Estácio

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

André Luis Gonçalves Carvalhal - Matrícula: 20220318540

Campus Barra World
Vamos manter as informações! – 9001 – 2023.3

Objetivo da Prática

1. **Implementar persistência com base em JPA.**
2. **Implementar regras de negócio na plataforma JEE, através de EJBs.**
3. **Implementar sistema cadastral Web com base em Servlets e JSPs.**
4. **Utilizar a biblioteca Bootstrap para melhoria do design.**
5. **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.**

Link GitHub
https://github.com/ANDREC1986/RPG0017-202203185403.git

1º Procedimento | Criando o Banco de Dados

Projeto desenvolvido no NetBeans 19 arquivos criados já possuem o jakarta no lugar de javax, única alteração foi no produto.java, demais códigos gerados foram gerados automaticamente, então não colados para economizar espaço!

Produto.java

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

import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author andre
 */
@Entity
@Table(name = "produto")
@NamedQueries({
    @NamedQuery(name = "Produto.findAll", query = "SELECT p FROM Produto p"),
    @NamedQuery(name = "Produto.findByIdProduto", query = "SELECT p FROM Produto p
WHERE p.idProduto = :idProduto"),
    @NamedQuery(name = "Produto.findByNome", query = "SELECT p FROM Produto p
WHERE p.nome = :nome"),
    @NamedQuery(name = "Produto.findByQuantidade", query = "SELECT p FROM Produto p
WHERE p.quantidade = :quantidade"),
    @NamedQuery(name = "Produto.findByPrecoDeVenda", query = "SELECT p FROM Produto p
WHERE p.precoDeVenda = :precoDeVenda")})
```

```
public class Produto implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idProduto")
    private Integer idProduto;
    @Basic(optional = false)
    @Column(name = "nome")
    private String nome;
    @Basic(optional = false)
    @Column(name = "quantidade")
    private int quantidade;
    // @Max(value=?)  @Min(value=?)//if you know range of your decimal fields consider using these
annotations to enforce field validation
    @Basic(optional = false)
    @Column(name = "preco_de_venda")
    private Float precoDeVenda;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idProduto")
    private Collection<Movimento> movimentoCollection;

    public Produto() {
    }

    public Produto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public Produto(Integer idProduto, String nome, int quantidade, Float precoDeVenda) {
        this.idProduto = idProduto;
        this.nome = nome;
        this.quantidade = quantidade;
        this.precoDeVenda = precoDeVenda;
    }

    public Integer getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }
```

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

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    public Float getPrecoDeVenda() {
        return precoDeVenda;
    }

    public void setPrecoDeVenda(Float precoDeVenda) {
        this.precoDeVenda = precoDeVenda;
    }

    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idProduto != null ? idProduto.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Produto)) {
            return false;
        }
        Produto other = (Produto) object;
        if ((this.idProduto == null && other.idProduto != null) || (this.idProduto != null &&
!this.idProduto.equals(other.idProduto))) {
            return false;
        }
        return true;
    }
```

```
    @Override
    public String toString() {
        return "cadastroee.model.Produto[ idProduto=" + idProduto + " ]";
    }

}
```

ServletProdutoFC.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 java.io.IOException;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.util.List;

/**
 *
 * @author andre
 */
public class ServletProdutoFC extends HttpServlet {
    @EJB
    ProdutoFacadeLocal facade;

    /**
     * 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
     */
```

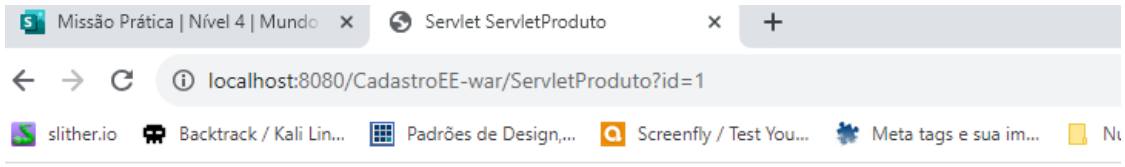
```
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
      String destino = "ProdutoLista.jsp";
      if("listar".equals(request.getParameter("acao"))) {
         List<Produto> produtos = facade.findAll();
         request.setAttribute("produtos", produtos);
         }
         else if("incluir".equals(request.getParameter("acao"))) {
            Produto produto = new Produto();
            produto.setNome(request.getParameter("nome"));
            produto.setPrecoDeVenda(Float.valueOf(request.getParameter("preco")));
            produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
            facade.create(produto);
            List<Produto> list = facade.findAll();
            request.setAttribute("produtos", list);
         }
         else if("alterar".equals(request.getParameter("acao"))) {
            int id = Integer.parseInt(request.getParameter("id"));
            Produto produto = facade.find(id);
            produto.setNome(request.getParameter("nome"));
            produto.setPrecoDeVenda(Float.valueOf(request.getParameter("preco")));
            produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
            facade.edit(produto);
            List<Produto> list = facade.findAll();
            request.setAttribute("produtos", list);
         }
         else if("excluir".equals(request.getParameter("acao"))) {
            Produto produto = facade.find(Integer.valueOf(request.getParameter("id")));
            facade.remove(produto);
            List<Produto> list = facade.findAll();
            request.setAttribute("produtos", list);
         }
         else if("formIncluir".equals(request.getParameter("acao"))) {
            Produto produto = new Produto();
            request.setAttribute("acao","incluir");
            request.setAttribute("produto", produto);
            destino = "ProdutoDados.jsp";
         }
         else if("formAlterar".equals(request.getParameter("acao"))) {
            int id = Integer.parseInt(request.getParameter("id"));
            Produto produto = facade.find(id);
            request.setAttribute("acao", "alterar");
            request.setAttribute("produto", produto);
            destino = "ProdutoDados.jsp";
         }
      request.getRequestDispatcher(destino).forward(request, response);
   }
```

```
    // <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>

}
```



Servlet ServletProduto at /CadastroEE-war

Banana

Conclusão:

a.) Como é organizado um projeto corporativo no NetBeans?

No netbeans o projeto empresarial é organizado em uma estrutura que possui um pacto principal que gerencia os demais. Sendo eles um pacote de estrutura ejb e outro de web application. Ademais a organização segue o padrão de Source Packages and Libraries como qualquer outro projeto.

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

JPA (Java Persistence API) é a ferramenta do java que fornece o mapeamento de bancos de dados, simplificando as operações de CRUD sem que seja necessário criar comandos SQL manualmente. Servindo como uma camada model do projeto.

EJBs (Enterprise JavaBeans) criam as regras de negócio do projeto, controlando as transações, sessões e entidades. Fazendo um papel de camada controller no Projeto.

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

O Netbeans é capaz de gerar as entidades JPA ao mapear o banco de dados automaticamente, ao ser solicitado para criar entidades com base em um banco específico. Posteriormente ele é capaz de criar a estrutura EJB das entidades mapeadas.

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

Servlets são os responsáveis em processar requisições HTTP do lado do servidor. O Netbeans fornece um ambiente completo para a criação, edição, depuração e implementação dos mesmo, permitindo um ambiente integrado com servidores também implementados no mesmo.

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

Através da injeção de dependência com a anotação @EJB, então o pool pode gerenciar as chamadas, garantindo a integridade dos dados.

2º Procedimento | Interface Cadastral com Servlet e JSPs

ServletProdutoFC.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 java.io.IOException;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.util.List;

/**
 *
 * @author andre
 */
public class ServletProdutoFC extends HttpServlet {
    @EJB
    ProdutoFacadeLocal facade;

    /**
     * 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
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        String destino = "ProdutoLista.jsp";
        if("listar".equals(request.getParameter("acao"))) {
            List<Produto> produtos = facade.findAll();
            request.setAttribute("produtos", produtos);
            }
            else if("incluir".equals(request.getParameter("acao"))) {
```

```
            Produto produto = new Produto();
            produto.setNome(request.getParameter("nome"));
            produto.setPrecoDeVenda(Float.valueOf(request.getParameter("preco")));
            produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
            facade.create(produto);
            List<Produto> list = facade.findAll();
            request.setAttribute("produtos", list);
        }
        else if("alterar".equals(request.getParameter("acao"))) {
            int id = Integer.parseInt(request.getParameter("id"));
            Produto produto = facade.find(id);
            produto.setNome(request.getParameter("nome"));
            produto.setPrecoDeVenda(Float.valueOf(request.getParameter("preco")));
            produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
            facade.edit(produto);
            List<Produto> list = facade.findAll();
            request.setAttribute("produtos", list);
        }
        else if("excluir".equals(request.getParameter("acao"))) {
            Produto produto = facade.find(Integer.valueOf(request.getParameter("id")));
            facade.remove(produto);
            List<Produto> list = facade.findAll();
            request.setAttribute("produtos", list);
        }
        else if("formIncluir".equals(request.getParameter("acao"))) {
            Produto produto = new Produto();
            request.setAttribute("acao","incluir");
            request.setAttribute("produto", produto);
            destino = "ProdutoDados.jsp";
        }
        else if("formAlterar".equals(request.getParameter("acao"))) {
            int id = Integer.parseInt(request.getParameter("id"));
            Produto produto = facade.find(id);
            request.setAttribute("acao", "alterar");
            request.setAttribute("produto", produto);
            destino = "ProdutoDados.jsp";
        }
       request.getRequestDispatcher(destino).forward(request, response);
    }

    // <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>
}
```

ProdutoLista.jsp

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

```
</head>
<body>
   <h1>Listagem de Produtos</h1>
   <a href="ServletProdutoFC?acao=formIncluir">Incluir Novo Produto</a>

   <table border="1">
      <tr>
         <th>ID</th>
         <th>Nome</th>
         <th>Quantidade</th>
         <th>Preço</th>
         <th>Ações</th>
      </tr>
      <%
      List<Produto> produtos = (List<Produto>) request.getAttribute("produtos");
      for (Produto produto : produtos) {
      out.println("<tr><td>"+produto.getIdProduto()+"</td>");
      out.println("<td>"+produto.getNome()+"</td>");
      out.println("<td>"+produto.getQuantidade()+"</td>");
      out.println("<td>"+produto.getPrecoDeVenda()+"</td>");
      out.println("<td><a
href='ServletProdutoFC?acao=formAlterar&id="+produto.getIdProduto()+"'>Alterar</a> <a
href='ServletProdutoFC?acao=excluir&id="+produto.getIdProduto()+"'>Excluir</a></td>");
         }
      %>
   </table>
</body>
</html>
```

Listagem de Produtos

Incluir Novo Produto

ID	Nome	Quantidade	Preço	Ações
1	Banana	100	5.0	Alterar Excluir
3	Laranja	500	2.0	Alterar Excluir
4	Manga	80	4.1	Alterar Excluir

ProdutoDados.jsp

```
<%--
    Document   : ProdutoDados
    Created on : 1 de nov. de 2023, 19:03:39
    Author     : andre
--%>

<%@ page import = "cadastroee.model.Produto" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
<body>
        <h1>Dados do Produto</h1>
    <%
            Produto produto = Produto.class.cast(request.getAttribute("produto"));
    %>
    <form action="ServletProdutoFC" method="POST">
        <div>
            <label for="nome" >Nome:</label>
            <input type="text" name="nome" id="nome" required value="<%
                if (produto.getNome() != null) {
                out.print(produto.getNome());
                }
            %>">
        <label for="quantidade" >Quantidade:</label>
        <input type="number" name="quantidade" id="quantidade" required value = "<%
            if(produto.getQuantidade() != 0) {
                out.print(produto.getQuantidade());
            }
              %>">
        <label for="preco" >Preço de Venda:</label>
        <input type="text" name="preco" id="preco" required value="<%
            if(produto.getPrecoDeVenda() != null) {
                out.print(produto.getPrecoDeVenda());
            }
        %>">
        <input type="hidden" name="id" id="id" value="<% out.print(produto.getIdProduto());%>">
        <input type="hidden" name="acao" id="acao"
value=<%out.print(request.getAttribute("acao"));%>>
        <input type="submit" value="<%
            String acao = request.getAttribute("acao").toString();
            acao = acao.substring(0, 1).toUpperCase() + acao.substring(1);
            out.print(acao);
        %> Produto">
    </form>
```

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

Conclusão:

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

Front Controller é um padrão de design de software que centraliza as solicitações de entrada, roteamento e controle de fluxo. Em aplicativo Web Java o Front Controller geralmente é implementado utilizando um Servlet para centralizar as requisições e redirecionar o cliente para o a View adequada.

b) Quais as diferenças e semelhanças entre Servlets e JSPs?

Ambos fazem parte da plataforma Java EE, capazes de gerar páginas dinâmicas e geralmente utilizadas em conjunto. Porém o Servlet são classes direcionadas a lógica de programação e podem responder solicitações HTTP diretamente e em diversos formatos(XML, JSON, JSP, etc..). Já os JSP são páginas HTML que misturam o HTML com a linguagem Java, sendo estes mais recomendados para criação da interface da aplicação Web Java.

c) 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?

O Redirecionamento simples é uma resposta do servidor que instrui o cliente a fazer uma nova solicitação a uma URL diferentes, criando uma solicitação distinta. Já o RequestDispatcher é um encaminhamento interno do servidor, que envia a solicitação do usuário para um novo recurso (Servlet, JSP etc..) podendo atrelar atributos a solicitação, para ser processado pelo recurso solicitado. Não havendo a necessidade de se iniciar uma nova solicitação.

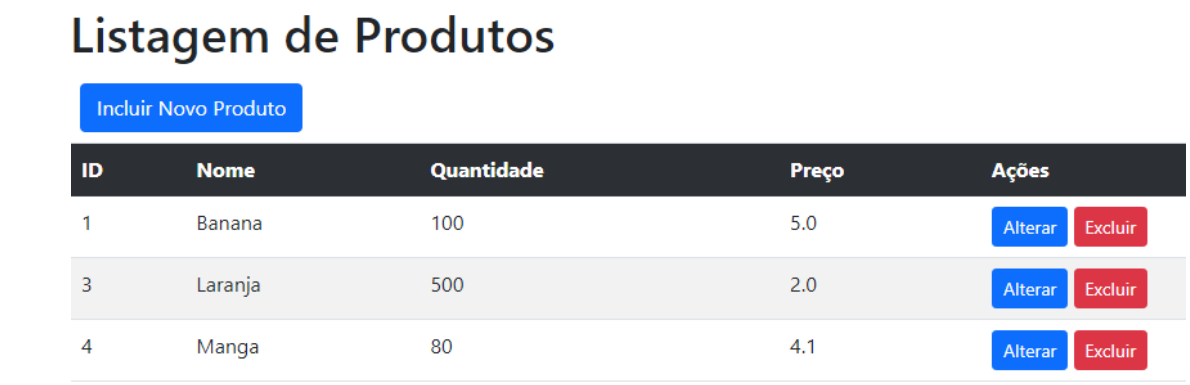
3º Procedimento | Melhorando o Design da Interface

ProdutoLista.JSP

```
<%@ page import = "cadastroee.model.Produto" %>
<%@ page import = "java.util.List" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Lista de Produtos</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCO
mLASjC" crossorigin="anonymous">
</head>
<body>
    <h1>Listagem de Produtos</h1>
    <a class="btn btn-primary m-2" href="ServletProdutoFC?acao=formIncluir">Incluir Novo
Produto</a>

    <table border="1" class="table table-striped">
        <tr class="table-dark">
            <th>ID</th>
            <th>Nome</th>
            <th>Quantidade</th>
            <th>Preço</th>
            <th>Ações</th>
        </tr>
        <%
        List<Produto> produtos = (List<Produto>) request.getAttribute("produtos");
        for (Produto produto : produtos) {
        out.println("<tr><td>"+produto.getIdProduto()+"</td>");
        out.println("<td>"+produto.getNome()+"</td>");
        out.println("<td>"+produto.getQuantidade()+"</td>");
        out.println("<td>"+produto.getPrecoDeVenda()+"</td>");
        out.println("<td><a class='btn btn-primary btn-sm'
href='ServletProdutoFC?acao=formAlterar&id="+produto.getIdProduto()+"'>Alterar</a> <a
class='btn btn-danger btn-sm'
href='ServletProdutoFC?acao=excluir&id="+produto.getIdProduto()+"'>Excluir</a></td>");
            }
        %>
    </table>
</body>
```

```
</html>
```

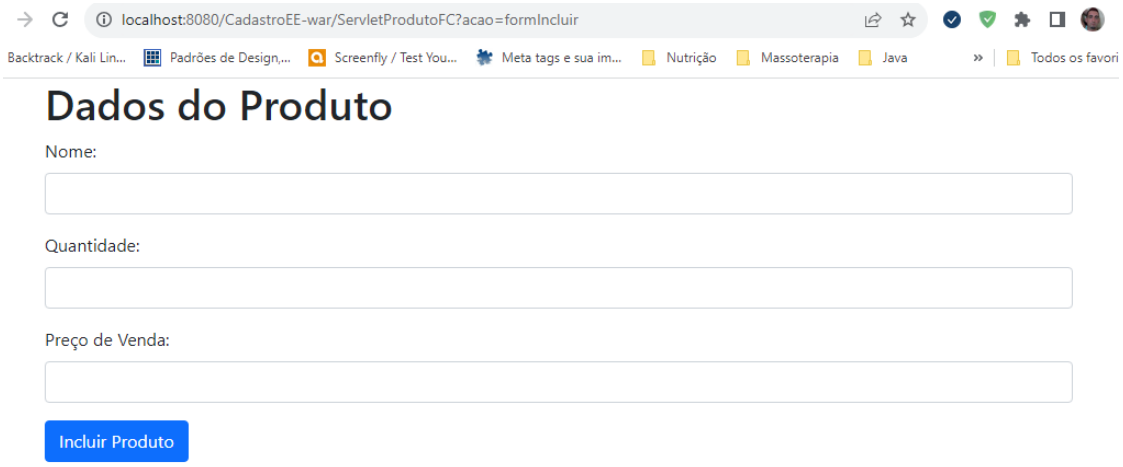


ProdutoDados.jsp

```
<%--
    Document   : ProdutoDados
    Created on : 1 de nov. de 2023, 19:03:39
    Author     : andre
--%>

<%@ page import = "cadastroee.model.Produto" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmL
ASjC" crossorigin="anonymous">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIa
xVXM" crossorigin="anonymous"></script>
        <title>JSP Page</title>
    </head>
<body class="container">
        <h1>Dados do Produto</h1>
    <%
            Produto produto = Produto.class.cast(request.getAttribute("produto"));
    %>
```

```
   <form class="form" action="ServletProdutoFC" method="POST">
      <div class="mb-3">
         <label for="nome" class="form-label">Nome:</label>
         <input class="form-control" type="text" name="nome" id="nome" required value="<%
            if (produto.getNome() != null) {
            out.print(produto.getNome());
            }
         %>">
      </div>
      <div class="mb-3">
      <label for="quantidade" class="form-label">Quantidade:</label>
      <input class="form-control" type="number" name="quantidade" id="quantidade" required
value = "<%
         if(produto.getQuantidade() != 0) {
            out.print(produto.getQuantidade());
         }
           %>">
      </div>
      <div class="mb-3">
      <label for="preco" class="form-label">Preço de Venda:</label>
      <input class="form-control" type="text" name="preco" id="preco" required value="<%
         if(produto.getPrecoDeVenda() != null) {
            out.print(produto.getPrecoDeVenda());
         }
      %>">
      </div>
      <input type="hidden" name="id" id="id" value="<%
out.print(produto.getIdProduto());%>">
      <input type="hidden" name="acao" id="acao"
value=<%out.print(request.getAttribute("acao"));%>>
      <input class="btn btn-primary" type="submit" value="<%
         String acao = request.getAttribute("acao").toString();
         acao = acao.substring(0, 1).toUpperCase() + acao.substring(1);
         out.print(acao);
      %> Produto">
   </form>
</body>
</html>
```



Conclusão:

a.) Como o framework Bootstrap é utilizado?

O bootstrap é utilizado através da inclusão do CSS e JS do mesmo, então a implementação das classes do Bootstrap para personalizar da pagina.

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

O Bootstrap garante a independência estrutural do HTML pois se baseie em classes, não necessitando de alterações na semântica do HTML (<nav>,<buton> , etc..).

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

Ao utilizar um padrão de grip, o bootstrap permite que o desenvolvedor organize o conteúdo em colunas e linhas, sem alterar a estrutura do HTML. Além de recurso que permite sinalizar a resolução em que um recurso deve ou não ser exibido.