Estácio	**Universidade Estácio de Sá** Desenvolvimento Full Stack Nível 4 – Vamos integrar sistemas Turma 2022.3 – 3º Semestre
Nome:	Luan Augusto Vieira Bandeira
Repositório:	https://github.com/luanguto/nivel-4-mundo-3

Objetivos da prática

Implementar persistência com base em JPA.

Implementar regras de negócio na plataforma JEE, através de EJBs.

Implementar sistema cadastral Web com base em Servlets e JSPs.

Utilizar a biblioteca Bootstrap para melhoria do design.

No final do exercício, o aluno terá criado todos os elementos necessários para exibição e entrada de dados na plataforma Java Web, tornando-se capacitado para lidar com contextos reais de aplicação.

Procedimento 1 – Camadas de Persistência e Controle

Codigos:

ServletProduto.java

```
package cadastroee.servlet;


import cadastroee.dao.ProdutoDAO;
import cadastroee.model.Produto;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

@WebServlet (name = "servletProduto" , value = "/ServletProduto")
public class ServletProduto extends HttpServlet {
    private static final long serialVersionUID = 1L ;
    private ProdutoDAO produtoDAO = new ProdutoDAO();
    private Produto produto = new Produto();
    public ServletProduto() {super(); }
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException{
        String acao = request.getParameter("acao");
```

```
        if(acao != null && acao.equals("lista")) {
            lista(request, response);
        }



}
    protected void lista(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException{
        List<Produto> produtos = produtoDAO.lista();
        PrintWriter w = response.getWriter();
        produtos.forEach( p -> w.println(p.getNome()));
    }
}
```

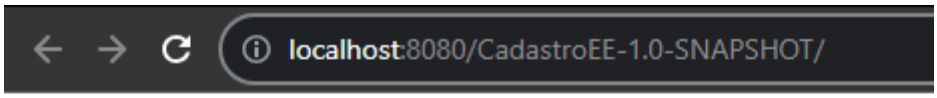
Produto.java

```
package cadastroee.model;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
public class Produto {
    private int idProduto;
    private String nome;
    private int quantidade;
    private float precoVenda;
}
```

ProdutoDAO.java

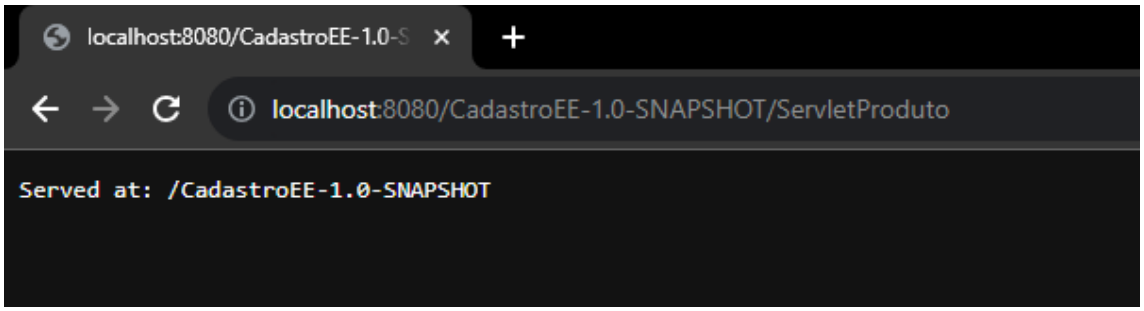
```
package cadastroee.dao;

import cadastroee.model.Produto;

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;


public class ProdutoDAO {
    private String driver =
"com.microsoft.sqlserver.jdbc.SQLServerDriver";
    private String url =
"jdbc:sqlserver://localhost:1433;database=Loja;user=Loja;password=Loja
;encrypt=true;trustServerCertificate=true;loginTimeout=10;";

    private Connection conectar() {
        Connection con = null;
```

```
        try {
            Class.forName(driver);
            con = DriverManager.getConnection(url);
            return con;
        } catch (Exception e) {
            System.out.println(e);
            return null;
        }
    }

    public List<Produto> lista() {
        List<Produto> produtos = new ArrayList<>();
        String read = "select * from produtos";
        try {
            Connection con = conectar();
            PreparedStatement prepared = con.prepareStatement(read);
            ResultSet res = prepared.executeQuery();

            while (res.next()) {
                int idProduto = res.getInt("idProduto");
                String nome = res.getString("nome");
                int quantidade = res.getInt("quantidade");
                float precoVenda = res.getFloat("precoVenda");
                produtos.add(new Produto(idProduto, nome, quantidade,
precoVenda));
            }

            con.close();
            return produtos;

        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```



Hello World!

Hello Servlet



Análise e Conclusão:

Como é organizado um projeto corporativo no NetBeans?

Um projeto corporativo no NetBeans é organizado como um projeto de aplicação empresarial que contém módulos EJB e projetos de aplicativos web.

Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?

JPA é usada para mapeamento objeto-relacional e EJB para lógica de negócios e serviços de back-end.

Como o NetBeans viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?

NetBeans fornece ferramentas e assistentes para criar e gerenciar entidades JPA e EJBs, melhorando a eficiência do desenvolvedor.

O que são Servlets, e como o NetBeans oferece suporte à construção desse tipo de componentes em um projeto Web?

Servlets são componentes do lado do servidor para processamento de solicitações web, e o NetBeans oferece assistentes e modelos para criar e configurar Servlets facilmente.

Como é feita a comunicação entre os Serlvets e os Session Beans do pool de EJBs?

A comunicação é realizada através de injeção de dependência usando a anotação @EJB ou por meio de lookup JNDI, ambos suportados pelo NetBeans.

2º Procedimento

ServletProdutoFC.java

```
package cadastroee.servlet;

import cadastroee.controller.ProdutoFacadeLocal;
import cadastroee.model.Produto;

import jakarta.ejb.EJB;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

@WebServlet("/ServletProdutoFC")
public class ServletProdutoFC extends HttpServlet {

    @EJB
    ProdutoFacadeLocal facade;

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");

        String acao = request.getParameter("acao");
        String destino = null;

        if (acao != null) {
            switch (acao) {
                case "listar":
                    List<Produto> produtos = facade.findAll();
                    request.setAttribute("produtos", produtos);
                    destino = "ProdutoLista.jsp";
                    break;

                case "formAlterar":
                    int idAlterar =
Integer.parseInt(request.getParameter("id"));
                    Produto produtoAlterar = facade.find(idAlterar);
                    request.setAttribute("produto", produtoAlterar);
                    destino = "ProdutoDados.jsp";
                    break;

                case "formIncluir":
                    destino = "ProdutoDados.jsp";
                    break;

                case "excluir":
                    int idExcluir =
Integer.parseInt(request.getParameter("id"));
                    facade.remove(facade.find(idExcluir));
                    produtos = facade.findAll();
                    request.setAttribute("produtos", produtos);
```

```
                    destino = "ProdutoLista.jsp";
                    break;

                case "alterar":
                    int idUpdate =
Integer.parseInt(request.getParameter("id"));
                    Produto produtoUpdate = facade.find(idUpdate);

produtoUpdate.setNome(request.getParameter("nome"));

produtoUpdate.setQuantidade(Integer.parseInt(request.getParameter("qua
ntidade")));

produtoUpdate.setPrecoVenda(Float.parseFloat(request.getParameter("pre
coVenda")));
                    facade.edit(produtoUpdate);
                    produtos = facade.findAll();
                    request.setAttribute("produtos", produtos);
                    destino = "ProdutoLista.jsp";
                    break;

                case "incluir":
                    Produto novoProduto = new Produto();
                    novoProduto.setNome(request.getParameter("nome"));

novoProduto.setQuantidade(Integer.parseInt(request.getParameter("quant
idade")));

novoProduto.setPrecoVenda(Float.parseFloat(request.getParameter("preco
Venda")));
                    facade.create(novoProduto);
                    produtos = facade.findAll();
                    request.setAttribute("produtos", produtos);
                    destino = "ProdutoLista.jsp";
                    break;
            }
        }

        if (destino != null) {
            RequestDispatcher dispatcher =
request.getRequestDispatcher(destino);
            dispatcher.forward(request, response);
        }
    }

    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
```

```
        return "Servlet de controle para operações de Produto";
    }
}
```

ProdutoLista.jsp

```
<%@ page import="java.util.List" %>
<%@ page import="cadastroee.model.Produto" %>
<%@ page import="java.text.DecimalFormat" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Lista de Produtos</title>
</head>
<body>

<h1>Lista de Produtos</h1>

<a href="ServletProdutoFC?acao=formIncluir">Incluir Novo Produto</a>

<%
    List<Produto> produtos = (List<Produto>)
request.getAttribute("produtos");

    if (produtos != null && !produtos.isEmpty()) {
%>
<table border="1">
    <tr>
        <th>ID</th>
        <th>Nome</th>
        <th>Quantidade</th>
        <th>Preço</th>
        <th>Ações</th>
    </tr>
    <% for (Produto produto : produtos) { %>
    <tr>
        <td><%= produto.getId() %></td>
        <td><%= produto.getNome() %></td>
        <td><%= produto.getQuantidade() %></td>
        <td><%= new
DecimalFormat("#.##").format(produto.getPrecoVenda()) %></td>
        <td>
            <a href="ServletProdutoFC?acao=formAlterar&id=<%=
produto.getId() %>">Alterar</a>
            |
            <a href="ServletProdutoFC?acao=excluir&id=<%=
produto.getId() %>">Excluir</a>
        </td>
    </tr>
    <% } %>
</table>
<%
} else {
%>
<p>Nenhum produto encontrado.</p>
<%
    }
%>
```

```
</body>
</html>
```

ProdutosDados.jsp

```
<%@ page import="cadastroee.model.Produto" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Cadastro de Produto</title>
</head>
<body>
<%
    Produto produto = (Produto) request.getAttribute("produto");
    String acao = (produto == null) ? "incluir" : "alterar";
%>
<h2><%= produto == null ? "Incluir Produto" : "Alterar Produto"
%></h2>
<form action="ServletProdutoFC" method="post">
    <!-- Campo oculto para definir a ação a ser realizada -->
    <input type="hidden" name="acao" value="<%= acao %>">

    <!-- Campo oculto para enviar o ID do produto quando a ação for
alterar -->
    <% if ("alterar".equals(acao)) { %>
    <input type="hidden" name="id" value="<%= produto.getId() %>">
    <% } %>

    <div>
        <label for="nome">Nome:</label>
        <input type="text" id="nome" name="nome" value="<%= produto !=
null ? produto.getNome() : "" %>" required>
    </div>

    <div>
        <label for="quantidade">Quantidade:</label>
        <input type="number" id="quantidade" name="quantidade"
value="<%= produto != null ? produto.getQuantidade() : "" %>"
required>
    </div>

    <div>
        <label for="precoVenda">Preço de Venda:</label>
        <input type="text" id="precoVenda" name="precoVenda"
value="<%= produto != null ? produto.getPrecoVenda() : "" %>"
required>
    </div>

    <div>
        <input type="submit" value="<%= produto == null ? "Incluir
Produto" : "Salvar Alterações" %>">
    </div>
</form>
</body>
</html>
```

Análise e Conclusão:

Como funciona o padrão Front Controller, e como ele é implementado em um aplicativo Web Java, na arquitetura MVC?

É um Servlet que centraliza o processamento de requisições em aplicações MVC, direcionando para os controladores adequados.

Quais as diferenças e semelhanças entre Servlets e JSPs?

Servlets lidam com lógica de negócios; JSPs focam na apresentação. Ambos rodam no servidor e podem gerar conteúdo dinâmico.

Qual a diferença entre um redirecionamento simples e o uso do método forward, a partir do RequestDispatcher? Para que servem parâmetros e atributos nos objetos HttpRequest?

sendRedirect inicia um novo pedido e muda a URL; forward continua o mesmo pedido sem mudar a URL. Parâmetros são dados de entrada do pedido; atributos são usados para passar dados entre as operações no servidor durante o processamento da requisição.

3º Procedimento

ProdutoLista.jps

```
<%@ page import="java.util.List" %>
<%@ page import="cadastroee.model.Produto" %>
<%@ page import="java.text.DecimalFormat" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.
min.css" rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bu
ndle.min.js" integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
crossorigin="anonymous"></script>
    <title>Lista de Produtos</title>
</head>
<body class="container">

<h1>Lista de Produtos</h1>

<a href="ServletProdutoFC?acao=formIncluir" class="btn btn-primary m-
2">Incluir Novo Produto</a>


<%
    List<Produto> produtos = (List<Produto>)
request.getAttribute("produtos");

    if (produtos != null && !produtos.isEmpty()) {
%>
<table class="table table-striped">
    <thead class="table-dark">
    <tr>
        <th>ID</th>
        <th>Nome</th>
        <th>Quantidade</th>
        <th>Preço</th>
        <th>Ações</th>
    </tr>
    </thead>
    <tbody>
        <% for (Produto produto : produtos) { %>
    <tr>
        <td><%= produto.getId() %></td>
        <td><%= produto.getNome() %></td>
        <td><%= produto.getQuantidade() %></td>
        <td><%= new
DecimalFormat("#.##").format(produto.getPrecoVenda()) %></td>
        <td>
            <a href="ServletProdutoFC?acao=formAlterar&id=<%=
produto.getId() %>" class="btn btn-primary btn-sm">Alterar</a>
            |
            <a href="ServletProdutoFC?acao=excluir&id=<%=
produto.getId() %>" class="btn btn-danger btn-sm">Excluir</a>
        </td>
```

```
    </tr>
    <% } %>
</table>
<%
} else {
%>
<p>Nenhum produto encontrado.</p>
<%
    }
%>

</body>
</html>
```

ProdutoDados.jps

```
<%@ page import="cadastroee.model.Produto" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.
min.css" rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bu
ndle.min.js" integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
crossorigin="anonymous"></script>
    <title>Cadastro de Produto</title>
</head>
<body class="container">
<%
    Produto produto = (Produto) request.getAttribute("produto");
    String acao = (produto == null) ? "incluir" : "alterar";
%>
<h2><%= produto == null ? "Incluir Produto" : "Alterar Produto"
%></h2>
<form action="ServletProdutoFC" method="post" class="form">
    <input type="hidden" name="acao" value="<%= acao %>">

    <% if ("alterar".equals(acao)) { %>
    <input type="hidden" name="id" value="<%= produto.getId() %>">
    <% } %>

    <div class="mb-3">
        <label for="nome" class="form-label">Nome:</label>
        <input type="text" id="nome" name="nome" class="form-control"
value="<%= produto != null ? produto.getNome() : "" %>" required>
    </div>

    <div class="mb-3">
        <label for="quantidade" class="form-label">Quantidade:</label>
        <input type="number" id="quantidade" name="quantidade"
class="form-control" value="<%= produto != null ?
produto.getQuantidade() : "" %>" required>
    </div>

    <div class="mb-3">
```

```
        <label for="precoVenda" class="form-label">Preço de
Venda:</label>
        <input type="text" id="precoVenda" name="precoVenda"
class="form-control" value="<%= produto != null ?
produto.getPrecoVenda() : "" %>" required>
    </div>

    <div class="mb-3">
        <input type="submit" class="btn btn-primary" value="<%=
produto == null ? "Incluir Produto" : "Salvar Alterações" %>">
    </div>
</form>
</body>
</html>
```

Análise e Conclusão:

Como o framework Bootstrap é utilizado?

O framework Bootstrap é utilizado para o desenvolvimento de interfaces web. Ele oferece um conjunto de folhas de estilo CSS e componentes JavaScript para criar elementos de interface de usuário como formulários, botões, navegação, e sistemas de grid, com design responsivo e compatibilidade entre navegadores.

Por que o Bootstrap garante a independência estrutural do HTML?

O Bootstrap promove independência estrutural ao separar o design (CSS) do conteúdo (HTML), permitindo que os desenvolvedores mudem o design sem alterar o HTML. As classes predefinidas do Bootstrap podem ser aplicadas a qualquer elemento HTML sem a necessidade de estilos CSS adicionais.

Qual a relação entre o Boostrap e a responsividade da página?

O Bootstrap é fundamentalmente ligado à responsividade de páginas, oferecendo um sistema de grid flexível e classes de utilidade que ajustam os elementos da página de acordo com o tamanho da tela do dispositivo. Isso facilita a criação de sites que se adaptam bem a dispositivos móveis, tablets e desktops.