



Bônus: Aprenda Fundamentos

(Spring e desenvolvimento de sistemas)

Capítulo: API REST, camadas, CRUD, exceções, validações

<https://devsuperior.com.br>

1

DISCLAIMER

Este é um **BÔNUS** concedido aos alunos do Bootcamp Spring, sem custo adicional.

O objetivo deste bônus é ensinar os seguintes fundamentos:

- Injeção de dependência
- DTO, padrão camadas
- JPA: entidades associadas, lazy loading

Vale ressaltar também que o objetivo deste bônus **NÃO É** implementar um projeto completo, mas sim abordar os fundamentos citados acima.

Bons estudos!

2

2

API REST

Conceitos importantes

3

- **API - Application Programming Interface:** é o conjunto de funcionalidades que são expostas por uma aplicação/módulo.
 - Outra aplicação/módulo pode acessar essa API.
- É um contrato entre um provedor e um consumidor de funcionalidades.

4

4

- API Web: é uma API que está disponibilizada via web. As funcionalidades são acessadas por meio de endpoints web (host, porta, rota, parâmetros, corpo (payload), cabeçalhos) usando protocolo HTTP.
- API REST: é uma API Web que está em conformidade com as restrições do padrão REST.

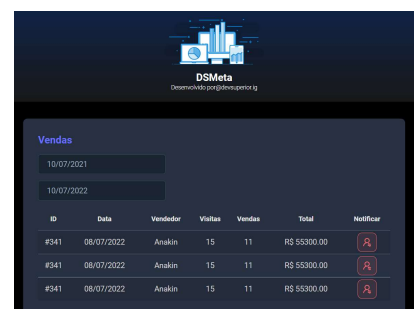
5

5

Back end + banco de dados (servidor)



Front end (navegador) (cliente)



← requisições web (HTTP/JSON) →

Back end: é "todo" sistema que roda do lado do servidor

API: é o conjunto de funcionalidades que são expostas pelo back end

6

Padrão REST

- Cliente/servidor com HTTP
- Comunicação stateless
- Cache
- Interface uniforme, formato padronizado
- Sistema em camadas
- Código sob demanda (opcional)

<https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>

7

7

Recursos e URLs

As funcionalidades/informações de um sistema web são organizadas na forma de **RECURSOS**

URL - Universal Resource Locator

A URL deve acessar os recursos pelo nome:

GET: host:port/products	(obter produtos)
GET: host:port/products?page=3	(obter produtos da página 3)
GET: host:port/products/1	(obter produto id 1)
GET: host:port/products/1/categories	(obter categorias do produto id 1)

8

8

Padrões de URL

A ação desejada deve ser expressa pelo verbo HTTP e não pela rota

ERRADO:

GET: host:port/insertProduct

GET: host:port/listProduct

CORRETO:

POST: host:port/products

GET: host:port/products

9

9

Verbos (métodos) HTTP mais utilizados

GET - obter recurso

POST - criar novo recurso

PUT - salvar recurso de forma idempotente

DELETE - deletar recurso

Operação idempotente = não causa novos efeitos se executada mais de uma vez

<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>

10

10

Códigos de resposta HTTP

- Respostas de informação (100-199)
- Respostas de sucesso (200-299)
- Redirecionamentos (300-399)
- Erros do cliente (400-499)
- Erros do servidor (500-599)

<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status>

11

11

Padrão camadas

Organizando a aplicação em camadas com responsabilidades definidas

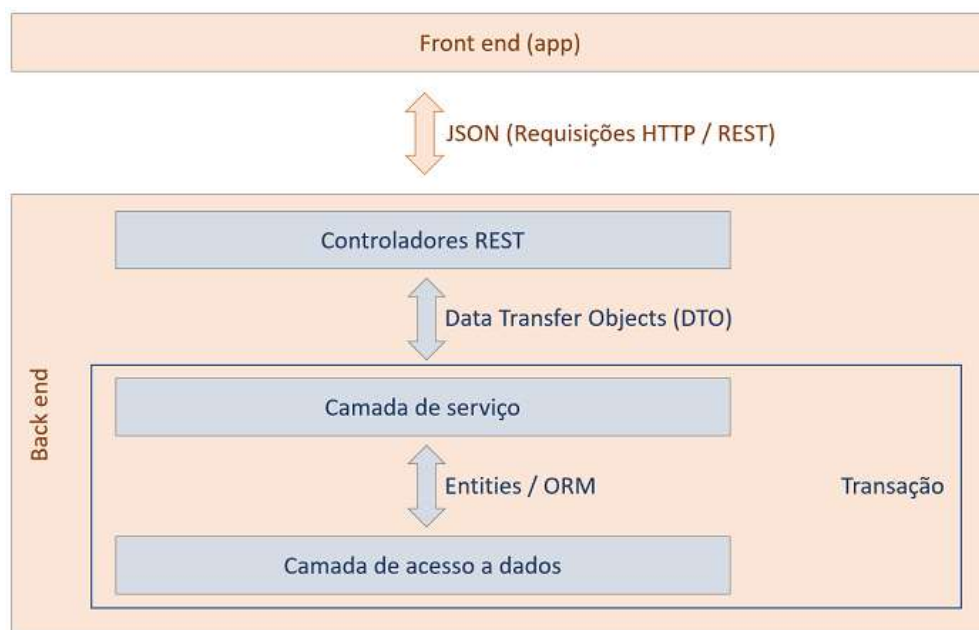
12

Padrão camadas

- Consiste em organizar os **componentes** do sistema em partes denominadas camadas
- Cada camada possui uma responsabilidade específica
- Componentes de uma camada só podem depender de componentes da mesma camada, ou da camada mais abaixo

13

13



14

Responsabilidades

- Controlador: responder interações do usuário
 - No caso de uma API REST, essas “interações” são as requisições
- Service: realizar operações de negócio.
 - Um método da camada Service deve ter um SIGNIFICADO relacionado ao negócio, podendo executar várias operações. Exemplo: registrarPedido [verificar estoque, salvar pedido, baixar estoque, enviar email]
- Repository: realizar operações “individuais” de acesso ao banco de dados

15

15

DTO - Data Transfer Object

- Data Transfer Object
- É um objeto SIMPLES para transferência de dados
- Não é gerenciado por uma lib de ORM / acesso a dados
- Pode conter outros DTO's aninhados
 - Nunca aninhe uma entity dentro de um DTO

16

16

Pra quê usar DTO?

- Projeção de dados
 - Segurança
 - Economia de tráfego
 - Flexibilidade: permite que a API trafegue mais de uma representação dos dados
 - Para preencher uma combobox: { id: number, nome: string }
 - Para um relatório detalhado: { id: number, nome: string, salario: number, email: string, telefones: string[] }
- Separação de responsabilidades
 - Service e repository: transação e monitoramento ORM
 - Controller: tráfego simples de dados

17

17

Como copiar dados da entity para o DTO?

- Cópia manual (set / construtor)
- Usar alguma lib que copia atributos de mesmo nome de um objeto para outro, por exemplo: ModelMapper
 - <https://www.baeldung.com/entity-to-and-from-dto-for-a-java-spring-application>

18

18

CRUD

Create, Retrieve, Update, Delete

19

CRUD - Create, Retrieve, Update, Delete

DSCommerce

Novo Usuário Sair

Cadastro de produtos

Novo

Nome do produto

ID	Preço	Nome	Descrição		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		
341	R\$ 5000,00	Computador Game...	Lorem ipsum dolor sit am...		

Carregar mais

DSCommerce

Novo Usuário Sair

DADOS DO PRODUTO

Nome

Preço

Imagem

Categorias

Descrição

Cancelar

Salvar

20

Operações de back end para um CRUD

- (C) Salvar um novo registro
- (R) Recuperar todos registros (paginados)
- (R) Recuperar um registro (dado um id)
- (U) Atualizar um registro (dado um id)
- (D) Deletar um registro (dado um id)

21

21

Exemplo de produto JSON

<https://gist.github.com/acenelio/d79622af48fbf1ddf99690c302cf2f62>

22

22

Sistema DSCommerce

Documento de requisitos:

<https://drive.google.com/drive/folders/1WTBggtq38cLeeQosPHjuhjSLxa94Lmx>

Exemplo de CRUD: caso de uso **Manter produtos**

23

23

Exceções

Tratamento de exceções com ControllerAdvice

24

Códigos de erro mais comuns

- 400 - Bad request (erro genérico)
- 401 - Unauthorized (falha na autenticação)
- 403 - Forbidden (acesso negado)
- 404 - Not found
- 409 - Conflict
- 415 - Unsupported Media Type
- 422 - Unprocessable entity

25

25

ControllerAdvice do Spring

Em uma classe com a annotation `@ControllerAdvice`, podemos definir tratamentos globais para exceções específicas, sem precisar ficar colocando try-catch em várias partes do código.

```
@ControllerAdvice
public class ControllerExceptionHandler {

    @ExceptionHandler(CustomException.class)
    public ResponseEntity<CustomError> customName(CustomException e, HttpServletRequest request) {
        HttpStatus status = HttpStatus.NOT_FOUND;
        CustomError err = new ...
        return ResponseEntity.status(status).body(err);
    }
}
```

Nota: no exemplo, `CustomError` e `CustomException` seriam tipos que nós criamos.

26

26

Validação

Validação de dados com Bean Validation

27

Bean Validation

<https://jakarta.ee/specifications/bean-validation/3.0/>

<https://jakarta.ee/specifications/bean-validation/3.0/apidocs/>

(acessar pacote constraints)

<https://javaee.github.io/tutorial/bean-validation.html>

28

28

Dependências Maven

```
<dependency>  
  <groupId>jakarta.validation</groupId>  
  <artifactId>jakarta.validation-api</artifactId>  
  <version>3.0.2</version>  
</dependency>
```

```
<dependency>  
  <groupId>org.hibernate</groupId>  
  <artifactId>hibernate-validator</artifactId>  
  <version>8.0.0.CR2</version>  
</dependency>
```