

FIAP GRADUAÇÃO

SISTEMAS DE INFORMAÇÃO

MICROSERVICE AND WEB ENGINEERING

Prof^a. Aparecida Castello Rosa
profaparecida.rosa@fiap.com.br

Agenda

- Continuar com o projeto produto-mvc
- Criar a camada Model
- Persistência de dados com JPA
- Banco de Dados H2.
- Entity (Entidade)
- Iniciando com *Bean Validation*

Objetivos

- Continuar com o projeto produto-mvc
- Criar camada model
- Entender a passagem de informação entre View, Controller e Model
- Implementar o método PostMapping
- Baixar os arquivos anexados à aula

Aula anterior

Aula Anterior – Erro 405

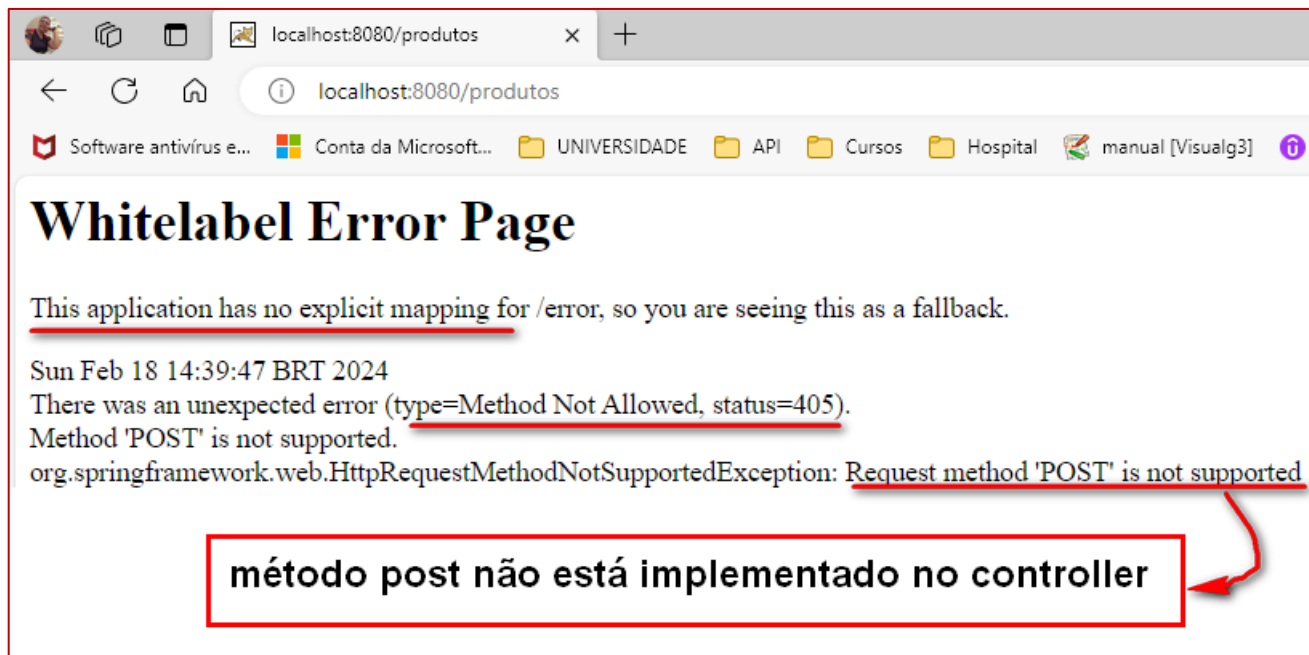


Cadastro de Produto

Nome do Produto:

Descrição:

Valor - R\$:



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun Feb 18 14:39:47 BRT 2024

There was an unexpected error (type=Method Not Allowed, status=405).

Method 'POST' is not supported.

org.springframework.web.HttpRequestMethodNotSupportedException: Request method 'POST' is not supported

método post não está implementado no controller

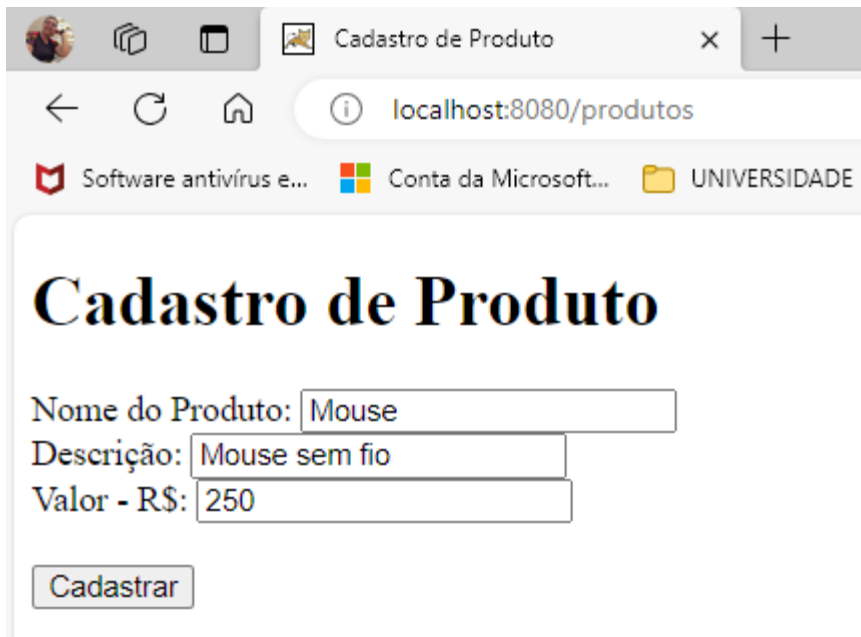
Aula Anterior

- Implementando o método PostMapping – provisoriamente

```
10 // -- código omitido --
11 @Controller // Gerenciado pelo Spring
12 @RequestMapping("/produtos") //mapeando URL
13 public class ProdutoController {
14
15     // Método ou verbo HTTP
16     @GetMapping
17     public String adicionarProduto(){
18
19         // As Views ficam dentro da pasta templates
20         //retorna caminho da página cadastro HTML
21         return "produto/novo-produto";
22     }
23
24     // receber dados da View para cadastrar produto
25     @PostMapping()
26     public String insertProduto(){
27         //provisório
28         return "redirect:/produtos";
29     }
30 }
```

Aula Anterior

- Depois de clicar no botão Cadastrar, provisoriamente, carregamos novamente a página novo-produto.html, pelo redirecionamento da rota.



Cadastro de Produto

Nome do Produto:

Descrição:

Valor - R\$:



Cadastro de Produto

Nome do Produto:

Descrição:

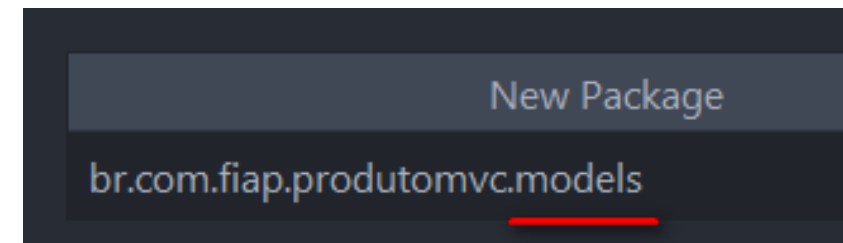
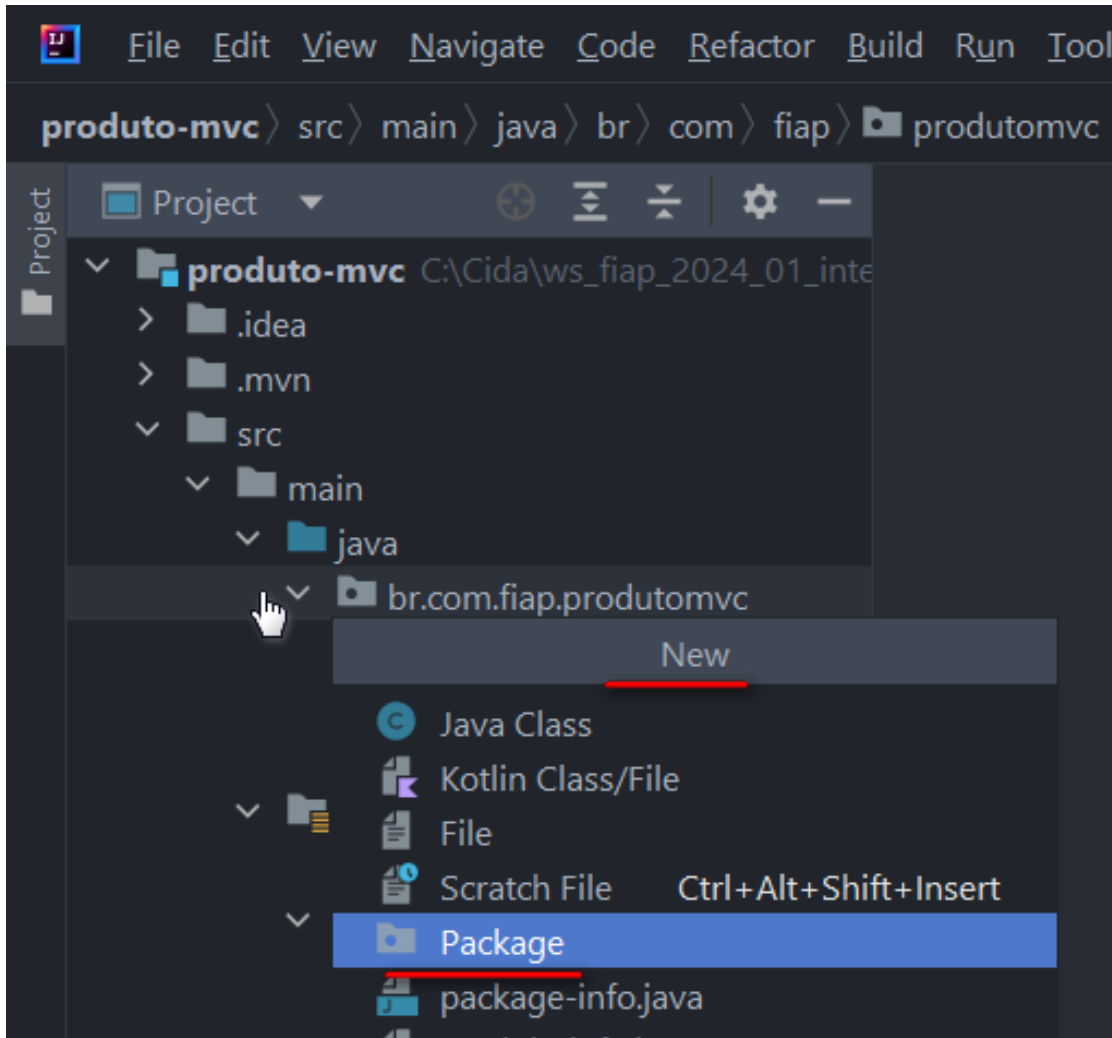
Valor - R\$:

Model

Camada Model

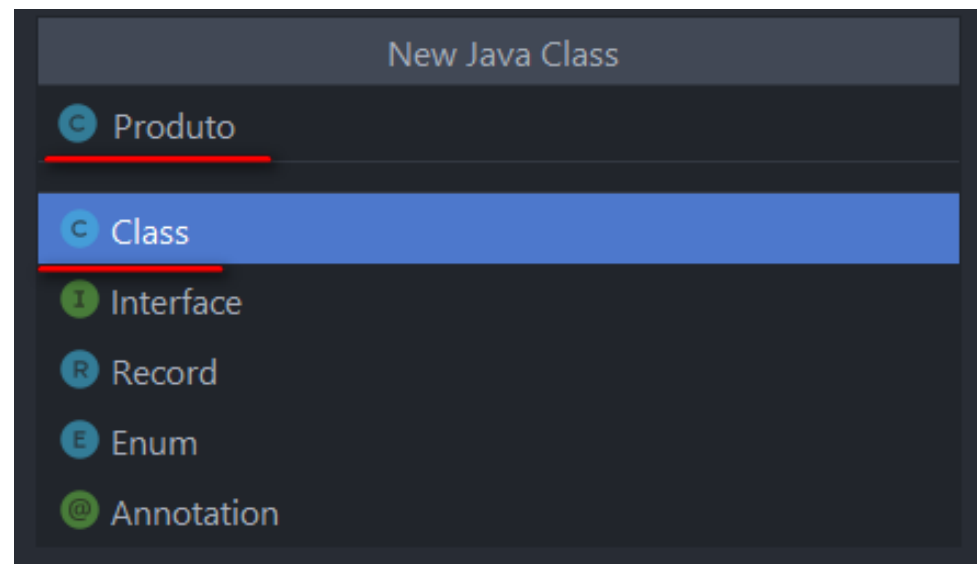
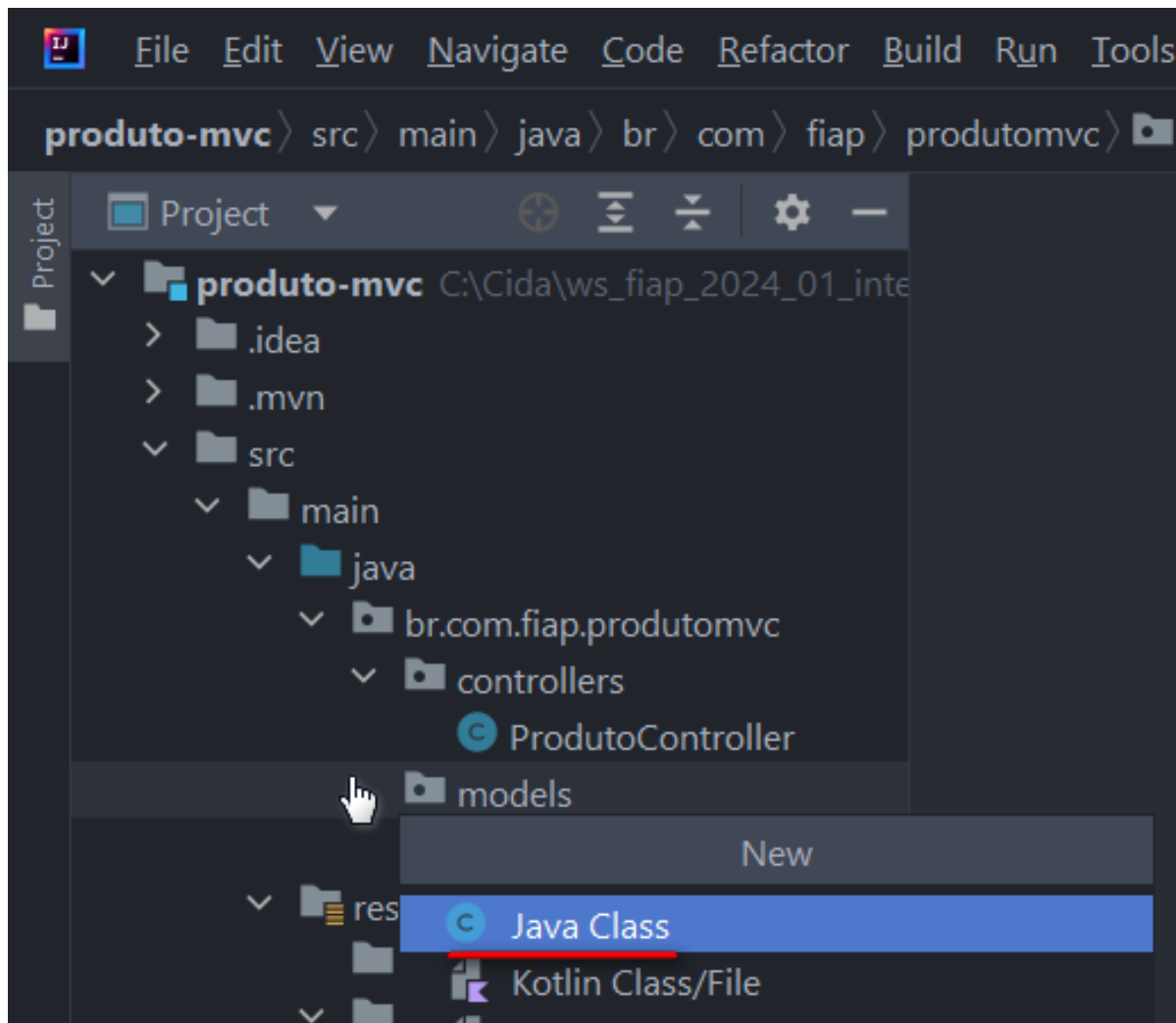
Package models

- Tecla de atalho: ALT + Insert



class Produto

Tecla de atalho: ALT + Insert



Class Produto

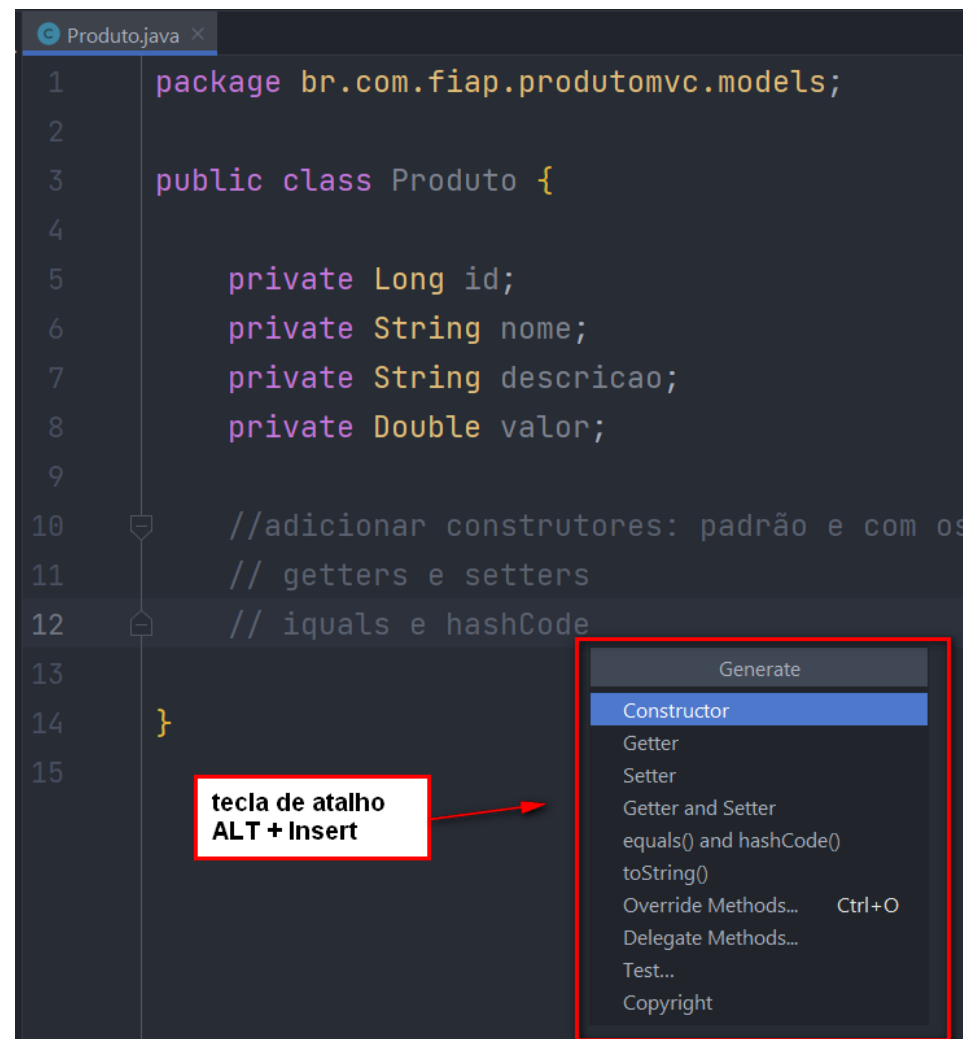
```
Produto.java x
1  package br.com.fiap.produtomvc.models;
2
3  public class Produto {
4
5      private Long id;
6      private String nome;
7      private String descricao;
8      private Double valor;
9
10     //adicionar construtores: padrão e com os atributos
11     // getters e setters
12     // equals e hashCode
13 }
```

Produto

- id : Long
- nome : String
- descricao : String
- valor : Double

Class Produto

- **Incluir**
 - construtores com e sem argumentos
 - Getter and setter
 - equals() and hashCode()
 - toString()



```

1 package br.com.fiap.produtomvc.models;
2
3 public class Produto {
4
5     private Long id;
6     private String nome;
7     private String descricao;
8     private Double valor;
9
10    //adicionar construtores: padrão e com os
11    // getters e setters
12    // equals e hashCode
13
14 }
15

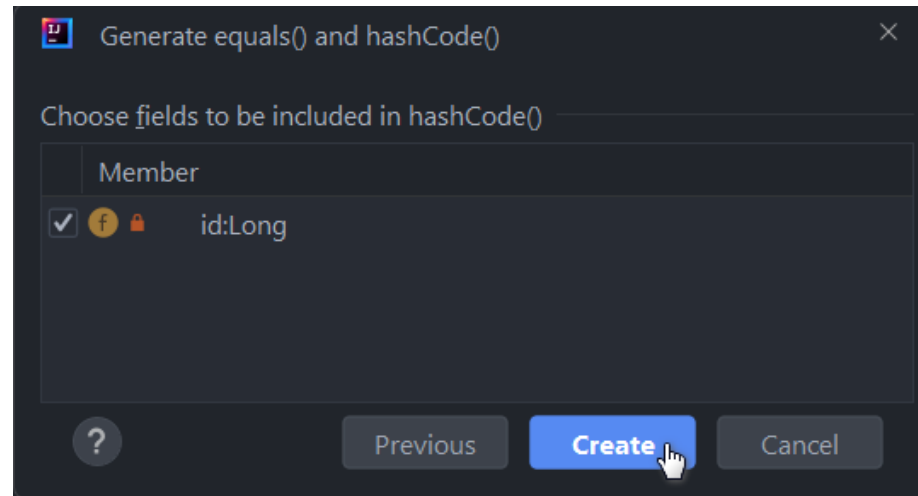
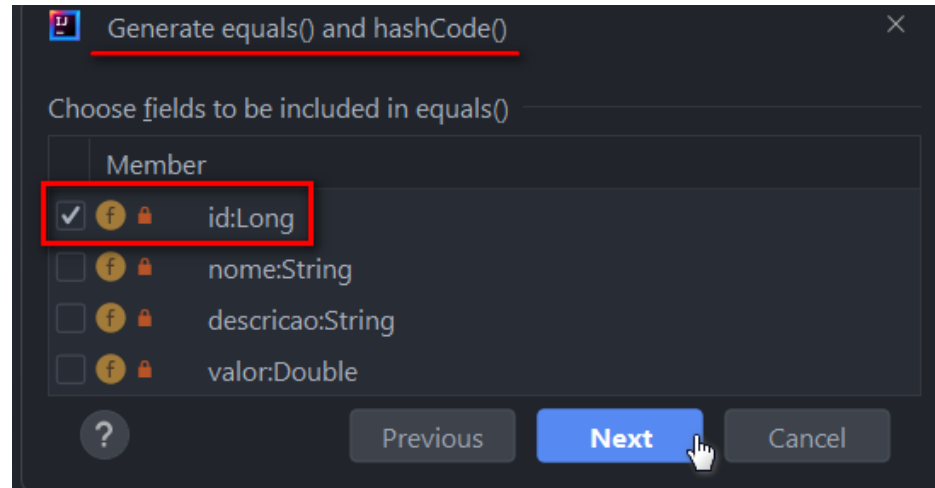
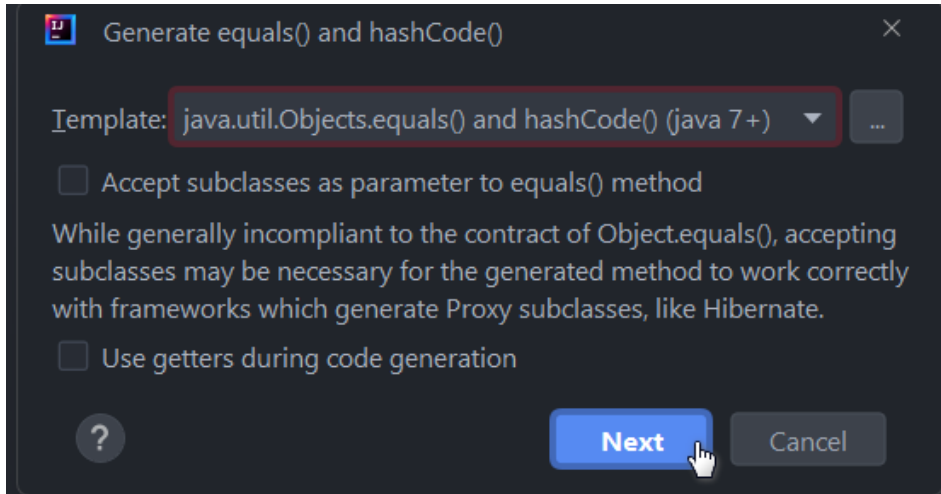
```

tecla de atalho
ALT + Insert

Generate

- Constructor
- Getter
- Setter
- Getter and Setter
- equals() and hashCode()
- toString()
- Override Methods... Ctrl+O
- Delegate Methods...
- Test...
- Copyright

IntelliJ - Gerando equals() and hashCode()



Spring Data JPA e H2 Database

Dependências no pom.xml

Adicionar as dependências ao projeto



Project

☐ Gradle - Groovy

☐ Gradle - Kotlin

☒ **Maven**

Language

☒ **Java**

☐ Kotlin

☐ Groovy

Spring Boot

☐ 3.3.0 (SNAPSHOT)

☐ 3.3.0 (M1)

☐ 3.2.3 (SNAPSHOT)

☒ **3.2.2**

☐ 3.1.9 (SNAPSHOT)

☐ 3.1.8

Project Metadata

Group

Artifact

Name

Dependencies

ADD ... CTRL + B

Spring Data JPA

SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database

SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

! pom.xml - Dependências JPA e H2

The screenshot shows the Spring Initializr web application interface. On the left, a file explorer shows the project structure: demo.zip, .gitignore, .mvn, HELP.md, mvnw, mvnw.cmd, pom.xml (highlighted), and src. On the right, the pom.xml content is displayed. A red box highlights the following dependencies:

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>
</dependencies>
```

A red arrow points from a yellow text box to the highlighted dependencies. The text box contains the following text:

Copiar as dependências do JPA e H2 e colar no pom.xml

At the bottom of the interface, there are buttons for 'DOWNLOAD CTRL + ⌘' and 'CLOSE ESC'.

Alterar - pom.xml - Dependências JPA e H2

```
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>  
  
<dependency>  
<groupId>com.h2database</groupId>  
<artifactId>h2</artifactId>  
<scope>runtime</scope>  
</dependency>
```

Atualizar o Maven

Problemas pom.xml

Se o pom.xml apresentar problema, inserir o código abaixo, dentro da tag `<plugins></plugins>`

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-resources-plugin</artifactId>
<version>3.1.0</version>
</plugin>
```

código omitido

```
</dependencies>
```

```
<build>
```

```
<plugins>
```

```
<plugin>
```

```
<groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-maven-plugin</artifactId>
```

```
</plugin>
```

```
<!-- Failed to execute goal org.apache.maven.plugins:ma
```

```
<plugin>
```

```
<groupId>org.apache.maven.plugins</groupId>
```

```
<artifactId>maven-resources-plugin</artifactId>
```

```
<version>3.1.0</version>
```

```
</plugin>
```

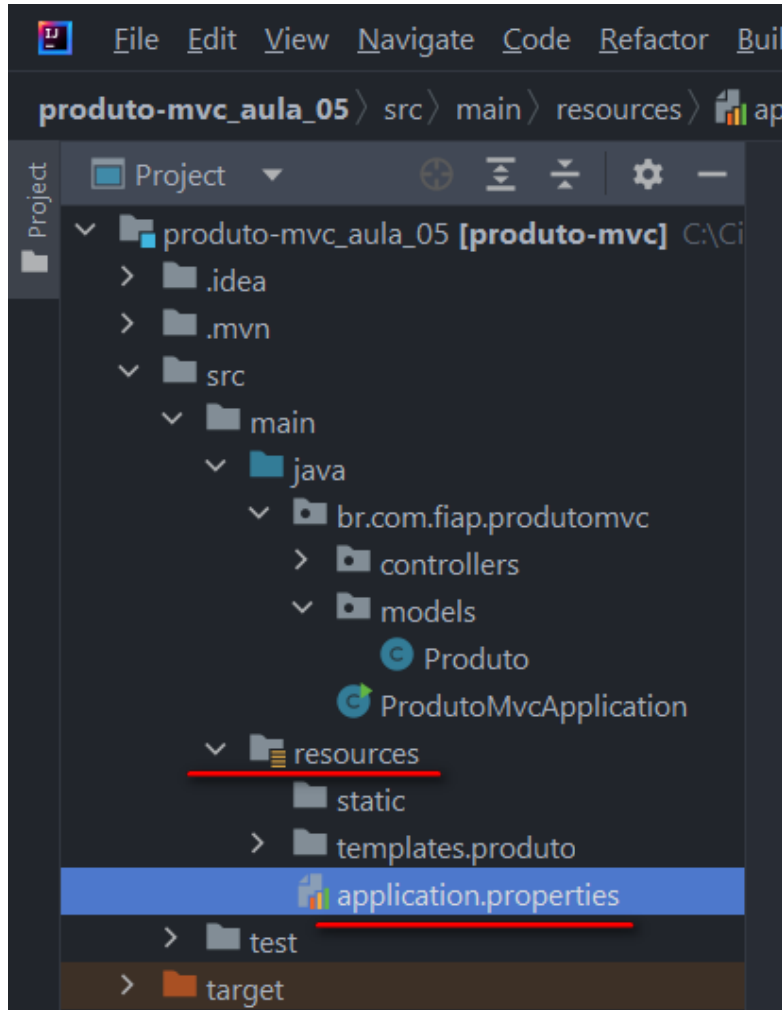
```
</plugins>
```

código omitido

application.properties

Configurações para o H2

application.properties - profiles



- No arquivo *application.properties* podemos inserir algumas configurações da aplicação. Esse arquivo já existe, só precisamos editá-lo.

application.properties

alterar a porta - se tiver problemas no laboratório

server.port = 8081

spring.profiles.active=test

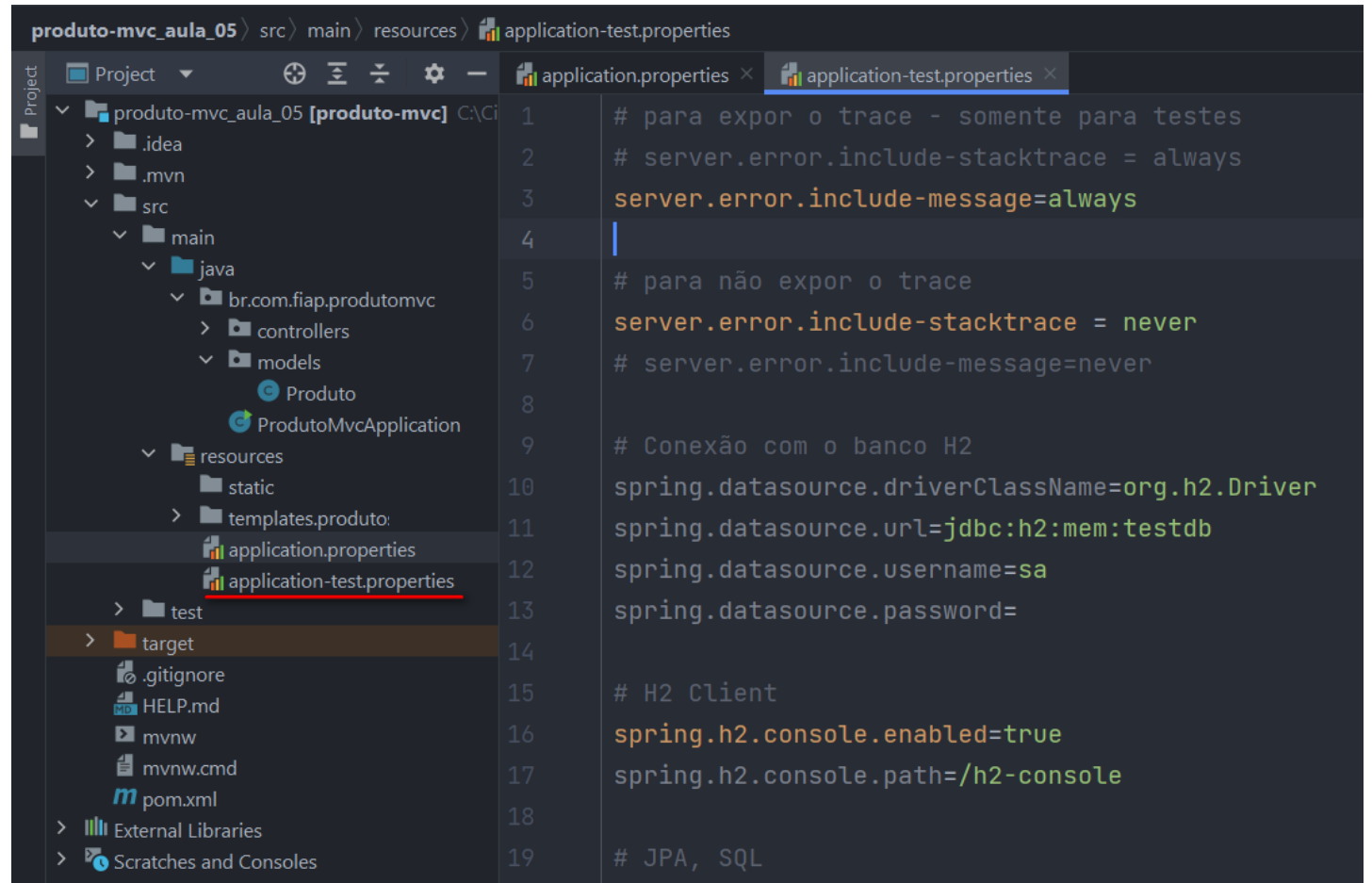
spring.jpa.open-in-view=false

A screenshot of a code editor window titled 'application.properties'. The window contains five lines of text. The first two lines are commented out with '#'. The third line is empty. The fourth and fifth lines are active Spring properties. The text is as follows:

```
1 # alterar a porta - se tiver problemas no laboratório
2 # server.port = 8081
3
4 spring.profiles.active=test
5 spring.jpa.open-in-view=false
```

application-test.Properties – Profile Test

Baixar este arquivo
em apostilas e
adicionar em
resources



The screenshot shows an IDE with a project named 'produto-mvc_aula_05'. The left sidebar displays the project structure, with 'resources/application-test.properties' highlighted. The main editor window shows the content of this file, which is a properties file for testing. The file contains comments in Portuguese and configuration for a H2 database and JPA.

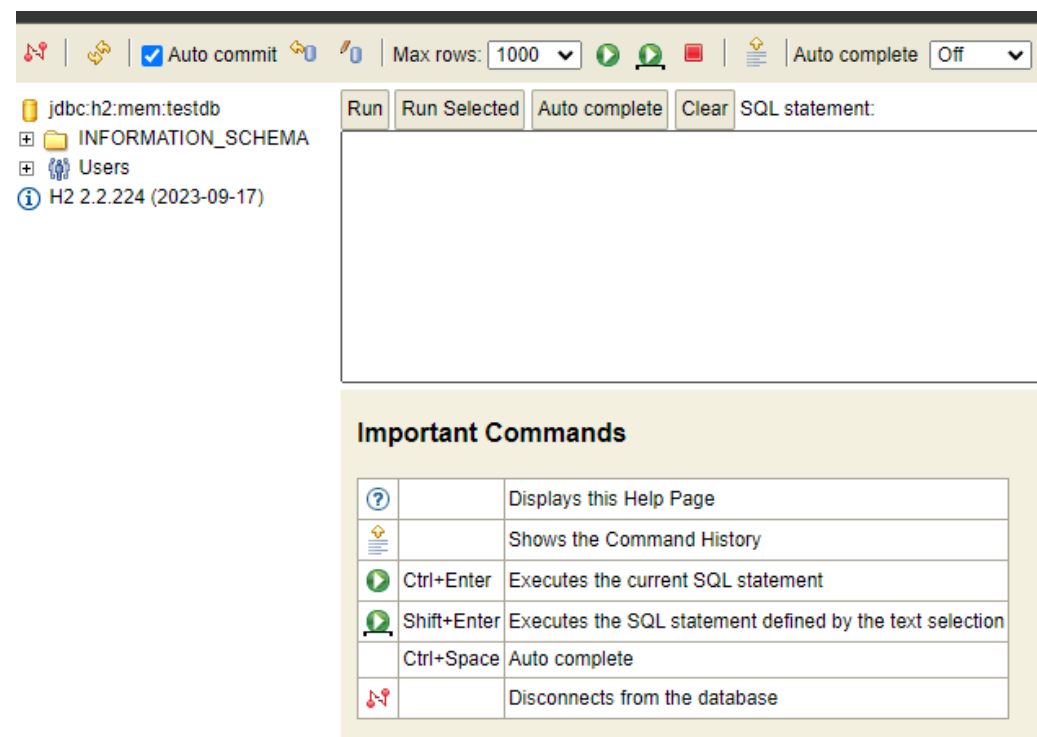
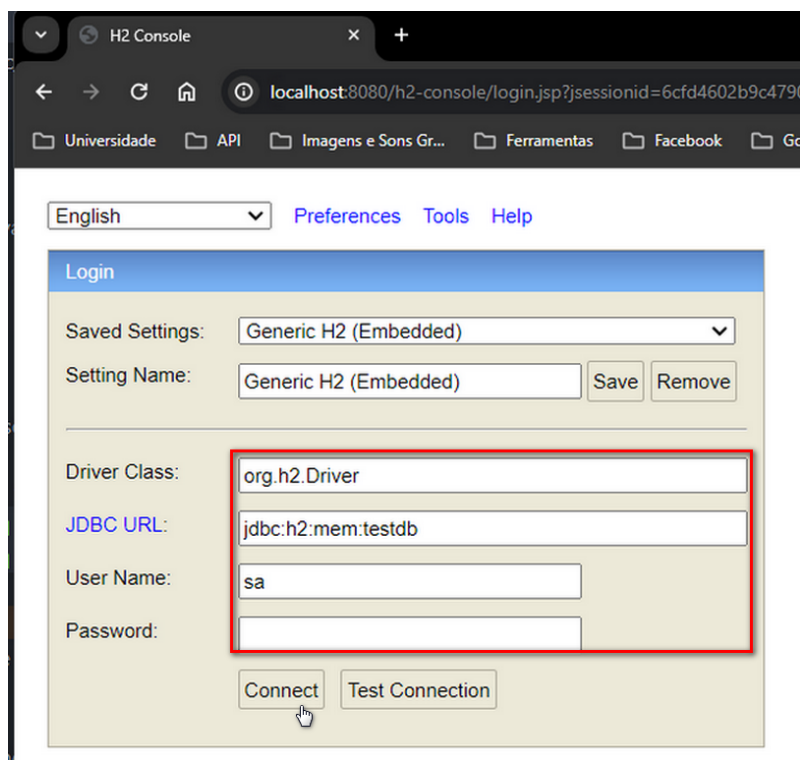
```
1 # para expor o trace - somente para testes
2 # server.error.include-stacktrace = always
3 server.error.include-message=always
4
5 # para não expor o trace
6 server.error.include-stacktrace = never
7 # server.error.include-message=never
8
9 # Conexão com o banco H2
10 spring.datasource.driverClassName=org.h2.Driver
11 spring.datasource.url=jdbc:h2:mem:testdb
12 spring.datasource.username=sa
13 spring.datasource.password=
14
15 # H2 Client
16 spring.h2.console.enabled=true
17 spring.h2.console.path=/h2-console
18
19 # JPA, SQL
```

<https://docs.spring.io/spring-boot/docs/1.2.0.M1/reference/html/boot-features-profiles.html>

Testar o H2 no navegador

Iniciar a aplicação

<http://localhost:8080/h2-console>



Model - Entity

Entity (Entidade) do banco de dados

I **Annotations**

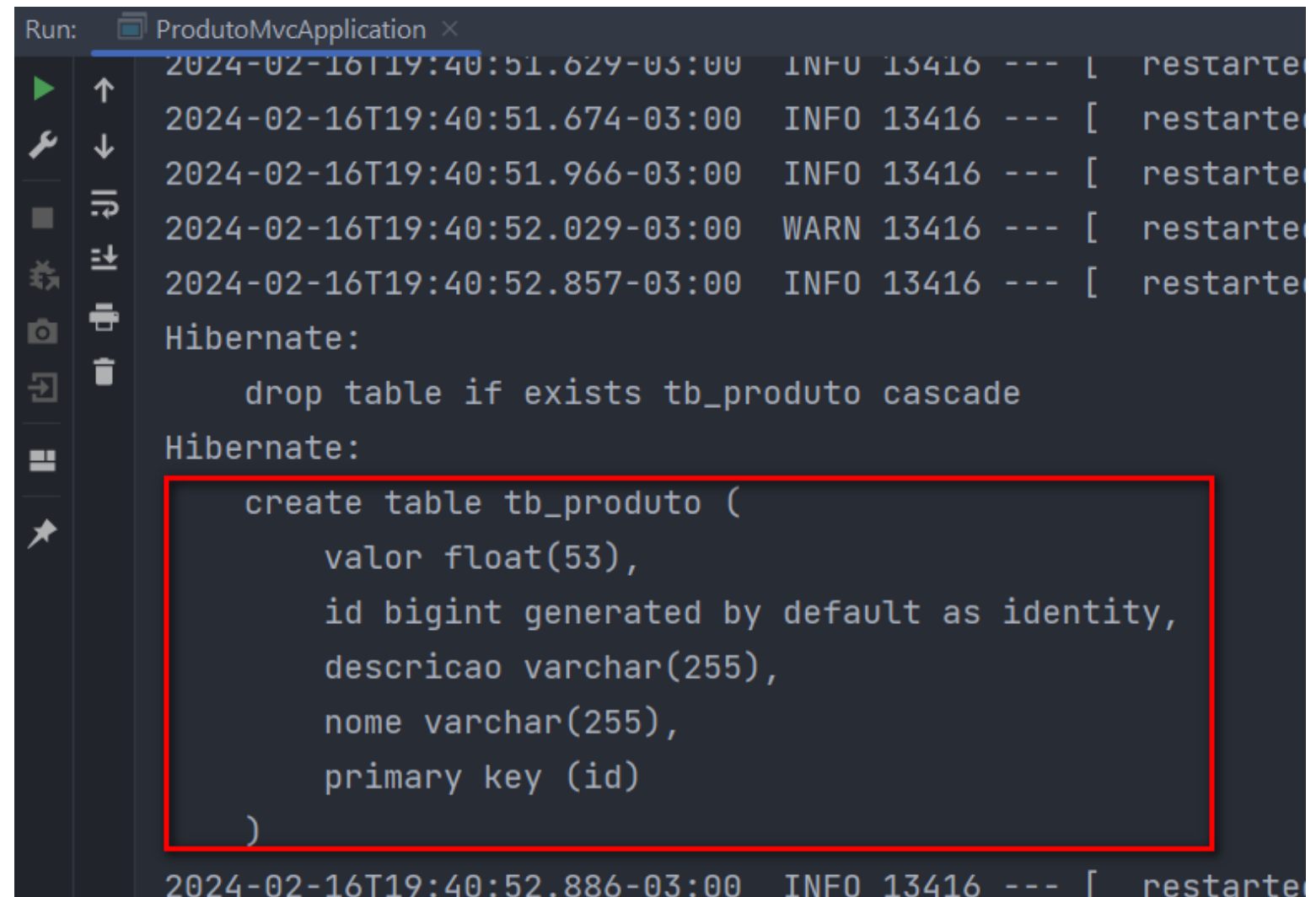
Anotação (Data)	Objetivo
@Entity	Identifica a classe como mapeamento de tabela no BD
@Id	Identifica o atributo que representa a coluna para PK na tabela
@Repository	Classes de acesso a banco de dados
@Column	Altera propriedades da coluna no banco de dados
@Table	Altera o nome da table no banco de dados
@GeneratedValue	Usa uma estratégia para geração automática de número identificador único no atributo

Alterar class Produto

```
Produto.java x
1 package br.com.fiap.produtomvc.models;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.persistence.Id;
7 import jakarta.persistence.Table;
8
9 import java.util.Objects;
10
11 @Entity
12 @Table(name = "tb_produto")
13 public class Produto {
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private Long id;
17     private String nome;
18     private String descricao;
19     private Double valor;
20     // código omitido
```

■ Executar a aplicação

- Saída no console

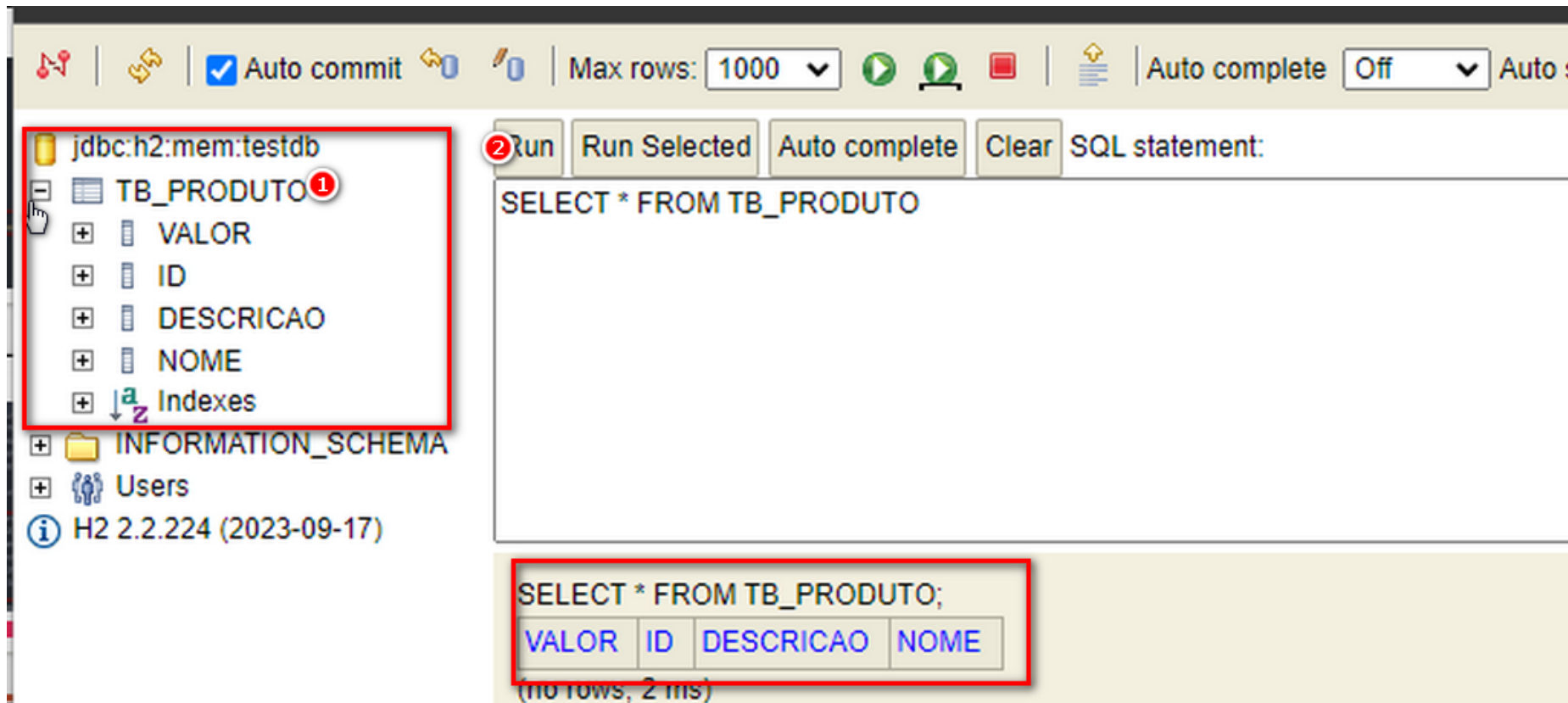


The screenshot shows a console window titled "Run: ProdutoMvcApplication". The output includes several log entries with timestamps, log levels (INFO, WARN), and messages. A red box highlights the SQL commands executed by Hibernate to create the database table.

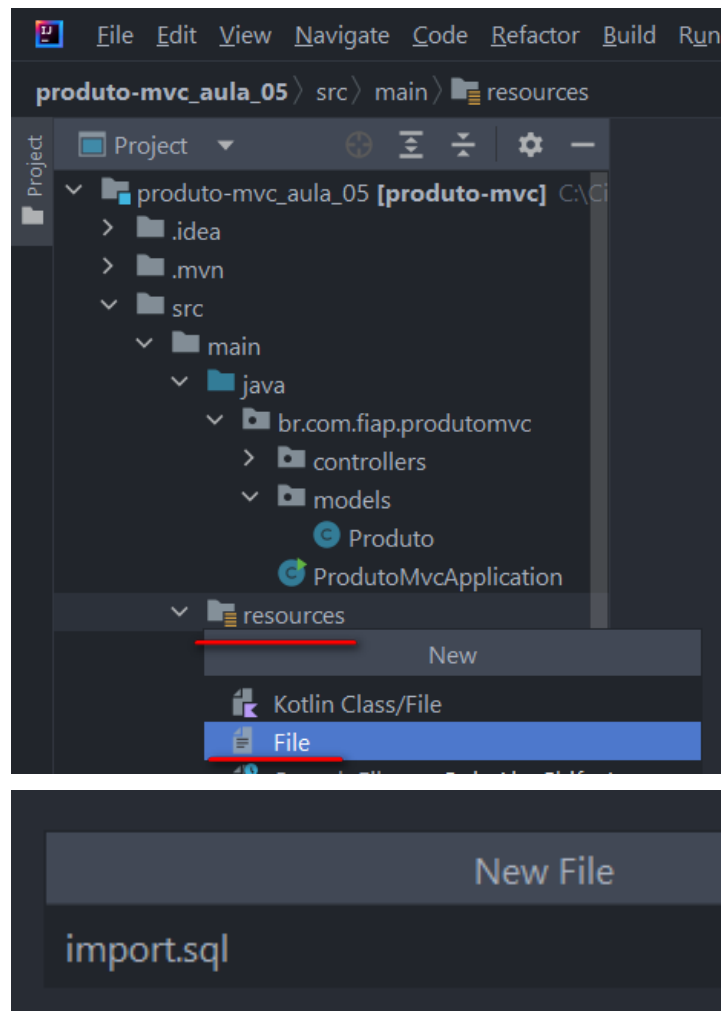
```
Run: ProdutoMvcApplication x
2024-02-16T19:40:51.629-03:00 INFO 13416 --- [ restarted
2024-02-16T19:40:51.674-03:00 INFO 13416 --- [ restarted
2024-02-16T19:40:51.966-03:00 INFO 13416 --- [ restarted
2024-02-16T19:40:52.029-03:00 WARN 13416 --- [ restarted
2024-02-16T19:40:52.857-03:00 INFO 13416 --- [ restarted
Hibernate:
    drop table if exists tb_produto cascade
Hibernate:
    create table tb_produto (
        valor float(53),
        id bigint generated by default as identity,
        descricao varchar(255),
        nome varchar(255),
        primary key (id)
    )
2024-02-16T19:40:52.886-03:00 INFO 13416 --- [ restarted
```

Testar o H2

<http://localhost:8080/h2-console>



Seed para o banco de dados



Criar arquivo **import.sql** na pasta *resources* (se der erro, renomear o arquivo para **data.sql**).

Vamos escrever o script para inserir alguns dados no DB.

import.sql

```
INSERT INTO tb_produto(nome, descricao, valor) VALUES('Mouse  
Microsoft', 'Mouse sem fio', 250.0);
```

```
INSERT INTO tb_produto(nome, descricao, valor) VALUES('Smartphone  
Samsung Galaxy A54 5G', 'Samsung Galaxy A54 5G', 1799.0);
```

```
INSERT INTO tb_produto(nome, descricao, valor) VALUES('Smart TV',  
'Smart TV LG LED 65 polegadas', 3999);
```

Seed DB

```
import.sql x
1 INSERT INTO tb_produto(nome, descricao, valor) VALUES('Mouse Microsoft', 'Mouse sem fio', 250.0);
2 INSERT INTO tb_produto(nome, descricao, valor) VALUES('Smartphone Samsung Galaxy A54 5G', 'Samsung Galaxy A54 5G', 1799.0);
3 INSERT INTO tb_produto(nome, descricao, valor) VALUES('Smart TV', 'Smart TV LG LED 65 polegadas', 3999);
```

```
Hibernate:
    drop table if exists tb_produto cascade
Hibernate:
    create table tb_produto (
        valor float(53),
        id bigint generated by default as identity,
        descricao varchar(255),
        nome varchar(255),
        primary key (id)
    )
Hibernate: INSERT INTO tb_produto(nome, descricao, valor) VALUES('Mouse Microsoft', 'Mouse sem fio', 250.0)
Hibernate: INSERT INTO tb_produto(nome, descricao, valor) VALUES('Smartphone Samsung Galaxy A54 5G', 'Samsung Galaxy A54 5G', 1799.0)
Hibernate: INSERT INTO tb_produto(nome, descricao, valor) VALUES('Smart TV', 'Smart TV LG LED 65 polegadas', 3999)
2024-02-18T19:16:59.562-03:00 INFO 14428 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManager
```


Testar no H2

<http://localhost:8080/h2-console>

The screenshot displays the H2 Database Console interface. On the left, a tree view shows the database structure: jdbc:h2:mem:testdb, TB_PRODUTO, INFORMATION_SCHEMA, Users, and H2 2.2.224 (2023-09-17). The main area shows the SQL statement `SELECT * FROM TB_PRODUTO` entered in the 'SQL statement:' field. Below the statement, the execution results are displayed as a table with 3 rows and 4 columns: VALOR, ID, DESCRICAO, and NOME. The results show three products: a mouse, a smartphone, and a smart TV. The interface also includes a toolbar with buttons for 'Run', 'Run Selected', 'Auto complete', and 'Clear', along with settings for 'Auto commit' and 'Max rows'.

VALOR	ID	DESCRICAO	NOME
250.0	1	Mouse sem fio	Mouse Microsoft
1799.0	2	Samsung Galaxy A54 5G	Smartphone Samsung Galaxy A54 5G
3999.0	3	Smart TV LG LED 65 polegadas	Smart TV

(3 rows, 2 ms)

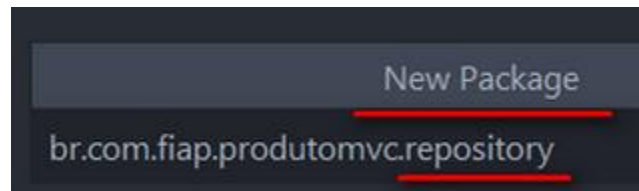
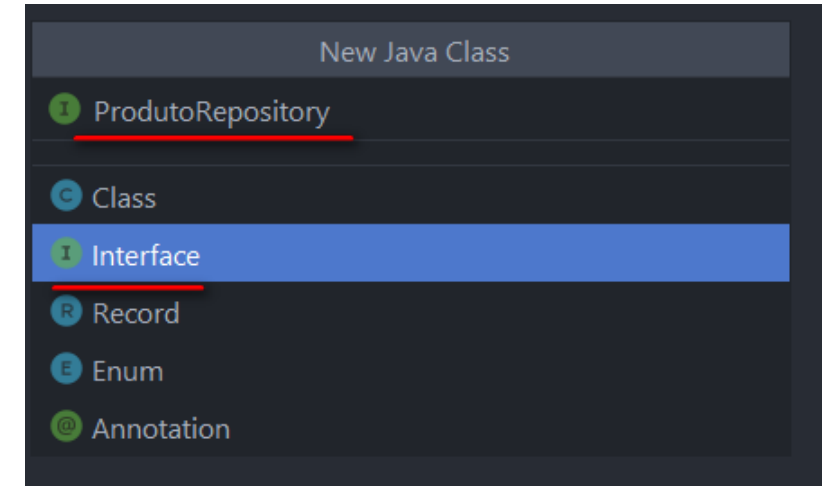
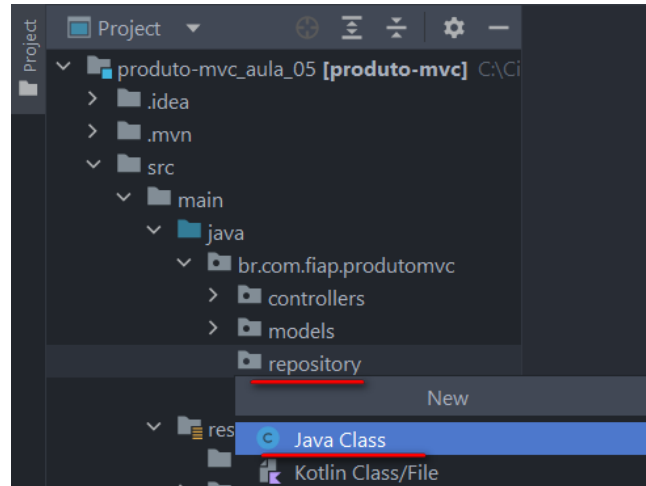
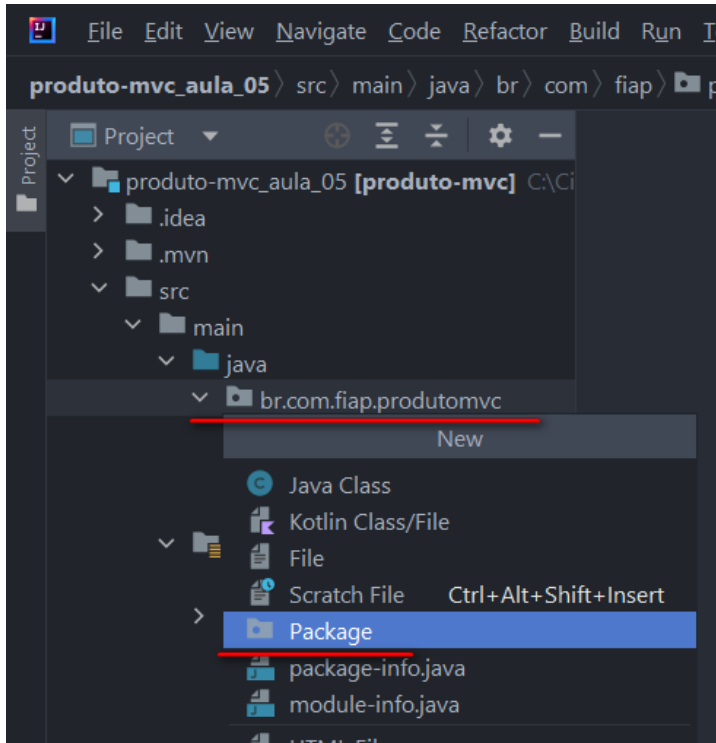
Camada de Acesso a Dados - *Repository*

Repositório para acesso a dados

■ Camada de Acesso a Dados - JPA

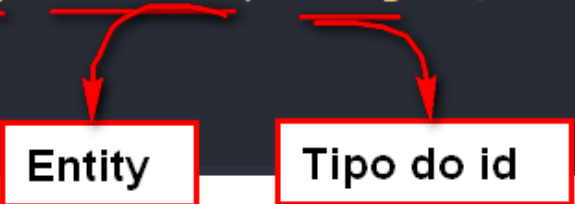
- O **Spring Data JPA** é um framework para facilitar a criação dos nossos repositórios.
- Implementa consultas e interações com a base de dados.
- Assim, métodos como `findAll()`, `save()`, `delete()`, entre outros podem ser utilizados sem a necessidade de implementação.
- Criar uma **interface** que ***extends*** ***JpaRepository*** do Spring Data para utilizar os métodos já prontos para realizar as transações (operações) com o banco de dados.

■ Criar a interface ProdutoRepository



interface ProdutoRepository

```
ProdutoRepository.java x
1 package br.com.fiap.produtomvc.repository;
2
3 import br.com.fiap.produtomvc.models.Produto;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository // anotação opcional
8 public interface ProdutoRepository extends JpaRepository<Produto, Long> {
9
10 }
```



Alterar View novo-produto.html

```
7 <!--código omitido-->
8 <body>
9 <h1>Cadastro de Produto</h1>
10 <form name="form-cadastro" action="#" th:action="@{/produtos/salvar}" th:object="${produto}" method="post">
11 <label for="nome">Nome do Produto:</label>
12 <input id="nome" name="nome" th:field="*{nome}" placeholder="Nome do produto"> <br />
13 <label for="descricao">Descrição:</label>
14 <input id="descricao" name="descricao" th:field="*{descricao}" placeholder="Descrição do produto">
15 <br />
16 <label for="valor">Valor - R$:</label>
17 <input type="text" id="valor" name="valor" th:field="*{valor}" placeholder="Valor do produto">
18 <br /><br />
19 <input type="submit" value="Salvar">
20 </form>
21 <!--código omitido-->
```

URL mapeada - /produtos/salvar

objeto referenciado no controller

alterar para type="text"

th:field = corresponde aos atributos do model (class Produto)

Alterar class ProdutoController

```
ProdutoController.java x
12 // -- código omitido --
13 @Controller // Gerenciado pelo Spring
14 @RequestMapping("/produtos") //mapeando URL
15 public class ProdutoController {
16
17     //URL - localhost:8080/produtos/novo
18     @GetMapping("/novo")
19     @ public String adicionarProduto(Model model){
20         //Model é injetado pelo Spring
21         //instância um objeto Produto vazio, associado à chave produto
22         model.addAttribute("produto", new Produto());
23         //model -> enviar o obj. Produto para a view
24         return "produto/novo-produto";
25     }
26
27     // receber dados do form da View novo-produto.html
28     //URL - localhost:8080/produtos/salvar
29     @PostMapping("/salvar")
30     @ public String insertProduto(Produto produto){ //recebe objeto
31         System.out.println(produto.toString());
32         //provisório - redireciona para localhost:8080/produtos/novo
33         return "redirect:/produtos/novo";
34     }
35 }
```

Validation

Validation in Spring Boot

■ *Validation* – Adicionar dependência pom.xml

FIAP

Dependencies

ADD DEPENDENCIES... CTRL + B

Validation

I/O

Bean Validation with Hibernate validator.

```
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-validation</artifactId>  
</dependency>
```

Atualizar o Maven

■ *Constraints in Bean Validation*

<i>Annotation</i>	Objetivo	Exemplo
@NotNull	O valor do campo ou propriedade não pode ser nulo.	@NotNull(message = "Campo requerido")
@Null	O valor do campo ou propriedade deve ser nulo.	
@NotEmpty	O valor do campo ou propriedade não pode ser vazio.	@NotEmpty(message = "A senha deve ser informada")
@NotBlank	O valor do campo ou propriedade não pode ser nulo ou vazio.	@NotBlank(message = "Campo obrigatório")

■ Constraints in Bean Validation

Annotation	Objetivo	Exemplo
@Past	O valor do campo ou propriedade deve ser uma data passada em relação à atual.	
@Future	O valor do campo ou propriedade deve ser uma data futura em relação à atual.	
@Length	Valida o tamanho mínimo e máximo de um campo.	@Length(min = 4, max = 100)
@Max	O valor do campo ou propriedade deve ser menor ou igual ao especificado no atributo value da anotação.	@Max(35)
@Min	O valor do campo ou propriedade deve ser maior ou igual ao especificado no atributo value da anotação.	@Min(18)

■ Constraints in Bean Validation

<i>Annotation</i>	<i>Objetivo</i>	<i>Exemplo</i>
@Size	O valor do campo ou propriedade deve estar dentro dos limites informados na anotação por meio dos atributos min e max. Aplica-se a Strings, Collections e Arrays.	@Size(min=2, max=30); @Size(min = 5, max = 60, message = "Deve ter entre 5 e 60 caracteres"); @Size(min = 4, message = "O login deve ter no mínimo 4 caracteres")
@Positive	O valor do campo ou propriedade deve ser um número positivo	@Positive(message = "Preço deve ser um valor positivo")
@PastOrPresent	O valor do campo ou propriedade deve ser uma data passada ou atual	@PastOrPresent(message = "A data do produto não pode ser futura")
@Email	O valor do campo ou propriedade deve ser um e-mail válido	@Email(message = "Favor entrar um email válido")

■ Constraints in Bean Validation

Diferenças entre `@NotNull`, `@NotEmpty` e `@NotBlank`

`jakarta.persistence`

Anotação	Objetivo
<code>@NotNull</code>	Não permite um valor nulo, porém permite um valor vazio. Deve ser utilizada para campos de outros tipos como, por exemplo, <code>Float</code> .
<code>@NotEmpty</code>	Não permite valor nulo e além disso seu tamanho deve ser maior que zero. Espaços em brancos são considerados na verificação do tamanho.
<code>@NotBlank</code>	Não permite valor nulo e o comprimento (sem considerar espaços em branco) deve ser maior que zero. Verifica se uma <code>String</code> é diferente de nulo e também não é vazia.

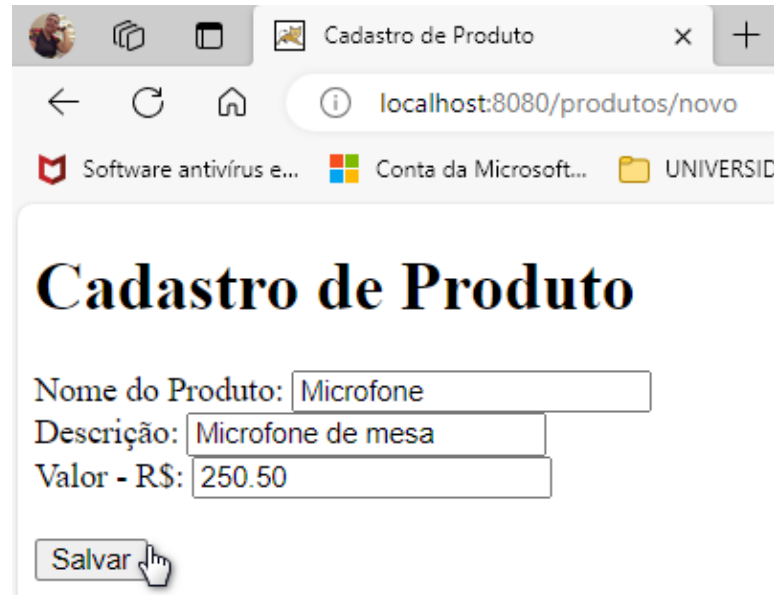
- Vazio é diferente de *null*, pois existe uma referência de memória.
- Sendo assim para validar um campo `String` de preenchimento obrigatório, não vazio, é indicado o uso de `@NotBlank`.

<https://www.baeldung.com/java-bean-validation-not-null-empty-blank>

Alterar class Produto

```
Produto.java x
16 // -- código omitido
17 @Entity
18 @Table(name = "tb_produto")
19 public class Produto {
20
21     @Id
22     @GeneratedValue(strategy = GenerationType.IDENTITY)
23     private Long id;
24
25     @NotBlank(message = "Campo requerido")
26     @Size(min = 3, message = "O nome deve ter no mínimo 3 caracteres")
27     private String nome;
28
29     @NotBlank(message = "Campo requerido")
30     @Column(columnDefinition = "TEXT") //para textos longos
31     private String descricao;
32
33     @NotNull(message = "Campo requerido")
34     @Positive(message = "O valor deve ser positivo")
35     private Double valor;
36 // -- código omitido
```

Testar a aplicação

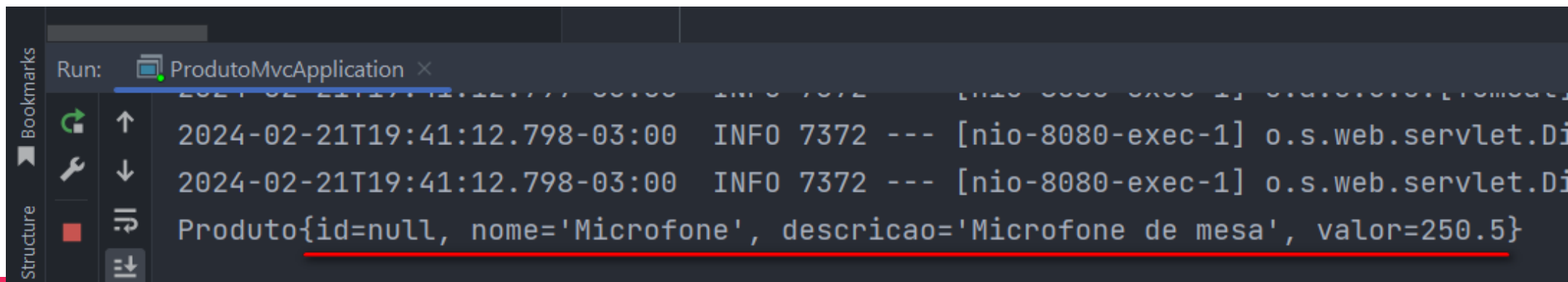


Cadastro de Produto

Nome do Produto:

Descrição:

Valor - R\$:



```
Run: ProdutoMvcApplication x
2024-02-21T19:41:12.798-03:00 INFO 7372 --- [nio-8080-exec-1] o.s.web.servlet.D
2024-02-21T19:41:12.798-03:00 INFO 7372 --- [nio-8080-exec-1] o.s.web.servlet.D
Produto{id=null, nome='Microfone', descricao='Microfone de mesa', valor=250.5}
```



Copyright © 2024
Prof^a. Aparecida de Fátima Castello Rosa

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).