

Tecnologias Para Back-End

Prof. JUNIO FIGUEIRÊDO

JUNIOINF@GMAIL.COM

AULA 10 – REQUISIÇÕES DELETE...

Requisições Delete



Spring

- Vamos criar um o novo método, **para exclusão**, em "**MedicoController.java**".

- O método se chamará **excluir**

```
public void excluir() {  
  
}
```

- Acima dele, a anotação será **@DeleteMapping**. Que irá receber um **ID**
- Vamos abrir parênteses e aspas após e anotação e passar o complemento da URL.
- Para que seja um parâmetro dinâmico, passaremos ("/{id}").

```
@DeleteMapping("/{id}")
```

Spring

- Como iremos fazer para capturar esse ID, que é dinâmico que está chegando pela URL ???
- Basta recebe-lo como parâmetro no método `excluir`
- Se passamos apenas o `Long id`, dessa forma o **Spring não assume** que o `Long id` e `{id}` **são a mesma coisa**, ele não irá associar.
- Para solucionar vamos adicionar no parâmetro do método a seguinte anotação `@PathVariable`

```
@DeleteMapping("/{id}")  
public void excluir(@PathVariable Long id){  
}
```

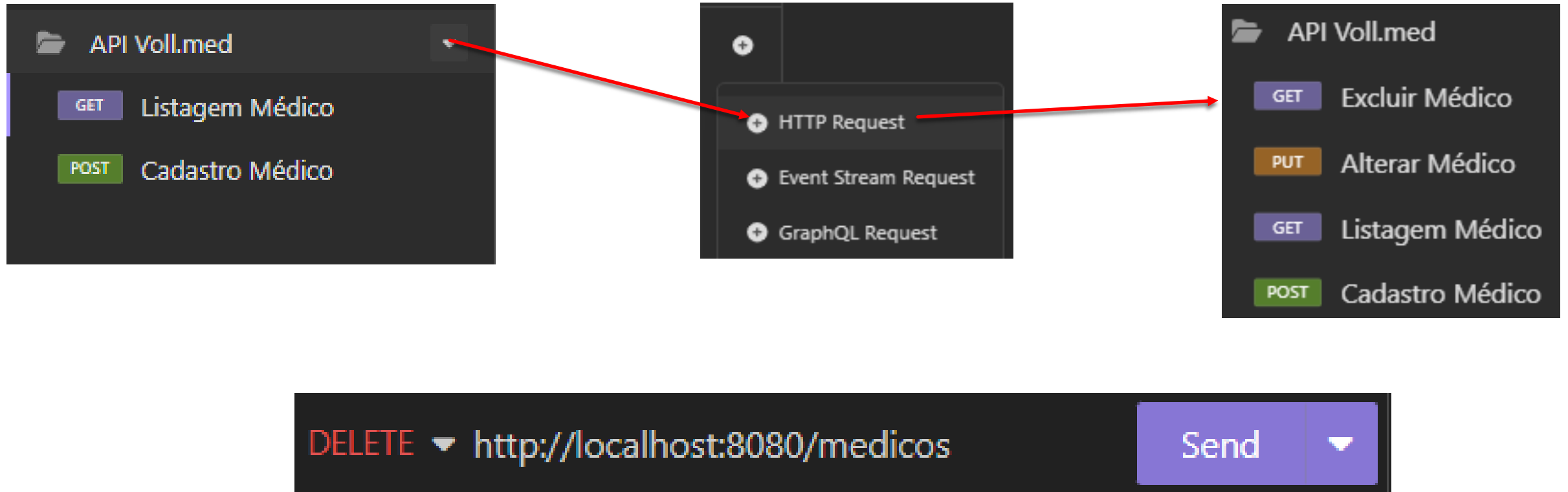
Spring

- Também adicionaremos o método `@Transactional` abaixo de `@DeleteMapping`
- Com a ajuda de `repository`, também passaremos `.deleteById(id)` para fazer o delete no banco de dados.

```
@DeleteMapping("/{id}")
@Transactional
public void excluir(@PathVariable Long id){
    repository.deleteById(id);
}
```

Spring

- Agora basta salvar.
- Vamos ao Insomnia, Vamos criar uma nova requisição.



Spring

- Ainda no Insomnia, selecione a requisição Listagem Médico.

GET ▼ http://localhost:8080/medicos

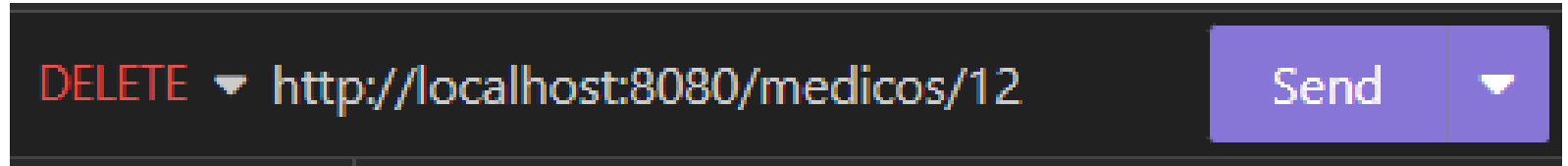
Send ▼

- Realize a requisição
- Por exemplo, vamos excluir o id = 12.
- Vá na requisição de Excluir

```
{
  "id": 14,
  "nome": "Ana Julia Maria Luiza Carlo",
  "email": "jsanamafoaeeee.ferreira@voll.med",
  "crm": "571034",
  "especialidade": "ORTOPEDIA"
},
{
  "id": 12,
  "nome": "Ana Maria Luiza Carlo",
  "email": "anamafoaeeee.ferreira@voll.med",
  "crm": "471034",
  "especialidade": "ORTOPEDIA"
},
{
  "id": 8,
  "nome": "Antonio Carlos",
  "email": "joaeeee.ferreira@voll.med",
  "crm": "871034",
  "especialidade": "ORTOPEDIA"
},
}
```

Spring

- Na requisição Excluir.
- Clique em **Send**.
- Veja que a resposta foi **200** **200 OK**
- Para confirmar a exclusão basta, verificar na **requisição Listagem Médico**
- Fizemos a exclusão tradicional.
- Depois vamos fazer a exclusão lógica.



Exclusão Lógica



Spring

- Vamos fazer uma **exclusão lógica**.
- Em outras palavras, **não vamos apagar** o médico do banco de dados, mas **marcá-lo como inativo**.
- Vamos acessar "**src > main > resources > db.migration**".
- Nessa pasta, **vamos criar a migration** "**VXX__alter-table-medicos-add-column-ativo.sql**".
- Agora vamos abrir a migration.

```
alter table medicos add ativo tinyint;  
update medicos set ativo =1;
```

```
alter table medicos add ativo tinyint;  
update medicos set ativo =1;
```

Spring

- Essa nova migration será responsável por alterar a tabela de médicos, adicionando a coluna chamada "ativo", do tipo tinyint.
- É hora de atualizar a entidade JPA.
- Vamos declarar o atributo `private Boolean ativo`.
- Na entidade medico.

```
@Embedded
private Endereco endereco;

private Boolean ativo;
```

Spring

- Atualizaremos, também, o construtor que recebe DadosCadastroMedico.
- Isso na Entidade medico.
- Nele, adicionaremos `this.ativo = true;`

```
public Medico(DadosCadastroMedico dados) {  
    this.ativo = true;  
    this.nome = dados.nome();  
    this.email = dados.email();  
}
```

Spring

- Vamos voltar à funcionalidade de excluir de "MedicoController.java".
- Nela, vamos carregar a entidade, inativá-lo, configurar o atributo como "false" e disparar o update no banco de dados.
- Vamos substituir `repository.deleteById(id);`
- Por var medico : `Medico = repository.getReferenceById(id);`.

```
public void excluir(@PathVariable Long id){  
    repository.deleteById(id);  
    var medico = repository.getReferenceById(dados.id());  
}
```

Spring

- Ficará assim

```
public void excluir(@PathVariable Long id){  
    var medico = repository.getReferenceById(dados.id());  
}
```

- Remover `dados.id()` do parâmetro, ficando apenas `id`. `getReferenceById(id);`
- Ainda no método `excluir`, vamos adicionar o seguinte método.


```
public void excluir(@PathVariable Long id){  
    var medico = repository.getReferenceById(id);  
    medico.excluir();  
}
```

- Vamos `criar o método excluir` na entidade `medico`.

Spring

- Criando o método `excluir` na entidade `medico`. **Alt + Enter**

```
medico.excluir();
```

 Create method 'excluir' in 'Medico'

Press Ctrl+Q to toggle preview

```
// Medico.java  
public void excluir() {  
  
}
```

- No método `excluir` fica assim:

```
public void excluir() {  
    this.ativo = false;  
}
```



```
public Medico(DadosCadastroMedico dados) {  
    this.ativo = true;  
    this.nome = dados.nome();  
    this.email = dados.email();  
    this.telefone = dados.telefone();  
    this.crm = dados.crm();  
    this.especialidade = dados.especialidade();  
    this.endereco = new Endereco(dados.endereco());  
}
```

Spring

- Retorne ao Insomnia e tente excluir algum médico pelo id.
- Veja no banco se o médico está com o `ativo = false`.
- Depois veja como está a requisição do Listar.
- A listagem, porém, continua apresentando médicos `ativos` e `inativos`. Vamos `alterar isso`, para exibir `apenas os ativos`. Vamos voltar ao `método lista`, em `"MedicoController.java"`, para `atualizar a listagem`.
- Nesse método, vamos substituir `findAll` por `findAllByAtivoTrue`:

Spring

```
return repository.findByAtivoTrue()
```

- Como esse comando não existe no repositório.
- Temos condições de criar devido ao padrão de nomenclatura do Spring Data.
- Vamos “Alter + Enter”... Para criar esse método (query/comando).


```
True(paginacao).map(DadosListagem::toEntity)
```

⚠ Create method 'findByAtivoTrue' in 'MedicoRepository'

⚠ Rename reference

Spring

```
public interface MedicoRepository extends JpaRepository<Medico, Long> {  
    1 usage  
    ObservationFilter findByAtivoTrue(Pageable paginacao);  
}
```



- Vamos arrumar **esse retorno**, tem que ser um **Page<Medico>**
- Ficando assim.

```
public interface MedicoRepository extends JpaRepository<Medico, Long> {  
    1 usage  
    Page<Medico> findByAtivoTrue(Pageable paginacao);  
}
```

Spring

- O Spring irá montar uma consulta com um atributo **Ativo = true**
- **Select * from medico where ativo = true**
- Faça um requisição de Listagem, veja que o médico que foi excluído não aparece na listagem!!!!
- Finalizamos o CRUD médico...
- Faça o mesmo com Paciente

Spring

