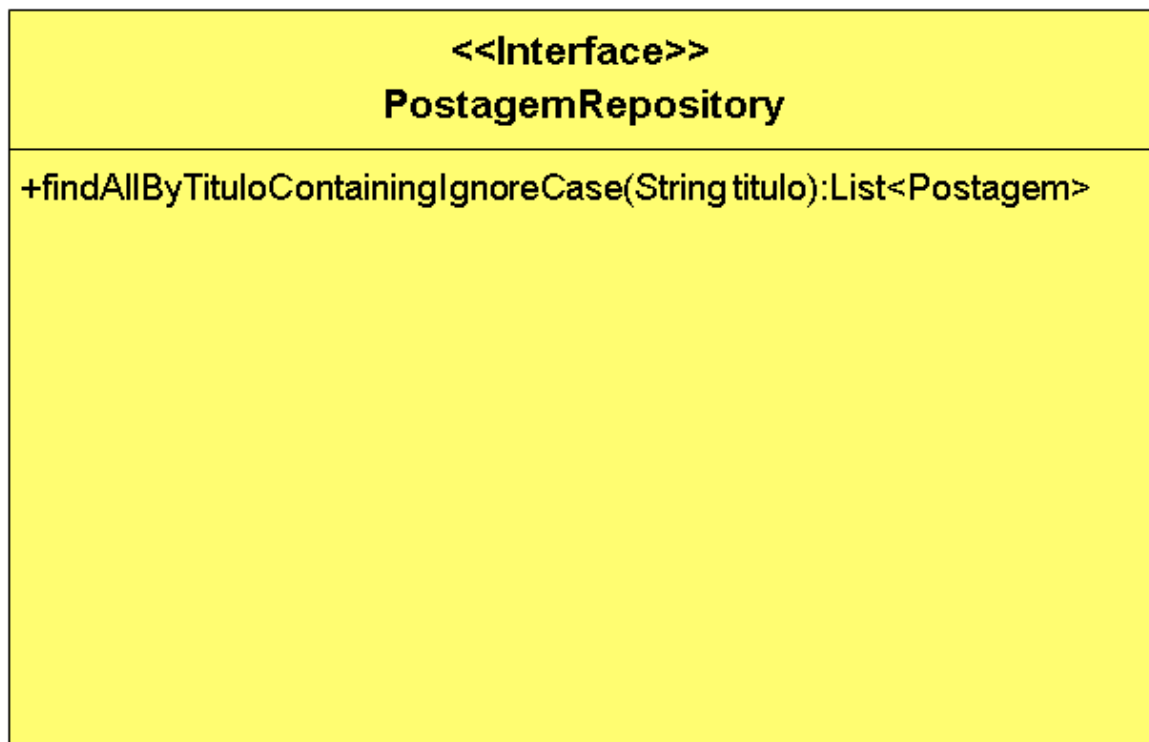
Vamos continuar a construção da nossa Classe Controladora implementando o **Método `getByTitulo(String titulo)`**, que retornará todos os Objetos da Classe Postagem persistidos no Banco de dados, cujo Atributo titulo contenha a String enviada no parâmetro titulo do Método.

Postagem
-id: Long -titulo: String -texto: String -data: LocalDateTime
+ getAll():ResponseEntity<List<Postagem>> + getById(Long id):ResponseEntity<Postagem> + getByTitulo(String nome):ResponseEntity<List<Postagem>> + post(Postagem postagem):ResponseEntity<Postagem> + put(Postagem postagem):ResponseEntity<Postagem> + delete(Long id):void



Passo 01 - Implementar a Query Method

Para implementarmos o Método de Consulta por título **`getByTitulo(String titulo)`** será necessário criar uma **Query Method** na Interface `PostagemRepository`. Desta forma, o Diagrama de Classes da nossa Interface sofrerá uma alteração:



Query Method são Métodos de Consulta personalizados, que permitem criar consultas específicas com qualquer Atributo da Classe associada a Interface Repositório (Postagem). Como a Interface JpaRepository possui apenas um Método de consulta específico pelo id (**findById(Long id)**), que é um Atributo comum em todas as Classes Model do Projeto, através das Query Methods podemos ampliar as nossas opções de consulta. As Query Methods são declaradas na Interface Repositório e implementadas nas Classes Controladoras e de Serviços (Service, que veremos mais a frente).

Na prática, o que as Query Methods fazem são criar instruções SQL através de Palavras Chave, que combinadas com os Atributos da Classe, geram consultas personalizadas.

Exemplo 01:

Query Method

```
public optional <Postagem> findByTitulo(String titulo);
```

Instrução SQL equivalente

```
SELECT * FROM tb_postagens WHERE titulo = "titulo";
```

Palavra Chave		Instrução SQL
find	→	SELECT
By	→	WHERE
Titulo	→	Atributo da Classe Postagem
String titulo	→	Parâmetro do Método contendo o título que você deseja procurar.

Como esta consulta retornará apenas um Objeto da Classe Postagem ou um Objeto Nulo, caso a consulta não encontre nada, o Método foi assinado com apenas um Objeto da Classe Postagem do tipo **Optional** para evitar o erro **NullPointerException** (Objeto Nulo).

Exemplo 02:

Query Method

```
public List <Postagem> findAllByTituloContainingIgnoreCase(@Param("titulo")
String titulo);
```

Instrução SQL equivalente

```
SELECT * FROM tb_postagens WHERE titulo LIKE "%titulo%";
```

Palavra Chave		Instrução SQL
find	→	SELECT
All	→	*
By	→	WHERE
Titulo	→	Atributo da Classe Postagem
Containing	→	LIKE "%titulo%"
IgnoreCase	→	Ignorando letras maiúsculas ou minúsculas
@Param("titulo")	→	Define a variável String titulo como um parâmetro da consulta. Esta anotação é obrigatório em consultas do tipo Like.
String titulo	→	Parâmetro do Método contendo o título que você deseja procurar.

Como esta consulta retornará um ou mais Objetos da Classe Postagem, o Método foi assinado com uma **Collection List** de Objetos da Classe Postagem.



ATENÇÃO: A instrução *FROM tb_postagens* será inserida pelo JPA ao checar o nome da tabela gerada pela Classe Postagem.



DICA: Acesse o [Guia do JPA](#) e explore outras opções de Query Methods (Métodos de Consulta), com exemplos implementados. Pedimos apenas que não implemente os exemplos no Projeto Blog Pessoal.

O Exemplo 02 será criado dentro da **Interface PostagemRepository** e será implementado na Classe PostagemController. Veja o código abaixo, implementado na Interface:

```

10
11 @Repository
12 public interface PostagemRepository extends JpaRepository<Postagem, Long> {
13
14     public List <Postagem> findAllByTituloContainingIgnoreCase(@Param("titulo") String titulo);
15
16 }
17

```

Observe que a consulta foi adicionada na linha 14 da Interface PostagemRepository.



[Documentação: Query Methods](#)



[Interface PostagemRepository](#)



Passo 02 - Criar o Método getByTitulo(String titulo)

Vamos implementar o Método **getByTitulo(String titulo)** na Classe Postagem Controller.

Traçando um paralelo com o MySQL, seria o equivalente a instrução: `SELECT * FROM tb_postagens where titulo like "%titulo%";`.

```

36
37 @GetMapping("/titulo/{titulo}")
38 public ResponseEntity<List<Postagem>> getByTitulo(@PathVariable String titulo){
39     return ResponseEntity.ok(postagemRepository.findAllByTituloContainingIgnoreCase(titulo));
40 }
41

```

Linha 37: a anotação **@GetMapping** indica que o Método `getAll()`, responderá a todas as requisições do tipo **HTTP GET**, enviadas no endereço <http://localhost:8080/postagens/titulo/postagem>.



ATENÇÃO: O Endereço deste Endpoint será composto pelo Endereço do Recurso (**@RequestMapping**) + a variável de caminho indicada na anotação **@GetMapping**. Lembre-se que não pode existir dois ou mais Métodos do tipo **GET** com o mesmo endereço.

Linha 38: O Método **getByTitulo(String titulo)** será do tipo **ResponseEntity** porque ele responderá a **Requisição HTTP** (HTTP Request), com uma **Resposta HTTP** (HTTP Response).

✓ **<List<Postagem>>**: O Método além de retornar um objeto da Classe **ResponseEntity** (OK → 200), no parâmetro body (Corpo da Resposta), será retornado um Objeto da Classe **List (Collection)**, contendo todos os Objetos da Classe **Postagem** persistidos no Banco de dados, na tabela **tb_postagens**, cujo Atributo titulo contenha a String enviada como parâmetro do Método.

Linha 39: return

ResponseEntity.ok(postagemRepository.findAllByTituloContainingIgnoreCase(String titulo)); Executa o Método **findAllByTituloContainingIgnoreCase(String titulo)** (Método personalizado, criado na Interface **PostagemRepository**), e exibe o resultado (**<List<Postagem>>**) no Corpo da Resposta. Como a List sempre será gerada (vazia ou não), o Método sempre retornará o **Status 200 → OK**.

Para concluir, não esqueça de Salvar o código (**File → Save All**) e verificar se o Projeto está em execução



[Documentação: @GetMapping](#)

[Documentação: ResponseEntity](#)

[Documentação: HttpStatus](#)



[Documentação: Collection List](#)



[Documentação: Java Generics](#)



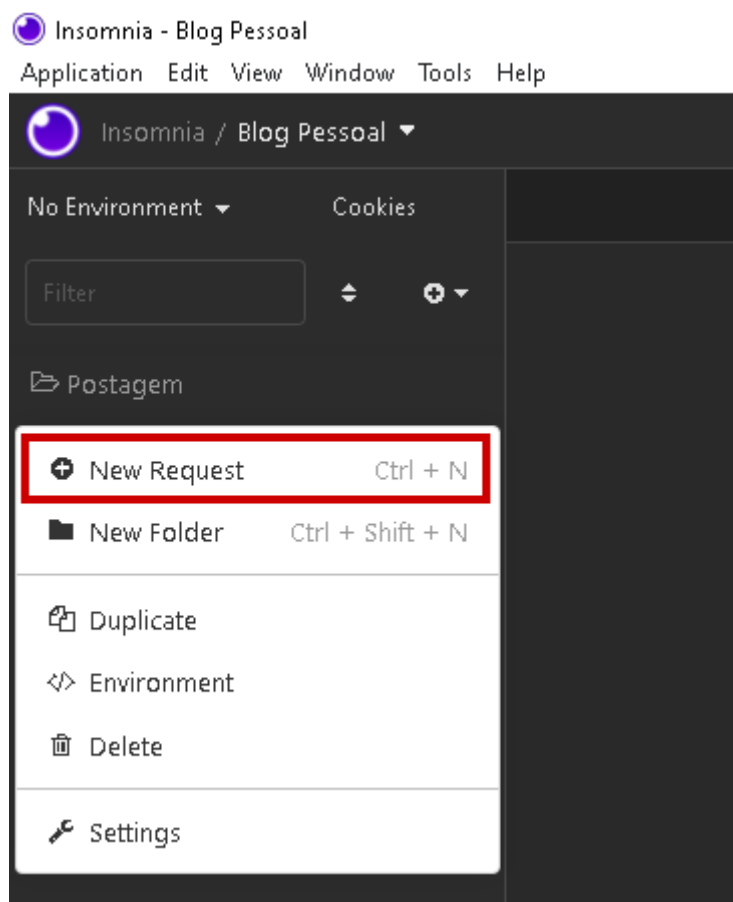
[Classe PostagemController](#)



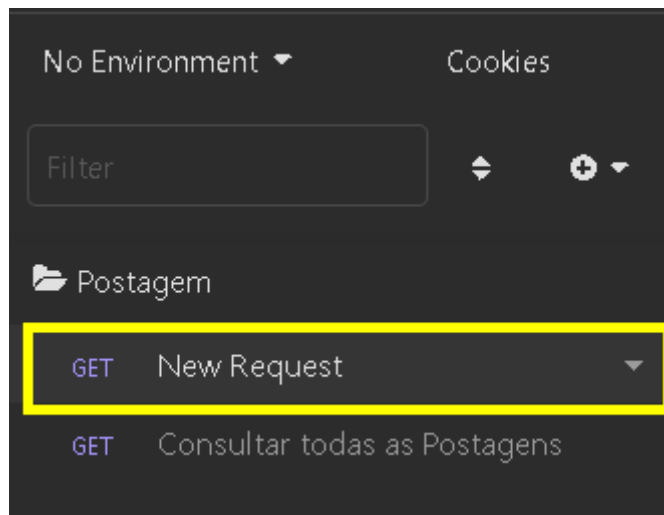
Passo 03 - Testar no Insomnia

Agora vamos criar a Requisição para o **Método getByTitulo(String titulo)**:

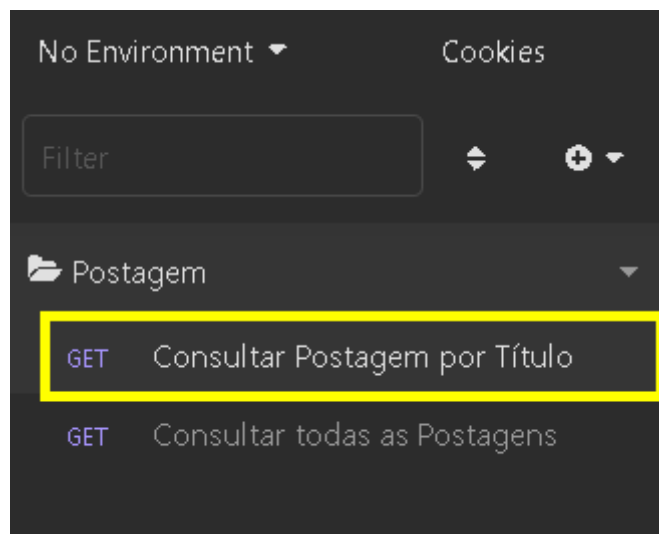
1. Clique com o botão direito do mouse sobre a **Pasta Postagem** para abrir o menu e clique na opção **New Request**.



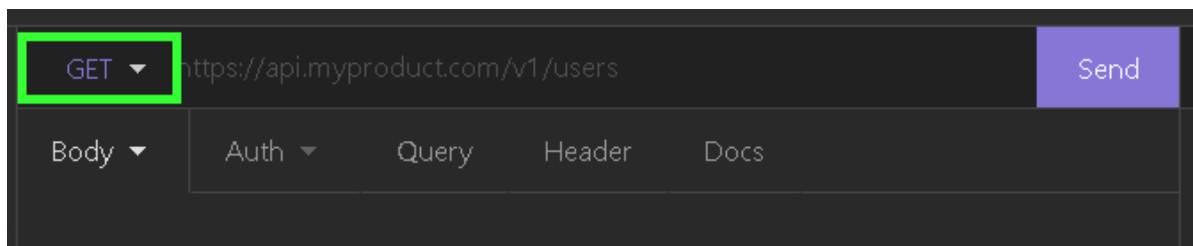
2. Será criada uma nova Requisição (New Request) dentro da pasta **Postagem**.



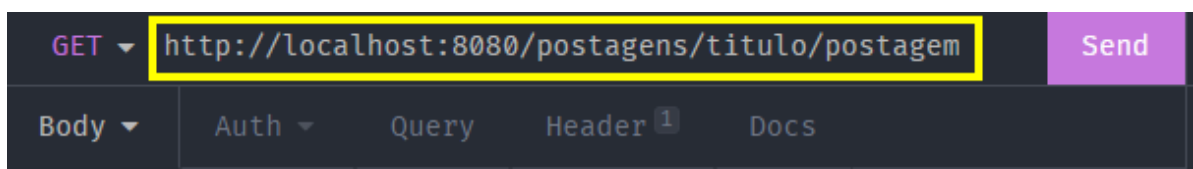
3. Dê um duplo clique sobre a nova requisição (**New Request**), informe o nome da requisição (indicado na imagem abaixo na cor amarela) e pressione a tecla **enter** do seu teclado.



4. Selecione o Método HTTP que será utilizado (**GET**) na requisição, indicado na imagem abaixo na cor verde.




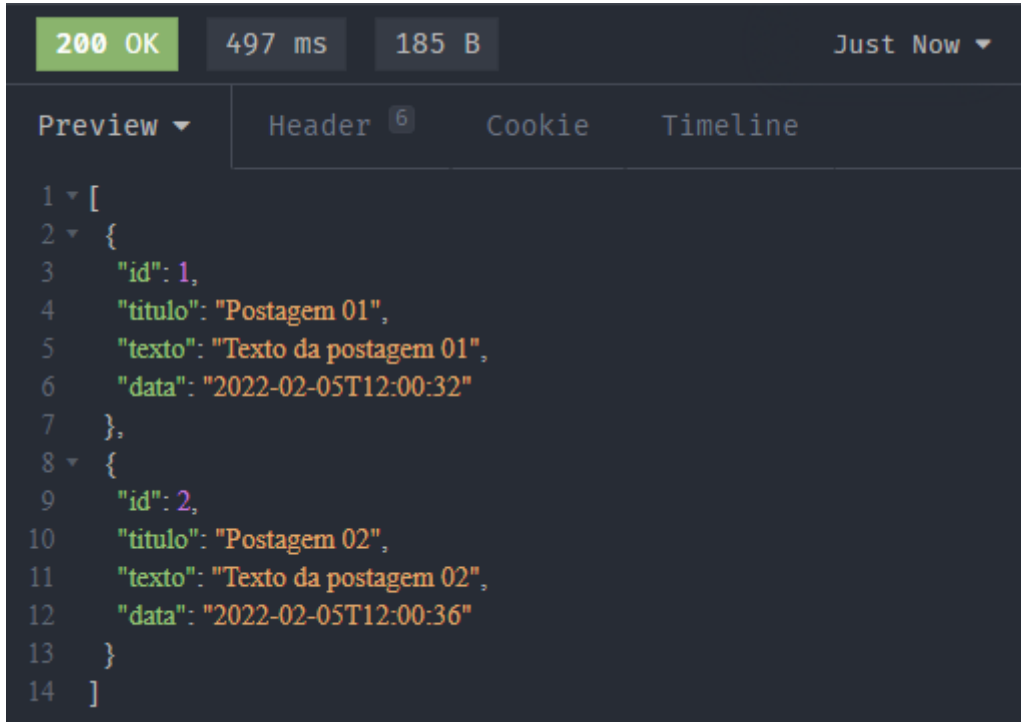
5. Configure a requisição conforme a imagem abaixo:



6. No item marcado em amarelo na imagem acima, informe o endereço (endpoint) da Requisição. A requisição **Consultar Postagem por Título** foi configurada da seguinte maneira:

- A primeira parte do endereço (<http://localhost:8080>) é o endereço do nosso servidor local. Quando a API estiver na nuvem, ele será substituído pelo endereço da aplicação na nuvem (<http://nomedaaplicacao.herokuapp.com>).

- A segunda parte do endereço é o **endpoint** configurado na anotação **@RequestMapping**, em nosso caso **/postagens**.
 - A terceira parte (**/titulo/postagem**), **titulo** é apenas um indicativo do conteúdo da variável que deverá ser preenchida. A palavra **postagem** é o conteúdo da variável de caminho (**@PathVariable**) **titulo**. Informe o texto que você deseja pesquisar.
7. Para testar a requisição, com a aplicação rodando, clique no botão .
8. O resultado da requisição você confere na imagem abaixo:



```
1 [
2   {
3     "id": 1,
4     "titulo": "Postagem 01",
5     "texto": "Texto da postagem 01",
6     "data": "2022-02-05T12:00:32"
7   },
8   {
9     "id": 2,
10    "titulo": "Postagem 02",
11    "texto": "Texto da postagem 02",
12    "data": "2022-02-05T12:00:36"
13  }
14 ]
```

9. Observe que a aplicação além de exibir os dados de todos os Objetos da Classe **Postagem** persistidos no Banco de dados, no Corpo da Resposta, respeitando o critério informado na consulta (palavra **postagem**), ela também retornará um **HTTP Status 200 → OK** (indicado em verde na imagem acima), informando que a Requisição foi bem sucedida!



[Código fonte do projeto](#)

 [Voltar](#)