

FIAP GRADUAÇÃO

SISTEMAS DE INFORMAÇÃO

MICROSERVICE AND WEB ENGINEERING

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

■ Agenda

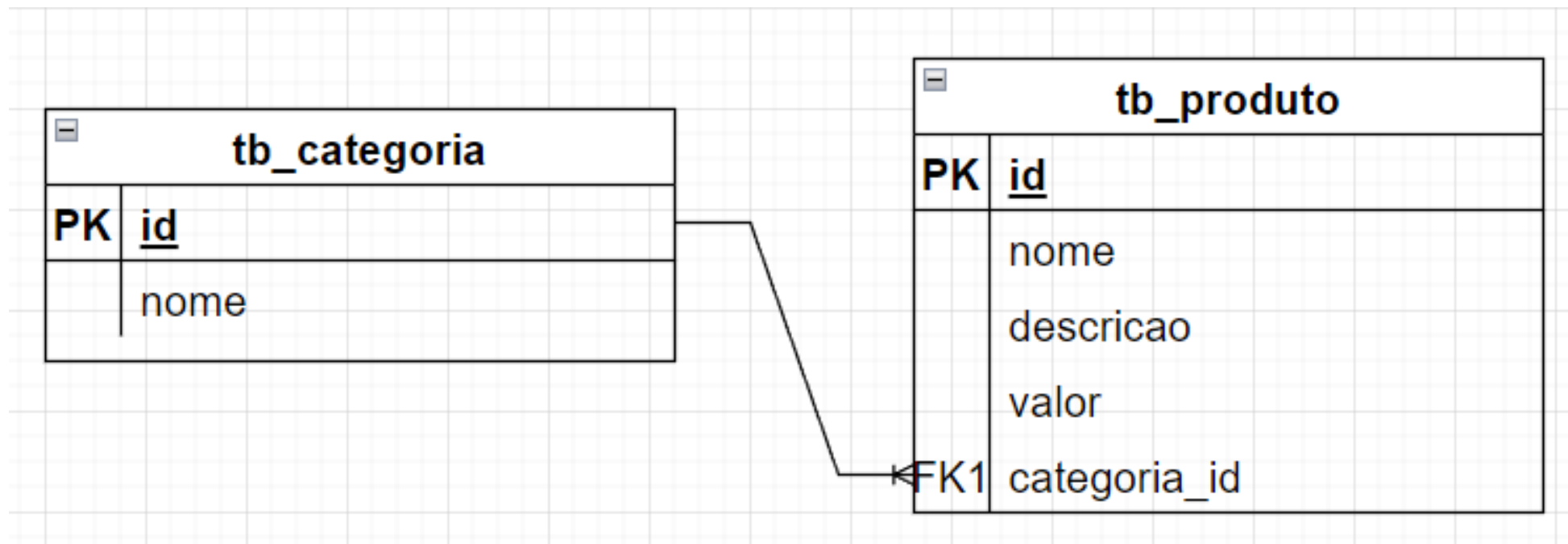
- Continuar com o projeto produto-mvc
- Relacionamento OneToMany

Objetivos

- Continuar com o projeto produto-mvc
- Entender e implementar relacionamento OneToMany de produto e categoria.

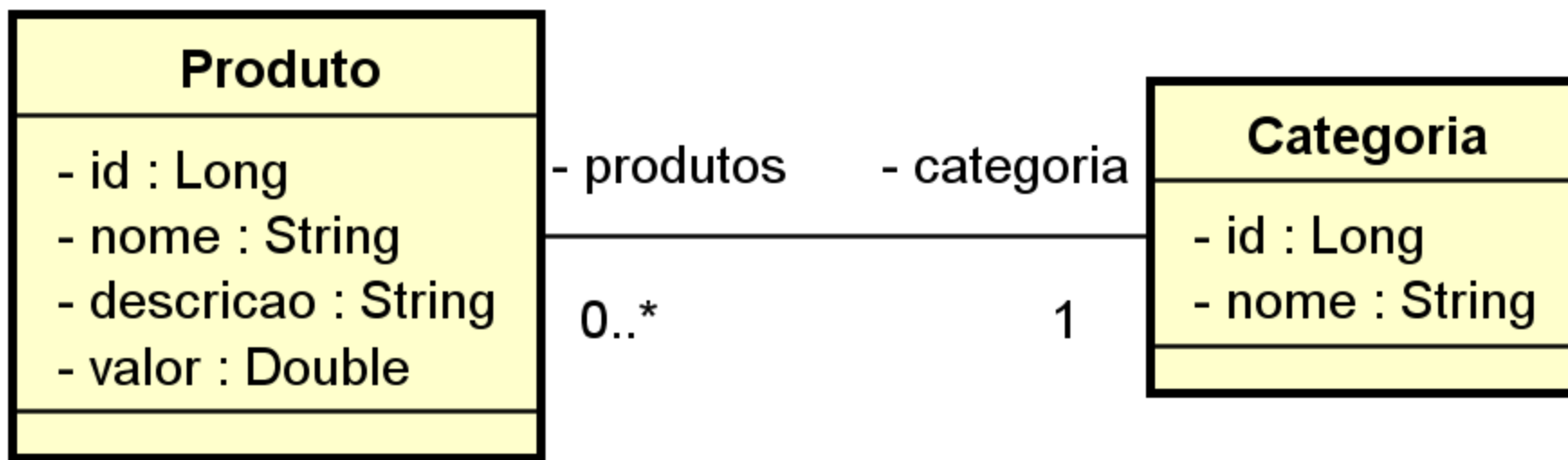
Hibernate JPA Relacionamento

- Modelo de Entidade Relacionamento



Hibernate JPA Relacionamento

- Diagrama de classes - atualizando a modelagem.



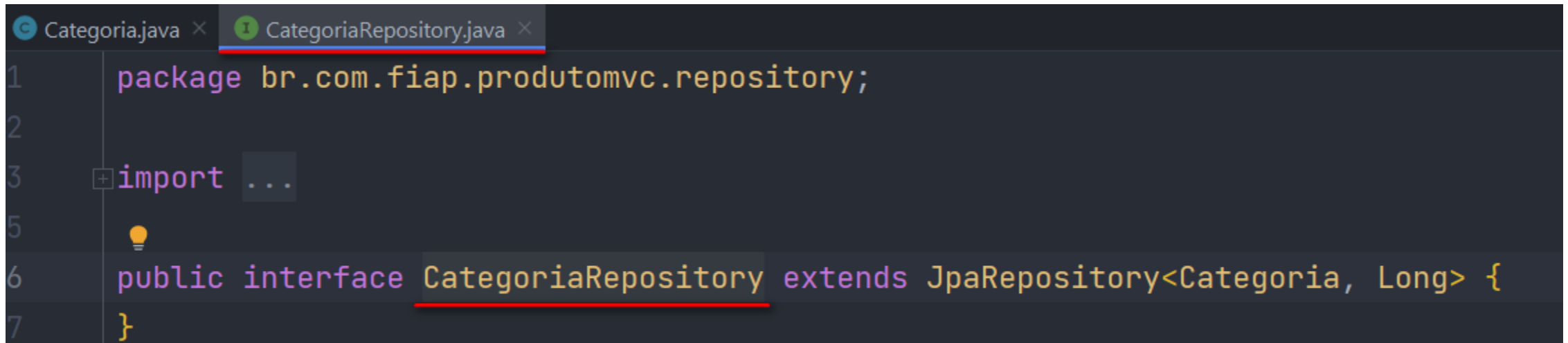
Back-end

O que foi implementado na aula anterior

Entity Categoria

```
Categoria.java x
13  @Entity
14  @Table(name = "tb_categoria")
15  public class Categoria {
16
17      @Id
18      @GeneratedValue(strategy = GenerationType.IDENTITY)
19      private Long id;
20
21      @NotBlank(message = "Campo requerido")
22      @Size(min = 3, message = "O nome deve ter no mínimo 3 caracteres")
23      private String nome;
24      //construtores, getters and setters
25      //equal and hashCode, toString
26      //código omitido
```


Interface CategoriaRepository



```
1 package br.com.fiap.produtomvc.repository;
2
3 import ...
4
5
6 public interface CategoriaRepository extends JpaRepository<Categoria, Long> {
7 }
```

Seed DB

```
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_produto(nome, descricao, valor) VALUES('Mouse Microsoft', 'Mouse sem fio', 250.0);
9 INSERT INTO tb_produto(nome, descricao, valor) VALUES('Smartphone Samsung Galaxy A54 5G', 'Samsung Galaxy A54 5G', 1799.0);
10 INSERT INTO tb_produto(nome, descricao, valor) VALUES('Smart TV', 'Smart TV LG LED 65 polegadas', 3999);
```

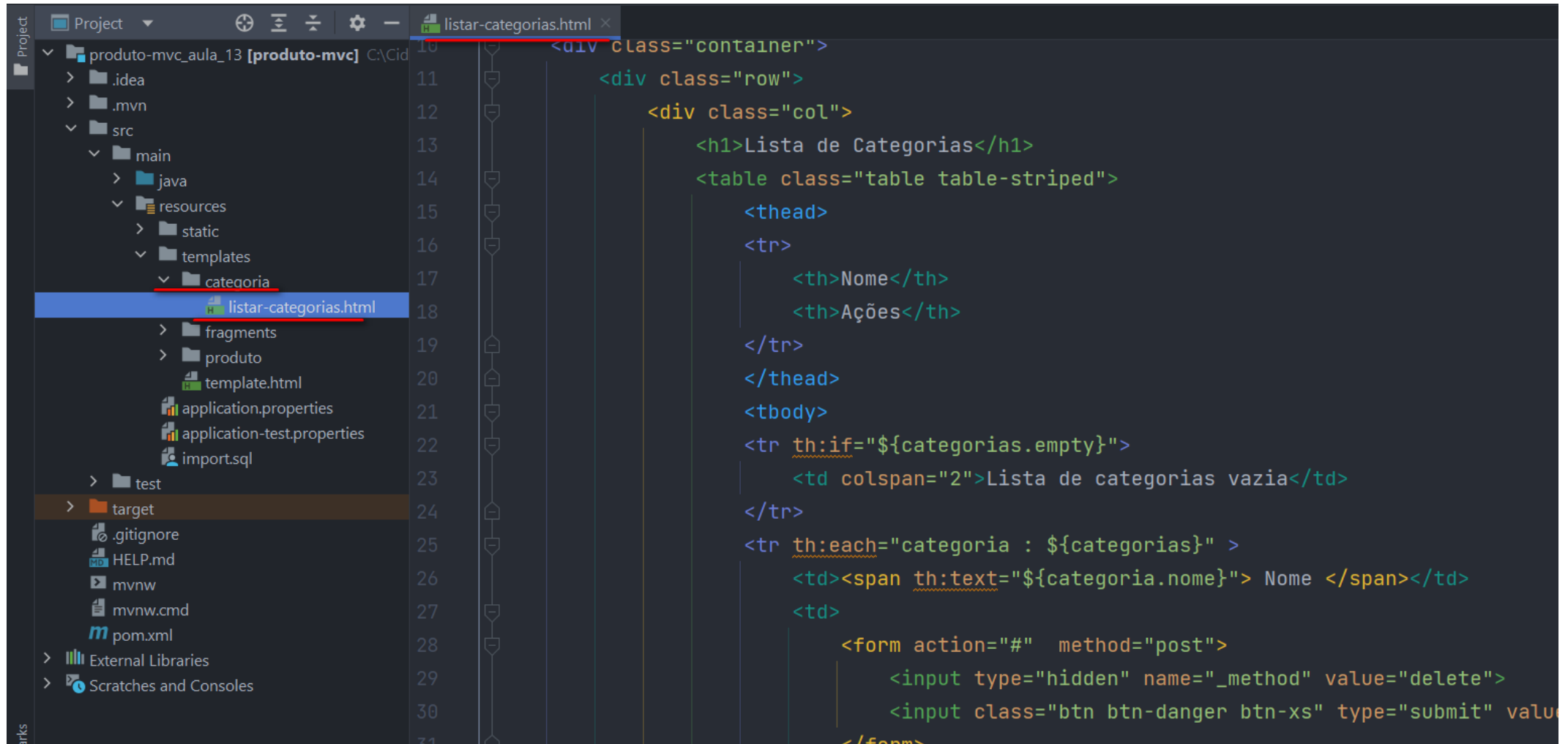
Controller CategoriaController

```
import.sql x CategoriaController.java x
10 @Controller
11 @RequestMapping("/categorias")
12 public class CategoriaController {
13
14     @Autowired
15     private CategoriaRepository repository;
16
17     @GetMapping()
18     @ public String findAll(Model model){
19         model.addAttribute("categorias", repository.findAll());
20         return "/categoria/listar-categorias";
21     }
22 }
```

Front-end

O que foi implementado na aula anterior

View listar-categorias.html



The screenshot shows an IDE with a project named 'produto-mvc_aula_13'. The file explorer on the left shows the project structure, with 'listar-categorias.html' selected under the 'categoria' folder. The main editor displays the HTML code for this file, which includes a container div, a row div, a column div, an h1 title, a table with a header and body, and a form for deleting categories.

```
10 <div class="container">
11   <div class="row">
12     <div class="col">
13       <h1>Lista de Categorias</h1>
14       <table class="table table-striped">
15         <thead>
16           <tr>
17             <th>Nome</th>
18             <th>Ações</th>
19           </tr>
20         </thead>
21         <tbody>
22           <tr th:if="${categorias.empty}">
23             <td colspan="2">Lista de categorias vazia</td>
24           </tr>
25           <tr th:each="categoria : ${categorias}" >
26             <td><span th:text="${categoria.nome}"> Nome </span></td>
27             <td>
28               <form action="#" method="post">
29                 <input type="hidden" name="_method" value="delete">
30                 <input class="btn btn-danger btn-xs" type="submit" value="Excluir">
31               </form>

```

Atualização navbar

```
listar-categorias.html x navbar.html x
12         aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navig
13         <span class="navbar-toggler-icon"></span>
14     </button>
15     <div class="collapse navbar-collapse" id="navbarNav">
16         <ul class="navbar-nav">
17             <li class="nav-item">
18                 <a class="nav-link active" aria-current="page" href="/">Home</a>
19             </li>
20             <li class="nav-item">
21                 <a class="nav-link" href="/produtos/form">Cadastrar</a>
22             </li>
23             <li class="nav-item">
24                 <a class="nav-link" href="/produtos">Produtos</a>
25             </li>
26             <li class="nav-item">
27                 <a class="nav-link" href="/categorias">Categorias</a>
28             </li>
```

Mapeamento Objeto Relacional

ORM

Mapeamento Objeto Relacional

Existem três tipos de relacionamento entre entidades no banco de dados.

Um para Um (1:1) - Um relacionamento entre duas entidades em que cada entidade tem exatamente um registro correspondente na outra entidade.

Um para Muitos (1:N) - Um relacionamento entre duas entidades em que uma entidade pode ter vários registros correspondentes na outra entidade.

Muitos para Muitos (N:N) - Um relacionamento entre duas entidades em que cada entidade pode ter vários registros correspondentes na outra entidade.

Mapeamento Objeto Relacional

Anotação	Objetivo
@OneToOne	Esta anotação é usada para especificar um relacionamento unidirecional entre duas entidades, onde uma entidade tem uma referência para outra entidade.
@OneToMany	Esta anotação é usada para especificar um relacionamento unidirecional de um para muitos entre duas entidades, onde uma entidade tem uma referência para várias entidades. Por exemplo, um professor pode ter vários alunos. O registro de uma entidade está relacionado com vários registros de outra entidade. O mappedBy fica dentro do @OneToMany e é usado no lado fraco do relacionamento e faz com que o relacionamento seja bidirecional.
@ManyToOne	Esta anotação é usada para especificar um relacionamento unidirecional de muitos para um entre duas entidades, onde várias entidades têm uma referência para uma entidade. Por exemplo, vários alunos podem ter o mesmo professor. (vários registros de uma entidade estão relacionados com um registro de outra entidade)
@ManyToMany	Esta anotação é usada para especificar um relacionamento bidirecional de muitos para muitos entre duas entidades, onde várias entidades têm referências para várias entidades. Por exemplo, vários alunos podem ter vários professores.

■ Mapeamento Objeto Relacional

Um relacionamento **unidirecional** indica que apenas uma das entidades possui referência para a outra. Por exemplo, um relacionamento @OneToMany unidirecional indica que apenas a entidade de origem possui referência à entidade de destino, enquanto a entidade de destino não possui referência à entidade de origem.

Um relacionamento **bidirecional** indica que ambas as entidades possuem referências à outra. Por exemplo, um relacionamento @OneToOne bidirecional indica que ambas as entidades possuem referências à outra e que qualquer alteração em uma entidade afeta a outra.

Implementando Relacionamento

Produto x Categoria
Back-end

Atualizar Class Produto

```
11 // código omitido
12 @Entity
13 @Table(name = "tb_produto")
14 public class Produto {
15
16     @ {...}
18     private Long id;
19
20     @ {...}
22     private String nome;
23
24     @ {...} //para textos longos
26     private String descricao;
27
28     @ {...}
30     private Double valor;
31
32     //Relacionamento
33     @ManyToOne
34     @JoinColumn(name = "categoria_id", nullable = false) //PK
35     private Categoria categoria;
36 }
```

@ManyToOne

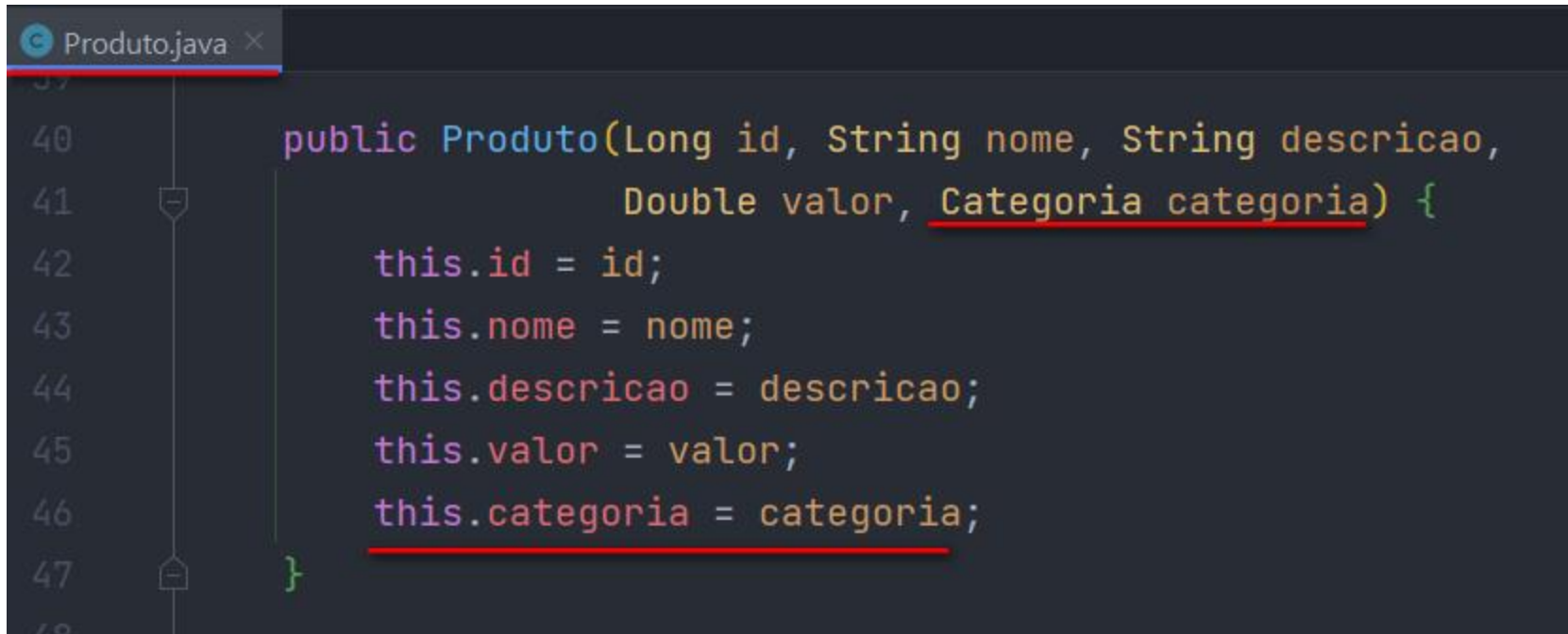
Indica o relacionamento N:1

@JoinColumn

Mapeia a chave estrangeira (FK) e indica o lado forte do relacionamento

- Se o relacionamento termina com **ToOne**, só há uma **única** entidade relacionada e utilizamos `@JoinColumn`.

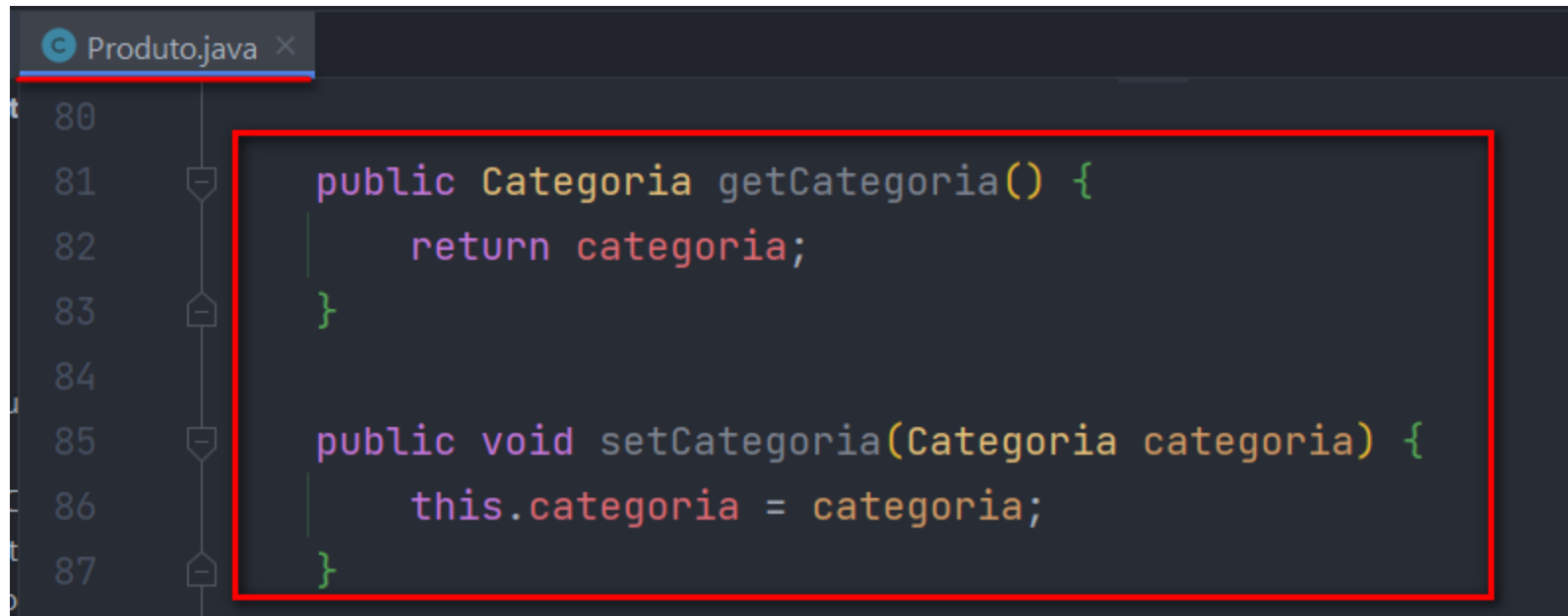
Class Produto – atualizar construtor



The screenshot shows a code editor with a file named 'Produto.java'. The code defines a constructor for the 'Produto' class. The constructor signature is 'public Produto(Long id, String nome, String descricao, Double valor, Categoria categoria)'. The body of the constructor contains five assignment statements: 'this.id = id;', 'this.nome = nome;', 'this.descricao = descricao;', 'this.valor = valor;', and 'this.categoria = categoria;'. The line 'this.categoria = categoria;' is underlined in red. The line numbers 40, 41, 42, 43, 44, 45, 46, 47, and 48 are visible on the left side of the editor.

```
40     public Produto(Long id, String nome, String descricao,  
41                     Double valor, Categoria categoria) {  
42         this.id = id;  
43         this.nome = nome;  
44         this.descricao = descricao;  
45         this.valor = valor;  
46         this.categoria = categoria;  
47     }  
48
```

■ Class Produto – adicionar get e set



```
80  
81 public Categoria getCategoria() {  
82     return categoria;  
83 }  
84  
85 public void setCategoria(Categoria categoria) {  
86     this.categoria = categoria;  
87 }
```

Atualizar Class Categoria

```
Categoria.java x
10
11 @Entity
12 @Table(name = "tb_categoria")
13 public class Categoria {
14
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private Long id;
18
19     @NotBlank(message = "Campo requerido")
20     @Size(min = 3, message = "O nome deve ter no mínimo 3 caracteres")
21     private String nome;
22
23     //Relacionamento
24     @OneToMany(mappedBy = "categoria")
25     private List<Produto> produtos = new ArrayList<>();
26 }
```

@OneToMany

Indica o relacionamento 1:N

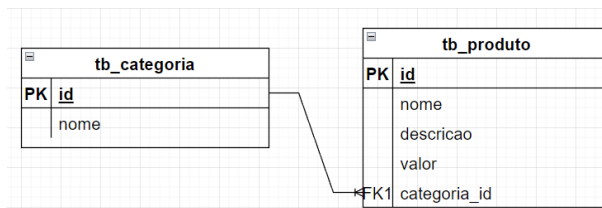
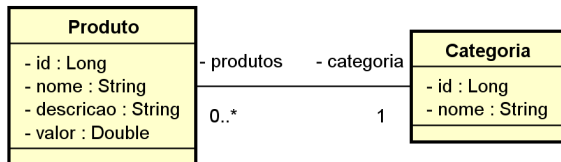
- Se o relacionamento termina com **ToMany**, então você tem uma **lista** de entidades relacionadas e utiliza o `mappedBy`.

■ Class Categoria – adicionar get



```
Categoria.java x
47      public void setNome(String nome) { this.nome = nome; }
50
51      public List<Produto> getProdutos() {
52          return produtos;
53      }
```


Categoria x Produto



```

10
11 @Entity
12 @Table(name = "tb_categoria")
13 public class Categoria {
14
15     @{...}
16     private Long id;
17
18     @{...}
19     private String nome;
20
21     //Relacionamento
22     @OneToMany(mappedBy = "categoria")
23     private List<Produto> produtos = new ArrayList<>();
  
```

```

11 // código omitido
12 @Entity
13 @Table(name = "tb_produto")
14 public class Produto {
15
16     @{...}
17     private Long id;
18
19     @{...}
20     private String nome;
21
22     @{...} //para textos longos
23     private String descricao;
24
25     @{...}
26     private Double valor;
27
28     //Relacionamento
29     @ManyToOne
30     @JoinColumn(name = "categoria_id", nullable = false) //PK
31     private Categoria categoria;
32
33
34
35
36
  
```

Atualizar Seed DB

```
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_produto(nome, descricao, valor, categoria_id) VALUES('Mouse Microsoft', 'Mouse sem fio', 250.0, 5);
9 INSERT INTO tb_produto(nome, descricao, valor, categoria_id) VALUES('Smartphone Samsung Galaxy A54 5G', 'Samsung Galaxy A54 5G', 1799.0, 1);
10 INSERT INTO tb_produto(nome, descricao, valor, categoria_id) VALUES('Smart TV', 'Smart TV LG LED 65 polegadas', 3999, 2);
```

Testar H2

jdbc:h2:mem:testdb

Run Run Selected Auto complete Clear

⊕ TB_CATEGORIA
⊕ TB_PRODUTO
⊕ INFORMATION_SCHEMA
⊕ Users
H2 2.2.224 (2023-09-17)

```
SELECT * FROM TB_CATEGORIA|
```

```
SELECT * FROM TB_CATEGORIA;
```

ID	NOME
1	Smartphone
2	Smart TV
3	Notebook
4	Tablet
5	Mouse
6	Teclado

(6 rows, 0 ms)

jdbc:h2:mem:testdb

Run Run Selected Auto complete Clear SQL statement:

⊕ TB_CATEGORIA
⊕ TB_PRODUTO
⊕ INFORMATION_SCHEMA
⊕ Users
H2 2.2.224 (2023-09-17)

```
SELECT * FROM TB_PRODUTO|
```

```
SELECT * FROM TB_PRODUTO;
```

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

(3 rows, 0 ms)

■ Mapeamento Objeto Relacional

@OneToMany	Indica o relacionamento 1:N
@ManyToOne	Indica o relacionamento N:1
@JoinColumn	Mapeia a chave estrangeira (FK) e indica o lado forte do relacionamento
mappedBy	Indica o lado fraco do relacionamento e torna-o Bidirecional

Para mapear relacionamento **Bidirecionais**, siga essa regra:

- Se o relacionamento termina com **ToMany**, então você tem uma **lista** de entidades relacionadas e utiliza o `mappedBy`.
- Se o relacionamento termina com **ToOne**, só há uma **única** entidade relacionada e utilizamos `@JoinColumn`.

Mapeamento Objeto Relacional

Primitive Types

Mapping type	Java type	ANSI SQL Type
integer	int or java.lang.Integer	INTEGER
long	long or java.lang.Long	BIGINT
short	short or java.lang.Short	SMALLINT
float	float or java.lang.Float	FLOAT
double	double or java.lang.Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	java.lang.String	CHAR(1)
string	java.lang.String	VARCHAR
byte	byte or java.lang.Byte	TINYINT
boolean	boolean or java.lang.Boolean	BIT
yes/no	boolean or java.lang.Boolean	CHAR(1) ('Y' or 'N')
true/false	boolean or java.lang.Boolean	CHAR(1) ('T' or 'F')

Fonte: https://www.tutorialspoint.com/hibernate/hibernate_mapping_types.htm

■ Mapeamento Objeto Relacional

Date and Time Types

Mapping type	Java type	ANSI SQL Type
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
timestamp	java.util.Date or java.sql.Timestamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE

Fonte: https://www.tutorialspoint.com/hibernate/hibernate_mapping_types.htm

Mapeamento Objeto Relacional

Estratégia de geração da chave primária através do atributo `strategy` de `@GeneratedValue`.



A JPA suporta quatro estratégias, definidas no enum `GenerationType`:

Estratégia	Descrição
<code>GenerationType.AUTO</code>	Valor padrão, deixa com o provedor de persistência a escolha da estratégia mais adequada de acordo com o banco de dados.
<code>GenerationType.IDENTITY</code>	Informamos ao provedor de persistência que os valores a serem atribuídos ao identificador único serão gerados pela coluna de auto incremento do banco de dados. Assim, um valor para o identificador é gerado para cada registro inserido no banco. Alguns bancos de dados podem não suportar essa opção.
<code>GenerationType.SEQUENCE</code>	Informamos ao provedor de persistência que os valores serão gerados a partir de uma sequence. Caso não seja especificado um nome para a sequence, será utilizada uma sequence padrão, a qual será global, para todas as entidades. Caso uma sequence seja especificada, o provedor passará a adotar essa sequence para criação das chaves primárias. Alguns bancos de dados podem não suportar essa opção.
<code>GenerationType.TABLE</code>	Com a opção <code>TABLE</code> é necessário criar uma tabela para gerenciar as chaves primárias. Por causa da sobrecarga de consultas necessárias para manter a tabela atualizada, essa opção é pouco recomendada.

Fonte: <https://www.devmedia.com.br/jpa-como-usar-a-anotacao-generatedvalue/38592>

Consulte também: Entendendo a geração de chaves com JPA: <https://www.alura.com.br/artigos/entendendo-a-geracao-de-chaves-com-jpa>

Atualizando ProdutoController

Back-end

I Acrescentar em ProdutoController

```
ProdutoController.java x
17
18 @Controller
19 @RequestMapping("/produtos")
20 public class ProdutoController {
21
22     @Autowired
23     private ProdutoRepository repository;
24
25     @Autowired
26     private CategoriaRepository categoriaRepository;
27
28     // adicionando atributo categorias do model
29     // para popular ComboBox da View
30     @ModelAttribute("categorias")
31     public List<Categoria> categorias(){
32         return categoriaRepository.findAll();
33     }
}
```

Atualizações

Front-end

Atualizar View novo-produto.html

```
41      th:if="${#fields.hasErrors('descricao')}"></span>
42    </div>
43  </div>
44  <!--adicionando e populando ComboBox Categoria-->
45  <div class="form-row">
46    <div class="row mb-3">
47      <label class="col-sm-1 col-form-label">Categoria:</label>
48      <div class="col-sm-3">
49        <select class="chosen-select form-control" data-placeholder="categoria" id="categoria"
50          name="categoria" th:field="*{categoria}">
51          <option th:each="categoria : ${categorias}"
52            th:text="${categoria.nome}"
53            th:value="${categoria.id}">
54          </option>
55        </select>
56      </div>
57      <div class="col-sm-5">
58        <span class="mensagem" th:errors="*{categoria}"
59          th:if="${#fields.hasErrors('categoria')}"></span>
60      </div>
61    </div>
62  </div>
63
64  <div class="row mb-3">
65    <label class="col-sm-1 col-form-label" for="valor">Valor:</label>
66    <div class="col-sm-2">
```

Testar app

FIAP Home Cadastrar Produtos Categorias

Cadastro de Produtos


Nome:

Descrição:

Categoria:

Valor:

- Smartphone
- Smart TV
- Notebook
- Tablet
- Mouse
- Teclado



Atualizar View editar-produto.html

```
editar-produto.html x
38      <span class="mensagem" th:if="${#fields.hasErrors('descricao')}" th:errors="mensagem"
39    </div>
40  </div>
41  <!--adicionando e populando ComboBox Categoria-->
42  <div class="form-row">
43    <div class="row mb-3">
44      <label class="col-sm-1 col-form-label">Categoria:</label>
45      <div class="col-sm-3">
46        <select th:field="*{categoria}" id="categoria" name="categoria"
47              class="chosen-select form-control" data-placeholder="categoria">
48          <option th:each="categoria : ${categorias}"
49                th:value="${categoria.id}"
50                th:text="${categoria.nome}">
51            </option>
52        </select>
53      </div>
54      <div class="col-sm-5">
55        <span class="mensagem" th:errors="*{categoria}"
56              th:if="${#fields.hasErrors('categoria')}"></span>
57      </div>
58    </div>
59  </div>
60
61  <div class="row mb-3">
62    <label for="valor" class="col-sm-1 col-form-label">Valor:</label>
```

Atualizar View listar-produtos.html

```
listar-produtos.html x
13 <h1>Lista de Produtos</h1>
14 <table class="table table-striped">
15   <thead>
16     <tr>
17       <th>Nome</th>
18       <th>Descrição</th>
19       <th>Categoria</th>
20       <th>Valor</th>
21       <th>Ações</th>
22     </tr>
23   </thead>
24   <tbody>
25     <tr th:if="${produtos.empty}">
26       <td colspan="5">Lista de produtos vazia</td>
27     </tr>
28     <tr th:each="produto : ${produtos}" >
29       <td><span th:text="${produto.nome}"> Nome </span></td>
30       <td><span th:text="${produto.descricao}"> Descrição </span></td>
31       <td><span th:text="${produto.categoria.nome}"> Categoria </span></td>
32       <td><span th:text="${produto.valor}"> Valor </span></td>
```

Testar App

FIAP Home Cadastrar Produtos Categorias

Lista de Produtos

Nome	Descrição	Categoria	Valor	Ações
Mouse Microsoft	Mouse sem fio	Mouse	250.0	<button>Excluir</button> <button>Editar</button>
Smartphone Samsung Galaxy A54 5G	Samsung Galaxy A54 5G	Smartphone	1799.0	<button>Excluir</button> <button>Editar</button>
Smart TV	Smart TV LG LED 65 polegadas	Smart TV	3999.0	<button>Excluir</button> <button>Editar</button>

FIAP Home Cadastrar Produtos Categorias

Alteração de Produto

Nome:

Descrição:

Categoria:

Valor:

Alterar

- Smartphone
- Smart TV
- Notebook
- Tablet**
- Mouse
- Teclado

Lista de Produtos

Nome	Descrição	Categoria	Valor	Ações
Mouse Microsoft	Mouse sem fio	Mouse	250.0	<button>Excluir</button> <button>Editar</button>
Smartphone Samsung Galaxy A54 5G	Samsung Galaxy A54 5G	Smartphone	1799.0	<button>Excluir</button> <button>Editar</button>
Smart TV	Smart TV LG LED 65 polegadas	<u>Tablet</u>	3999.0	<button>Excluir</button> <button>Editar</button>

Implementar funcionalidades de Categorias

Exercício – Back-end

■ Implementar CRUD para Categoria

- Incluir categoria.
- Alterar categoria.
- Excluir categoria.

Implementar em CategoriaController

```
CategoriaController.java x listar-categorias.html x nova-categoria.html x novo-produto.html x
16 @RequestMapping("/categorias")
17 public class CategoriaController {
18
19     @Autowired
20     private CategoriaRepository repository;
21
22     @GetMapping("/form")
23     public String loadFormCategoria(Model model) {
24         model.addAttribute("categoria", new Categoria());
25         return "categoria/nova-categoria";
26     }
27
28     @PostMapping()
29     @Transactional
30     public String insert(@Valid Categoria categoria,
31                         BindingResult result,
32                         RedirectAttributes attributes) {
33         if (result.hasErrors()) {
34             return "categoria/nova-categoria";
35         }
36         repository.save(categoria);
37         attributes.addFlashAttribute("mensagem", "Categoria salva com sucesso!");
38         return "redirect:/categorias";
39     }
}
```

Implementar em CategoriaController

```
CategoriaController.java x listar-categorias.html x nova-categoria.html x novo-produto.html x
40
41 @GetMapping()
42 @Transactional(readOnly = true)
43 @ public String findAll(Model model) {
44     model.addAttribute("categorias", repository.findAll());
45     return "/categoria/listar-categorias";
46 }
47
48 @GetMapping("/{id}")
49 @Transactional(readOnly = true)
50 @ public String findById(@PathVariable("id") Long id, Model model) {
51
52     Categoria categoria = repository.findById(id).orElseThrow(
53         () -> new IllegalArgumentException("Recurso inválido - " + id)
54     );
55
56     model.addAttribute("categoria", categoria);
57     return "/categoria/editar-categoria";
58 }
```

Implementar em CategoriaController

```
CategoriaController.java x listar-categorias.html x nova-categoria.html x novo-produto.html x
59
60 @PutMapping("/{id}")
61 @ public String update(@PathVariable("id") Long id,
62                        @Valid Categoria categoria,
63                        BindingResult result) {
64
65     if (result.hasErrors()) {
66         categoria.setId(id);
67         return "/categoria/editar-categoria";
68     }
69     repository.save(categoria);
70     return "redirect:/categorias";
71 }
72
73 @DeleteMapping("/{id}")
74 public String delete(@PathVariable("id") Long id, Model model) {
75     if (!repository.existsById(id)) {
76         throw new IllegalArgumentException("Recurso inválido - id: " + id);
77     }
78     try {
79         repository.deleteById(id);
80     } catch (DataIntegrityViolationException e) {
81         throw new IllegalArgumentException("Falha de integridade referencial - id: " + id);
82     }
83     return "redirect:/categorias";
84 }
```

Implementação e atualização das Views

Front-end

Views

- Implementar nova-categoria.html
- Implementar editar-categoria.html

Atualizar listar-categorias.html

```
listar-categorias.html x
21 <tbody>
22 <tr th:if="${categorias.empty}">
23     <td colspan="2">Lista de categorias vazia</td>
24 </tr>
25 <tr th:each="categoria : ${categorias}" >
26     <td><span th:text="${categoria.nome}"> Nome </span></td>
27     <td>
28         <form th:action="@{/categorias/{id}(id=${categoria.id})}" th:object="${categoria}" method="post">
29             <input type="hidden" name="_method" value="delete">
30             <input class="btn btn-danger btn-xs" type="submit" value="Excluir"
31                 th:data-confirm-delete="|Deseja excluir ${categoria.nome}?|"
32                 onclick="if (!confirm(this.getAttribute('data-confirm-delete'))) return false"/>
33         </form>
34     </td>
35     <td>
36         <form action="#" th:action="@{/categorias/{id}(id=${categoria.id})}" th:object="${categoria}" method="get">
37             <input class="btn btn-primary btn-xs" type="submit" value="Editar">
38         </form>
39     </td>
40 </tr>
41 </tbody>
```

Atualizar navbar

```
template.html x navbar.html x
16 <ul class="navbar-nav">
17   <li class="nav-item">
18     <a class="nav-link active" aria-current="page" href="/">Home</a>
19   </li>
20   <li class="nav-item">
21     <a class="nav-link" href="/produtos/form">Cadastrar Produto</a>
22   </li>
23   <li class="nav-item">
24     <a class="nav-link" href="/categorias/form">Cadastrar Categorias</a>
25   </li>
26   <li class="nav-item">
27     <a class="nav-link" href="/produtos">Listar Produtos</a>
28   </li>
29
30   <li class="nav-item">
31     <a class="nav-link" href="/categorias">Listar Categorias</a>
32   </li>
```


Melhorando Layout da navbar

Front-end

Incluir em template.html

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
```

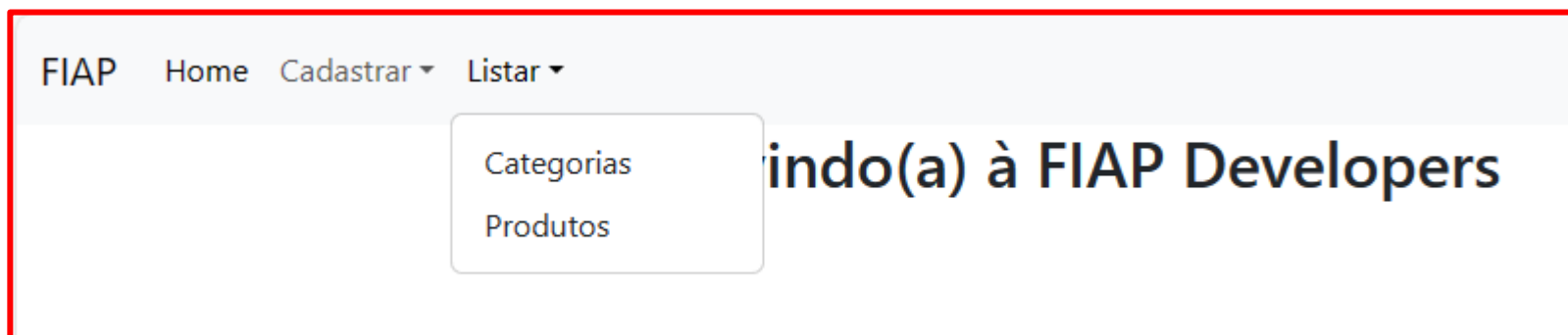
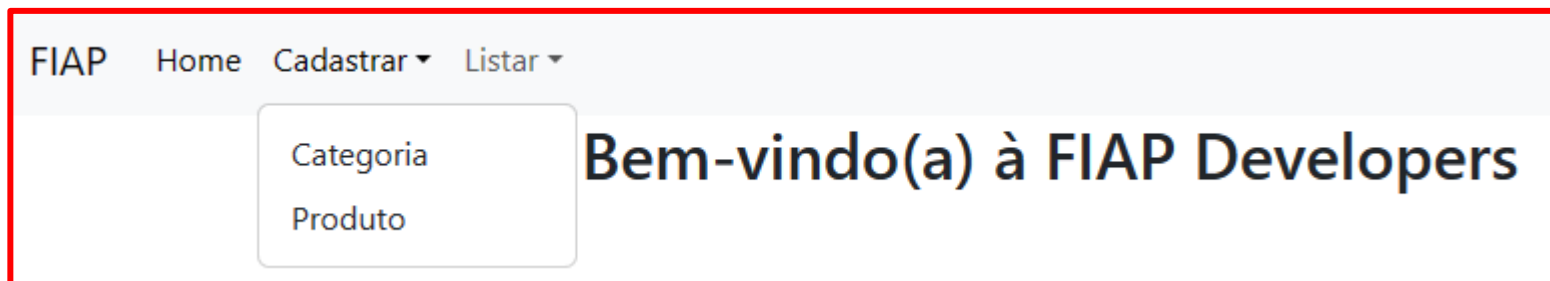


```
template.html x
9      <meta charset="UTF-8">
10     <!--      adicionando bootstrap e jquery-->
11     <!--      th - apontando para os arquivos-->
12     <link rel="stylesheet" th:href="@{/bootstrap-5.3.2-dist/css/bootstrap.min.css}">
13     <script th:src="@{/jquery-3.6.0-dist/jquery-3.6.0.min.js}" ></script>
14     <script th:ref="@{/bootstrap-5.3.2-dist/js/bootstrap.min.js}"></script>
15
16     <!--      incluir -->
17     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
18
```

Substituir os links na navbar

```
template.html x navbar.html x
32
33 <li class="nav-item dropdown">
34   <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Cadastrar</a>
35   <ul class="dropdown-menu">
36     <li><a class="dropdown-item" href="/categorias/form">Categoria</a></li>
37     <li><a class="dropdown-item" href="/produtos/form">Produto</a></li>
38   </ul>
39 </li>
40
41 <li class="nav-item dropdown">
42   <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Listar</a>
43   <ul class="dropdown-menu">
44     <li><a class="dropdown-item" href="/categorias">Categorias</a></li>
45     <li><a class="dropdown-item" href="/produtos">Produtos</a></li>
46   </ul>
47 </li>
```

Testar app





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).