Aula 2: Boas Práticas em APIs Rest com Spring Boot 3

September 18, 2025

1 Introdução

Nesta aula, ajustaremos o projeto da API Rest desenvolvida no curso anterior, focando em boas práticas relacionadas ao protocolo HTTP. Usaremos o `Visual Studio Code` ou sua IDE preferida para implementar essas melhorias. As requisições CRUD (Create, Read, Update, Delete) foram testadas no Insomnia, simulando a comunicação entre o aplicativo mobile e a API back-end. Nosso objetivo é padronizar os retornos HTTP usando a classe `ResponseEntity` do Spring Boot.

2 Boas Práticas no Protocolo HTTP

Uma API Rest deve seguir as convenções do protocolo HTTP, incluindo o uso correto de verbos e códigos de status. No curso anterior, implementamos as operações CRUD com os seguintes verbos:

- **Cadastrar**: POST
- **Listar**: GET
- **Atualizar**: PUT
- **Excluir**: DELETE

Porém, os retornos dos métodos no controlador (`MedicoController`) utilizavam `void`, resultando no código HTTP 200 (OK) por padrão, mesmo em cenários onde outros códigos seriam mais apropriados. Vamos ajustar isso.

3 Ajustando o Método de Exclusão

No método de exclusão, originalmente usávamos `void`, o que retornava 200 OK sem corpo de resposta. O código HTTP 204 (No Content) é mais adequado, pois indica sucesso sem retorno de conteúdo.

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

Após essa alteração, ao disparar a requisição `DELETE http://localhost:8080/medicos/2` no Insomnia, o retorno será 204 No Content, conforme esperado.

4 Ajustando o Método de Listagem

O método de listagem já retornava dados, mas não usava `ResponseEntity`. Ajustaremos para explicitar o código 200 OK e manter a paginação.

```
@GetMapping
public ResponseEntity<Page<DadosListagemMedico>> listar(@PageableDefault(size =
    10, sort = {"nome"}) Pageable paginacao) {
    var page =
        repository.findAllByAtivoTrue(paginacao).map(DadosListagemMedico::new);
    return ResponseEntity.ok(page);
}
```

Aqui, usamos `ResponseEntity.ok(page)` para retornar o código 200 com os dados paginados.

5 Ajustando o Método de Atualização

No método de atualização, originalmente `void`, retornaremos os dados atualizados do médico usando um DTO (`DadosDetalhamentoMedico`) e o código 200 OK.

```
@PutMapping
@Transactional
public ResponseEntity atualizar(@RequestBody @Valid DadosAtualizacaoMedico dados) {
    var medico = repository.getReferenceById(dados.id());
    medico.atualizarInformacoes(dados);
    return ResponseEntity.ok(new DadosDetalhamentoMedico(medico));
}
```

5.1 Criando o DTO DadosDetalhamentoMedico

Para evitar expor entidades JPA diretamente, criamos o DTO `DadosDetalhamentoMedico` no pacote `med.voll.api.medico`:

```
package med.voll.api.medico;

import med.voll.api.endereco.Endereco;

public record DadosDetalhamentoMedico(Long id, String nome, String email, String
    crm, String telefone, Especialidade especialidade, Endereco endereco) {
    public DadosDetalhamentoMedico(Medico medico) {
        this(medico.getId(), medico.getNome(), medico.getEmail(), medico.getCrm(),
            medico.getTelefone(), medico.getEspecialidade(), medico.getEndereco());
    }
}
```

Este DTO encapsula todas as informações do médico e é usado no retorno do método de atualização.

6 Próximos Passos

Na próxima aula, ajustaremos o método de cadastro para usar o código HTTP 201 (Created) e implementaremos um endpoint para detalhamento de médicos. Continue praticando no `Visual Studio Code` ou sua IDE preferida!

7 Dica do Professor

- **Aprofunde-se em códigos HTTP**: Consulte a documentação da Mozilla sobre códigos de status HTTP (`https://developer.mozilla.org/en-US/docs/Web/HTTP/Status`) para entender os cenários de uso de 200, 201, 204, etc.
- **Comunidade no X**: Siga perfis como @SpringGuru e @JavaPro no X para dicas sobre Spring Boot e boas práticas em APIs Rest.
- **Pratique**: Crie um endpoint adicional em seu projeto para testar diferentes códigos HTTP, como 201 para criação e 404 para recursos não encontrados.