Tecnologias Para Back-End

Prof. JUNIO FIGUEIRÊDO

JUNIOINF@GMAIL.COM

AULA 13 – DETALHANDO DADOS DA API

Detalhando Dados na API

Spring

- Padronizamos os retornos para ResponseEntity no controller, e para cada operação do CRUD devolver o código HTTP adequado.
- **Cadastro** devolve 201 Created (Cabeçalho , Corpo)
- **Excluir** devolve 204 No Content.(Sem Corpo).
- **Alteração** devolve 200 OK (Com Corpo)
- **Exibir/Listagem** devolve 200 OK (Com Corpo)

Spring

- Header do método Cadastro de médico:

NAME	VALUE
Location	http://localhost:8080/medicos/30
Content-Type	application/json
Transfer-Encoding	chunked
Date	Mon, 13 May 2024 04:00:51 GMT

- Temos o CRUD e ficamos com a funcionalidade de detalhes de um médico pendente.

- O cabeçalho location, ao cadastrarmos um médico, ele nos devolve o endereço para acessarmos o recurso criado na API, sendo o http://localhost:8080/medicos/30.

Spring

- Vamos implementar uma nova requisição no Insomnia chamada de DetalharMedico, Sendo GET, como estamos trazendo registro da API, é uma requisição do tipo GET.
- Retorne para o MedicoController, localize o método excluir

```
@DeleteMapping("/{id}")
@Transactional
public ResponseEntity excluir(@PathVariable Long id){
    var medico = repository.getReferenceById(id);
    medico.excluir();
    return ResponseEntity.noContent().build();
}
```

- Copie, logo após, abaixo d'ele cole.

Spring

- Teremos dois métodos excluir!!!!!
- Vamos arrumar o segundo.
- Ao invés de **@DeleteMapping** usaremos **@GetMapping** com o complemento /{id}.
- Vamos remover o @Transactional, porque é um método de leitura, e alteramos o nome do método para detalhar.

```
@GetMapping("/{id}")
public ResponseEntity detalhar(@PathVariable Long id){
    var medico = repository.getReferenceById(id);
    medico.excluir();
    return ResponseEntity.noContent().build();
}
```

- Na implementação, **precisamos carregar o médico do banco de dados** e, por isso, a linha com a variável **medico permanece** e **a linha medico.excluir() podemos remover**.

```
@GetMapping("/{id}")
public ResponseEntity detalhar(@PathVariable Long id){
    var medico = repository.getReferenceById(id);
    return ResponseEntity.noContent().build();
}
```

Spring

- Por fim, no return precisamos devolver um conteúdo.
- Em razão disso, alteramos de .noContent() para .ok() e removeremos o .build().

```
@GetMapping("/{id}")
public ResponseEntity detalhar(@PathVariable Long id){
    var medico = repository.getReferenceById(id);
    return ResponseEntity.ok();
}
```

- Como parâmetro do método ok(), criamos um DTO, sendo o mesmo de detalhamento: new DadosDetalhamentoMedico().

- Em DadosDetalhamentoMedico() vamos passar como parâmetro o médico que acabamos de carregar no banco de dados.

```
@GetMapping("/{id}")
public ResponseEntity detalhar(@PathVariable Long id){
    var medico = repository.getReferenceById(id);
    return ResponseEntity.ok(new DadosDetalhamentoMedico(medico));
}
```

- Já tínhamos tudo implementado, até o DTO DadosDetalhamentoMedico. Logo, tivemos somente que usar o mesmo DTO utilizado no cadastro e na atualização.

- Note que, eventualmente, podemos reaproveitar um DTO, podendo ser usado em diversos métodos no controller.
- Salve, levante o servidor!!!!
- Retorne ao Insomnia, dispare a requisição para detalhar o médico.
- O código devolvido foi o 200 OK, com os campos no corpo da resposta.
- A funcionalidade de detalhar está funcionando!
- Podemos aplicar a mesma lógica para a funcionalidade de pacientes. Inclusive, até em nosso projeto.

Spring

- Voltaremos ao IntelliJ, no arquivo do controller.
- Conseguimos fazer as alterações necessárias para a melhoria do nosso código, todos os métodos estão padronizados retornando o ResponseEntity.
- Além disso, em cada método devolvemos o código HTTP mais adequado para aquela determinada situação.
- Por exemplo, o código 201 **para o cadastro**, o código 200 para o **método de listagem**, **atualizar** e **detalhar** e 204 para o **método de excluir**.
- Em cada um desses códigos, podemos ter um corpo da resposta e cabeçalhos do protocolo HTTP.

Spring