- DESENVOLVIMENTO FULL STACK- TURMA 9001
- Disciplina: RPG0017 - Vamos Integrar Sistemas
- Semestre Letivo: 2025.01
- Repositorio Git: https://github.com/Elena-Gudimenko/Missao-4-Mundo-3.git
- ELENA VICTOROVNA GUDIMENKO, MATRICULA: 2024.0277.9826

Missão Prática | Nível 4| Mundo 3

Missão Prática | Nível 4 | Mundo 3

Implementação de sistema cadastral com interface Web, baseado nas tecnologias deServlets, JPA e JEE.

Procedimento 1: Camadas de Persistência e Controle

Procedimento 2: Interface Cadastral com Servlet e JSPs

Procedimento 3: Melhorando o Design da Interface

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.

Códigos:

Procedimento 1: Camadas de Persistência e Controle

- **ServletProduto.java**

/*

 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this
template
 */
package cadastroee.servlets;

import cadastroee.controller.ProdutoFacadeLocal;
import jakarta.ejb.EJB;
import java.io.IOException;
import java.io.PrintWriter;
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.util.List;

@WebServlet(name = "ServletProduto", urlPatterns = {"/ServletProduto"})
public class ServletProduto extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @EJB
    private ProdutoFacadeLocal facade;

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

```
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Lista de Produtos</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Lista de Produtos " + request.getContextPath() + "</h1>");

            try {
            List<cadastroee.model.Produto> produtos = facade.findAll(); // импорт нужен!
            out.println("<ul>");
            for (cadastroee.model.Produto p : produtos) {
                out.println("<li>" + p.getNome() + "</li>"); // предполагаем, что у Produto есть
getNome()
            }
            out.println("</ul>");
        } catch (Exception e) {
            out.println("<p>Erro ao recuperar produtos: " + e.getMessage() + "</p>");
        }

            out.println("</body>");
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
```

```
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Procedimento 2: Interface Cadastral com Servlet e JSPs

- ***ServletProduto.java***

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
 */
package cadastroee.servlets;

import cadastroee.controller.ProdutoFacadeLocal;
import jakarta.ejb.EJB;
import java.io.IOException;
import java.io.PrintWriter;
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.util.List;

@WebServlet(name = "ServletProduto", urlPatterns = {"/ServletProduto"})
public class ServletProduto extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
```

```
    @EJB
    private ProdutoFacadeLocal facade;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Lista de Produtos</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Lista de Produtos " + request.getContextPath() + "</h1>");

            try {
            List<cadastroee.model.Produto> produtos = facade.findAll(); // импорт нужен!
            out.println("<ul>");
            for (cadastroee.model.Produto p : produtos) {
                out.println("<li>" + p.getNome() + "</li>"); // предполагаем, что у Produto есть
getNome()
            }
            out.println("</ul>");
        } catch (Exception e) {
            out.println("<p>Erro ao recuperar produtos: " + e.getMessage() + "</p>");
        }

            out.println("</body>");
            out.println("</html>");
        }
    }
```

```
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
```

```
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

- ***ServletProdutoFrontController.java***

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
 */
package cadastroee.servlets;

import cadastroee.controller.ProdutoFacadeLocal;
import cadastroee.model.Produto;
import jakarta.ejb.EJB;
import jakarta.servlet.RequestDispatcher;
import java.io.IOException;
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.util.List;


@WebServlet(name = "ServletProdutoFrontController", urlPatterns = {"/ServletProdutoFC"})
public class ServletProdutoFrontController extends HttpServlet {
```

```
@EJB
private ProdutoFacadeLocal facade;


protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {


    request.setCharacterEncoding("UTF-8");


    String acao = request.getParameter("acao");
    if (acao == null) {
        acao = "listar";
    }


    String destino;


    if (acao.equals("listar")) {
        List<Produto> produtos = facade.findAll();
        request.setAttribute("produtos", produtos);
        destino = "ProdutoLista.jsp";
    } else if (acao.equals("formIncluir")) {
        destino = "ProdutoDados.jsp";


    } else if (acao.equals("formAlterar")) {
        String idStr = request.getParameter("id");
        if (idStr != null) {
            Integer id = Integer.valueOf(idStr);
            Produto produto = facade.find(id);
            request.setAttribute("produto", produto);
        }
        destino = "ProdutoDados.jsp";
```

```
    } else if (acao.equals("excluir")) {
        String idStr = request.getParameter("id");
        if (idStr != null) {
            Integer id = Integer.valueOf(idStr);
            Produto produto = facade.find(id);
            if (produto != null) {
                facade.remove(produto);
            }
        }
        List<Produto> produtos = facade.findAll();
        request.setAttribute("produtos", produtos);
        destino = "ProdutoLista.jsp";

    } else if (acao.equals("alterar")) {
        String idStr = request.getParameter("id");
        if (idStr != null) {
            Integer id = Integer.valueOf(idStr);
            Produto produto = facade.find(id);
            if (produto != null) {
                produto.setNome(request.getParameter("nome"));

produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
                produto.setPrecoVenda(Float.valueOf(request.getParameter("precoVenda")));
                facade.edit(produto);
            }
        }
        List<Produto> produtos = facade.findAll();
        request.setAttribute("produtos", produtos);
        destino = "ProdutoLista.jsp";

    } else if (acao.equals("incluir")) {
```

```
            Produto produto = new Produto();
            produto.setNome(request.getParameter("nome"));
            produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
            produto.setPrecoVenda(Float.parseFloat(request.getParameter("precoVenda")));
            facade.create(produto);
            List<Produto> produtos = facade.findAll();
            request.setAttribute("produtos", produtos);
            destino = "ProdutoLista.jsp";

        } else {
            List<Produto> produtos = facade.findAll();
            request.setAttribute("produtos", produtos);
            destino = "ProdutoLista.jsp";
        }


        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 "Short description";
    }// </editor-fold>
}
```

- ***ProdutoLista.jsp***

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Lista de Produtos</title>
    <style>
        table {
            border-collapse: collapse;
            width: 80%;
            margin: 20px auto;
        }
        th, td {
            border: 1px solid #aaa;
            padding: 8px;
            text-align: center;
        }
        th {
            background-color: #eee;
        }
        a {
            text-decoration: none;
            color: blue;
        }
        a:hover {
            text-decoration: underline;
```

```
        }
        .incluir-link {
            margin: 20px auto;
            display: block;
            width: 80%;
            text-align: right;
        }
    </style>
</head>
<body>

<h2 style="text-align: center;">Lista de Produtos</h2>

<div class="incluir-link">
    <a href="ServletProdutoFC?acao=formIncluir">Incluir Novo Produto</a>
</div>

<table>
    <thead>
        <tr>
            <th>ID</th>
            <th>Nome</th>
            <th>Quantidade</th>
            <th>Preço Venda</th>
            <th>Ações</th>
        </tr>
    </thead>
    <tbody>
    <c:forEach var="produto" items="${produtos}">
        <tr>
            <td>${produto.id}</td>
            <td>${produto.nome}</td>
```

```
            <td>${produto.quantidade}</td>
            <td>${produto.precoVenda}</td>
            <td>
               <a href="ServletProdutoFC?acao=formAlterar&id=${produto.id}">Alterar</a> |
               <a href="ServletProdutoFC?acao=excluir&id=${produto.id}"
                  onclick="return confirm('Confirma exclusão do produto
${produto.nome}?');">Excluir</a>
            </td>
         </tr>
      </c:forEach>
      </tbody>
   </table>

   </body>
   </html>
```

- ***ProdutoDados.jsp***

```
<%@page import="cadastroee.model.Produto"%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
   <title>Cadastro de Produto</title>
   <style>
      form {
         width: 400px;
         margin: 30px auto;
         border: 1px solid #ccc;
         padding: 20px;
```

```
            border-radius: 8px;
            background-color: #f9f9f9;
        }
        label, input {
            display: block;
            width: 100%;
            margin-bottom: 10px;
        }
        input[type="submit"] {
            width: auto;
            padding: 8px 16px;
        }
    </style>
</head>
<body>

<%
    // Определим переменные Java
    Produto produto = (Produto) request.getAttribute("produto");
    String acao = (produto == null) ? "incluir" : "alterar";
%>

<h2 style="text-align: center;">
    <%= (acao.equals("incluir") ? "Incluir Novo Produto" : "Alterar Produto") %>
</h2>

<form action="ServletProdutoFC" method="post">
    <input type="hidden" name="acao" value="<%= acao %>"/>

    <c:if test="${produto != null}">
        <input type="hidden" name="id" value="${produto.id}"/>
    </c:if>
```

```
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome" value="${produto.nome}" required/>

    <label for="quantidade">Quantidade:</label>
    <input type="number" id="quantidade" name="quantidade"
value="${produto.quantidade}" required/>

    <label for="precoVenda">Preço de Venda:</label>
    <input type="number" id="precoVenda" name="precoVenda" step="0.01"
value="${produto.precoVenda}" required/>

    <input type="submit" value="<%= (acao.equals("incluir") ? "Incluir Produto" : "Salvar
Alterações") %>"/>
</form>

</body>
</html>
```

Procedimento 3: Melhorando o Design da Interface

- ***ProdutoLista.jsp***

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Lista de Produtos</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.6/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
4Q6Gf2aSP4eDXB8Miphtr37CMZZQ5oXLH2yaXMJ2w8e2ZtHTl7GptT4jmndRuHDT"
crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.6/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
j1CDi7MgGQ12Z7Qab0qlWQ/Qqz24Gc6BM0thvEMVjHnfYGF0rmFCozFSxQBxwHKO"
crossorigin="anonymous"></script>
```

```
</head>
<body class = "container">

<h2 style="text-align: center;">Lista de Produtos</h2>

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

<table class="table table-striped">
    <thead class="table-dark">
        <tr>
            <th>ID</th>
            <th>Nome</th>
            <th>Quantidade</th>
            <th>Preço Venda</th>
            <th>Ações</th>
        </tr>
    </thead>
    <tbody>
    <c:forEach var="produto" items="${produtos}">
        <tr>
            <td>${produto.id}</td>
            <td>${produto.nome}</td>
            <td>${produto.quantidade}</td>
            <td>${produto.precoVenda}</td>
            <td>
                <a href="ServletProdutoFC?acao=formAlterar&id=${produto.id}" class="btn btn-primary btn-sm">Alterar</a> |
                <a href="ServletProdutoFC?acao=excluir&id=${produto.id}"
                   class="btn btn-danger btn-sm"
```

```
                onclick="return confirm('Confirma exclusão do produto
${produto.nome}?');">Excluir</a>
            </td>
        </tr>
    </c:forEach>
    </tbody>
</table>

</body>
</html>
```

- ***ProdutoDados.jsp***

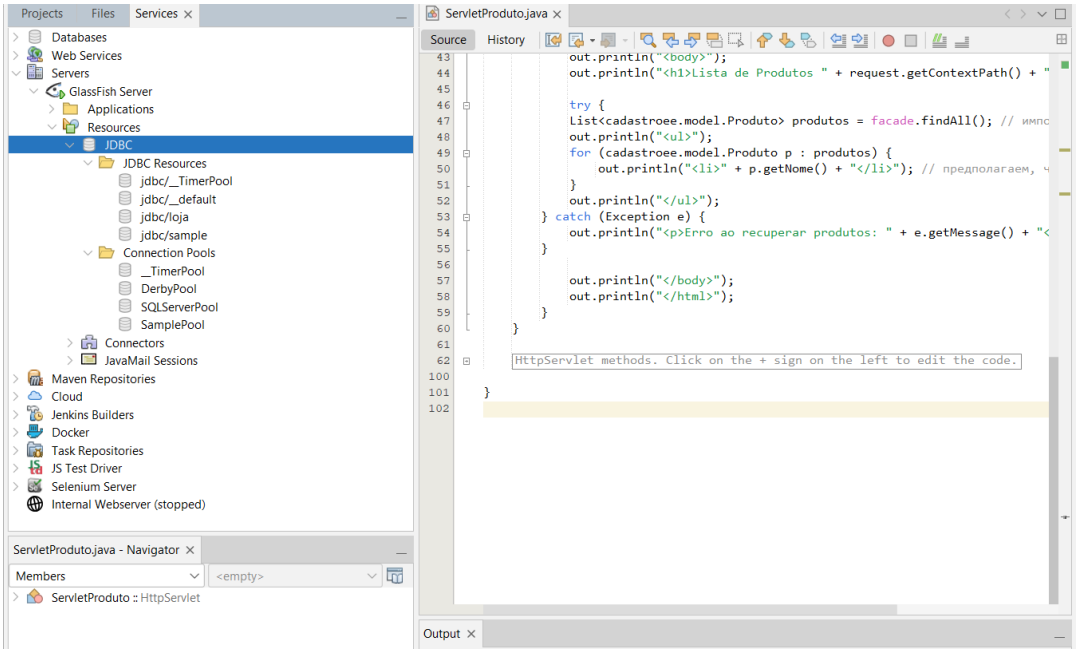
```
<%@page import="cadastroee.model.Produto"%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Cadastro de Produto</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.6/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
4Q6Gf2aSP4eDXB8Miphtr37CMZZQ5oXLH2yaXMJ2w8e2ZtHTl7GptT4jmndRuHDT"
crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.6/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
j1CDi7MgGQ12Z7Qab0qlWQ/Qqz24Gc6BM0thvEMVjHnfYGF0rmFCozFSxQBxwHKO"
crossorigin="anonymous"></script>
</head>
<body class="container">

<%
    // Определим переменные Java
    Produto produto = (Produto) request.getAttribute("produto");
    String acao = (produto == null) ? "incluir" : "alterar";
```

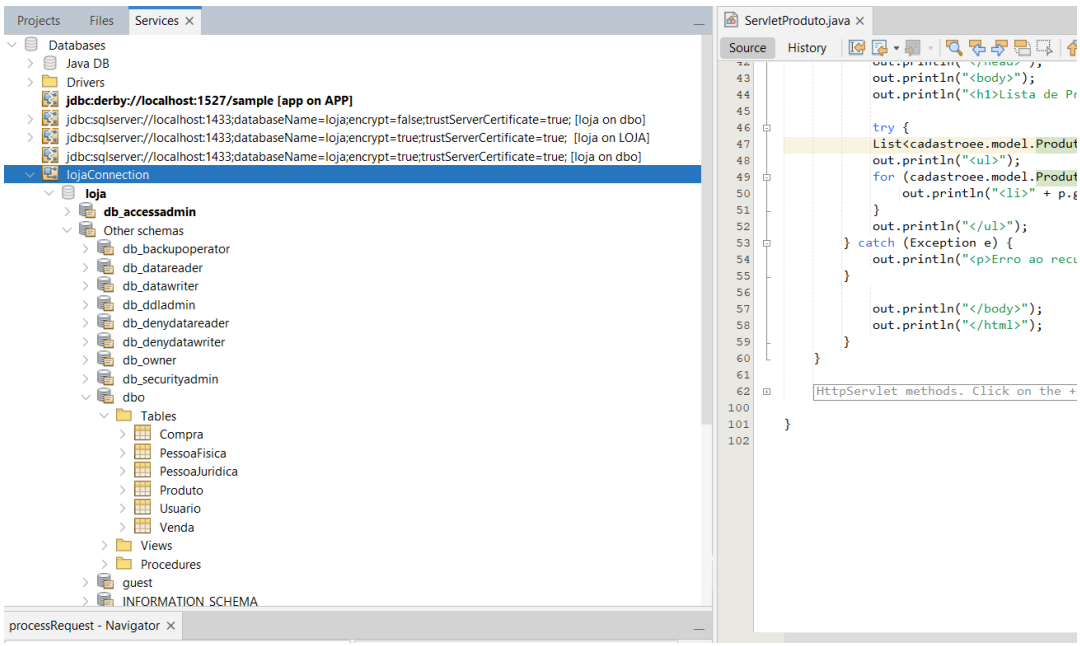
```
%>

<h2 style="text-align: center;">
    <%= (acao.equals("incluir") ? "Incluir Novo Produto" : "Alterar Produto") %>
</h2>

<form action="ServletProdutoFC" method="post" class="mx-auto mt-4 p-4 border rounded bg-light" style="max-width: 400px;">
    <input type="hidden" name="acao" value="<%= acao %>"/>

    <c:if test="${produto != null}">
        <input type="hidden" name="id" value="${produto.id}"/>
    </c:if>

    <div class="mb-3">
        <label for="nome" class="form-label">Nome:</label>
        <input type="text" id="nome" name="nome" value="${produto.nome}" required class="form-control"/>
    </div>

    <div class="mb-3">
        <label for="quantidade" class="form-label">Quantidade:</label>
        <input type="number" id="quantidade" name="quantidade" value="${produto.quantidade}" required class="form-control"/>
    </div>

    <div class="mb-3">
        <label for="precoVenda" class="form-label">Preço de Venda:</label>
        <input type="number" id="precoVenda" name="precoVenda" step="0.01" value="${produto.precoVenda}" required class="form-control"/>
    </div>

    <button type="submit" class="btn btn-primary">
```

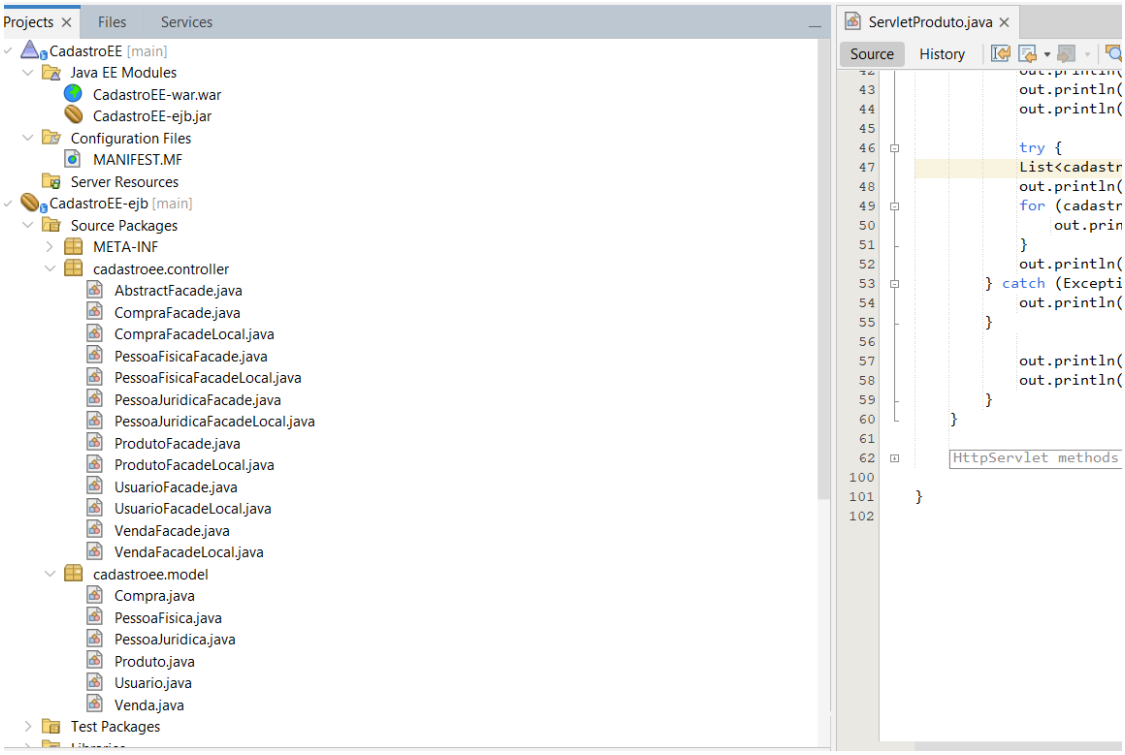
```
        <%= (acao.equals("incluir") ? "Incluir Produto" : "Salvar Alterações") %>
    </button>
</form>
</body>
</html>
```

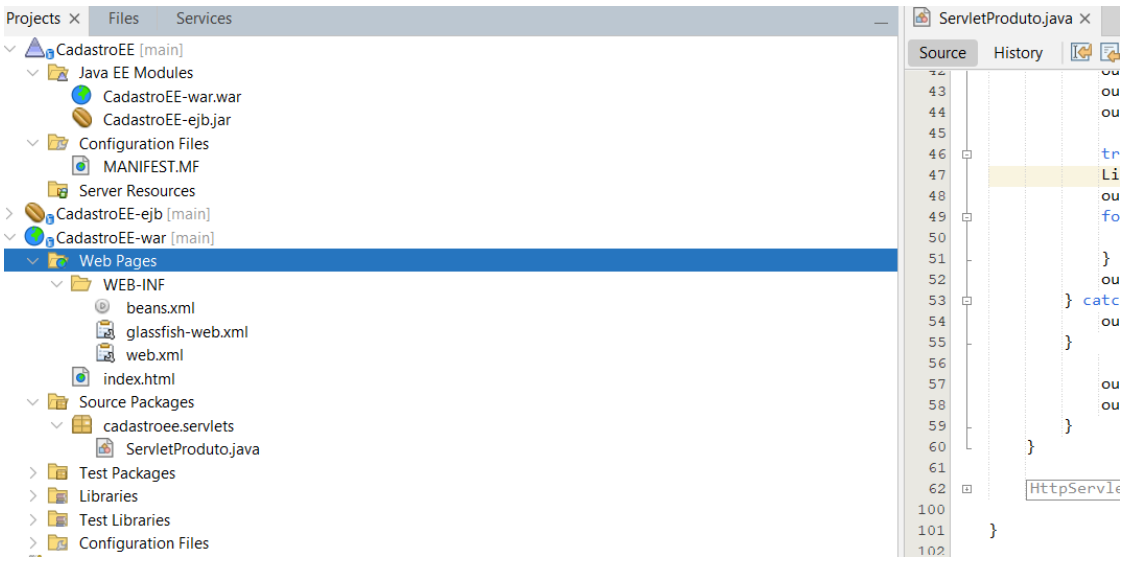
Resultados:

Procedimento 1: https://github.com/Elena-Gudimenko/Missao-4-Mundo-3/tree/main/Procedimento_01

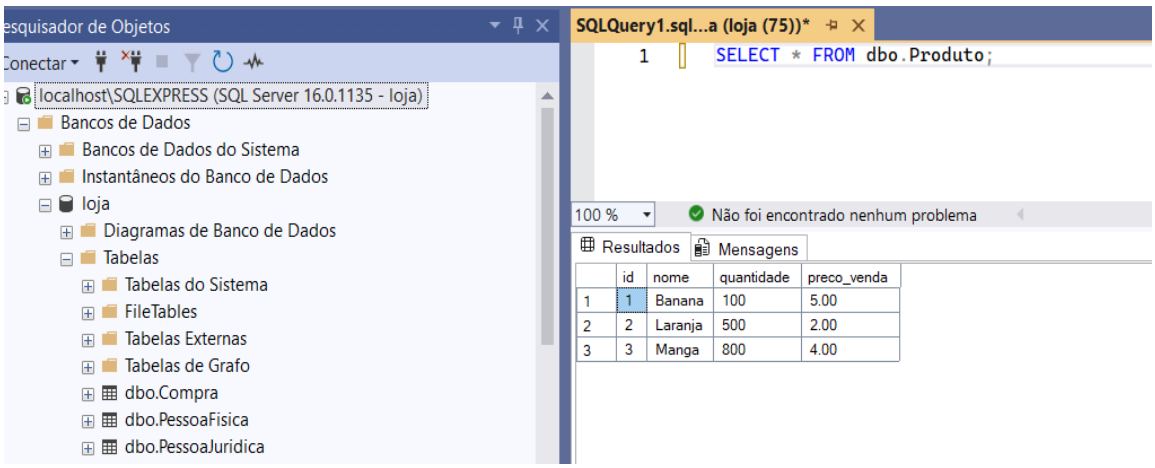



Projects
Files
Services
Databases
Java DB
Drivers
jdbc:derby://localhost:1527/sample [app on APP]
jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=false;trustServerCertificate=true; [loja on dbo]
jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true; [loja on LOJA]
jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true; [loja on dbo]
lojaConnection
loja
db_accessadmin
Other schemas
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
dbo
Tables
Compra
PessoaFisica
PessoaJuridica
Produto
Usuario
Venda
Views
Procedures
guest
INFORMATION_SCHEMA
processRequest - Navigator
ServletProduto.java
Source
History
out.println("<body>");
out.println("<h1>Lista de Pr
try {
List<cadastroee.model.Produt
out.println("<ul>");
for (cadastroee.model.Produt
out.println("<li>" + p.g
out.println("
");
} catch (Exception e) {
out.println("<p>Erro ao recu
out.println("</body>");
out.println("</html>");
HttpServlet methods. Click on the +


Projects
Files
Services
CadastroEE [main]
Java EE Modules
CadastroEE-war.war
CadastroEE-ejb.jar
Configuration Files
MANIFEST.MF
Server Resources
CadastroEE-ejb [main]
Source Packages
META-INF
cadastroee.controller
AbstractFacade.java
CompraFacade.java
CompraFacadeLocal.java
PessoaFisicaFacade.java
PessoaFisicaFacadeLocal.java
PessoaJuridicaFacade.java
PessoaJuridicaFacadeLocal.java
ProdutoFacade.java
ProdutoFacadeLocal.java
UsuarioFacade.java
UsuarioFacadeLocal.java
VendaFacade.java
VendaFacadeLocal.java
cadastroee.model
Compra.java
PessoaFisica.java
PessoaJuridica.java
Produto.java
Usuario.java
Venda.java
Test Packages
ServletProduto.java
Source
History
HttpServlet methods

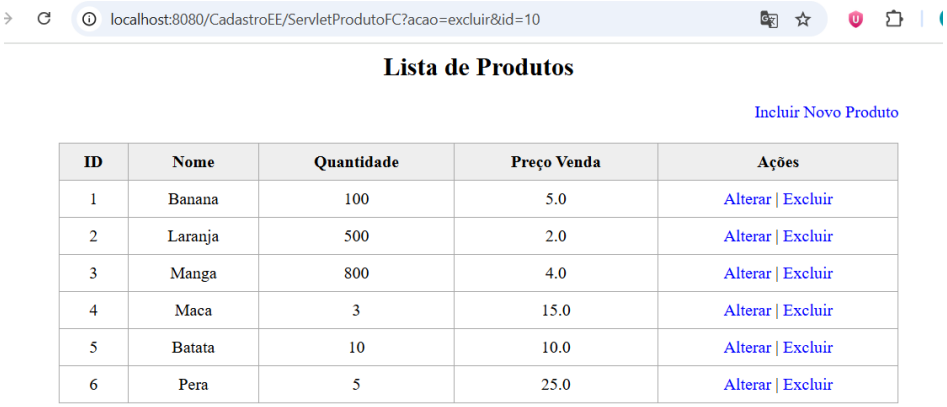


Teste do Servlet:
http://localhost:8080/CadastroEE-war/ServletProduto

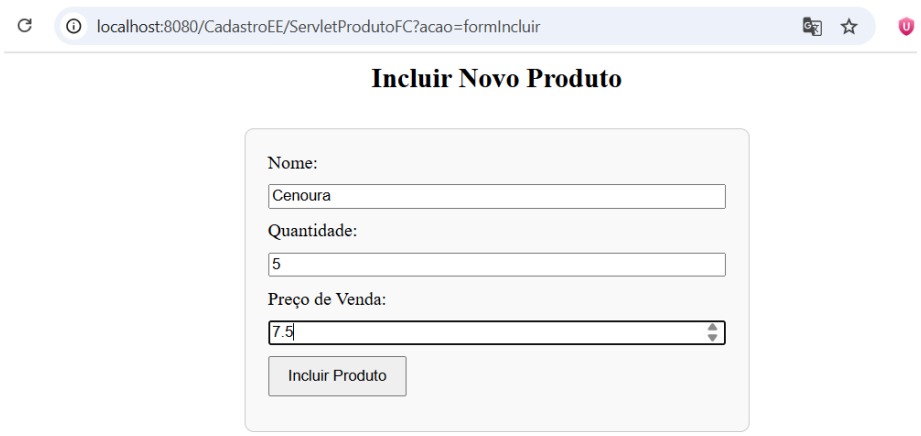


Procedimento 2: https://github.com/Elena-Gudimenko/Missao-4-Mundo-3/tree/main/Procedimento_02

Listar os produtos com a chamada para o endereço seguinte: http://localhost:8080/CadastroEE/ServletProdutoFC



Incluir Produto



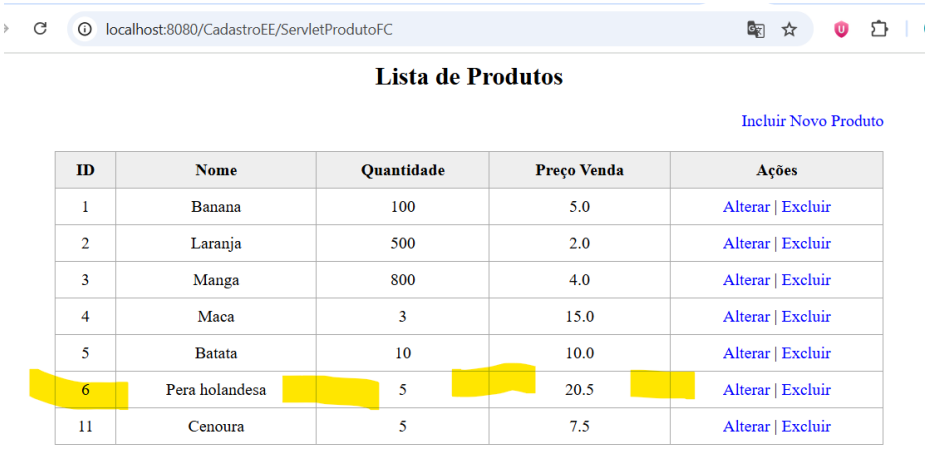
ID	Nome	Quantidade	Preço Venda	Ações
1	Banana	100	5.0	Alterar | Excluir
2	Laranja	500	2.0	Alterar | Excluir
3	Manga	800	4.0	Alterar | Excluir
4	Maca	3	15.0	Alterar | Excluir
5	Batata	10	10.0	Alterar | Excluir
6	Pera	5	25.0	Alterar | Excluir
11	Cenoura	5	7.5	Alterar | Excluir

Alterar Produto

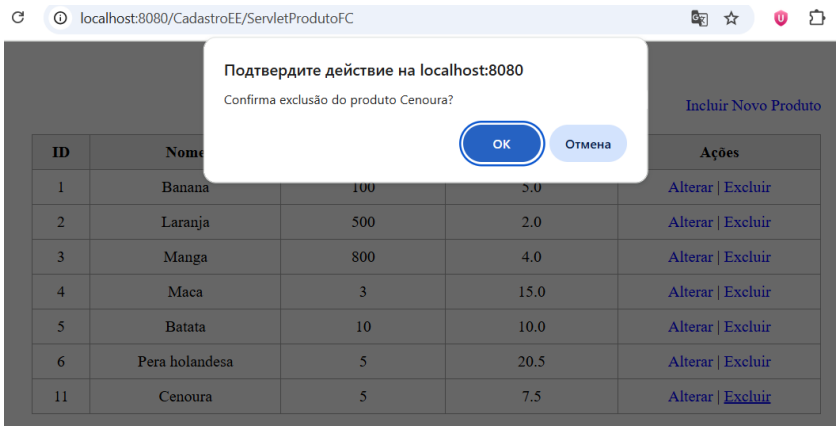
Lista de Produtos

Incluir Novo Produto

ID	Nome	Quantidade	Preço Venda	Ações
1	Banana	100	5.0	Alterar | Excluir
2	Laranja	500	2.0	Alterar | Excluir
3	Manga	800	4.0	Alterar | Excluir
4	Maca	3	15.0	Alterar | Excluir
5	Batata	10	10.0	Alterar | Excluir
6	Pera	5	25.0	Alterar | Excluir
11	Cenoura	5	7.5	Alterar | Excluir

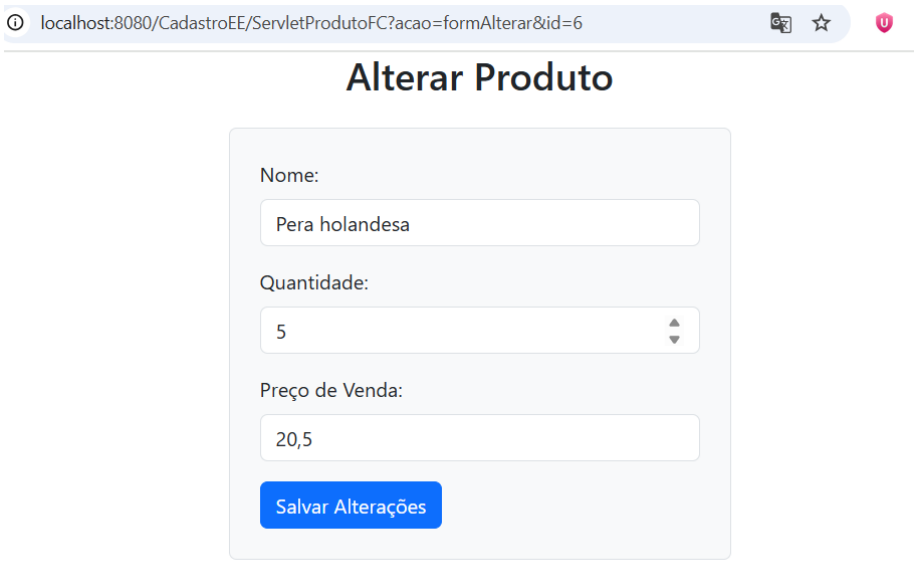


Excluir Produto



Procedimento 3: https://github.com/Elena-Gudimenko/Missao-4-Mundo-3/tree/main/Procedimento_03

Modificar as características de ProdutoLista.jsp e ProdutoDados.jsp por meio das bibliotecas do framework Bootstrap



Análise e Conclusão
Parte 1

1. Como é organizado um projeto corporativo no NetBeans?
Um projeto corporativo no NetBeans é organizado em múltiplos módulos que separam as responsabilidades da aplicação. No caso da plataforma Jakarta EE (Java EE), geralmente há ao menos dois módulos principais:

- O módulo **EJB** (Enterprise JavaBeans), onde ficam os componentes de negócio, como entidades JPA e session beans;
- O módulo **WAR**, responsável pela camada de apresentação e interface com o usuário, incluindo Servlets e páginas JSP.
  Essa separação promove melhor organização, reuso de código e facilita o gerenciamento de dependências.

2. Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?
A **JPA (Java Persistence API)** é responsável pelo mapeamento objeto-relacional, permitindo que objetos Java sejam diretamente persistidos e recuperados de um banco de dados relacional.
Já os **EJBs (Enterprise JavaBeans)**, especialmente os *Session Beans*, encapsulam a lógica de negócio da aplicação, oferecendo suporte a transações, segurança, injeção de dependências e gerenciamento de ciclo de vida.
Juntas, essas tecnologias permitem construir aplicações web robustas, escaláveis e de fácil manutenção.

3. Como o NetBeans viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?
O NetBeans oferece suporte completo ao desenvolvimento com JPA e EJB, com recursos como:

- **Geração automática de entidades JPA** a partir de tabelas do banco de dados;
- **Assistentes para criação de EJBs**, reduzindo o esforço manual;
- **Suporte à injeção de dependência via anotações**, com validação em tempo real;
- **Implantação direta em servidores como o GlassFish**, com feedback imediato.
  Essas funcionalidades tornam o desenvolvimento mais rápido, menos propenso a erros e mais acessível para estudantes e desenvolvedores.

4. O que são Servlets, e como o NetBeans oferece suporte à construção desse tipo de componentes em um projeto Web?
Servlets são componentes Java que rodam no servidor e respondem a requisições HTTP. Eles fazem parte da camada de controle de uma aplicação web, geralmente recebendo requisições do cliente, processando os dados (eventualmente consultando EJBs) e enviando uma resposta.
O NetBeans oferece suporte completo à criação de Servlets, com:

- Templates automáticos de criação;

- Gerenciamento de ciclo de vida via annotations (@WebServlet);
- Facilidade de execução e teste em servidores integrados.

5. Como é feita a comunicação entre os Servlets e os Session Beans do pool de EJBs?
A comunicação é feita através da **injeção de dependência**, utilizando a anotação @EJB.
Dentro do código do Servlet, o desenvolvedor injeta um Session Bean da camada EJB e pode chamar seus métodos diretamente, como no exemplo abaixo:

java

@EJB

private ProdutoBean produtoBean;

Esse mecanismo é gerenciado pelo contêiner de EJBs (como o GlassFish), que fornece uma instância do Bean já preparada, permitindo que o Servlet utilize os serviços de negócio sem precisar instanciar os objetos manualmente.

Parte 2

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

O padrão Front Controller centraliza o tratamento de todas as requisições do cliente em um único componente, geralmente um servlet, que atua como controlador principal. Esse controlador recebe todas as requisições, interpreta o parâmetro de ação (por exemplo, acao), decide qual lógica de negócio executar, e direciona a resposta para a página adequada (JSP).

Na arquitetura MVC (Model-View-Controller):

- Model (Modelo): representa a lógica de negócio e acesso a dados.
- View (Visão): é a interface com o usuário, geralmente implementada com JSPs.
- Controller (Controlador): é o Front Controller que recebe as requisições, manipula os dados via Model, e seleciona a View apropriada para exibir.

Assim, o Front Controller organiza e separa responsabilidades, facilitando a manutenção e o fluxo da aplicação.

2. Quais as diferenças e semelhanças entre Servlets e JSPs?

Semelhanças:

- Ambos são tecnologias Java para aplicações web que processam requisições HTTP.
- Executam no servidor, dentro de um container (ex: Tomcat, GlassFish).
- Podem trabalhar juntos em uma aplicação MVC.

Diferenças:

- Servlets: são classes Java puras que implementam a lógica de controle e processamento das requisições (métodos doGet, doPost).
- JSPs: são páginas que misturam HTML com código Java para gerar conteúdo dinâmico (visão). São compiladas em servlets automaticamente.

Os Servlets são usados para controle e processamento, enquanto os JSPs são mais adequados para a camada de apresentação.

3**. Qual a diferença entre um redirecionamento simples e o uso do método forward, a partir do RequestDispatcher?**

Redirecionamento (sendRedirect):

- É um redirecionamento do lado do cliente.
- O servidor responde ao navegador com um comando para carregar outra URL.
- O navegador faz uma nova requisição HTTP para essa URL.
- O endereço da URL no navegador muda.

Forward (RequestDispatcher.forward):

- É um redirecionamento do lado do servidor.
- A requisição é encaminhada internamente para outro recurso (Servlet ou JSP).
- Não gera uma nova requisição HTTP.
- O endereço da URL no navegador permanece o mesmo.

4. Para que servem parâmetros e atributos nos objetos HttpRequest?

Parâmetros: são valores enviados pelo cliente na requisição, via URL ou formulário (ex: request.getParameter("nome")). São sempre strings e usados para capturar dados de entrada do usuário.

Atributos: são objetos definidos pelo servidor ou aplicação e associados à requisição (ex: request.setAttribute("produto", produto)). Servem para passar informações entre componentes do servidor durante o processamento de uma única requisição, como entre um servlet e um JSP.

Parte 3

1. Como o framework Bootstrap é utilizado?

O Bootstrap é utilizado por meio da inclusão de suas bibliotecas CSS e JavaScript nas páginas HTML ou JSP, geralmente via links CDN. Ele fornece classes prontas que podem ser aplicadas diretamente aos elementos HTML, como botões, formulários, tabelas, entre outros, facilitando a criação de layouts modernos, organizados e responsivos sem a necessidade de escrever muito CSS manualmente.

2. Por que o Bootstrap garante a independência estrutural do HTML?

Porque o Bootstrap separa a lógica da estrutura HTML da estilização. Em vez de alterar o HTML para mudar o design, utilizamos classes CSS pré-definidas do Bootstrap. Isso mantém o HTML mais limpo e organizado, e permite que o visual da página seja controlado apenas pelas classes e estilos definidos pelo framework, sem modificar a estrutura básica dos elementos.

3. Qual a relação entre o Bootstrap e a responsividade da página?

O Bootstrap foi desenvolvido com foco em responsividade, ou seja, em adaptar o conteúdo da página a diferentes tamanhos de tela (computador, tablet, celular). Ele utiliza um sistema de grid flexível e classes específicas que ajustam automaticamente o layout conforme o tamanho do dispositivo. Isso garante que a interface funcione bem em qualquer resolução, sem a necessidade de criar várias versões da mesma página.