Programação de Aplicações Corporativas

Arquitetura MVC sem a utilização de Frameworks

Java Legado

- *View: JSP e JSTL*
- *Controller: Servlet despachando objetos Java*
- *DAO: Implementação de classes de acesso aos dados relacionais usando SQL*
- *Model: Java Bean (atributos privados, construtores, gest e sets)*

Contexto: Observe o exemplo de um JSP capturar e salvar um registro no BD;

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.sql.Connection" %>
<%@ page import="java.sql.DriverMana
<%@ page import="java.sql.PreparedSt
<%@ page import="java.sql.SQLExcepti
<html>
<head>
<title>Cadastro de Pessoa</title>
</head>
<body>
<h1>Cadastro de Pessoa</h1>
      <form action="cadastro.jsp" method="post">
      <label for="nome">Nome:</label>
      <input type="text" id="nome" name="nome" required>
      <br>
      <input type="submit" value="Cadastrar">
</form>
.....
```

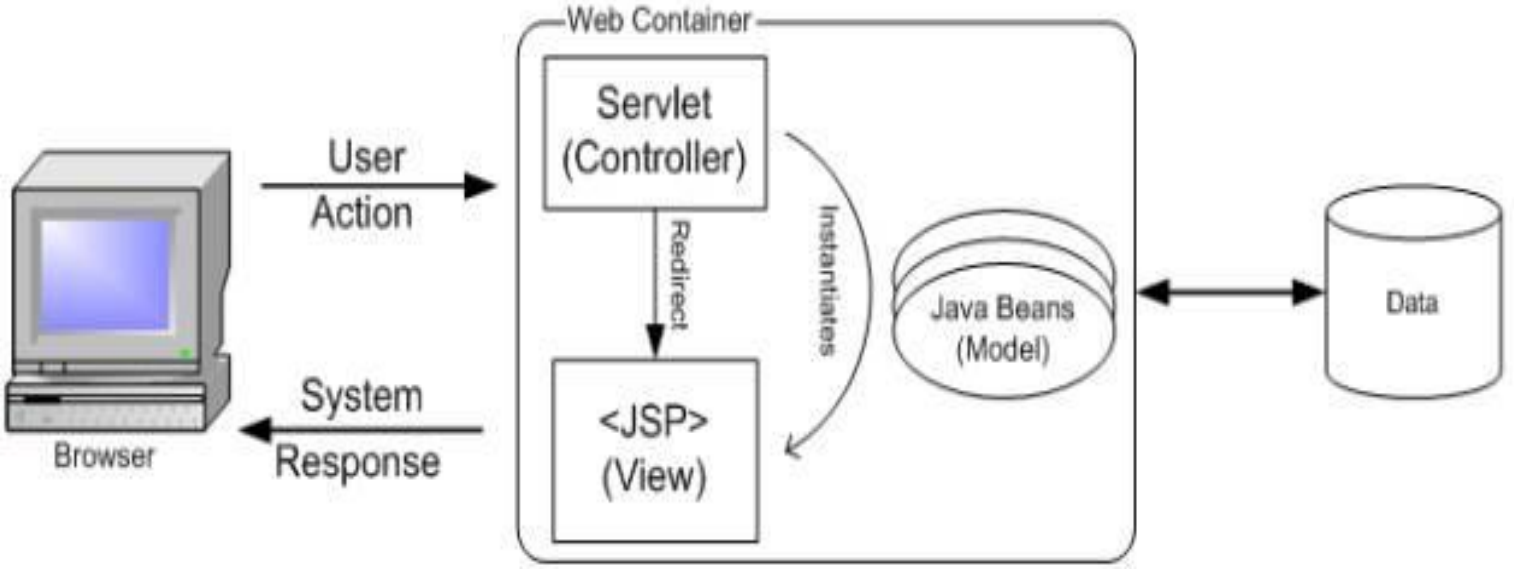
```
<% String nome = request.getParameter("nome");
try {
    // Cria uma conexão com o banco de dados
    Class.forName("com.mysql.jdbc.Driver");
    String url = "jdbc:mysql://localhost:3306/meu_banco_de_dados";
    String usuario = "usuario_do_banco";
    String senha = "senha_do_banco";
    Connection conexao = DriverManager.getConnection(url, usuario, senha);
    // Prepara a declaração SQL para inserir o nome no banco de dados
    String sql = "INSERT INTO pessoas (nome) VALUES (?)";
    PreparedStatement declaracao = conexao.prepareStatement(sql);
    declaracao.setString(1, nome);
    // Executa a declaração SQL e fecha a conexão com o banco de dados
    declaracao.executeUpdate();
    declaracao.close();
    conexao.close();
    // Redireciona para uma página de sucesso
    response.sendRedirect("sucesso.jsp?nome=" + nome);
} catch (ClassNotFoundException | SQLException e) {
    e.printStackTrace();
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
} %>
</body>
</html>
```

Imagine:

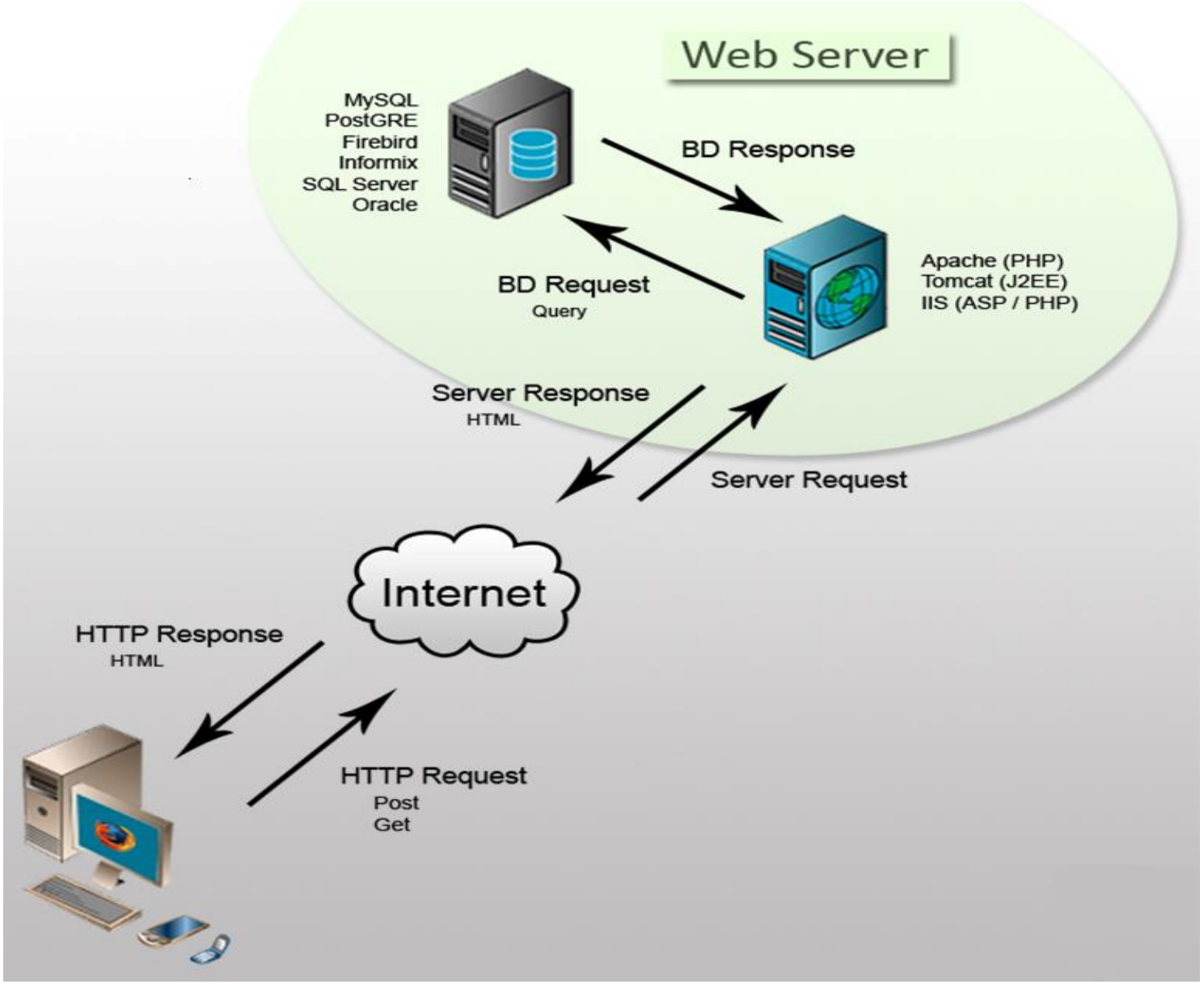
Se neste único arquivo fossem implementadas as funcionalidades de alterar e excluir...
Ou em diversas outras páginas teríamos a senha e o usuário do banco...

MVC – O que é?

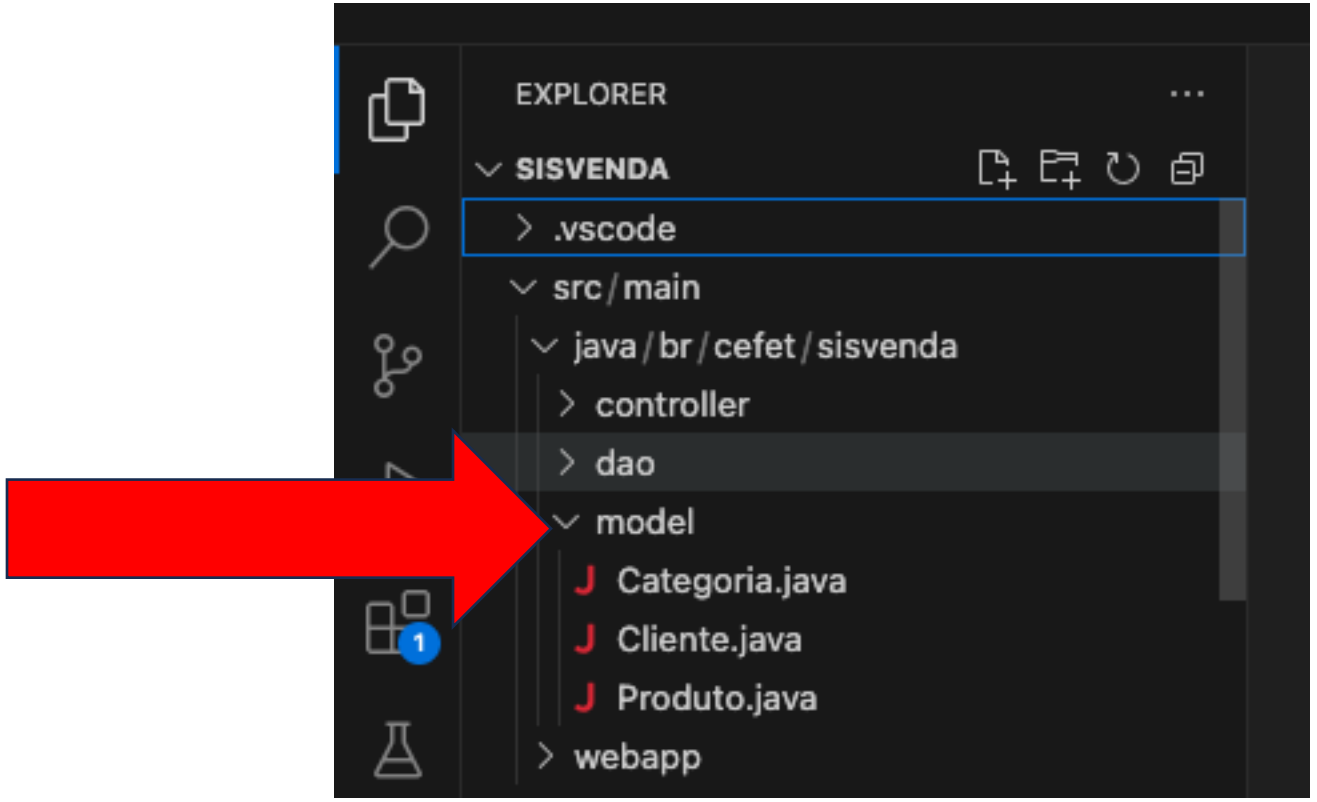
- **Model:** modelo de dados, camadas de acesso e algumas regras do negócio;
- **View:** representa tudo que envolve a interface do sistema (HTML, JSP e JS);
- **Controller:** Conecta a view a model.
  - Responsável por receber dados da view, processá-los e enviá-los a model;
  - Responsável por receber dados da model e enviar a view;

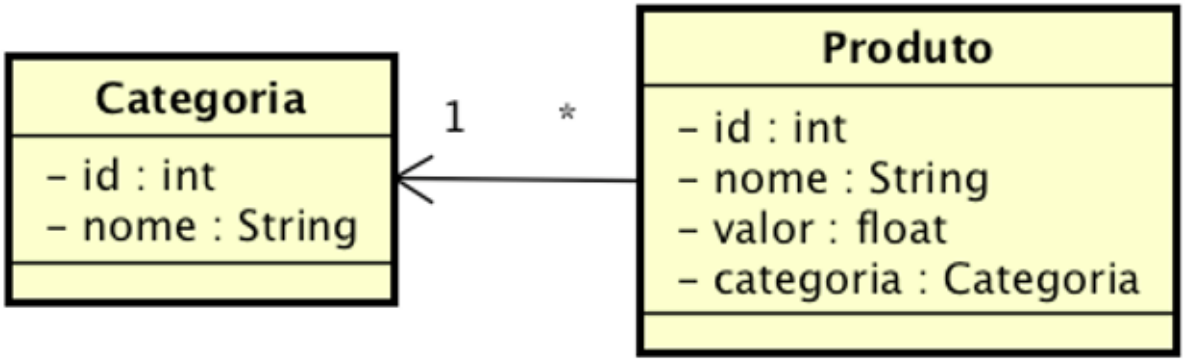


MVC – O que é?

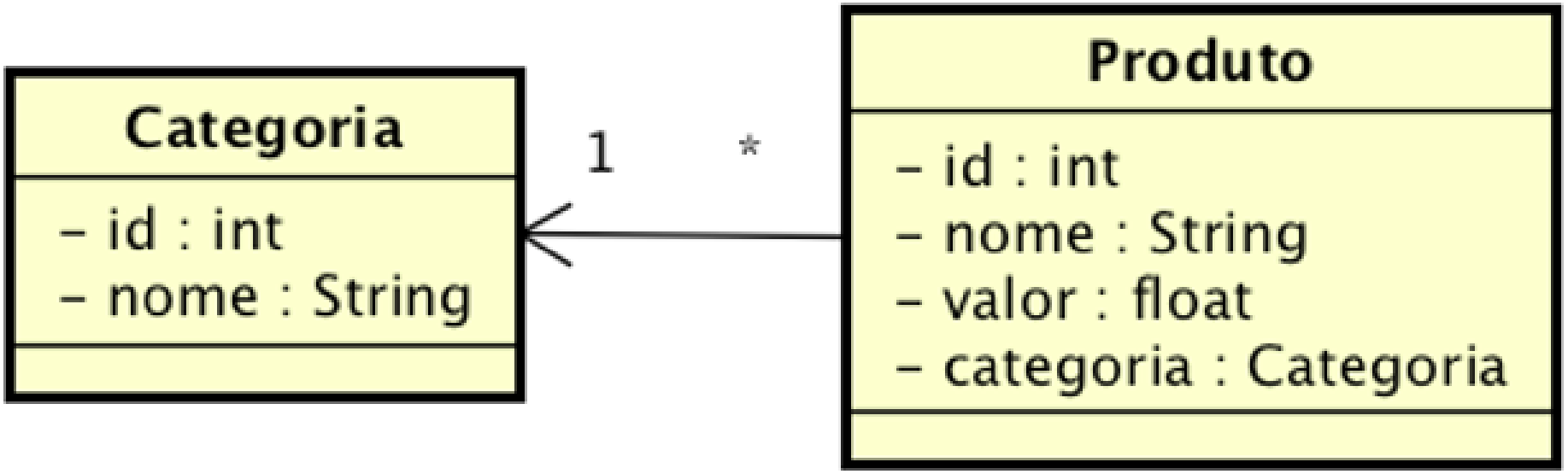

Web Server
MySQL
PostGRE
Firebird
Informix
SQL Server
Oracle
BD Response
BD Request
Query
Apache (PHP)
Tomcat (J2EE)
IIS (ASP / PHP)
Server Response
HTML
Server Request
Internet
HTTP Response
HTML
HTTP Request
Post
Get

Projeto Guiado
Camada Model





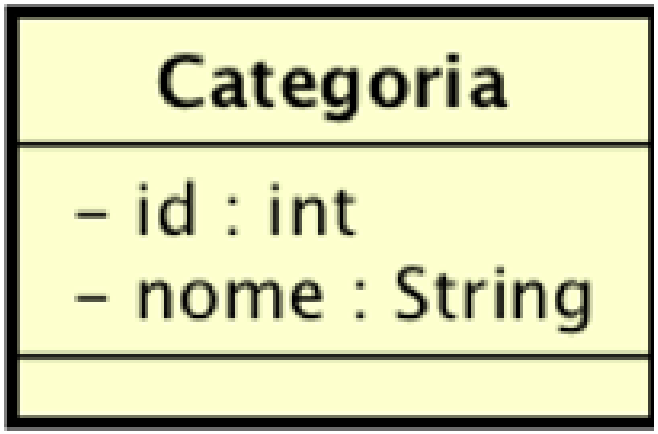
Camada Model



- Java Bean (model): Representa as entidades e lógicas do negócio;
- Encapsula as regras de como manipular os dados;
- Pode conter os atributos que representam os dados e;
    - Construtores;
    - Método GETs e SETs;
    - Outros métodos...

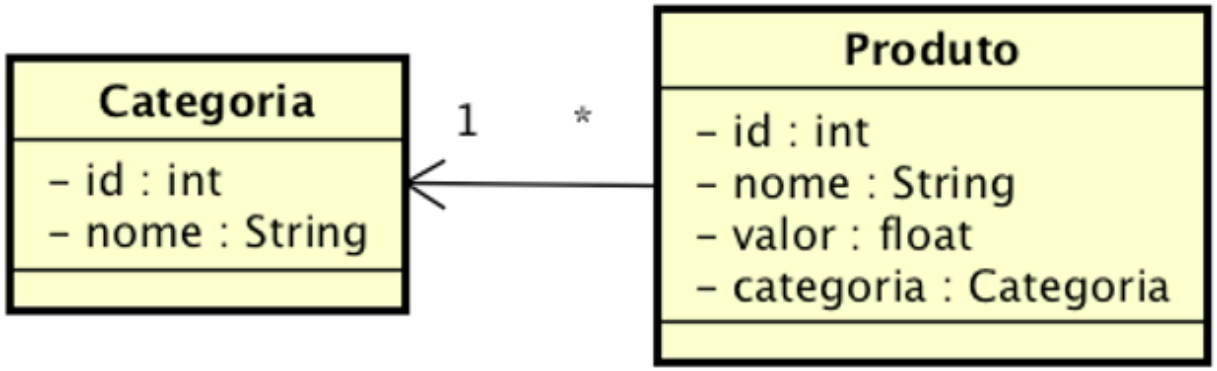
```java
package br.cefet.sisvenda.model;
public class Categoria {
    private int id;
    private String nome;

    public Categoria() {
    }

    public int getId() {
      return id;
    }

    public void setId(int id) {
      this.id = id;
    }

    public String getNome() {
      return nome;
    }

    public void setNome(String nome) {
      this.nome = nome;
    }}
```

Model
Java Bean – Classe Categoria

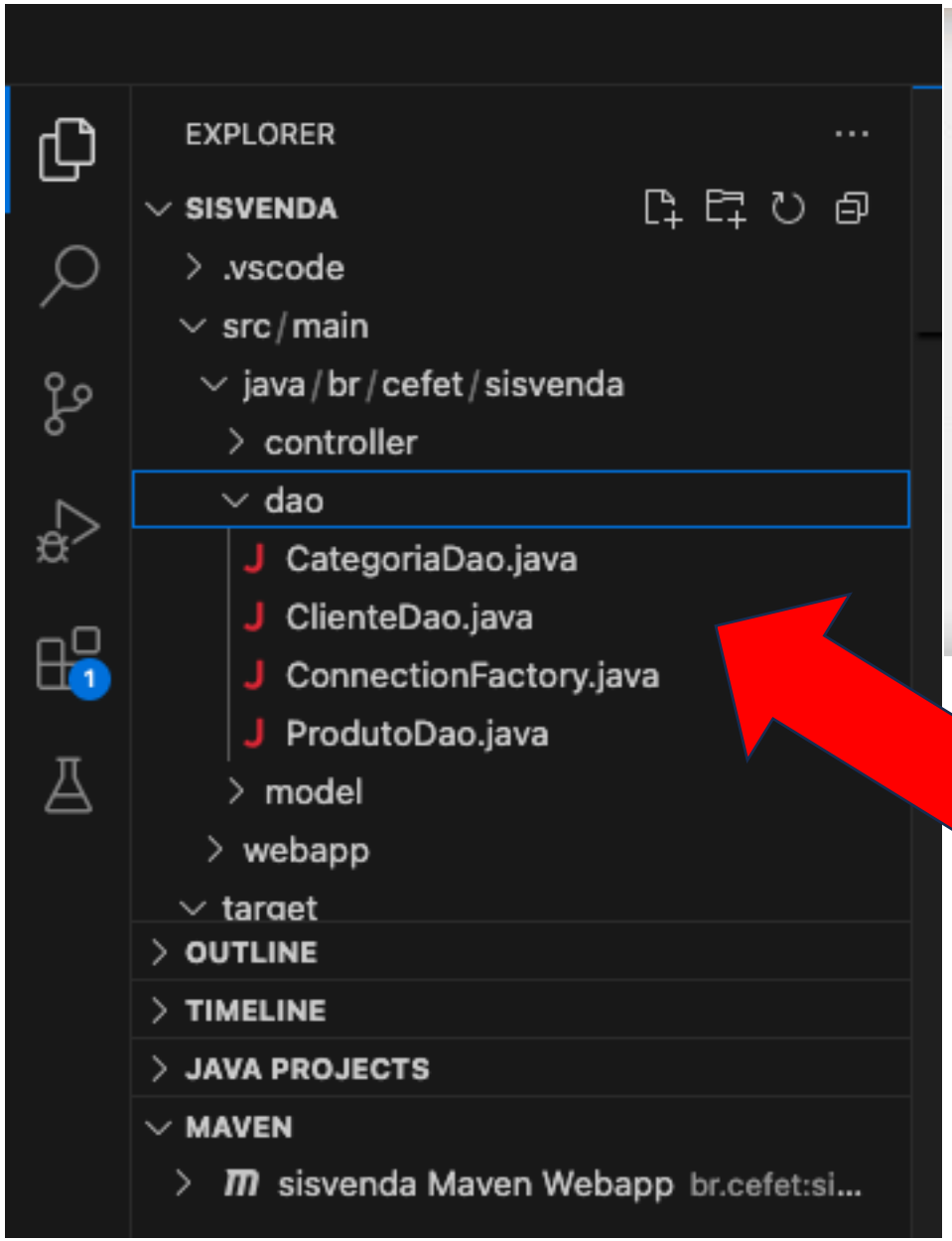


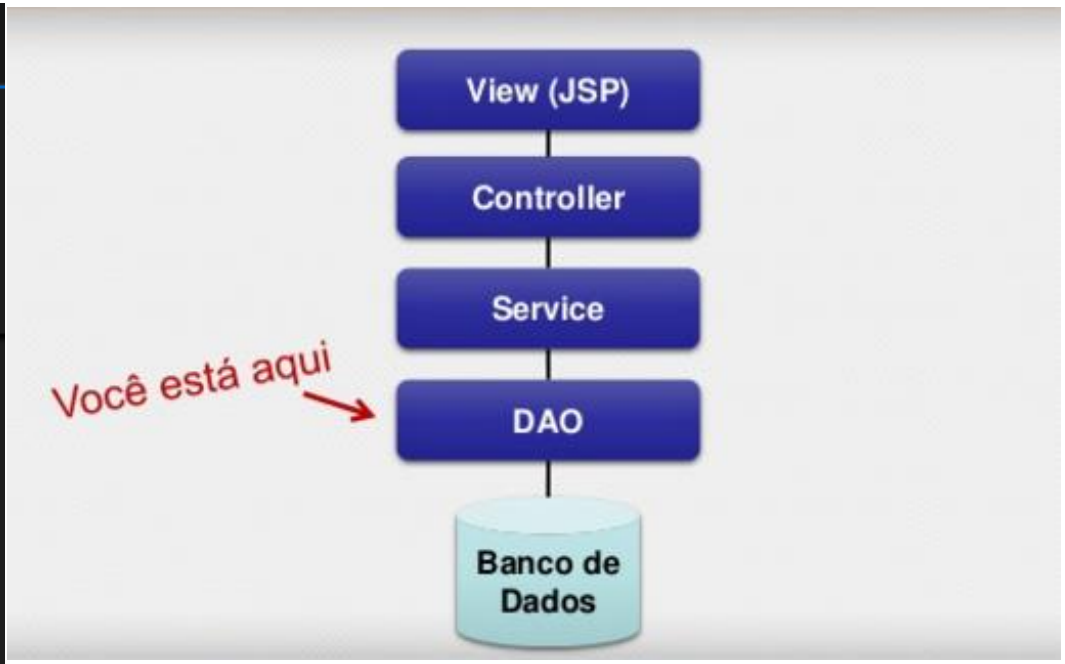
Model
Java Bean – Classe Produto

```
package br.cefet.sisvenda.model;
public class Produto {

	private int id;
	private String nome;
	private float valor;
	private Categoria categoria;
....

public float aumento(float percentual){
	if(percentual<=0 || percentual >100 )
		return 0;
	else
		return this.valor + this.valor * (percentual/100);
}
```



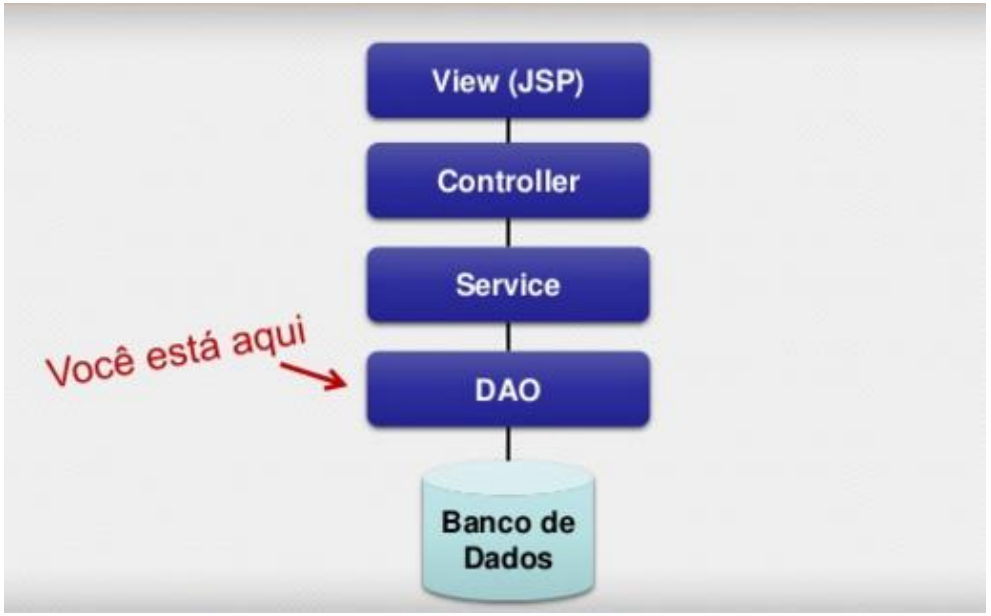
Projeto Guiado
Camada MODEL - DAO



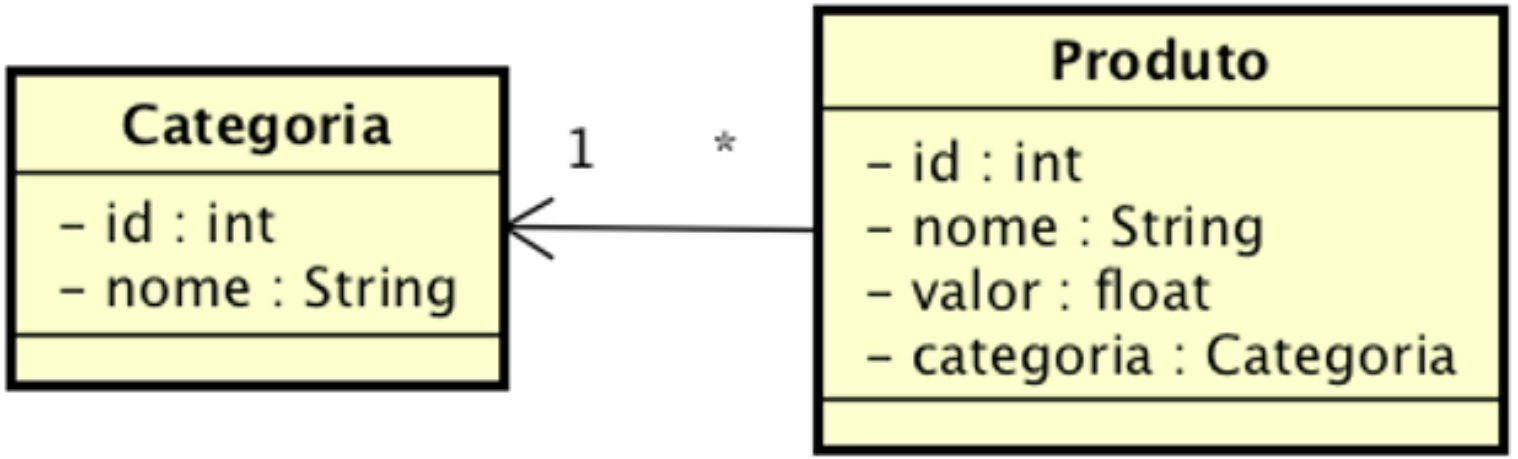


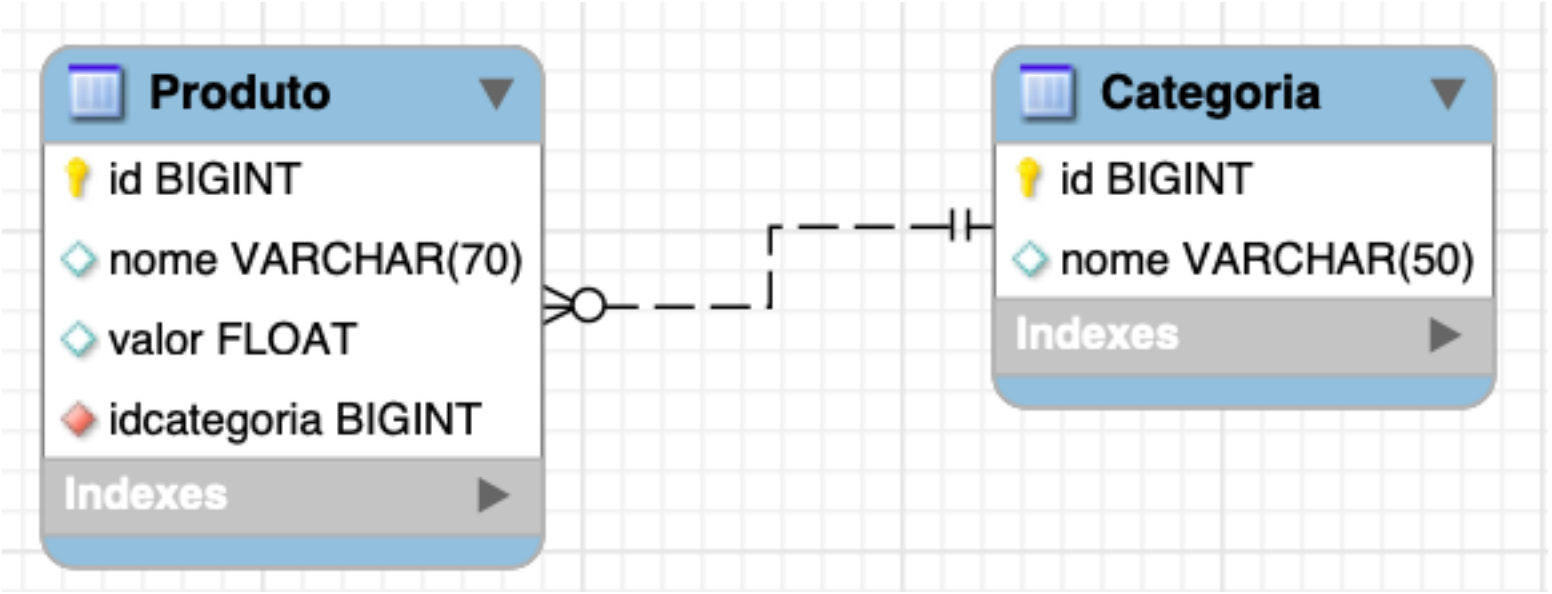
Model – DAO (Data Access Object)

- Padrão de acesso a dados muito difundido.
- Deve-se realizar a conversão do mundo OO (Classes) para o relacional (Tabelas).
- Métodos padrões provavelmente serão utilizados (incluir, alterar, excluir, listar uma linha da tabela e listar várias linhas da tabela).



DAO – Conversão Tabela X OO





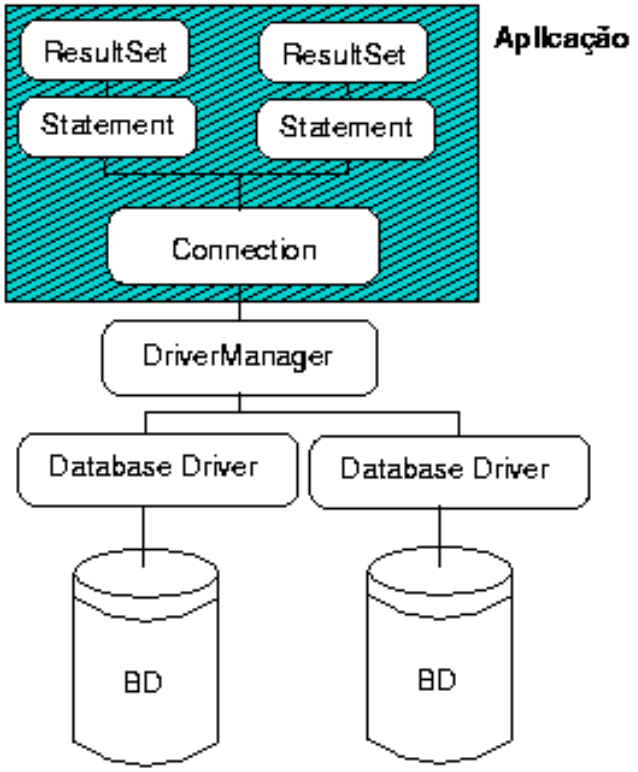
SQL

```
create database dbsis2025 character set utf8mb4 collate utf8mb4_bin;
use dbsis2025;

create table categoria(
id bigint primary key not null auto_increment,
nome varchar(30));

create table produto(
id bigint primary key not null auto_increment,
nome varchar(30),
valor float,
idcategoria bigint,
foreign key(idcategoria) references categoria(id));
```

DAO – Pacote java.sql

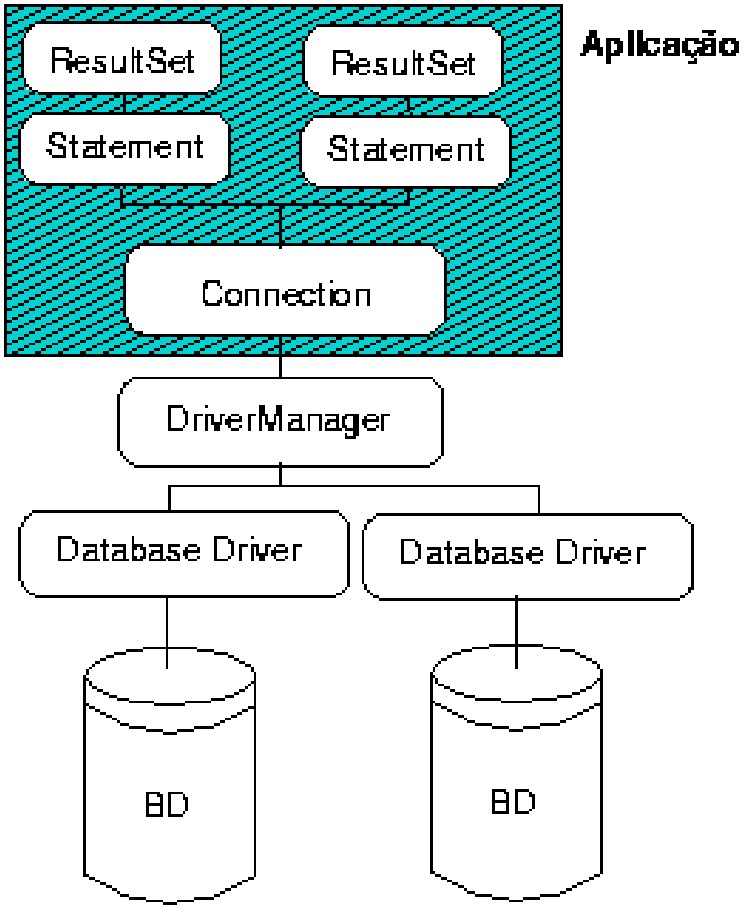
Classes Essencias

- **Connection:** Retorna uma conexão com o SGBD;
- **PreparedStatement:** Prepara o SQL para execução; Prepara e injetas os valores no SQL (respeitando “aspas”, formatação de data...);
- **ResultSet:** Recebe uma matriz com o resultado de um SQL (Select);



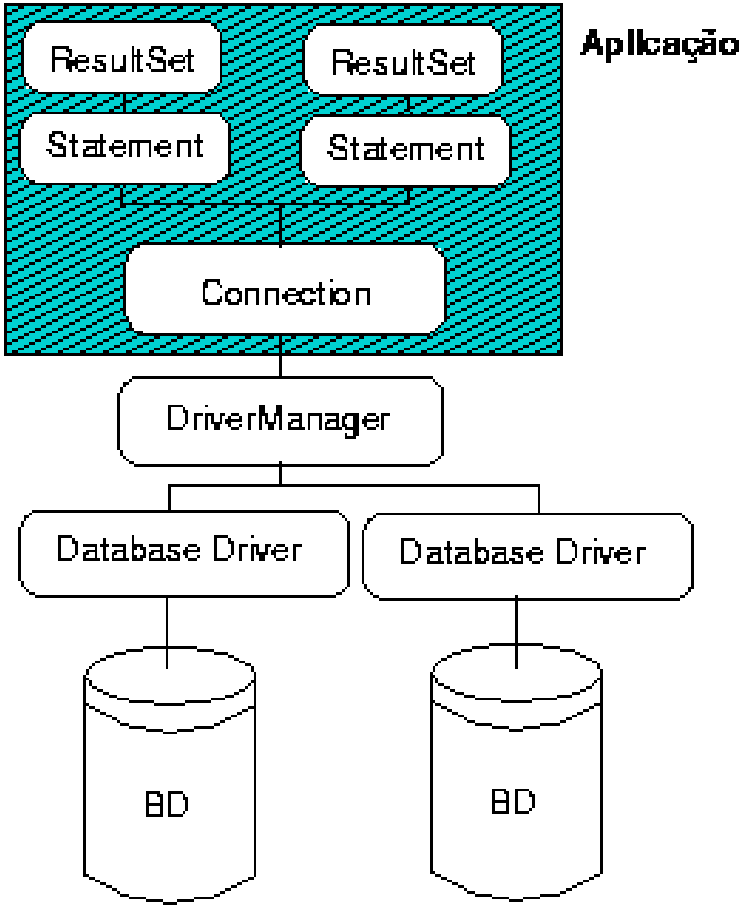
DAO - ConnectionFactory

- Classe que representa uma fábrica de conexões.
- É um padrão de projetos;
- A ideia é evitar a duplicação do código (usuário e senha);



DAO – ConnectionFactory (Exemplo)

```
public static Connection getConnection() {
try { 1
    Class.forName("com.mysql.jdbc.Driver"); 2
} catch (ClassNotFoundException e) { 3
    e.printStackTrace();} 4
try {
    return DriverManager.getConnection(”
          jdbc:mysql://ex1/ex2",“exe3",“exe4"); 5
} catch (SQLException e) { 6
    e.printStackTrace();
    return null;        }  7
```



DAO – ConnectionFactory (Exemplo)

1. Try – Tentar fazer, se der erro não para a execução do código;
2. Localiza o Driver Importado do Mysql (Build path)
3. Caso a linha 2, apresente um problema o programa não será encerrado.
4. Apresenta a pilha de erro na tela;
5. Retorna uma Conexão para quem chamar o método. Ex1: IP; Ex2: Database; Ex3: Login; Ex4: User.
6. Em uma conexão pode ocorrer um erro de SQL;
7. Caso não seja possível estabelecer uma conexão com o SGBD, o método irá retornar null;

DAO – CategoriaDAO (Incluir)

```
import java.sql.Connection; ...
public class CategoriaDao {

        private Connection con;

        public CategoriaDao(){ 1
                con = ConnectionFactory.getConnection(); 2
        }
        public void inserir(Categoria categoria) throws
SQLException{ 3
                ....
```

DAO – CategoriaDAO (Incluir)

```
public void inserir(Categoria categoria) throws SQLException{ 3
        String sql = "insert into categoria(nome) values(?)"; 4
        PreparedStatement stmt = con.prepareStatement(sql); 5
        stmt.setString(1, categoria.getNome()); 6
        stmt.execute(); 7
        stmt.close();      8
        con.close(); 9
}
```

DAO – CategoriaDAO (Excluir)

```
public void apagarUm(int id) throws SQLException {
        String sql = "delete from categoria where id = ?";
        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setInt(1, id);
        stmt.executeQuery();
        stmt.close();
        con.close();
}
```

DAO – CategoriaDAO (listarUm)

```
public Categoria listarUm(int id) throws SQLException {

    String sql = "select id, nome from categoria where id = ?";  0
    PreparedStatement stmt = con.prepareStatement(sql);
    stmt.setInt(1, id);

    ResultSet rs = stmt.executeQuery(); 1

    Categoria categoria = null; 2
    if(rs.next()) {.
        categoria = new Categoria(); 3
        categoria.setId(rs.getInt("id")); 4
        categoria.setNome(rs.getString("nome"));
    }
    stmt.close(); 5
    con.close();
    return categoria; 6
    }
```

0. O "?" indica a posição de inserção do valor;

1. *Executa e armazena o resultado da Query;*
2. *Define uma categoria;*
3. *Instancia uma categoria.*
4. *Atribui o código e nome;*
5. *Fecha o PreparedStatement;*
6. *Retorna UMACATEGORIA;.*

DAO – CategoriaDAO (listarTodos)

```
public List<Categoria>listarTodos() throws SQLException {

    String sql = "select id, nome from categoria"; 1
    PreparedStatement stmt = con.prepareStatement(sql);

    ResultSet rs = stmt.executeQuery();
    Categoria categoria = null;

    List<Categoria> categorias = new ArrayList<Categoria>(); 2
        while(rs.next()) {
        categoria = new Categoria(); 3
        categoria.setId(rs.getInt("id"));
        categoria.setNome(rs.getString("nome"));
        categorias.add(categoria); 4
        }

    stmt.close();
    con.close();
    return categorias; 5
    }
```

1. *SQL que retorna todas as categorias;*
2. *Lista de categorias;*
3. *Instancia UMA categoria*
4. *Adiciona UMA categoria em uma LISTA de categorias;*
5. *Retorna as CATEGORIAS;*

DAO – ProdutoDao (Inserir)

```
public void adicionar(Produto produto) throws SQLException {

    //Monta a string sql
    String sql = "insert produto(nome,valor, idcategoria) "
    + " values (?,?,?)";

    //Prepara a injeção dos valores
    PreparedStatement stmt = con.prepareStatement(sql);
    stmt.setString(1, produto.getNome());
    stmt.setFloat(2, produto.getValor());
    stmt.setInt(3, produto.getCategoria().getId());

    // Executa o sql
    stmt.execute();
    // Fecha as conexões
    stmt.close();
    con.close();
}
```

DAO – ProdutoDao (Listar)

```
public ArrayList<Produto> listarTodos() throws SQLException{ 0
        String sql ="select id, nome, valor, idcategoria from produto";
        PreparedStatement stmt = con.prepareStatement(sql);
        ResultSet rs = stmt.executeQuery(); 1
        ArrayList<Produto> produtos = new ArrayList<Produto>();  2
        while (rs.next()){ 3
                Produto produto = new Produto(); 4
                produto.setId(rs.getInt("codPro")); 5
                produto.setNome(rs.getString("nome")); 6
                produto.setValor(rs.getFloat(”valor"));
                CategoriaDao catDao = new CategoriaDao();
                produto.setCategoria(catDao.listarUm(rs.getInt("idcategoria"))
        );
                produtos.add(pro); 7 }    rs.close();    con.close();
        return produtos; } } 8
```

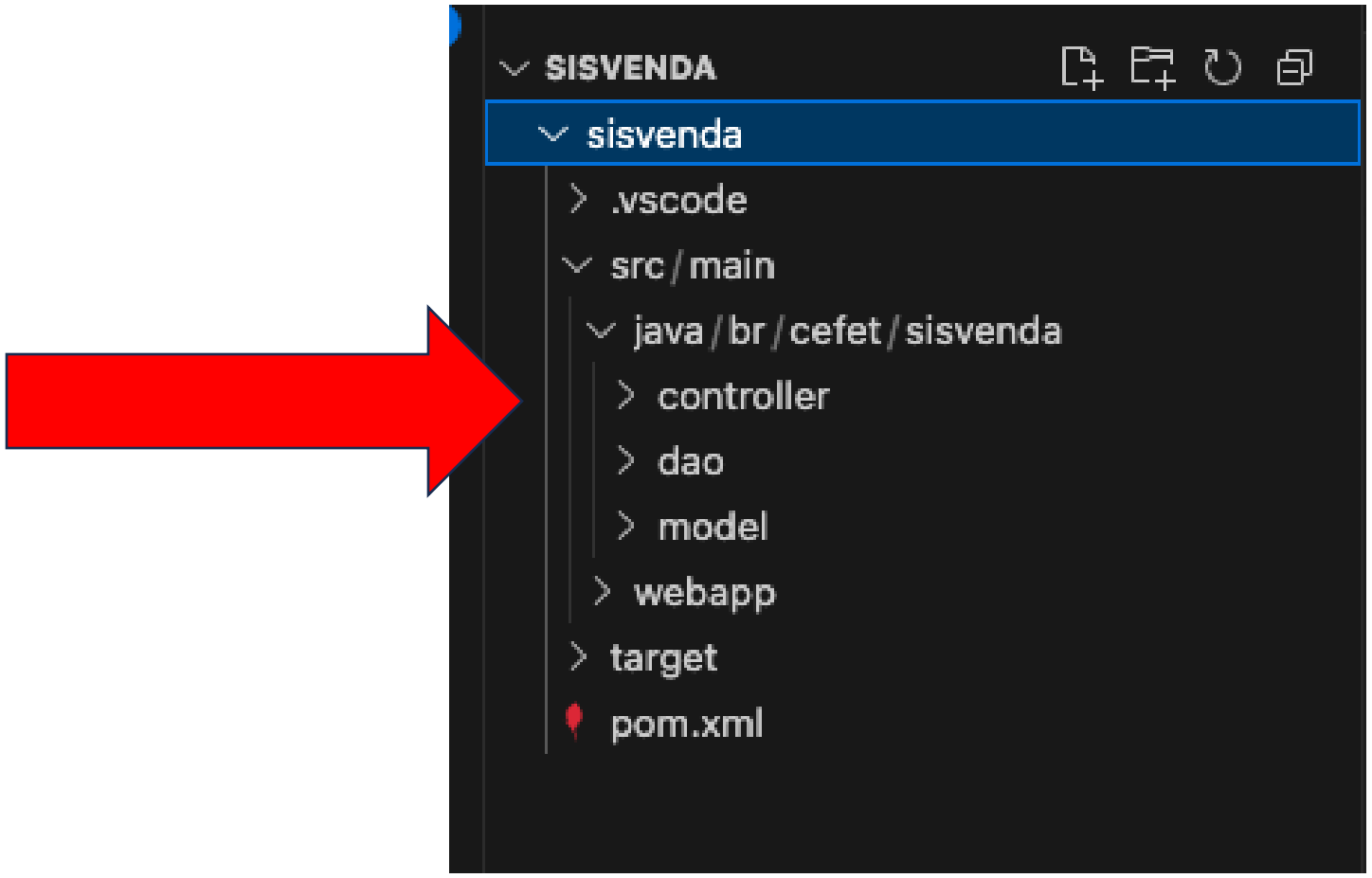
DAO – ProdutoDao (Listar)

0. O Método retorna um ArryList com Vários Produtos;
1. Executa o SQL e retorna o resultado do Select para dentro de ResultSet;
2. Instancia um ArrayList de Produtos;
3. Enquanto existir linhas para percorrer no ResultSet;
4. Instancia um Produto;
5. Atribui os valores dos campos SQL para os atributos do objeto;
6. Atribui os valores dos campos SQL para os atributos do objeto;
7. Adiciona o produto no ArrayList;
8. Retornar todos os produtos;

DAO – ProdutoDao (Listar com substring)

```
public List<Produto> listarTodos(String nome) throws SQLException {
    nome = "%" + nome + "%";
    String sql = "SELECT id, nome, valor, idcategoria"
    + " FROM produto WHERE nome LIKE ?";
    PreparedStatement stmt = con.prepareStatement(sql);
    stmt.setString(1, nome);
    ResultSet rs = stmt.executeQuery();

    Produto produto = null;
    List<Produto> produtos = new ArrayList<Produto>();

    while (rs.next()) {
        produto = new Produto();
        produto.setId(rs.getInt("id"));
        produto.setNome(rs.getString("nome"));
        produto.setValor(rs.getFloat("valor"));
        CategoriaDao catDao = new CategoriaDao();
        produto.setCategoria(catDao.listarUm(rs.getInt("idcategoria")));
        produtos.add(produto);
    }
    stmt.close();
    con.close();
    return produtos;}
```

Camada Controller



Servlet CategoriaServlet

- (Java Legado) - Não considerava a utilização de rotas padronizadas.
- Não existia JSON.
- Uma lista de objetos Java era encapsulada e despachada para view;
    - O método GET despacha um List<Categoria> para que um arquivo JSP possa listar os objetos em Java.

Verbo	Rota	Explicação
GET	/categorias?next=categorialistar.jsp	Lista todas as categorias. A variável next indica a página onde o despacho com a lista de categorias será realizado.
POST	/categorias	Inclui uma categoria

```
@WebServlet("/categorias")
public class CategoriaServlet extends HttpServlet {
}
```

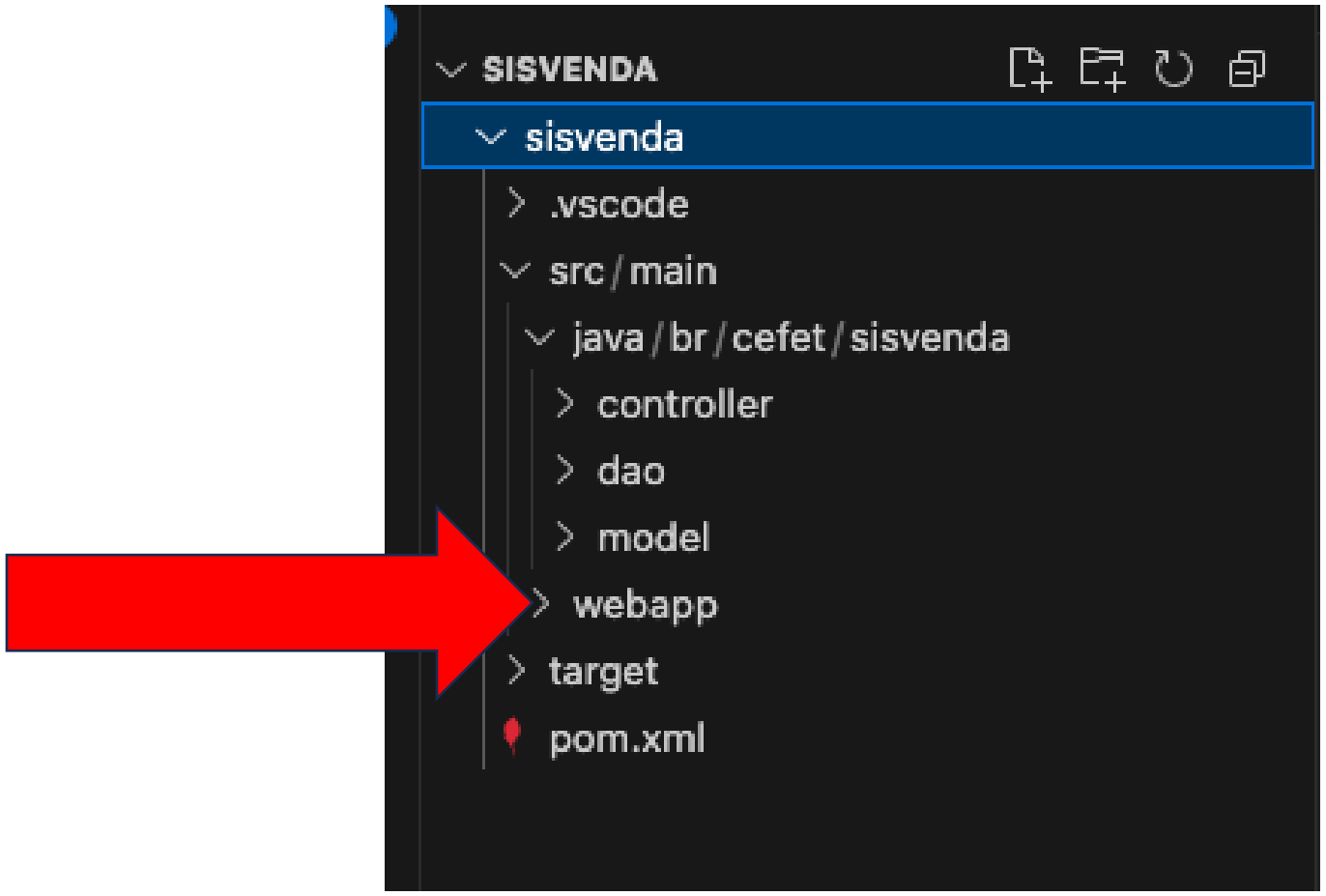
Verbo	Rota	Explicação
GET	/categorias?next=categorialistar.jsp	Lista todas as categorias.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{
        String next = request.getParameter("next");
        CategoriaDao catDao = new CategoriaDao();
        List<Categoria> categorias = null;
        try {
            categorias = catDao.listarTodos();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        request.setAttribute("categorias",categorias);
        RequestDispatcher rd = request.getRequestDispatcher(next);
        rd.forward(request, response);
}
```

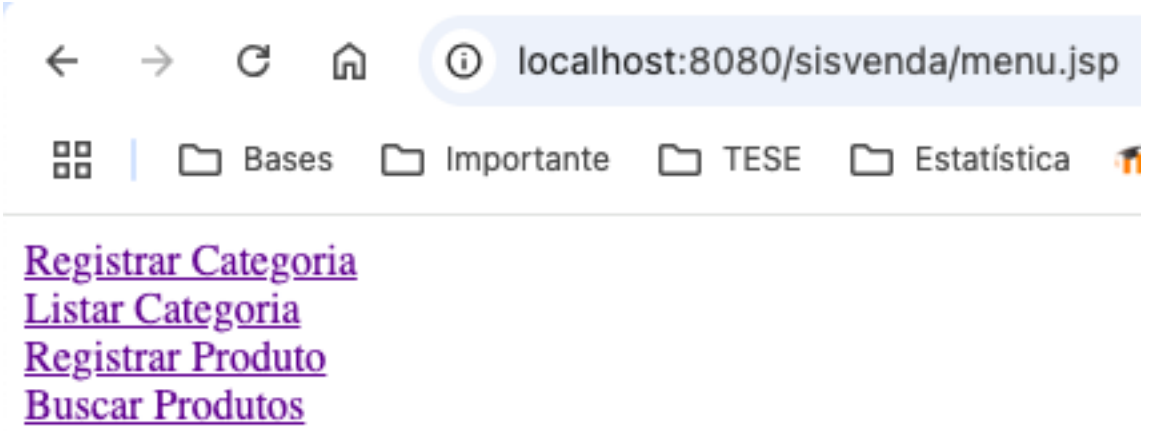
Verbo	Rota	Explicação
POST	/categorias	Incluir uma categoria

```
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException
{
    String nome = request.getParameter("nome");
    Categoria categoria = new Categoria();
    categoria.setNome(nome);
    CategoriaDao catDao = new CategoriaDao();
    try {
        catDao.inserir(categoria);
        }
    catch (SQLException e) {
        e.printStackTrace();
        }
}
```

Camada View


SISVENDA
sisvenda
.vscode
src / main
java / br / cefet / sisvenda
controller
dao
model
webapp
target
pom.xml

Menu (menu.jsp)



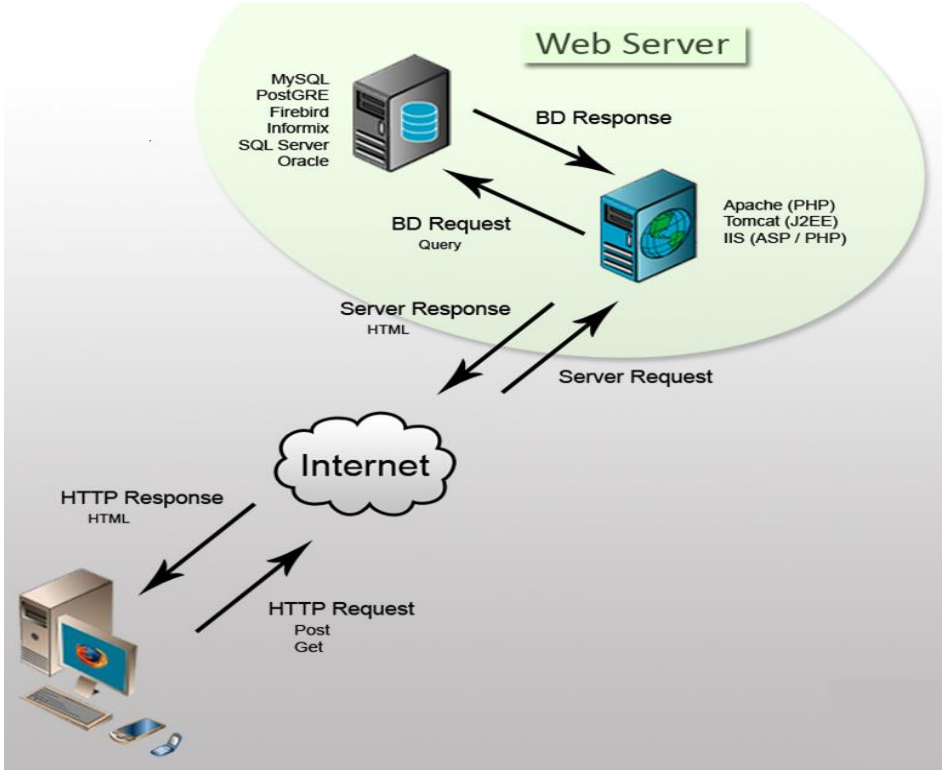

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
...
<title>SisVenda - Menu</title>
</head>
<body>
<a href="/sisvenda/categoriafrm.html"> Registrar Categoria</a> <br>
<a href="/sisvenda/categorias1?next=categorialistar.jsp"> Listar Categoria </a> <br>
<a href="/sisvenda/categorias1?next=produtofrm.jsp"> Registrar Produto<br>
```

Registrar Categoria (categoriafrm.html)


```
<meta charset="UTF-8">
<title> Registrar Categoria</title>
</head>
<body>
        <form action="/sisvenda/categorias" method="post">
            <p> Nome: <input type="text" name="nome"> </p>
            <input type="submit">
        </form>
</body>
```

Servlet – Despachando objetos para o JSP

- Uma utilizada era o despacho de objetos Java diretamente para as páginas (LEGADO). Porém, não é possível acessar um objeto em Java por meio de um simples HTML.
- Foi desenvolvido uma tecnologia que permite acessar esses objetos. Essa tecnologia é denominada JSP (Java Serve Page)
- JSP são páginas que utilizam as TAGs HTML mais alguns recursos para executar código Java.



JSTL (Java Server Pages Standard Template Library)

- As JSTLs são tags que auxiliam o programador com estruturas comuns de linguagens de programação, como o IF e o FOR.
- No cabeçalho da página JSP deveremos declarar o uso das TAGs:

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
```

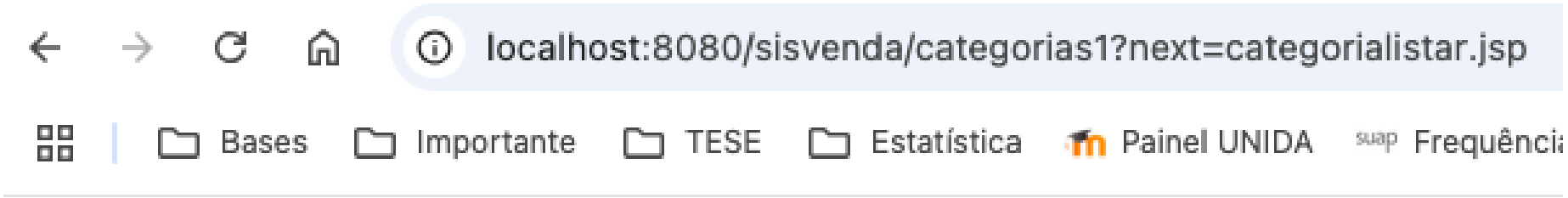
JSTL (Tags: set , if, out) – Exemplo

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<html> <head> <title> <c:if> Tag Example</title>
</head> <body>

<c:set var = "salary" scope = "session" value = "${2000*2}"/>
<c:if test = "${salary > 2000}">
        <p> My salary is: <c:out value = "${salary}"/><p>
</c:if>

</body> </html>
```

<c:set> Cria uma variável e atribui um valor;
<c:if> Define um condição de teste;
<c:out> Apresenta um valor na tela;

Camada View

Listar Categoria (categorialistar.jsp)



43 - Açougue

44 - Padaria

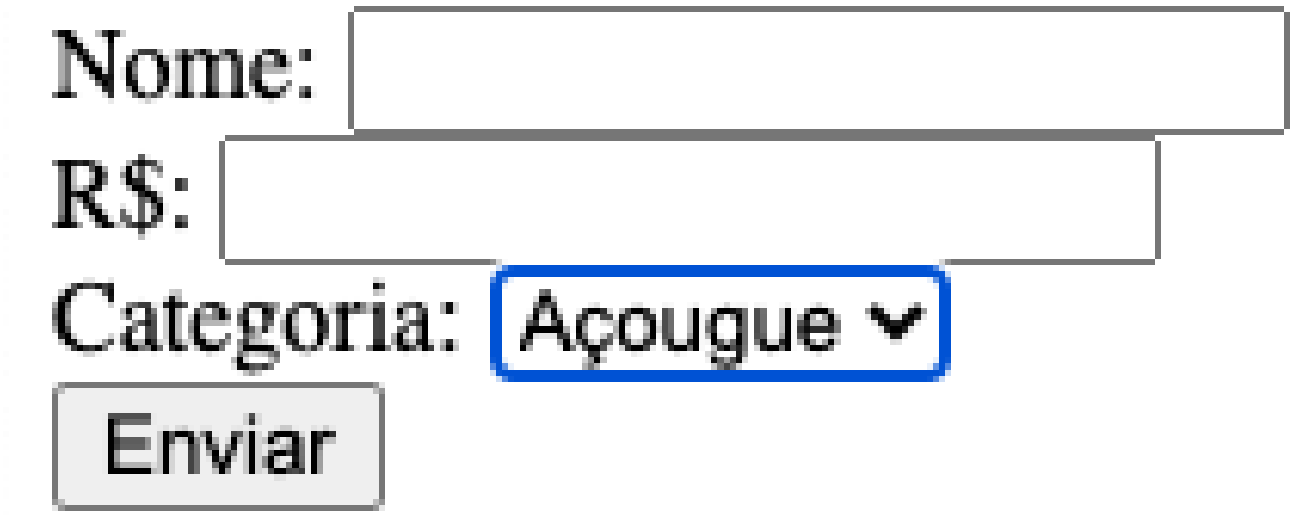
45 - Limpeza

Listar Categoria – categorialistar.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<!DOCTYPE html>
<html lang="pt">
<head>
<meta charset="UTF-8">
<title>Listar Categoria</title>
</head>
<body>
<c:forEach var="cat" items="${categorias}">
        <p> ${cat.id} - ${cat.nome} </p>
</c:forEach>
</body>
</html>
```

c:forEach: tag usada para percorrer um array de objetos.
items: lista de objetos para percorrer
var: objeto atual da lista

Camada View
Registrar Produto (produtofrm.jsp)

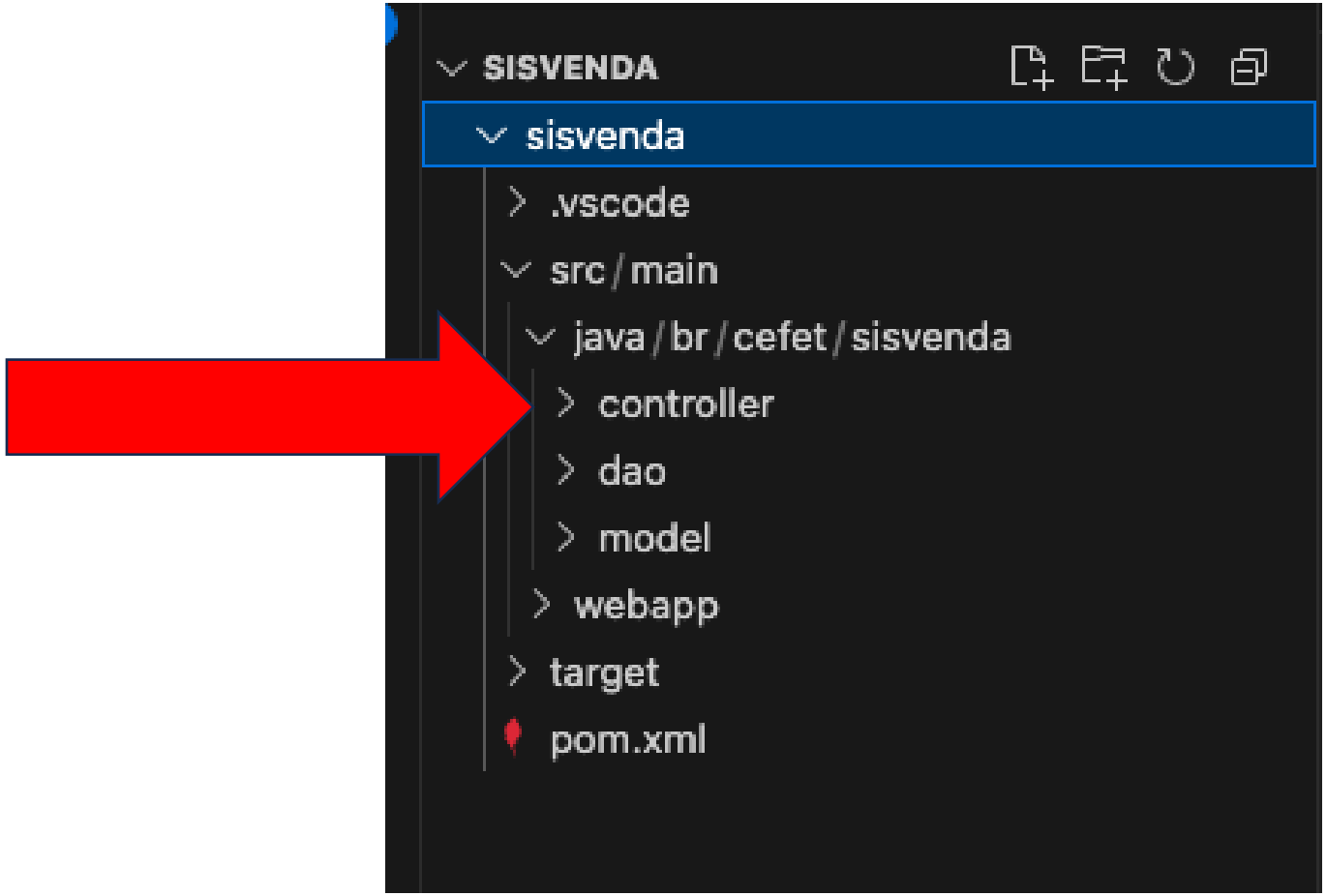


- Antes de entrar no produtofrm.jsp, a rota sisvenda/categorias1?next=categorialistar.jsp faz um despacho de todas as categorias para a view.

Registrar produto (produtofrm.jsp)

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<html>
<head>
<meta charset="UTF-8">
<title>Registrar Produto</title>
</head>
<body>
<form action="/sisvenda/produtos" method="post">
Nome: <input type="text" name="nome"> <br>
R$: <input type="text" name="valor"> <br>
Categoria: <select name="idCategoria">
<c:forEach var="categoria" items="${categorias}">
        <option value="${categoria.id}">${categoria.nome}</option>
</c:forEach>
</select> <br>
<input type="submit">
</form>
</body></html>
```

Camada Controller
ServletProduto



Servlet Produto

Verbo	Rota	Explicação
POST	/produtos	Inclui um produto.
GET	/produtos?q=texto	Listar produtos de acordo com uma string fornecida.

```
@WebServlet("/produtos")
public class ProdutoServlet extends HttpServlet {
}
```

ServletProduto (doPost)

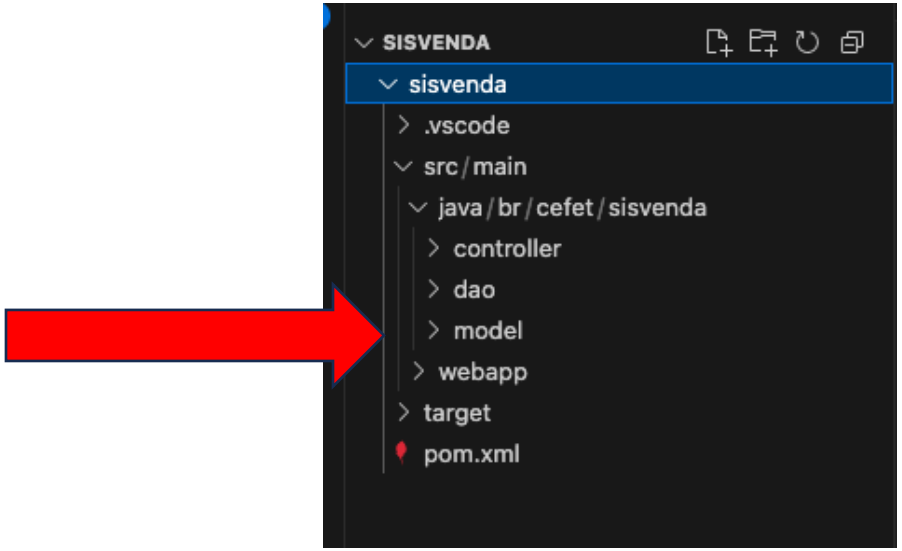
```
String nome = request.getParameter("nome");
float valor = Float.valueOf(request.getParameter("valor"));
int id = Integer.valueOf(request.getParameter("idCategoria"));
CategoriaDao catDao = new CategoriaDao();
Categoria categoria = null;
try {
        categoria = catDao.listarUm(id);
} catch (SQLException e) {
        e.printStackTrace();
}
Produto produto = new Produto();
produto.setNome(nome);
produto.setValor(valor);
produto.setCategoria(categoria);
ProdutoDao pDao = new ProdutoDao();
try {
        pDao.inserir(produto);
} catch (SQLException e) {
        e.printStackTrace();}
```

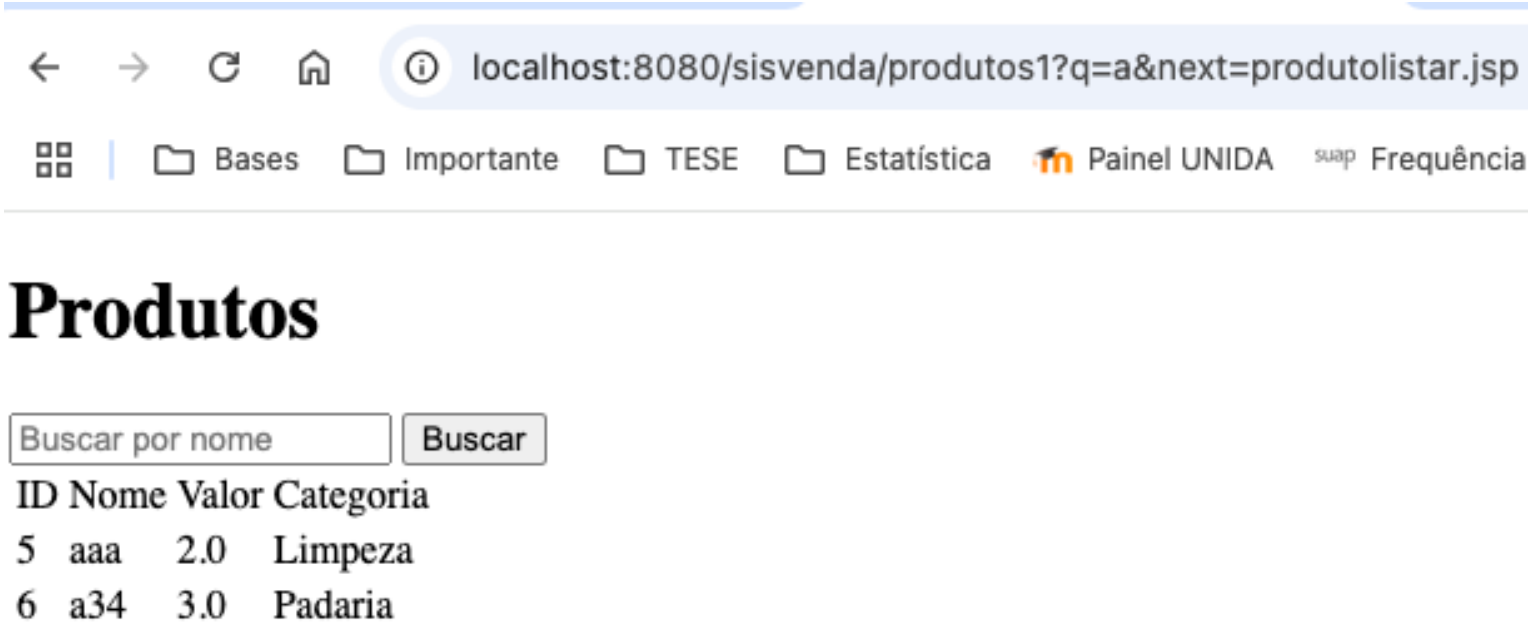
ServletProduto (doGet)

```
String query = request.getParameter("q");
String next = request.getParameter("next");
ProdutoDao prodDao = new ProdutoDao();
List<Produto> produtos = null;

try {
  produtos = prodDao.listar(query);
} catch (SQLException e) {
  e.printStackTrace();
}

request.setAttribute("produtos",produtos);
RequestDispatcher rd = request.getRequestDispatcher(next);
rd.forward(request, response);
```

Camada View

Listar Produtos (produtolistar.jsp)





Camada View

Listar Produtos (produtolistar.jsp)

```
<h1>Produtos</h1>
<form id="form-busca" method="get" action="/sisvenda/produtos1=">
    <input type="text" name="q" placeholder="Buscar por nome">
    <input type="hidden" name="next" value="produtolistar.jsp">
    <button type="submit">Buscar</button>
</form>
<table>
<tr>
    <td>ID</td>
    <td>Nome</td>
    <td>Valor</td>
    <td>Categoria</td>
</tr>
    <c:forEach var="p" items="${produtos}">
        <tr>
        <td>${p.id}</td>
        <td>${p.nome}</td>
        <td>${p.valor}</td>
        <td>${p.categoria.nome}</td>
        </tr>
    </c:forEach>
</table>
```