

FIAP GRADUAÇÃO

SISTEMAS DE INFORMAÇÃO

MICROSERVICE AND WEB ENGINEERING

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

I Agenda

- Continuar com o projeto produto-mvc
- Lombok
- Hibernate JPA DDL
- Conectando com Oracle

Objetivos

- Continuar com o projeto produto-mvc
- Utilizar Lombok
- Entender como funciona a criação do modelo de dados e demais itens do banco (DDL) automaticamente pelo Hibernate.
- Conectar com o Banco de Dados Oracle.

navbar

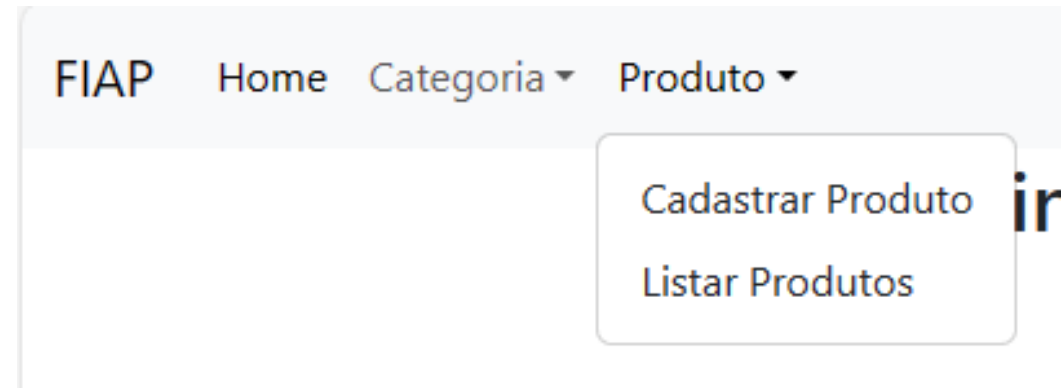
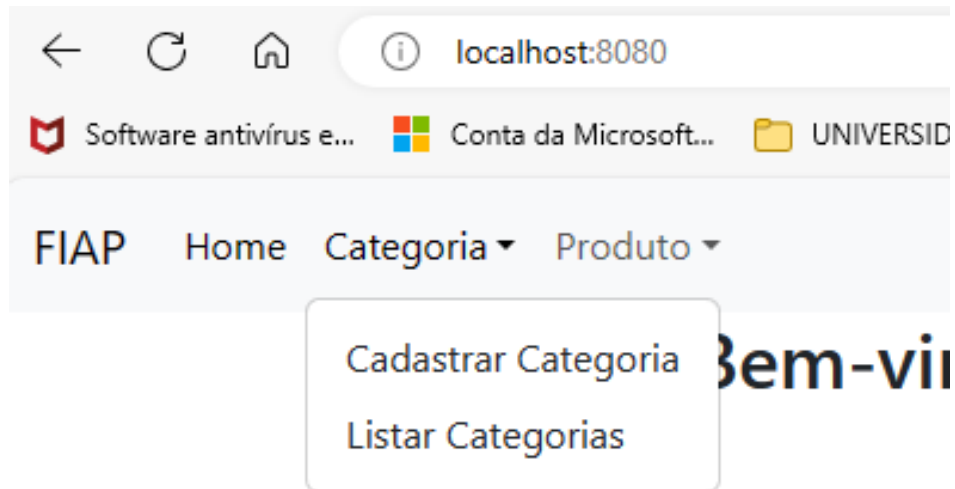
Ajustando

Refatorar navbar.html

```
navbar.html x
18 | <a class="nav-link active" aria-current="page" href="/">Home</a>
19 | </li>
20 |
21 | <li class="nav-item dropdown">
22 |   <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Categoria</a>
23 |   <ul class="dropdown-menu">
24 |     <li><a class="dropdown-item" href="/categorias/form">Cadastrar Categoria</a></li>
25 |     <li><a class="dropdown-item" href="/categorias">Listar Categorias</a></li>
26 |   </ul>
27 | </li>
28 |
29 | <li class="nav-item dropdown">
30 |   <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Produto</a>
31 |   <ul class="dropdown-menu">
32 |     <li><a class="dropdown-item" href="/produtos/form">Cadastrar Produto</a></li>
33 |     <li><a class="dropdown-item" href="/produtos">Listar Produtos</a></li>
34 |   </ul>
35 | </li>
```

I navbar

FIAP



Lombok

I Lombok - Instalar

- Precisa fazer a instalação do Lombok no STS e outras IDE's.
- Referências:
- https://www.youtube.com/watch?v=W0ywxkvc4_M
- <https://projectlombok.org/setup/eclipse>
- Lombok no IntelliJ
- <https://plugins.jetbrains.com/plugin/6317-lombok>

Lombok - Dependência



Project
☐ Gradle - Groovy
☐ Gradle - Kotlin
☒ **Maven**

Language
☒ **Java** ☐ Kotlin
☐ Groovy

Spring Boot
☐ 3.3.0 (SNAPSHOT) ☐ 3.3.0 (M3)
☐ 3.2.5 (SNAPSHOT) ☒ **3.2.4**
☐ 3.1.11 (SNAPSHOT) ☐ 3.1.10

Project Metadata

Dependencies ADD ... CTRL + B

Lombok **DEVELOPER TOOLS**
Java annotation library which helps to reduce boilerplate code.

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

Lombok - Dependência

demo.zip

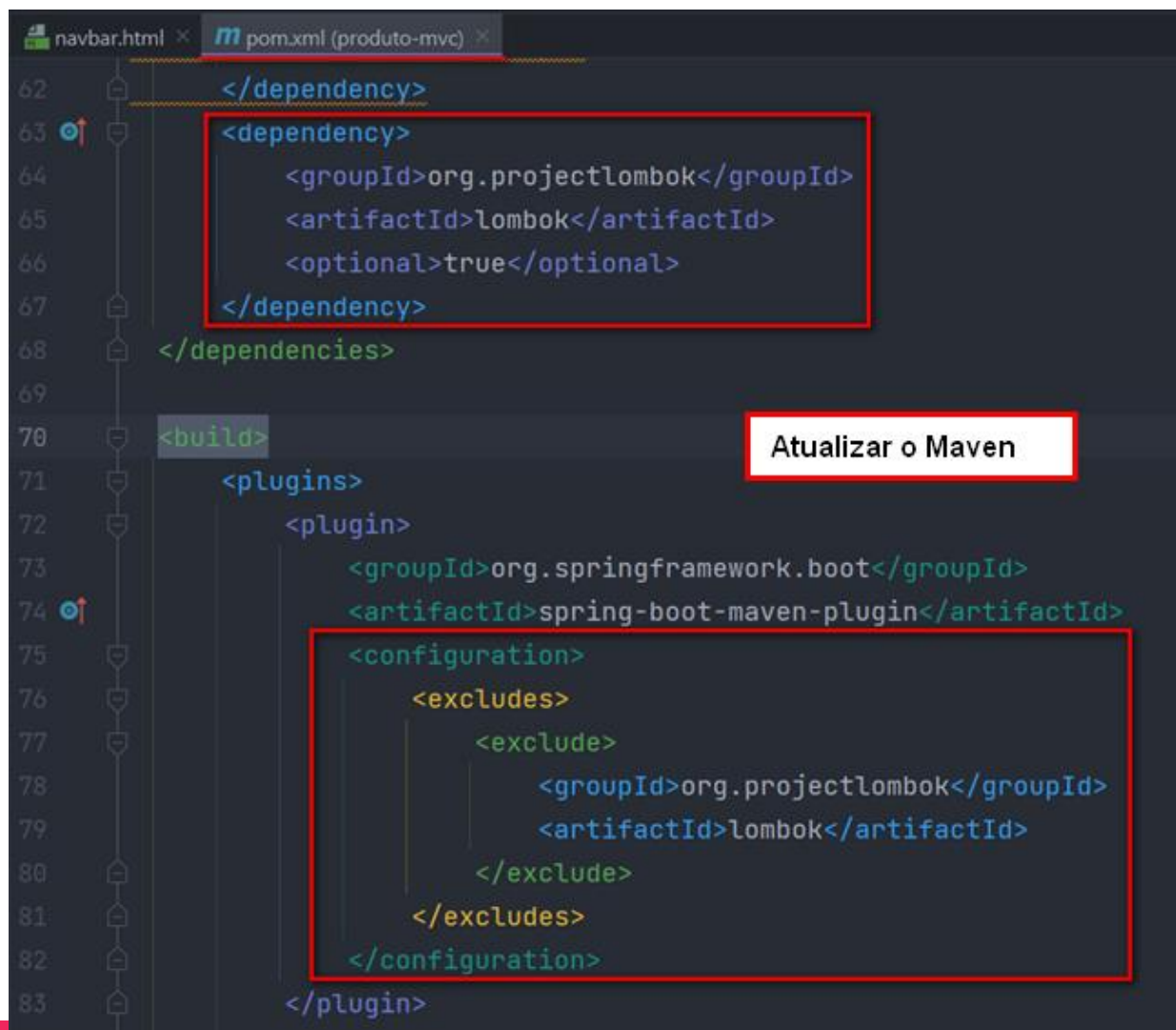
- .gitignore
- .mvn
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml**
- src

DOWNLOAD COPY

```
23 </dependency>
24
25 <dependency>
26   <groupId>org.projectlombok</groupId>
27   <artifactId>lombok</artifactId>
28   <optional>true</optional>
29 </dependency>
30 <dependency>
31   <groupId>org.springframework.boot</groupId>
32   <artifactId>spring-boot-starter-test</artifactId>
33   <scope>test</scope>
34 </dependency>
35 </dependencies>
36
37 <build>
38   <plugins>
39     <plugin>
40       <groupId>org.springframework.boot</groupId>
41       <artifactId>spring-boot-maven-plugin</artifactId>
42       <configuration>
43         <excludes>
44           <exclude>
45             <groupId>org.projectlombok</groupId>
46             <artifactId>lombok</artifactId>
47           </exclude>
48         </excludes>
49       </configuration>
50     </plugin>
```

copiar os códigos destacados e colar no pom.xml

Lombok – Dependência - pom.xml



The screenshot shows a code editor with a file named 'pom.xml (produto-mvc)'. The code is as follows:

```
62     </dependency>
63     <dependency>
64         <groupId>org.projectlombok</groupId>
65         <artifactId>lombok</artifactId>
66         <optional>true</optional>
67     </dependency>
68 </dependencies>
69
70 <build>
71     <plugins>
72         <plugin>
73             <groupId>org.springframework.boot</groupId>
74             <artifactId>spring-boot-maven-plugin</artifactId>
75             <configuration>
76                 <excludes>
77                     <exclude>
78                         <groupId>org.projectlombok</groupId>
79                         <artifactId>lombok</artifactId>
80                     </exclude>
81                 </excludes>
82             </configuration>
83         </plugin>
```

Two red boxes highlight specific parts of the code:

- The first box (lines 63-67) highlights the Lombok dependency configuration.
- The second box (lines 75-82) highlights the configuration for excluding Lombok from the Spring Boot Maven plugin.

A button labeled 'Atualizar o Maven' (Update Maven) is visible next to the build section.

Lombok Annotations

Anotação	Especificação
@Data	Gerar automaticamente os getters, setters, toString (não cria o construtor). Também podemos colocar o @Getter e o @Setter em cada um dos atributos, caso não deseja gerar todos eles com o @Data
@AllArgsConstructor	Criar um construtor com todos os atributos
@NoArgsConstructor	Criar um construtor sem atributos
@Getter	Gerar todos os <i>getters</i> da classe quando anotado acima da classe, e gera o <i>getter</i> de um atributo quando anotado acima de um atributo
@Setter	Gerar todos os <i>setters</i> da classe quando anotado acima da classe, e gera o <i>setters</i> de um atributo quando anotado acima de um atributo
@ToString	Gerar o método <i>toString</i> da classe
@EqualsAndHashCode	Gerar todos os <i>equalsAndHashCode</i> da classe. Para ignorar parâmetros podemos usar (exclude= {"nome-do-parâmetro-1", "nome-do-parâmetro-2", ...}). Em geral, devemos evitar usar o Lombok para gerar os métodos equals() e hashCode() para nossas entidades JPA.
@Builder	Criar objetos sem a necessidade de construtores e métodos <i>setter</i> , (caso os objetos sejam imutáveis)

<https://www.baeldung.com/intro-to-project-lombok>

<https://www.baeldung.com/lombok-ide>

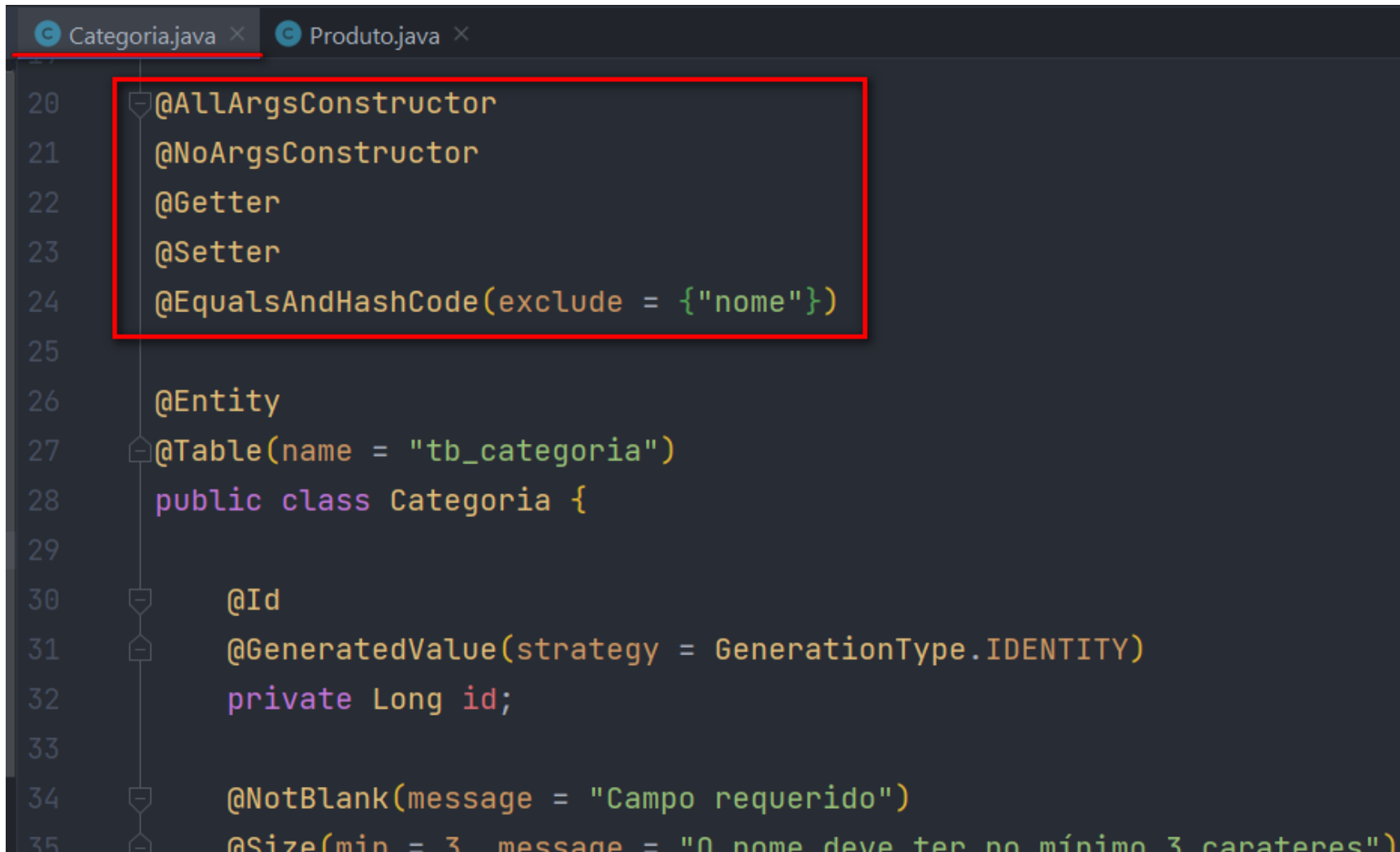
Usando Lombok

Classes Model

■ **Alterar** class Categoria

- Utilizaremos as anotações do Lombok.
- Excluir ou comentar os construtores, getters and setters, equals and hashCode.

Alterar class Categoria



The screenshot shows an IDE with two tabs: 'Categoria.java' and 'Produto.java'. The 'Categoria.java' tab is active, displaying the following code:

```
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35
```

A dropdown menu is open, showing the following annotations:

- @AllArgsConstructor
- @NoArgsConstructor
- @Getter
- @Setter
- @EqualsAndHashCode(exclude = {"nome"})

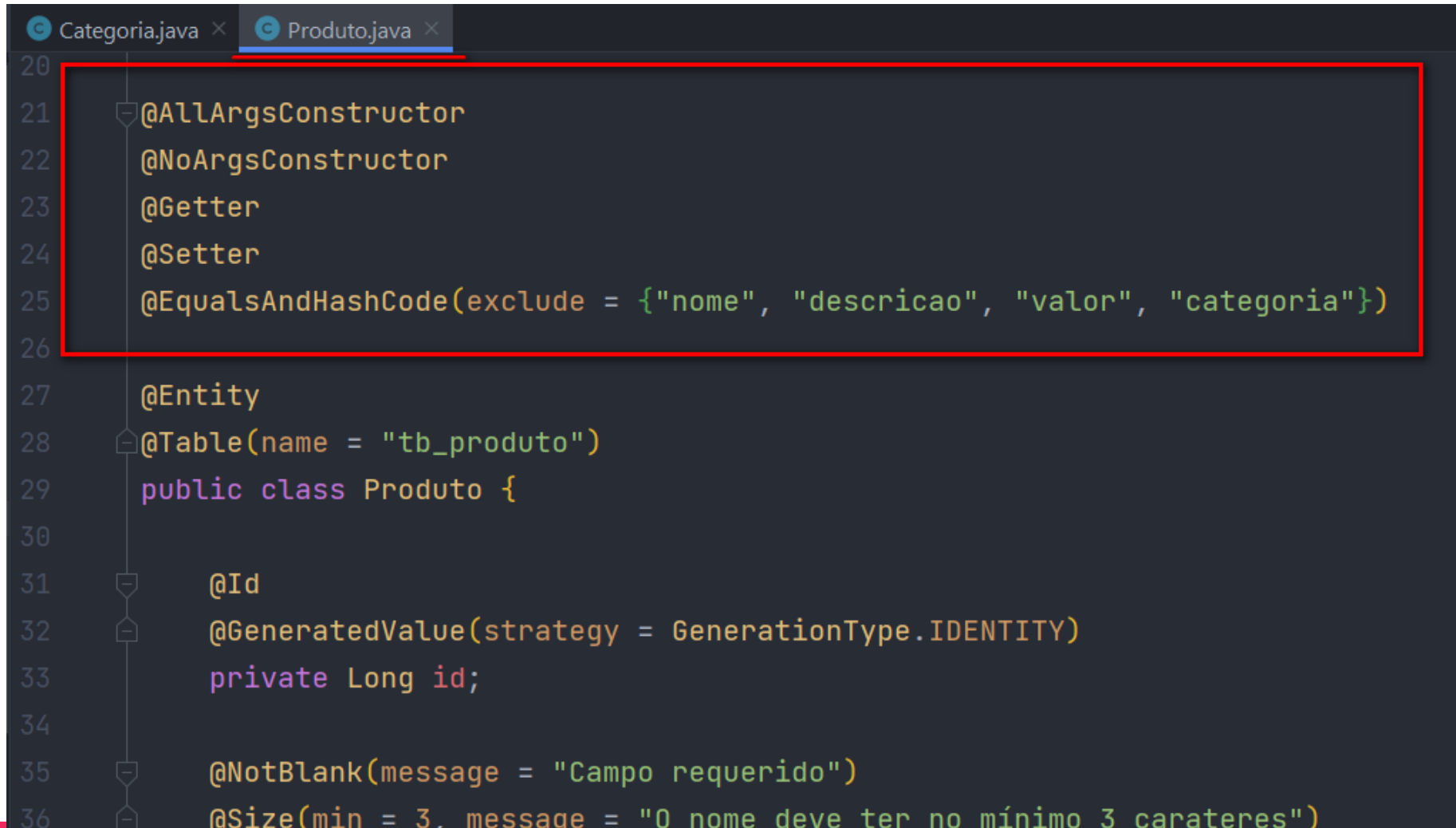
The code in the background includes:

```
@Entity  
@Table(name = "tb_categoria")  
public class Categoria {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    @NotBlank(message = "Campo requerido")  
    @Size(min = 3, message = "O nome deve ter no mínimo 3 caracteres")
```


■ Alterar class Produto

- Utilizaremos as anotações do Lombok.
- Excluir ou comentar os construtores, getters and setters, equals and hashCode.

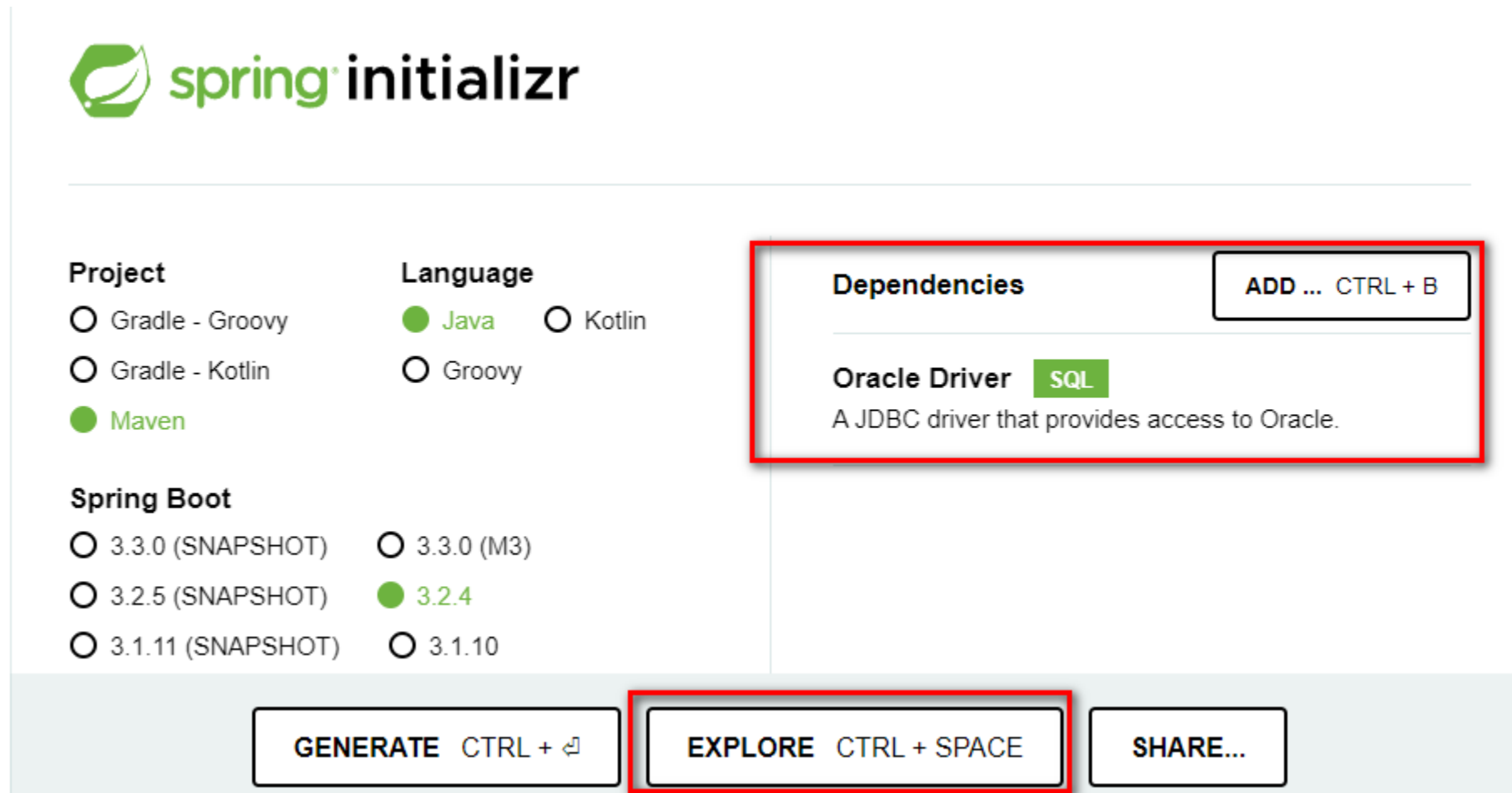
Alterar class Produto



```
20
21  @AllArgsConstructor
22  @NoArgsConstructor
23  @Getter
24  @Setter
25  @EqualsAndHashCode(exclude = {"nome", "descricao", "valor", "categoria"})
26
27  @Entity
28  @Table(name = "tb_produto")
29  public class Produto {
30
31      @Id
32      @GeneratedValue(strategy = GenerationType.IDENTITY)
33      private Long id;
34
35      @NotBlank(message = "Campo requerido")
36      @Size(min = 3, message = "O nome deve ter no mínimo 3 caracteres")
```

Oracle

Oracle - Dependência



The image shows the Spring Initializr web form. It has a header with the Spring logo and 'spring initializr' text. Below the header, there are three main sections: 'Project', 'Language', and 'Spring Boot'. The 'Project' section has three radio buttons: 'Gradle - Groovy', 'Gradle - Kotlin', and 'Maven' (which is selected). The 'Language' section has three radio buttons: 'Java' (selected), 'Kotlin', and 'Groovy'. The 'Spring Boot' section has six radio buttons arranged in two columns: '3.3.0 (SNAPSHOT)', '3.3.0 (M3)', '3.2.5 (SNAPSHOT)', '3.2.4' (selected), '3.1.11 (SNAPSHOT)', and '3.1.10'. To the right of these sections is a 'Dependencies' section, which is highlighted with a red box. It contains the text 'Oracle Driver' followed by a green button labeled 'SQL', and a description 'A JDBC driver that provides access to Oracle.' Below the 'Dependencies' section is a button labeled 'ADD ... CTRL + B'. At the bottom of the form, there are three buttons: 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE' (highlighted with a red box), and 'SHARE...'. The 'EXPLORE' button is highlighted with a red box.

spring initializr

Project

- ☐ Gradle - Groovy
- ☐ Gradle - Kotlin
- ☒ Maven

Language

- ☒ Java
- ☐ Kotlin
- ☐ Groovy

Spring Boot

- ☐ 3.3.0 (SNAPSHOT)
- ☐ 3.3.0 (M3)
- ☐ 3.2.5 (SNAPSHOT)
- ☒ 3.2.4
- ☐ 3.1.11 (SNAPSHOT)
- ☐ 3.1.10

Dependencies ADD ... CTRL + B

Oracle Driver SQL

A JDBC driver that provides access to Oracle.

GENERATE CTRL + G **EXPLORE** CTRL + SPACE **SHARE...**

Oracle - Dependência

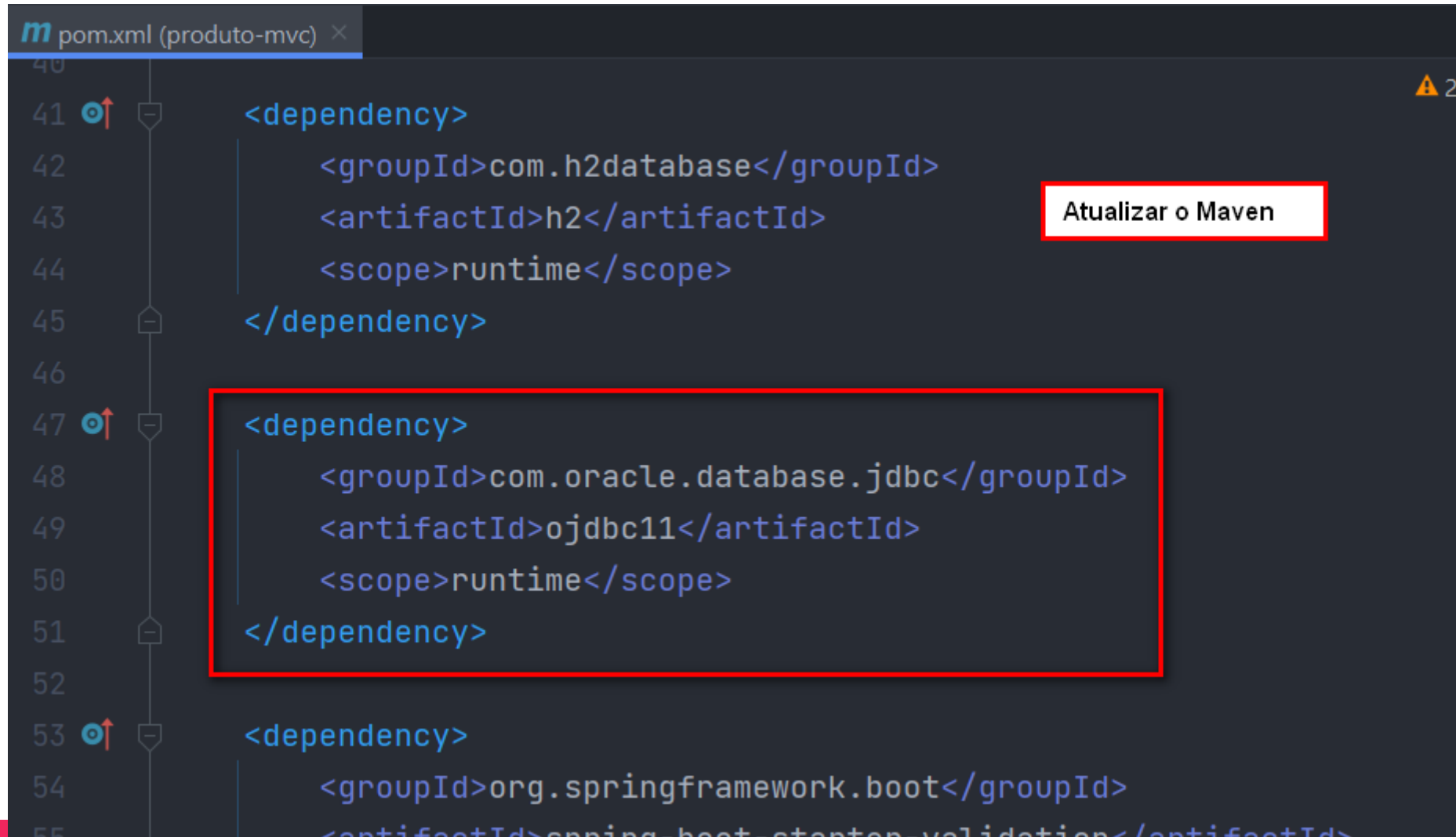
The screenshot shows a web-based interface for a Maven project. On the left, a file explorer for 'demo.zip' lists files: .gitignore, .mvn, HELP.md, mvnw, mvnw.cmd, **pom.xml** (highlighted), and src. A red box with the text 'copiar o código em destaque e colar no pom.xml' points to the pom.xml file. On the right, the pom.xml code is displayed in a text area. A red box highlights a new dependency being added: `<groupId>com.oracle.database.jdbc</groupId>`, `<artifactId>ojdbc11</artifactId>`, and `<scope>runtime</scope>`. Above the code area are 'DOWNLOAD' and 'COPY' buttons. Below the code area are 'DOWNLOAD' and 'CLOSE' buttons.

```
<artifactId>spring-boot-starter</artifactId>
</dependency>

<dependency>
  <groupId>com.oracle.database.jdbc</groupId>
  <artifactId>ojdbc11</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
</dependencies>

<build>
```

Oracle – Dependência – pom.xml



```
m pom.xml (produto-mvc) x
40
41 <dependency>
42     <groupId>com.h2database</groupId>
43     <artifactId>h2</artifactId>
44     <scope>runtime</scope>
45 </dependency>
46
47 <dependency>
48     <groupId>com.oracle.database.jdbc</groupId>
49     <artifactId>ojdbc11</artifactId>
50     <scope>runtime</scope>
51 </dependency>
52
53 <dependency>
54     <groupId>org.springframework.boot</groupId>
55     <artifactId>spring-boot-starter-validation</artifactId>
```

Atualizar o Maven

AULA 14 – Hibernate DDL - Conexão com Banco de Dados Oracle - Lombok

Preparar as classes para o Oracle

Atualizações

Alterar class Categoria

```
Categoria.java x
21  @Entity
22  @Table(name = "tb_categoria")
23  public class Categoria {
24
25      @Id
26      // @GeneratedValue(strategy = GenerationType.IDENTITY)
27      @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "categoria_seq")
28      @SequenceGenerator(name = "categoria_seq", sequenceName = "categoria_seq", allocationSize = 1)
29      private Long id;
30
31      @NotBlank(message = "Campo requerido")
32      @Size(min = 3, message = "O nome deve ter no mínimo 3 caracteres")
33      @Column(length = 150, nullable = false)
34      private String nome;
35
36      @OneToMany(mappedBy = "categoria")
37      private List<Produto> produtos = new ArrayList<>();
```


Alterar class Produto

```
Produto.java x
20 @Entity
21 @Table(name = "tb_produto")
22 public class Produto {
23
24     @Id
25     // @GeneratedValue(strategy = GenerationType.IDENTITY)
26     @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "produto_seq")
27     @SequenceGenerator(name = "produto_seq", sequenceName = "produto_seq", allocationSize = 1)
28     private Long id;
29
30     @NotBlank(message = "Campo requerido")
31     @Size(min = 3, message = "O nome deve ter no mínimo 3 caracteres")
32     @Column(length = 150, nullable = false)
33     private String nome;
34
35     @NotBlank(message = "Campo requerido")
36     // @Column(columnDefinition = "TEXT") //para textos longos
37     @Column(length = 250, nullable = false)
38     private String descricao;
39
40     @NotNull(message = "Campo requerido")
41     @Positive(message = "O valor deve ser positivo")
42     private Double valor;
43
44     @ManyToOne
45     @JoinColumn(name = "categoria_id", nullable = false) //PK
46     private Categoria categoria;
```

Seed do Banco de dados

Alterar import.sql

```
import.sql ×
1  INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Smartphone');
2  INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Smart TV');
3  INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Notebook');
4  INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Tablet');
5  INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Mouse');
6  INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Teclado');
7
8  INSERT INTO tb_produto(id, nome, descricao, valor, categoria_id) VALUES(produto_seq.NEXTVAL, 'Mouse Microsoft', 'Mouse sem fio', 250.0, 5);
9  INSERT INTO tb_produto(id, nome, descricao, valor, categoria_id) VALUES(produto_seq.NEXTVAL, 'Smartphone Samsung Galaxy A54 5G', 'Samsung Galaxy A54 5G', 1799.0, 1);
10 INSERT INTO tb_produto(id, nome, descricao, valor, categoria_id) VALUES(produto_seq.NEXTVAL, 'Smart TV', 'Smart TV LG LED 65 polegadas', 3999, 2);
```

Configuração para DB Oracle

Perfil dev

Hibernate - DDL

O Hibernate permite a criação do banco de dados inteiro se alterarmos a propriedade `spring.jpa.hibernate.ddl-auto`, os valores aceitos são:

Propriedade	Espedificação
validate	Faz uma validação sempre que a aplicação inicia e verifica se o banco coincide com as suas classes de entidade. Valida o schema, não faz mudanças no banco.
update	Sempre que a aplicação iniciar ele verifica se suas classes entidades estão de acordo com o banco. Caso não esteja, vai ser feito um <i>update</i> no banco adicionando novas colunas na tabela dessa entidade. Vale lembrar que o update não remove ou renomeia colunas. Por isso, não é recomendado de forma alguma utilizar o Spring Data para gerenciar a evolução do seu banco, para isso existem ferramentas próprias, como, por exemplo, o <i>FlyWay</i> .
create	Sempre que iniciar sua aplicação o Spring Data vai apagar tudo e recriar novamente, ou seja, cria o schema e apaga a estrutura e dados anteriores .
create-drop	Semelhante ao create , mas sempre que a aplicação é parada ele apaga tudo que foi criado, ou seja, cria o schema e depois apaga tudo ao terminar a sessão.
none	Basicamente é nenhuma das opções acima, ou seja, significa que você não quer que o Spring Data faça alterações no seu banco. Essa opção é a recomendada para produção , para que não ocorra problemas do Spring modificar seu banco.

OBS.: Não colocar **create**, **create-drop** e **update** em **PRODUÇÃO !!!**

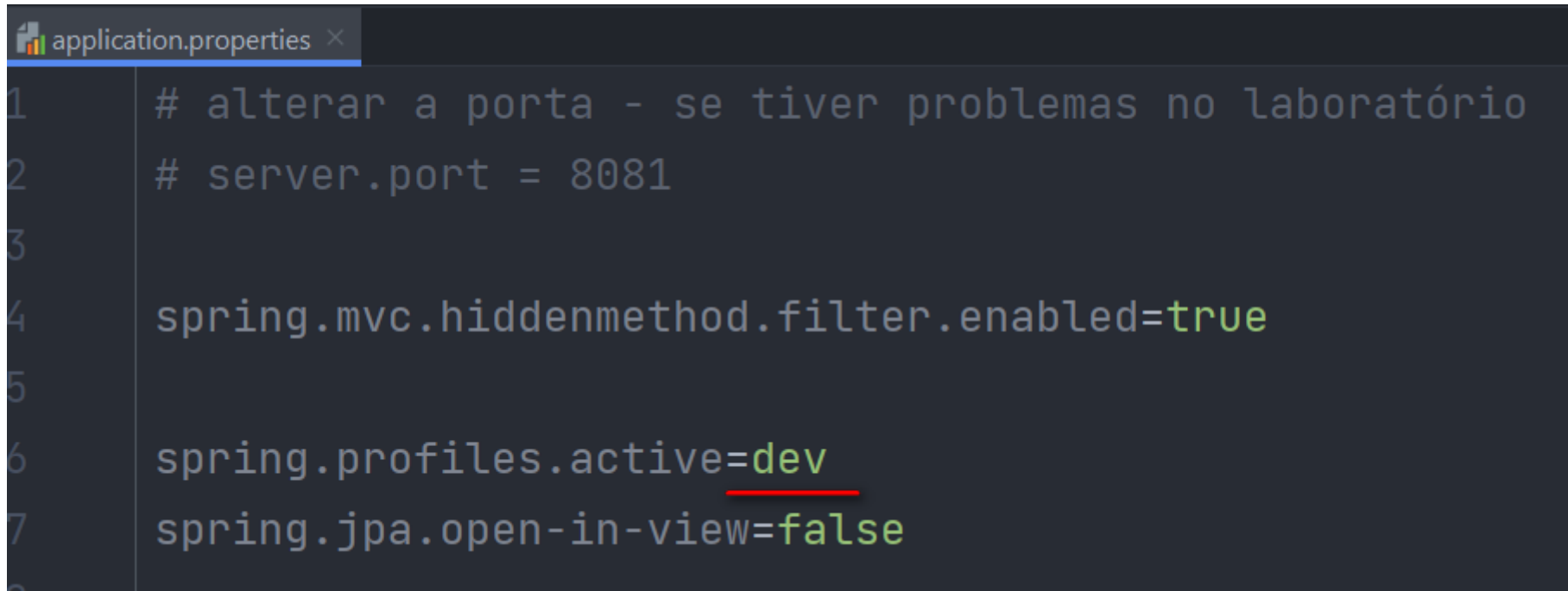
■ Configuração para DB Oracle

- Em resources criar o arquivo **application-dev.properties**

```
1  # perfil para homologação
2  #para criar o script para o DB automaticamente,
3  # descomentar as 4 linhas abaixo, executar a primeira vez e depois comentar
4  #vai ser criado um arquivo sql na raíz da aplicação
5
6  #spring.jpa.properties.jakarta.persistence.schema-generation.create-source=metadata
7  #spring.jpa.properties.jakarta.persistence.schema-generation.scripts.action=create
8  #spring.jpa.properties.jakarta.persistence.schema-generation.scripts.create-target=create.sql
9  #spring.jpa.properties.hibernate.hbm2ddl.delimiter=;
10
11 # Configurações básicas para conexão com o banco Oracle
12 spring.datasource.url=jdbc:oracle:thin:@oracle.fiap.com.br:1521:ORCL
13 spring.datasource.username= <seu login>
14 spring.datasource.password= <sua senha>
15 spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
16
17 # Dialeto utilizado para trabalhar no banco, cada banco possui um dialeto
18 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.OracleDialect
19
```

```
application-dev.properties x
20 # Informa ação DDL inicial do Hibernate - create, update, create-drop, validate, none
21 spring.jpa.hibernate.ddl-auto=create
22 #spring.jpa.hibernate.ddl-auto=update
23 #spring.jpa.hibernate.ddl-auto=none
24 # depois mudar para none
25
26 # Exibe / oculta o SQL executado
27 spring.jpa.show-sql=true
28
29 # Formata o sql exibido
30 spring.jpa.properties.hibernate.format_sql=true
31
32 # Seed do banco de dados
33 # Executa um SQL após inicialização do hibernate, funciona apenas com create
34 # spring.jpa.hibernate.ddl-auto=create
35 # Utilizamos o arquivo import.sql
36
37 # Se desejar criar um arquivo seed com nome diferente do import.sql
38 # utilize a propriedade abaixo
39 #spring.jpa.properties.hibernate.hbm2ddl.import_files=populate-database.sql
```

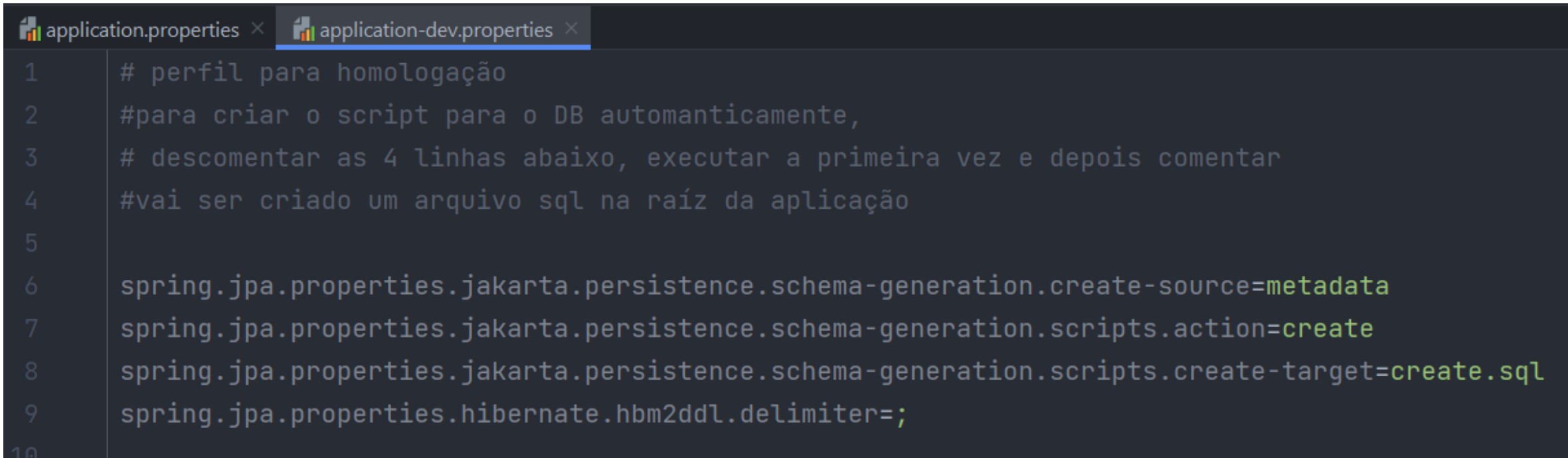

Alterar application.properties

A screenshot of a code editor window titled 'application.properties'. The editor shows a list of lines on the left (1-8) and the corresponding code on the right. The code is in a dark theme with syntax highlighting. Line 1: '# alterar a porta - se tiver problemas no laboratório'. Line 2: '# server.port = 8081'. Line 4: 'spring.mvc.hiddenmethod.filter.enabled=true'. Line 6: 'spring.profiles.active=dev' (with 'dev' underlined in red). Line 7: 'spring.jpa.open-in-view=false'.

```
1 # alterar a porta - se tiver problemas no laboratório
2 # server.port = 8081
3
4 spring.mvc.hiddenmethod.filter.enabled=true
5
6 spring.profiles.active=dev
7 spring.jpa.open-in-view=false
8
```

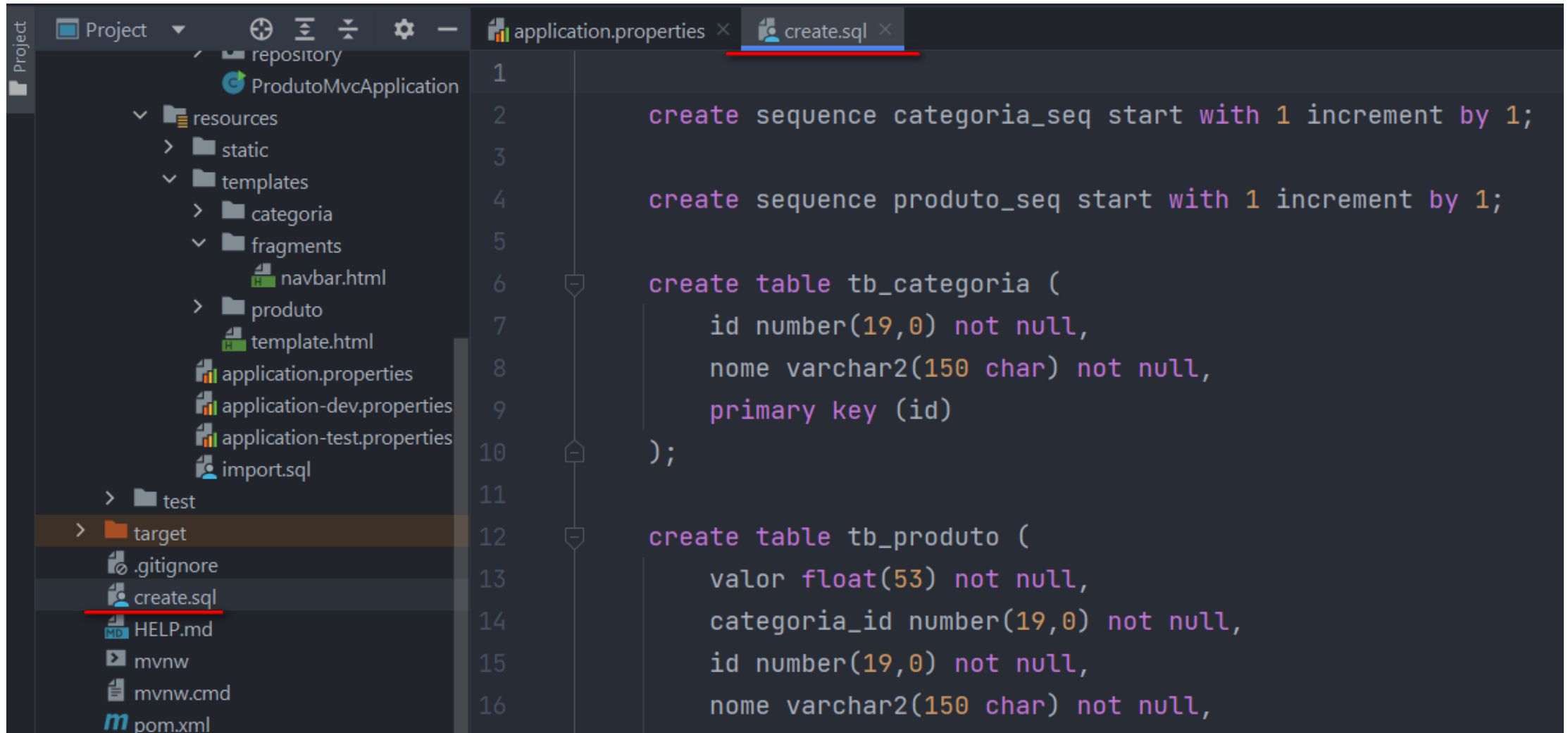
Testar o DB Oracle

- Descomentar as linhas para gerar o arquivo create.sql e comentar novamente



```
application.properties x application-dev.properties x
1  # perfil para homologação
2  #para criar o script para o DB automaticamente,
3  # descomentar as 4 linhas abaixo, executar a primeira vez e depois comentar
4  #vai ser criado um arquivo sql na raiz da aplicação
5
6  spring.jpa.properties.jakarta.persistence.schema-generation.create-source=metadata
7  spring.jpa.properties.jakarta.persistence.schema-generation.scripts.action=create
8  spring.jpa.properties.jakarta.persistence.schema-generation.scripts.create-target=create.sql
9  spring.jpa.properties.hibernate.hbm2ddl.delimiter=;
```

Arquivo gerado



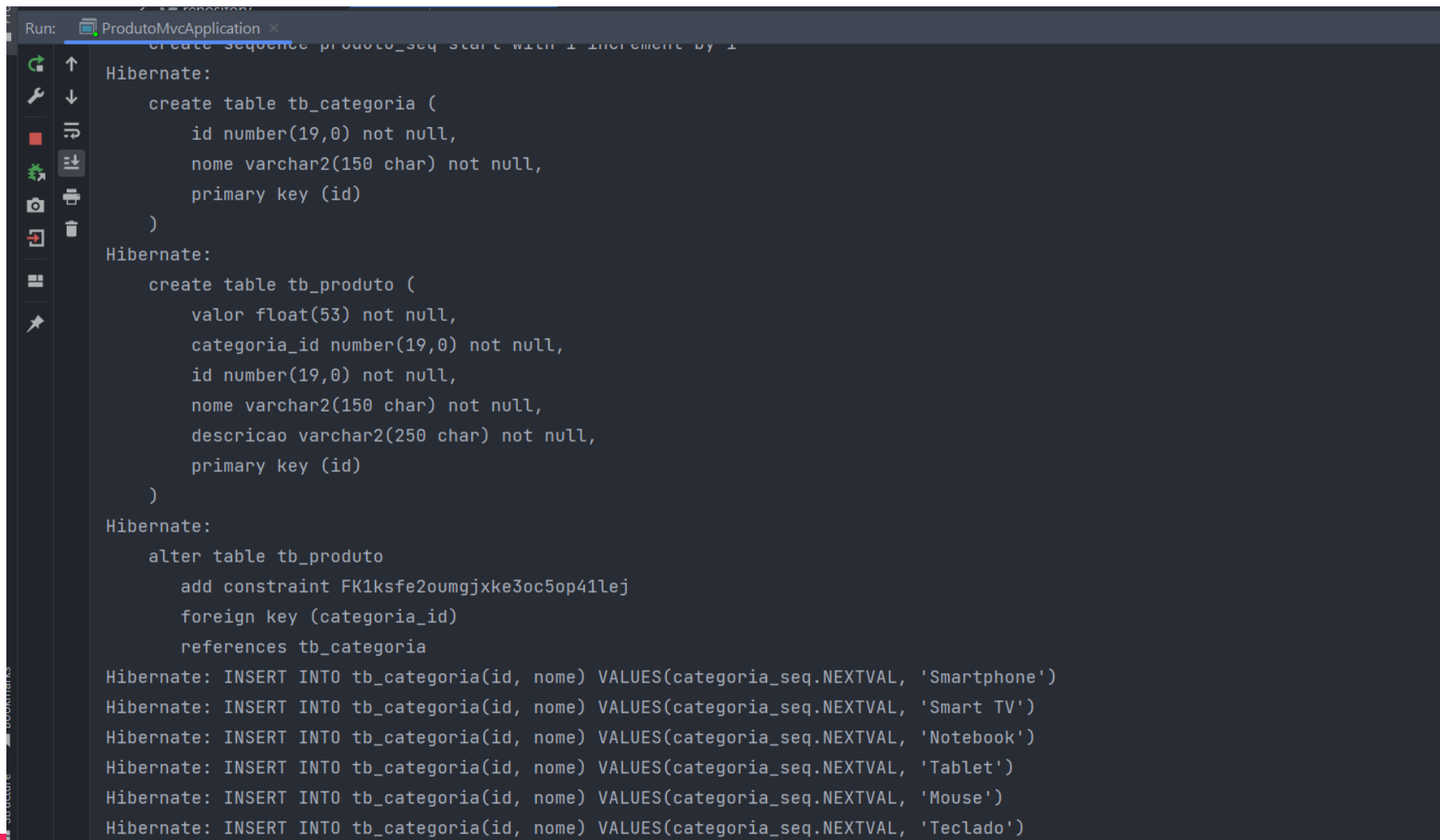
The screenshot shows an IDE with a project named 'ProdutoMvcApplication' in a 'repository' folder. The project structure includes 'resources' (static, templates, categoria, fragments, navbar.html, produto, template.html), 'application.properties', 'application-dev.properties', 'application-test.properties', 'import.sql', 'test', and 'target'. The 'create.sql' file is highlighted in the 'target' folder. The SQL code in the editor is as follows:

```
1  
2 create sequence categoria_seq start with 1 increment by 1;  
3  
4 create sequence produto_seq start with 1 increment by 1;  
5  
6 create table tb_categoria (  
7     id number(19,0) not null,  
8     nome varchar2(150 char) not null,  
9     primary key (id)  
10 );  
11  
12 create table tb_produto (  
13     valor float(53) not null,  
14     categoria_id number(19,0) not null,  
15     id number(19,0) not null,  
16     nome varchar2(150 char) not null,
```

Alterar application-dev.properties

```
application.properties × application-dev.properties ×
1  # perfil para homologação
2  #para criar o script para o DB automaticamente,
3  # descomentar as 4 linhas abaixo, executar a primeira vez e depois comentar
4  #vai ser criado um arquivo sql na raiz da aplicação
5
6  #spring.jpa.properties.jakarta.persistence.schema-generation.create-source=metadata
7  #spring.jpa.properties.jakarta.persistence.schema-generation.scripts.action=create
8  #spring.jpa.properties.jakarta.persistence.schema-generation.scripts.create-target=create.sql
9  #spring.jpa.properties.hibernate.hbm2ddl.delimiter=;
10
```

■ Executar novamente a Aplicação



The screenshot shows an IDE window titled 'Run: ProdutoMvcApplication x'. The main editor area displays a series of SQL scripts. The first script is a comment: 'Create Sequence produto_seq start with 1 increment by 1'. This is followed by two 'Hibernate:' blocks. The first block contains the DDL for 'tb_categoria' with columns 'id' (number(19,0) not null, primary key) and 'nome' (varchar2(150 char) not null). The second block contains the DDL for 'tb_produto' with columns 'valor' (float(53) not null), 'categoria_id' (number(19,0) not null), 'id' (number(19,0) not null, primary key), 'nome' (varchar2(150 char) not null), and 'descricao' (varchar2(250 char) not null). This is followed by an 'alter table' statement to add a foreign key constraint 'FK1ksfe2oumgjxke3oc5op41lej' on 'tb_produto' for 'categoria_id' referencing 'tb_categoria'. Finally, there are six 'Hibernate: INSERT INTO' statements for 'tb_categoria' with values for 'Smartphone', 'Smart TV', 'Notebook', 'Tablet', 'Mouse', and 'Teclado'.

```
Run: ProdutoMvcApplication x
Create Sequence produto_seq start with 1 increment by 1

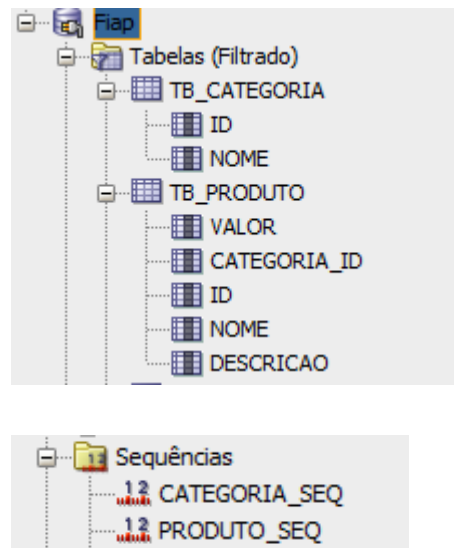
Hibernate:
  create table tb_categoria (
    id number(19,0) not null,
    nome varchar2(150 char) not null,
    primary key (id)
  )

Hibernate:
  create table tb_produto (
    valor float(53) not null,
    categoria_id number(19,0) not null,
    id number(19,0) not null,
    nome varchar2(150 char) not null,
    descricao varchar2(250 char) not null,
    primary key (id)
  )

Hibernate:
  alter table tb_produto
    add constraint FK1ksfe2oumgjxke3oc5op41lej
    foreign key (categoria_id)
    references tb_categoria

Hibernate: INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Smartphone')
Hibernate: INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Smart TV')
Hibernate: INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Notebook')
Hibernate: INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Tablet')
Hibernate: INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Mouse')
Hibernate: INSERT INTO tb_categoria(id, nome) VALUES(categoria_seq.NEXTVAL, 'Teclado')
```

Verificar no Oracle SQL Developer



Planilha Query Builder

```
select * from tb_categoria;
```

```
select * from tb_produto;
```

Resultado da Consulta x

Todas as Linhas Extraídas: 6 em 0,022 segundos

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

Planilha Query Builder

```
select * from tb_categoria;
```

```
select * from tb_produto;
```

Resultado da Consulta x

Todas as Linhas Extraídas: 3 em 0,019 segundos

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

Testar a Aplicação

FIAP Home Categoria ▾ Produto ▾

Lista de Categorias

Nome	Ações
Smartphone	<button>Excluir</button> <button>Editar</button>
Smart TV	<button>Excluir</button> <button>Editar</button>
Notebook	<button>Excluir</button> <button>Editar</button>
Tablet	<button>Excluir</button> <button>Editar</button>
Mouse	<button>Excluir</button> <button>Editar</button>
Teclado	<button>Excluir</button> <button>Editar</button>

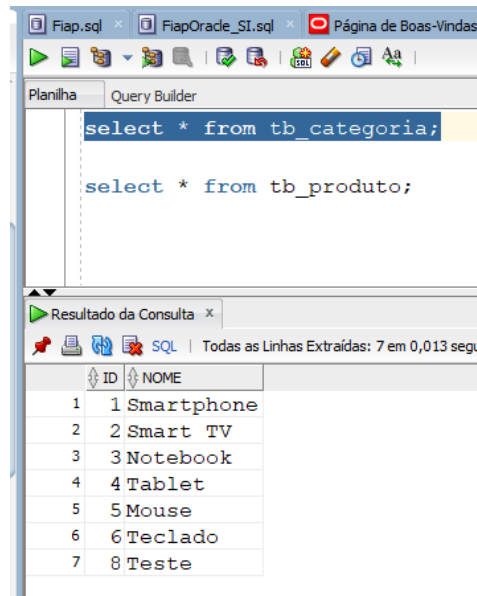
Testar a Aplicação

FIAP Home Categoria ▾ Produto ▾

Cadastro de Categoria

Nome:

Salvar



The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains two SQL statements: `select * from tb_categoria;` and `select * from tb_produto;`. The results pane shows the output of the first query, displaying a table with 7 rows and 2 columns: ID and NOME.

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

FIAP Home Categoria ▾ Produto ▾

Lista de Categorias

Nome	Ações	
Smartphone	<button>Excluir</button>	<button>Editar</button>
Smart TV	<button>Excluir</button>	<button>Editar</button>
Notebook	<button>Excluir</button>	<button>Editar</button>
Tablet	<button>Excluir</button>	<button>Editar</button>
Mouse	<button>Excluir</button>	<button>Editar</button>
Teclado	<button>Excluir</button>	<button>Editar</button>
Teste	<button>Excluir</button>	<button>Editar</button>

Testar a Aplicação

FIAP Home Categoria ▾ Produto ▾

Cadastro de Produtos

Nome:

Descrição:

Categoria:

Valor:

FIAP Home Categoria ▾ Produto ▾

Lista de Produtos

Nome	Descrição	Categoria	Valor	Ações	
Mouse Microsoft	Mouse sem fio	Mouse	250.0	<input type="button" value="Excluir"/>	<input type="button" value="Editar"/>
Smartphone Samsung Galaxy A54 5G	Samsung Galaxy A54 5G	Smartphone	1799.0	<input type="button" value="Excluir"/>	<input type="button" value="Editar"/>
Smart TV	Smart TV LG LED 65 polegadas	Smart TV	3999.0	<input type="button" value="Excluir"/>	<input type="button" value="Editar"/>
Teste	Teste	Teste	12.0	<input type="button" value="Excluir"/>	<input type="button" value="Editar"/>

Planilha Query Builder

```
select * from tb_categoria;  
select * from tb_produto;
```

Resultado da Consulta x

SQL | Todas as Linhas Extraídas: 4 em 0,013 segundos

	VALOR	CATEGORIA_ID	ID	NOME	DESCRICAO
1	250	5	1	Mouse Microsoft	Mouse sem fio
2	1799	1	2	Smartphone Samsung Galaxy A54 5G	Samsung Galaxy A54 5G
3	3999	2	3	Smart TV	Smart TV LG LED 65 polegadas
4	12	8	4	Teste	Teste

Alterar application-dev.properties

```
application-dev.properties x
19
20 # Informa ação DDL inicial do Hibernate - create, update, create-drop, validate, none
21 #spring.jpa.hibernate.ddl-auto=create
22 #spring.jpa.hibernate.ddl-auto=update
23 spring.jpa.hibernate.ddl-auto=none
24 # depois mudar para none
```



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).