

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
- Hibernate Lazy / Eager Loading
- Mapeando relacionamento Many to Many

Objetivos

- Continuar com o projeto produto-mvc – baixar projeto da aula 16
- Entender quando utilizar as propriedades de carregamento de objetos do Hibernate de forma adequada.
- Entender como funciona o mapeamento entidades com Hibernate em um relacionamento Many to Many.

■ Hibernate - Lazy / Eager Load

- Um dos pontos mais importantes para se preocupar em um software é a performance.
- Otimizações no SQL podem resultar em um ganho considerável !!!
- Vamos entender o funcionamento do Lazy e Eager Load da JPA que é configurado nos relacionamento entre entidades, pois o uso de maneira incorreta pode acarretar em uma performance desastrosa.

Hibernate - Lazy / Eager Load

EAGER é usado para carregar relacionamentos imediatamente junto com a entidade principal. Quando você carrega uma entidade, todas as entidades associadas a ela também são carregadas automaticamente.

LAZY carrega os relacionamentos apenas quando você acessa explicitamente o `getter` do relacionamento.

Isso significa que os dados associados não são carregados imediatamente, economizando recursos.

Resumindo:

EAGER: Carrega todos os dados do relacionamento.

LAZY: Não carrega os dados dos relacionamentos, a menos que explicitamente solicitado

■ Hibernate - Lazy / Eager Load

Prós e contras

EAGER

Conveniente devido ao carregamento antecipado

Evita `NullPointerException` inesperados

Menos eficiente

Tempo de carregamento maior

Maior consumo de memória

Pode afetar o desempenho por carregar muitos dados

LAZY

Mais eficiente

Tempo de carregamento menor

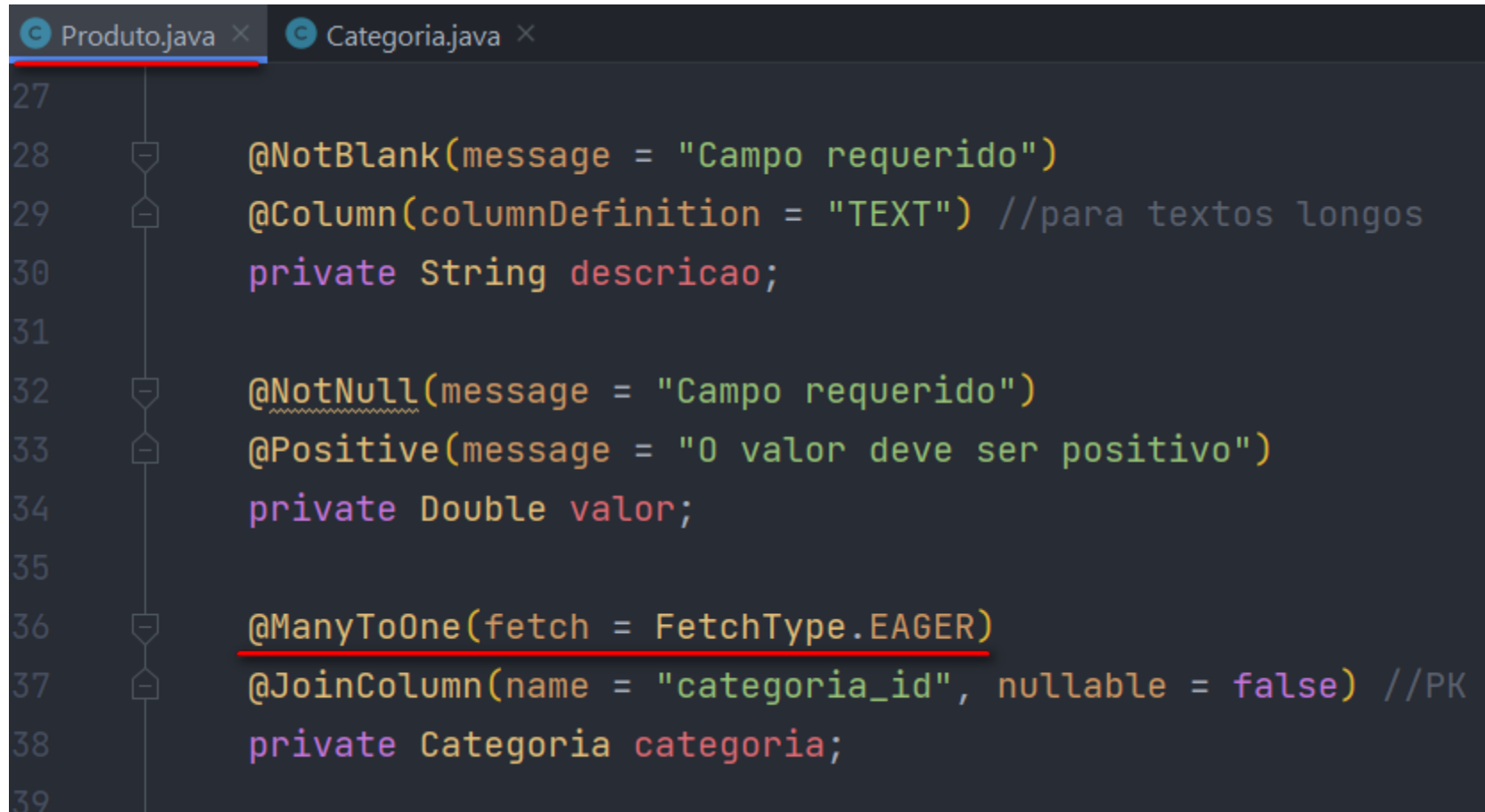
Menor consumo de memória

Pode afetar o desempenho de forma indesejada

Maior chance de `NullPointerException`



Alterar class Produto



```
27
28     @NotBlank(message = "Campo requerido")
29     @Column(columnDefinition = "TEXT") //para textos longos
30     private String descricao;
31
32     @NotNull(message = "Campo requerido")
33     @Positive(message = "0 valor deve ser positivo")
34     private Double valor;
35
36     @ManyToOne(fetch = FetchType.EAGER)
37     @JoinColumn(name = "categoria_id", nullable = false) //PK
38     private Categoria categoria;
39
```

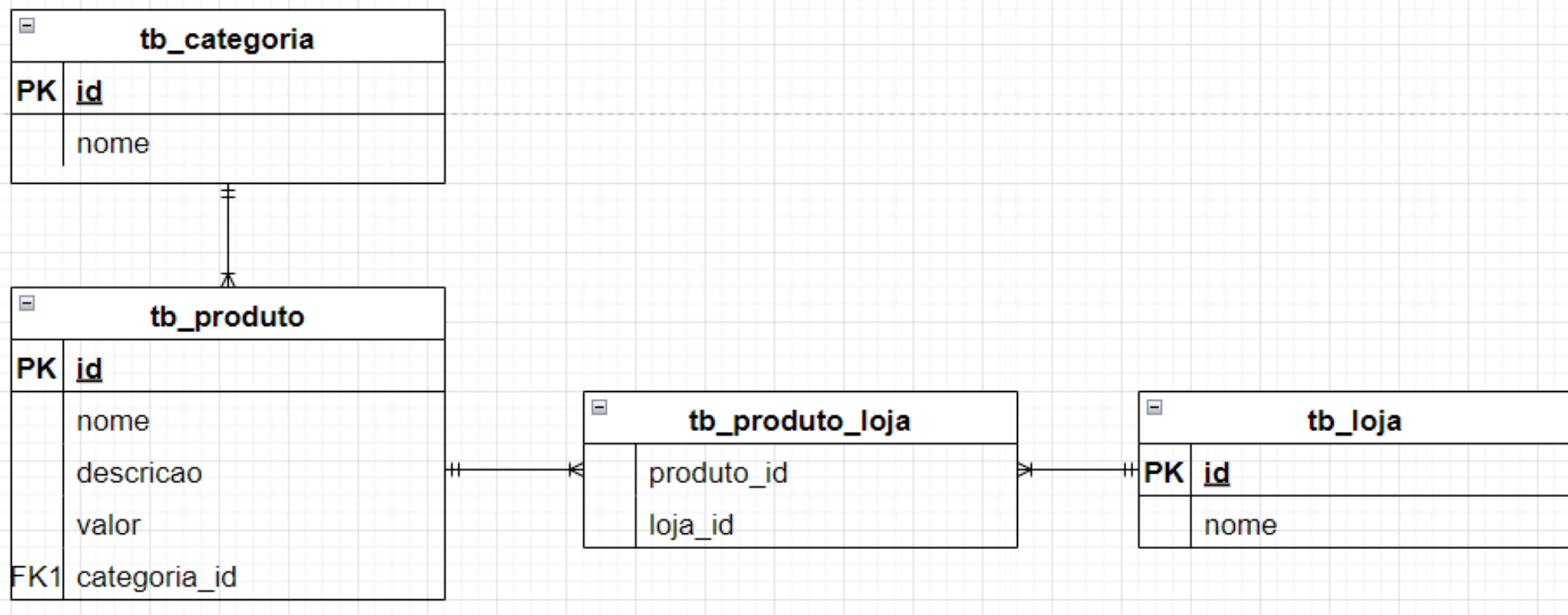

ManyToMany

■ Hibernate - Mapeamento Objeto Relacional

- Adicionar uma nova **entidade** Loja, onde 1 Produto pode ter N Lojas, e 1 Loja pode ter N Produtos:

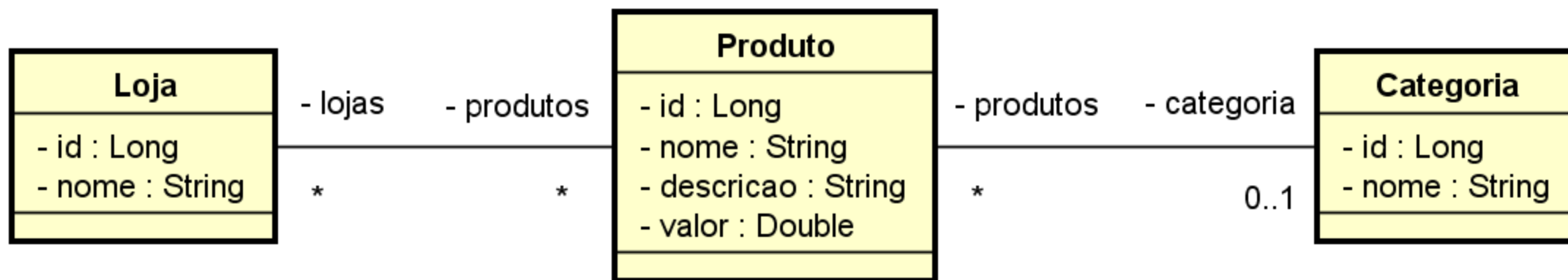
Hibernate JPA Relacionamento

- Modelo de Entidade Relacionamento



Hibernate JPA Relacionamento

- Diagrama UML



class Loja

```
Loja.java x
12  @AllArgsConstructor
13  @NoArgsConstructor
14  @Getter
15  @Setter
16
17  @Entity
18  @Table(name = "tb_loja")
19  public class Loja {
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
```

class Loja - continuação

```
Loja.java x
28
29 //relacionamento
30 //mapeamento na outra tabela
31 @ManyToMany(mappedBy = "lojas", fetch = FetchType.EAGER)
32 private Set<Produto> produtos = new HashSet<>(); // não permite valores duplicados
33
34 @Override
35 public boolean equals(Object o) {
36     if (this == o) return true;
37     if (o == null || getClass() != o.getClass()) return false;
38     Loja loja = (Loja) o;
39     return Objects.equals(id, loja.id);
40 }
41
42 @Override
43 public int hashCode() {
44     return Objects.hash(id);
45 }
46 }
```

Alterar class Produto

```
Loja.java x Produto.java x
41 @ManyToOne(fetch = FetchType.EAGER)
42 @JoinColumn(name = "categoria_id", nullable = false) //PK
43 private Categoria categoria;
44
45 //relacionamento
46 @ManyToMany(fetch = FetchType.EAGER)
47 @JoinTable(name = "tb_produto_loja", //tabela associativa
48           joinColumns = @JoinColumn(name = "produto_id"), //ref. FK - mesma entidade da classe
49           inverseJoinColumns = @JoinColumn(name = "loja_id")) //ref. PK - da outra classe
50 private Set<Loja> lojas = new HashSet<>(); // não permite valores duplicados
51
```

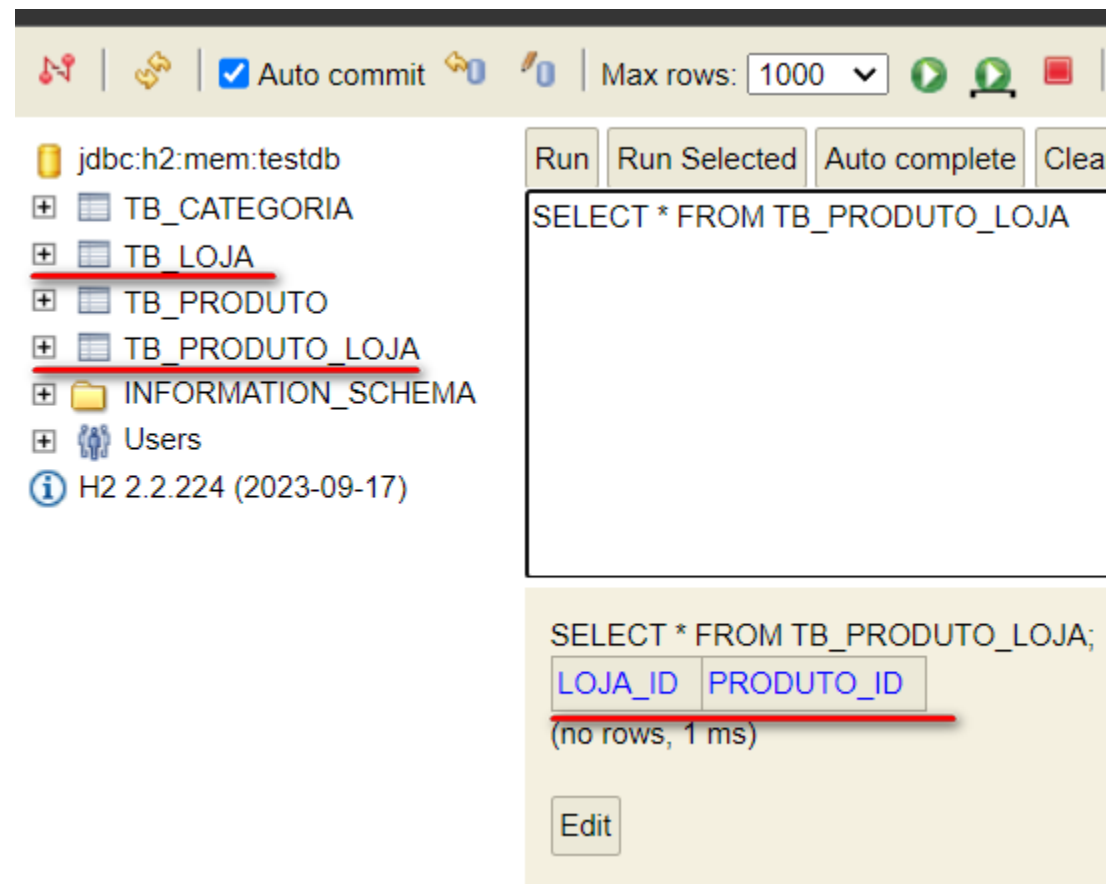
Entendendo o relacionamento

```
Produto.java
44
45 //relacionamento
46 @ManyToMany(fetch = FetchType.EAGER)
47 @JoinTable(name = "tb_produto_loja", //tabela associativa
48     joinColumns = @JoinColumn(name = "produto_id", //ref.
49     inverseJoinColumns = @JoinColumn(name = "loja_id")) //r
50 private Set<Loja> lojas = new HashSet<>(); // não permite valor
51

Loja.java
29 private String nome;
30
31 //relacionamento
32 //mapeamento na outra tabela
33 @ManyToMany(mappedBy = "lojas", fetch = FetchType.EAGER)
34 private Set<Produto> produtos = new HashSet<>(); // não permit
35
```


Executar aplicação

```
Run: ProdutoMvcApplication x
Hibernate:
    create table tb_loja (
        id bigint generated by default as identity,
        nome varchar(255),
        primary key (id)
    )
Hibernate:
    create table tb_produto (
        valor float(53) not null,
        categoria_id bigint not null,
        id bigint generated by default as identity,
        descricao TEXT,
        nome varchar(255),
        primary key (id)
    )
Hibernate:
    create table tb_produto_loja (
        loja_id bigint not null,
        produto_id bigint not null,
        primary key (loja_id, produto_id)
    )
Hibernate:
```



Atualizar import.sql

```
import.sql x
1  INSERT INTO tb_categoria(nome) VALUES('Smartphone');
2  INSERT INTO tb_categoria(nome) VALUES('Smart TV');
3  INSERT INTO tb_categoria(nome) VALUES('Notebook');
4  INSERT INTO tb_categoria(nome) VALUES('Tablet');
5  INSERT INTO tb_categoria(nome) VALUES('Mouse');
6  INSERT INTO tb_categoria(nome) VALUES('Teclado');
7
8  INSERT INTO tb_loja (nome) VALUES ('Americanas');
9  INSERT INTO tb_loja (nome) VALUES ('Fast Shop');
10 INSERT INTO tb_loja (nome) VALUES ('Magazine Luiza');
11 INSERT INTO tb_loja (nome) VALUES ('Submarino');
12
13 INSERT INTO tb_produto(nome, descricao, valor, categoria_id) VALUES('Mouse Microsoft', 'Mouse sem fio', 250.0, 5);
14 INSERT INTO tb_produto(nome, descricao, valor, categoria_id) VALUES('Smartphone Samsung Galaxy A54 5G', 'Samsung Galaxy A54 5G', 1799.0, 1);
15 INSERT INTO tb_produto(nome, descricao, valor, categoria_id) VALUES('Smart TV', 'Smart TV LG LED 65 polegadas', 3999, 2);
16 INSERT INTO tb_produto(nome, descricao, valor, categoria_id) VALUES('Teclado Microsof', 'Teclado sem fio', 278.50, 6);
17 INSERT INTO tb_produto(nome, descricao, valor, categoria_id) VALUES('Apple iPhone 15', 'Apple iPhone 15, 128G, Preto', 4999.00, 1);
18
19 INSERT INTO tb_produto_loja (produto_id, loja_id) VALUES (1, 1);
20 INSERT INTO tb_produto_loja (produto_id, loja_id) VALUES (1, 3);
21 INSERT INTO tb_produto_loja (produto_id, loja_id) VALUES (2, 1);
22 INSERT INTO tb_produto_loja (produto_id, loja_id) VALUES (2, 4);
23 INSERT INTO tb_produto_loja (produto_id, loja_id) VALUES (3, 2);
24 INSERT INTO tb_produto_loja (produto_id, loja_id) VALUES (4, 2);
25 INSERT INTO tb_produto_loja (produto_id, loja_id) VALUES (5, 2);
```

AULA 21 – Hibernate JPA Relacionamento Many to Many – parte 1

Executar aplicação

The screenshot shows the H2 database console interface. The left sidebar displays the database structure: jdbc:h2:mem:testdb, TB_CATEGORIA, TB_LOJA, TB_PRODUTO, TB_PRODUTO_LOJA, INFORMATION_SCHEMA, and Users. The main area contains the query `SELECT * FROM TB_LOJA;` and the results table below it.

ID	NOME
1	Americanas
2	Fast Shop
3	Magazine Luiza
4	Submarino

(4 rows, 2 ms)

The screenshot shows the H2 database console interface. The left sidebar displays the database structure: jdbc:h2:mem:testdb, TB_CATEGORIA, TB_LOJA, TB_PRODUTO, TB_PRODUTO_LOJA, INFORMATION_SCHEMA, and Users. The main area contains the query `SELECT * FROM TB_PRODUTO_LOJA;` and the results table below it.

LOJA_ID	PRODUTO_ID
1	1
1	2
2	3
2	4
2	5
3	1
4	2

(7 rows, 1 ms)

Adicionar LojaRepository

```
package br.com.fiap.produtomvc.repository;  
  
import br.com.fiap.produtomvc.models.Loja;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface LojaRepository extends JpaRepository<Loja, Long> {  
}
```

Adicionar LojaService

```
LojaService.java x
11
12 @Service
13 public class LojaService {
14
15     @Autowired
16     private LojaRepository repository;
17
18     @Transactional(readOnly = true)
19     public List<Loja> findAll(){
20         return repository.findAll();
21     }
22
23     @Transactional(readOnly = true)
24     public Loja findById(Long id){
25         Loja entity = repository.findById(id).orElseThrow(
26             () -> new IllegalArgumentException("Recurso não encontrado")
27         );
28         return entity;
29     }
}
```

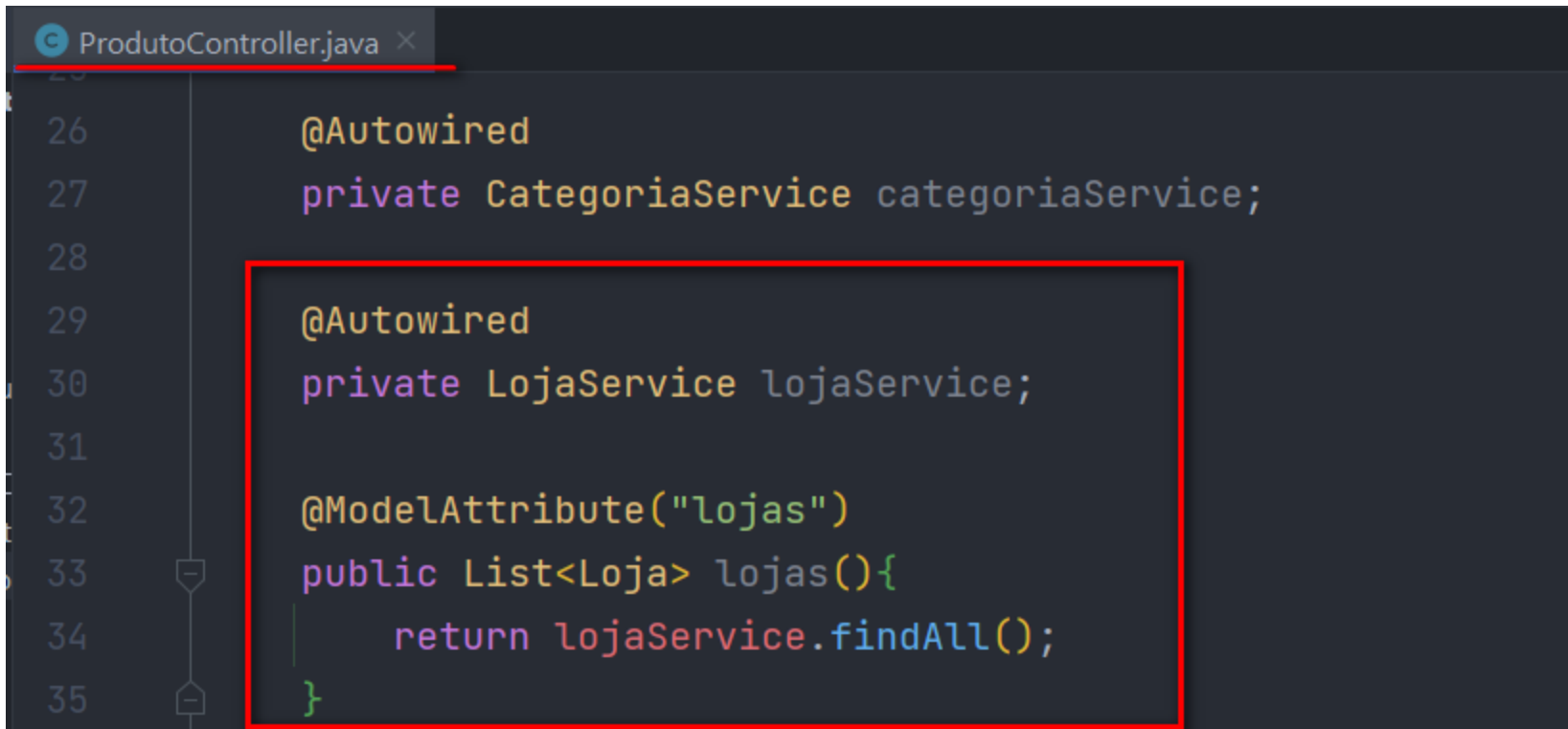
Adicionar LojaService

```
LojaService.java x
32 @ public Loja update(Long id, Loja entity){
33
34     try{
35         Loja loja = repository.getReferenceById(id);
36         loja.setNome(entity.getNome());
37         loja = repository.save(loja);
38         return loja;
39     } catch (EntityNotFoundException e){
40         throw new IllegalArgumentException("Recurso não encontrado");
41     }
42 }
43
44 @Transactional
45 public void delete(Long id){
46     if (!repository.existsById(id)) {
47         throw new IllegalArgumentException("Recurso não encontrado");
48     }
49
50     try{
51         repository.deleteById(id);
52     } catch (Exception e){
53         throw new IllegalArgumentException("Recurso não encontrado");
54     }
55 }
```

Alterar ProdutoService

```
ProdutoService.java x
66
67 @ private void copyToProduto(Produto entity, Produto produto) {
68     produto.setNome(entity.getNome());
69     produto.setDescricao(entity.getDescricao());
70     produto.setValor(entity.getValor());
71     produto.setCategoria(entity.getCategoria());
72
73     produto.getLojas().clear();
74     for (Loja loja : entity.getLojas()) {
75         Loja loja1 = new Loja();
76         loja1.setId(loja.getId());
77         produto.getLojas().add(loja1);
78     }
79 }
```

■ Incluir em ProdutoController



```
ProdutoController.java x
26      @Autowired
27      private CategoriaService categoriaService;
28
29      @Autowired
30      private LojaService lojaService;
31
32      @ModelAttribute("lojas")
33      public List<Loja> lojas(){
34          return lojaService.findAll();
35      }
```


Adicionar em ProdutoController

```
ProdutoController.java x
27     private CategoriaService categoriaService;
28
29     @Autowired
30     private LojaService lojaService;
31
32     @ModelAttribute("lojas")
33     public List<Loja> lojas(){
34         return lojaService.findAll();
35     }
36
37     @ModelAttribute("categorias")
38     public List<Categoria> categorias() {
39         return categoriaService.findAll();
40     }
```

Front-end

Editar View novo-produto.html

```
novo-produto.html x
61      </div>
62    </div>
63
64    <!--adicionando e populando ComboBox Loja-->
65    <div class="form-row">
66      <div class="row mb-3">
67        <label class="col-sm-1 col-form-label">Loja:</label>
68        <div class="col-sm-3">
69          <select class="chosen-select form-control" data-placeholder="lojas" id="lojas" multiple="true"
70            name="lojas" th:field="${produto.lojas}">
71            <option th:each="loja :${lojas}"
72              th:text="${loja.nome}"
73              th:value="${loja.id}"/>
74          </select>
75        </div>
76        <div class="col-sm-5">
77          <span class="mensagem" th:errors="*{lojas}"
78            th:if="${#fields.hasErrors('lojas')}"></span>
79        </div>
80      </div>
81    </div>
```

Testar aplicação

FIAP Home Categoria ▼ Produto ▼

Cadastro de Produtos

Nome:

Descrição:

Categoria:

Loja:

Americanas
Fast Shop
Magazine Luiza
Submarino

Valor:

Salvar

FIAP Home Categoria ▼ Produto ▼

Cadastro de Produtos

Nome:

Descrição:

Categoria:

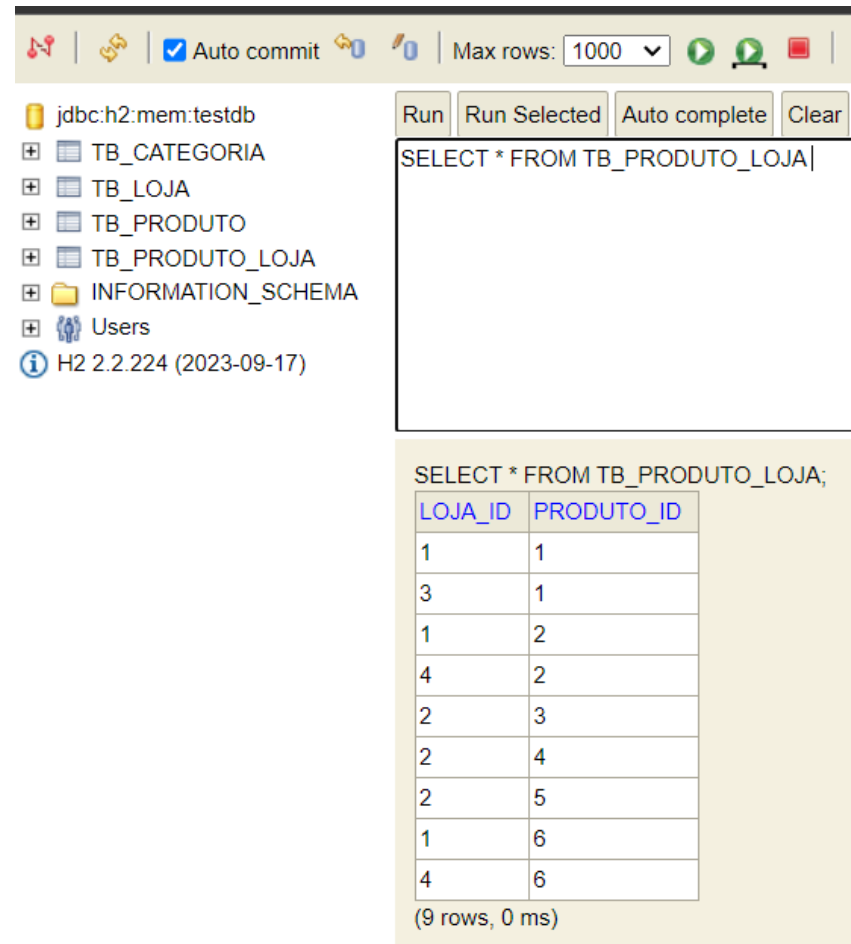
Loja:

Americanas
Fast Shop
Magazine Luiza
Submarino

Valor:

Salvar

Verificar DB



The screenshot shows a database client interface with a toolbar at the top containing icons for undo, redo, auto-commit, and other functions. Below the toolbar, the left sidebar displays a tree view of the database structure, including tables like TB_CATEGORIA, TB_LOJA, TB_PRODUTO, and TB_PRODUTO_LOJA, as well as an INFORMATION_SCHEMA and Users. The main area on the right contains a text input field with the SQL query `SELECT * FROM TB_PRODUTO_LOJA;` and buttons for `Run`, `Run Selected`, `Auto complete`, and `Clear`. Below the query input, the results of the query are displayed in a table format, showing 9 rows with columns `LOJA_ID` and `PRODUTO_ID`.

LOJA_ID	PRODUTO_ID
1	1
3	1
1	2
4	2
2	3
2	4
2	5
1	6
4	6

(9 rows, 0 ms)

■ Editar View editar-produto.html

```
Project
editar-produto.html x
58 </div>
59 </div>
60
61 <!--adicionando e populando ComboBox Loja-->
62 <div class="form-row">
63     <div class="row mb-3">
64         <label class="col-sm-1 col-form-label">Lojas:</label>
65         <div class="col-sm-3">
66             <select class="chosen-select form-control" data-placeholder="lojas" id="lojas" multiple="true"
67                 name="lojas" th:field="${produto.lojas}">
68                 <option th:each="loja :${lojas}"
69                     th:text="${loja.nome}"
70                     th:value="${loja.id}"/>
71             </select>
72         </div>
73         <div class="col-sm-5">
74             <span class="mensagem" th:errors="*{lojas}"
75                 th:if="${#fields.hasErrors('lojas')}"></span>
76         </div>
77     </div>
78 </div>
```

Testar aplicação

Alteração de Produto

Nome: Apple iPhone 15

Descrição: Apple iPhone 15, 128G, Preto

Categoria: Smartphone

Lojas: Americanas
Fast Shop
Magazine Luiza
Submarino

Valor: 4999.0

Alterar

Alteração de Produto

Nome: Mouse Microsoft

Descrição: Mouse sem fio

Categoria: Mouse

Lojas: Americanas
Fast Shop
Magazine Luiza
Submarino

Valor: 250.0

Alterar



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).