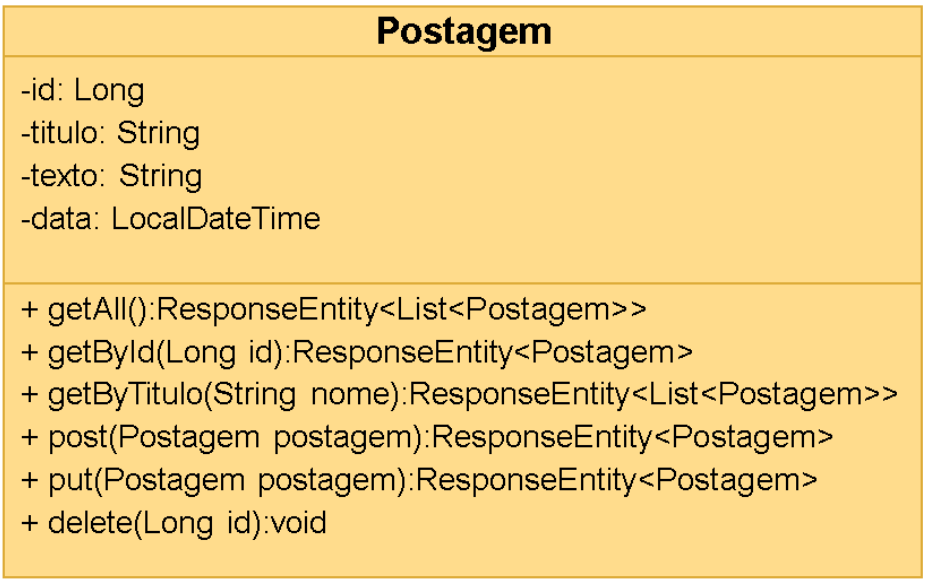
Projeto 02 - Blog Pessoal - CRUD 05

O que veremos por aqui:

1. Criar o método getById(Long id) para listar uma Postagem específica

1. O Recurso Postagem

Na etapa anterior, começamos a construir a Classe **PostagemController** e implementamos o Método **getAll()**, que retorna todos os Objetos da Classe Postagem persistidos no Banco de dados. Vamos continuar a construção da nossa Classe Controladora implementando o **Método getById(Long id)**.



🦶 Passo 01 - Criar o Método getById(Long id)

Vamos implementar o Método **getById(Long id)** na Classe Postagem Controller, que retornará um Objeto específico persistido no Banco de dados, identificado pelo **id** (Identificador único do Objeto). Traçando um paralelo com o MySQL, seria o equivalente a instrução: `SELECT * FROM tb_postagens where id = id;`.

```
29
30    @GetMapping("/{id}")
31    public ResponseEntity<Postagem> getById(@PathVariable Long id){
32        return postagemRepository.findById(id)
33                .map(resposta -> ResponseEntity.ok(resposta))
34                .orElse(ResponseEntity.status(HttpStatus.NOT_FOUND).build());
35    }
36
```

Para processar o Método findById(Long id), vamos utilizar dois recursos da Linguagem Java, que tornam o código mais limpo e assertivo. São os recursos **Optional e Expressões Lambda**.

Optional (map): É um contêiner para um valor que pode estar ausente. Em nosso contexto, tem como principal função evitar o erro do tipo **NullPointerException** (Objeto nulo), caso o Objeto Postagem procurado pelo método findById(id) não seja encontrado. O retorno do método NÃO pode ser nulo, por isso fazemos uso do Optional para que ele faça o encapsulamento e informe se o Objeto está presente ou não.

Expressões Lambda: Representam uma função anônima, ou seja, uma função lambda é uma função sem declaração, ou seja, não é necessário colocar um nome, um tipo de retorno e modificador de acesso. A ideia é que o método seja declarado no mesmo lugar em que será usado e a sua declaração seja implícita. As expressões lambda em Java tem a sintaxe definida como: (argumento) → (corpo). A Linguagem **Javascript/Typescript** possuem um recurso semelhante, que são as **Arrow Functions**.

Linha 30: A anotação **@GetMapping("/{id}")** mapeia todas as Requisições **HTTP GET**, enviadas para um endereço específico (**Endpoint**), dentro do Recurso Postagem, para um Método específico que responderá as requisições, ou seja, ele indica que o Método **getById(Long id)**, responderá a todas as requisições do tipo **HTTP GET**, enviadas no endereço **http://localhost:8080/postagens/id**, onde **id** é uma **Variável de Caminho** (Path Variable), que receberá o id da Postagem que será Consultada.

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 31: O Método **getById(@PathVariable Long id)** será do tipo **ResponseEntity** porque ele responderá Requisições HTTP (HTTP Request), com uma **Resposta HTTP** (HTTP Response). Observe que o Método possui um parâmetro do tipo **Long**, chamado **id**.

@PathVariable Long id: Esta anotação insere o valor enviado no endereço do endpoint, na Variável de Caminho **{id}**, no parâmetro do Método **getById(Long id)**;

Exemplo:

http://localhost:8080/postagens/1

Neste exemplo, o parâmetro **Long id**, do Método getById(Long id), receberá o valor 1 (Id que será procurado em tb_postagens)

ATENÇÃO: ***Por questões de boas práticas e legibilidade do código, a Variável de Caminho e o Parâmetro do Método getById devem possuir o mesmo nome.***

<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 Postagem**, apenas e somente se o Objeto procurado for encontrado no Banco de dados, na tabela **tb_postagens**. Observe que nesta linha também foi utilizado o recurso **Java Generics** para simplificar o retorno do Objeto.

Linha 32:

return postagemRepository.findById(Long id): Retorna a execução do método **findById(id)**, que é um **Método padrão da Interface JpaRepository**. O Método retornará **um Objeto da Classe Postagem** persistido no Banco de dados (<Postagem>), caso ele seja encontrado a partir do parâmetro Long id. Caso contrário, será retornado um Objeto Nulo.

.map(resposta → ResponseEntity.ok(resposta)): Se o Objeto da Classe Postagem for econtrado, o método **map** (Optional), mapeia no **Objeto resp** o Objeto Postagem retornado pelo método findById(id)), insere o Objeto mapeado no Corpo da Resposta do Método **ResponseEntity.ok(resp);** e retorna o HTTP Status **OK→200**.

.orElse(ResponseEntity.notFound().build());: Se o Objeto Postagem não for encontrado (Nulo), será retornado o HTTP Status **NOT FOUND → 404** (Não Encontrado!). O método **build()** constrói a Resposta com o HTTP Status retornado.


Documentação: *@GetMapping*


Documentação: *@PathVariable*


Documentação: *ResponseEntity*


Documentação: *HttpStatus*


Documentação: *.findById(Long id)*


Documentação: *Optional*


Documentação: *Expressões Lambda*


Documentação: *Long*


Artigo: *Classes Wrappers (Long)*

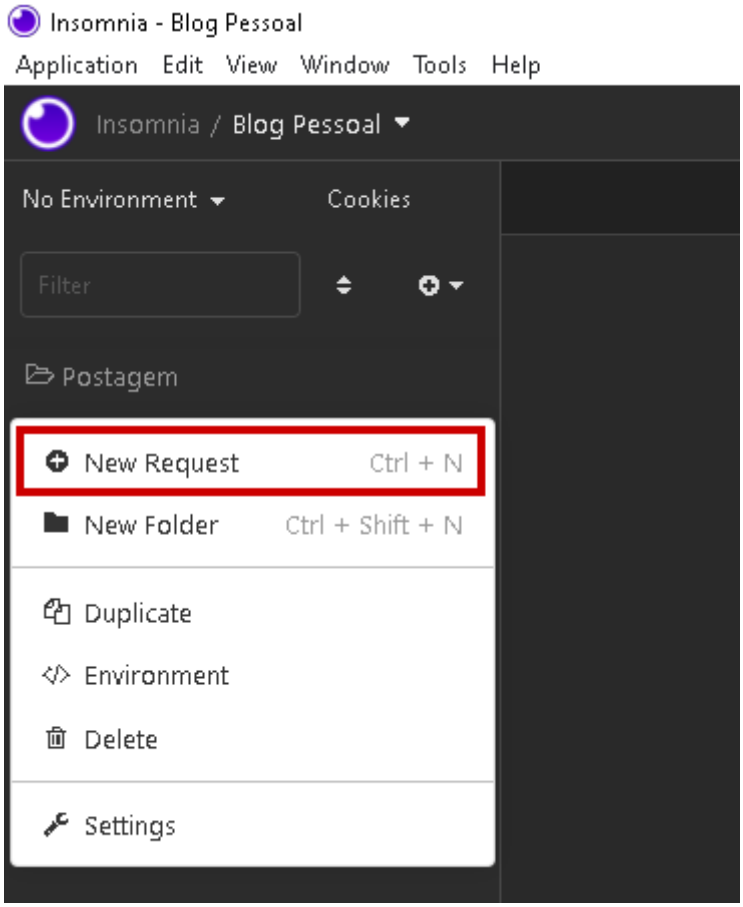
Para concluir, não esqueça de Salvar o código (**File → Save All**) e verificar se o Projeto está em execução


Código fonte do projeto

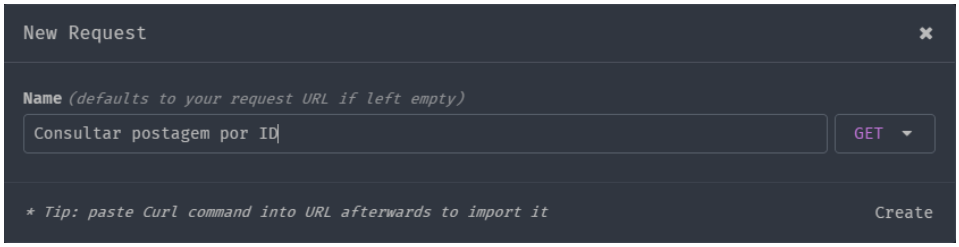
 Passo 02 - Testar no Insomnia

Agora vamos criar a Requisição para o **Método getById()**:

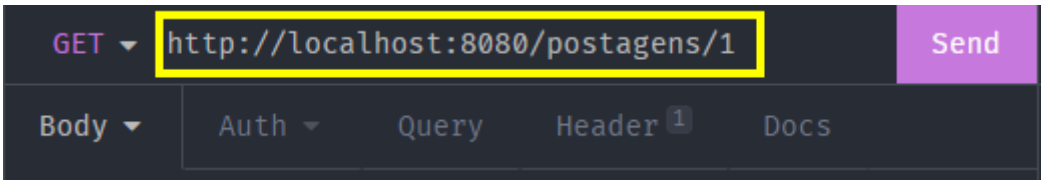
1. Clique com o botão direito do mouse sobre a **Pasta Postagem** para abrir o menu e clique na opção **New Request**.



2. Na janela que será aberta, informe o nome da requisição e o Método HTTP que será utilizado (**GET**). Clique no botão **Create** para concluir.

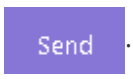


3. Configure a requisição conforme a imagem abaixo:



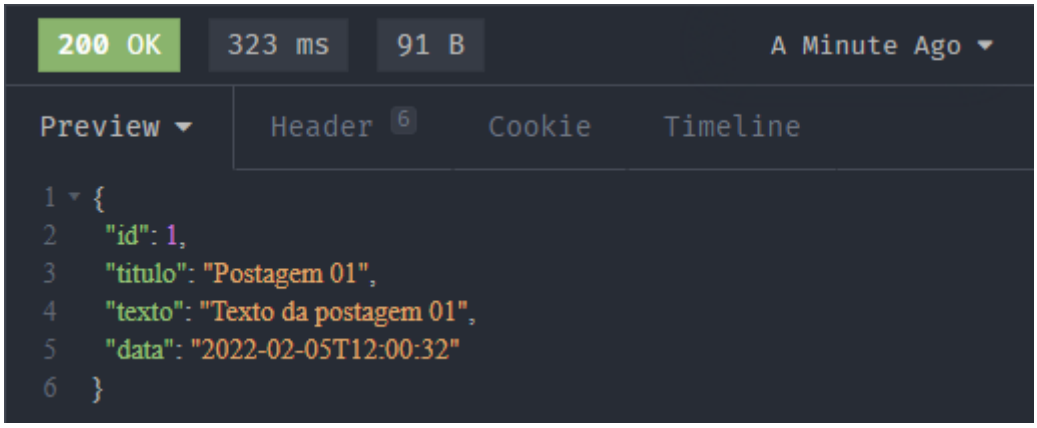
4. No item marcado em amarelo na imagem acima, informe o endereço (endpoint) da Requisição. A requisição **Consultar postagem por ID** foi configurada da seguinte maneira:

- 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 (**/1**) é a variável de caminho (**@PathVariable**) **id**. Informe o id que você deseja procurar.

5. Para testar a requisição, com a aplicação rodando, clique no botão 

.

6. O resultado da requisição você confere na imagem abaixo:



7. Observe que a aplicação quando encontra o Objeto no Banco de dados, além de exibir os dados do Objeto no Corpo da Resposta, respeitando o critério informado na consulta (id 1), ela também retorna um **HTTP Status 200 → OK** (indicado em verde na imagem acima), informando que a Requisição foi bem sucedida!
8. Caso o Objeto Postagem não seja encontrado, a aplicação retornará o **HTTP Status 404 → NOT FOUND** (Não encontrado), marcado em laranja na imagem abaixo: