Trabalho Prático | DGT2811 Desenvolvimento Back-End Corporativo com Java E Cloud

➤ 2º PROCEDIMENTO ◄

Curso: Desenvolvimento Full Stack

Polo: Polo - Vila Nova - Itu - Sp

Matrícula: 202404575268

Aluno: Rafael Lima de Medeiros

Turma: 2025.4 (5º Semestre)

Data: 14/02/2026

Github:
https://github.com/Rafaldm/Desenv-Back-end-Corporativo-Com-Java-e-Cloud

Objetivo: O objetivo deste trabalho foi criar um sistema cadastral Web em Java, usando Servlets, JSPs, EJBs e JPA, para aprender a organizar um projeto corporativo, implementar persistência de dados, regras de negócio e interfaces web de forma funcional e limpa.

Ánalise

1. Como funciona o padrão Front Controller e como ele é implementado em Java MVC?
O Front Controller é um servlet que recebe todas as ações do sistema, decide o que fazer e manda pra página certa. No MVC, ele controla a aplicação, deixando os JSPs só pra mostrar a interface e os EJBs só pra lógica de negócio.

2. Diferenças e semelhanças entre Servlets e JSPs
Servlets: processam dados e regras, rodam no servidor.
JSPs: mostram a interface, tipo páginas HTML dinâmicas.
Ambos fazem parte da aplicação web e podem trocar informações via request.

3. Diferença entre redirecionamento simples e forward; para que servem parâmetros e atributos no HttpRequest
Redirecionamento: envia o navegador pra outra URL, perdendo os dados da requisição.
Forward: vai pra outra página sem perder os dados, dentro do servidor.
Parâmetros: dados enviados pelo usuário, tipo ?id=1.
Atributos: dados que a gente passa entre servlet e JSP, tipo lista de produtos.

CÓDIGOS

Produto.java

```
package cadastroee.model;

import jakarta.persistence.*;
import java.io.Serializable;

@Entity
@Table(name = "produto")
public class Produto implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false)
    private String nome;

    private Integer quantidade;

    private Float precoVenda;

    public Produto() {
    }

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

    public Integer getId() {
        return id;
    }

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

    public String getNome() {
        return nome;
    }

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

    public Integer getQuantidade() {
        return quantidade;
    }

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

    public Float getPrecoVenda() {
        return precoVenda;
    }

    public void setPrecoVenda(Float precoVenda) {
        this.precoVenda = precoVenda;
    }
}
```

ProdutoFacadeLocal.java

```
package cadastroee.ejb;

import cadastroee.model.Produto;
import java.util.List;
import jakarta.ejb.Local;

@Local
public interface ProdutoFacadeLocal {

    void create(Produto produto);

    void edit(Produto produto);

    void remove(Produto produto);

    Produto find(Object id);

    List<Produto> findAll();
}
```

ProdutoFacade.java

```
package cadastroee.ejb;

import cadastroee.model.Produto;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import java.util.List;

@Stateless
public class ProdutoFacade implements ProdutoFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    public void create(Produto produto) {
        em.persist(produto);
    }

    @Override
    public void edit(Produto produto) {
        em.merge(produto);
    }

    @Override
    public void remove(Produto produto) {
        em.remove(em.merge(produto));
    }

    @Override
    public Produto find(Object id) {
        return em.find(Produto.class, id);
    }

    @Override
    public List<Produto> findAll() {
        return em.createQuery("SELECT p FROM Produto p", Produto.class)
                .getResultList();
    }
}
```

ServletProdutoFC.java

```
package cadastroee.servlets;

import jakarta.ejb.EJB;
import cadastroee.ejb.ProdutoFacadeLocal;
import cadastroee.model.Produto;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

public class ServletProdutoFC extends HttpServlet {

   @EJB
   private ProdutoFacadeLocal facade;

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

      String acao = request.getParameter("acao");
      String destino = "ProdutoLista.jsp";

      if (acao == null) {
         acao = "listar";
      }

      try {
         switch (acao) {
            case "listar":
               request.setAttribute("lista", facade.findAll());
               break;

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

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

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

novo.setQuantidade(Integer.parseInt(request.getParameter("quantid
ade")));

novo.setPrecoVenda(Float.parseFloat(request.getParameter("precoVe
nda")));
               facade.create(novo);
               request.setAttribute("lista", facade.findAll());
               break;

            case "alterar":
               Integer id = Integer.parseInt(request.getParameter("id"));
               Produto existente = facade.find(id);
               existente.setNome(request.getParameter("nome"));

existente.setQuantidade(Integer.parseInt(request.getParameter("qua
ntidade")));
existente.setPrecoVenda(Float.parseFloat(request.getParameter("prec
oVenda")));
 facade.edit(existente);
 request.setAttribute("lista", facade.findAll());
 break;
 case "excluir":
 Integer idExcluir = Integer.parseInt(request.getParameter("id"));
 Produto prodExcluir = facade.find(idExcluir);
 facade.remove(prodExcluir);
 request.setAttribute("lista", facade.findAll());
 break;
 }
 } catch (Exception e) {
 e.printStackTrace();
 }
 RequestDispatcher rd = request.getRequestDispatcher(destino);
 rd.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);
 }
}
```

ProdutoLista.jsp

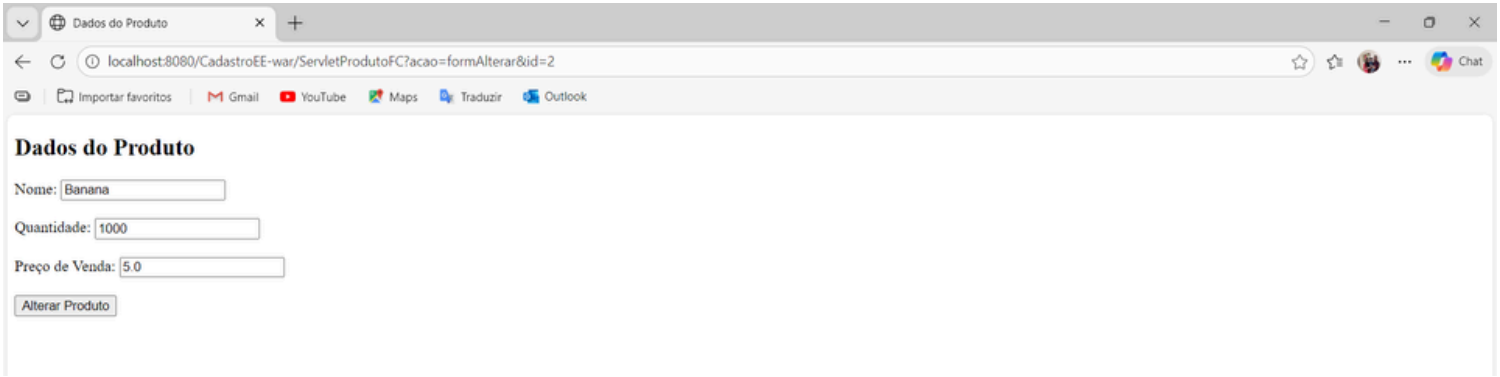
```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ page import="java.util.List" %>
<%@ page import="cadastroee.model.Produto" %>
<html>
<head>
   <title>Listagem de Produtos</title>
</head>
<body>
<h2>Listagem de produtos</h2>
<a href="ServletProdutoFC?acao=formIncluir">Novo produto</a>
<table border="1">
   <tr>
      <th>#</th>
      <th>Nome</th>
      <th>Quantidade</th>
      <th>Preço de venda</th>
      <th>Opções</th>
   </tr>
   <%
      List<Produto> lista = (List<Produto>) request.getAttribute("lista");
      if (lista != null) {
         for (Produto p : lista) {
   %>
   <tr>
      <td><%= p.getId() %></td>
      <td><%= p.getNome() %></td>
      <td><%= p.getQuantidade() %></td>
      <td><%= p.getPrecoVenda() %></td>
      <td>
         <a href="ServletProdutoFC?acao=formAlterar&id=
<%=p.getId()%>">alterar</a>
         <a href="ServletProdutoFC?acao=excluir&id=
<%=p.getId()%>">excluir</a>
      </td>
   </tr>
   <%     }
      }
   %>
</table>
</body>
</html>
```

ProdutoDados.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ page import="cadastroee.model.Produto" %>
<html>
<head>
   <title>Dados do Produto</title>
</head>
<body>
<h2>Dados do produto</h2>
<%
   Produto p = (Produto) request.getAttribute("produto");
   String acao = (p == null) ? "incluir" : "alterar";
%>
<form action="ServletProdutoFC" method="post">
   <input type="hidden" name="acao" value="<%=acao%>">
   <% if (p != null) { %>
      <input type="hidden" name="id" value="<%=p.getId()%>">
   <% } %>
   Nome: <input type="text" name="nome" value="<%= (p != null) ?
p.getNome() : "" %>"><br>
   Quantidade: <input type="number" name="quantidade" value="<%=
(p != null) ? p.getQuantidade() : "" %>"><br>
   Preço de venda: <input type="text" name="precoVenda" value="<%=
(p != null) ? p.getPrecoVenda() : "" %>"><br>
   <button type="submit"><%= (acao.equals("incluir")) ? "Adicionar
Produto" : "Alterar Produto" %></button>
</form>
</body>
</html>
```

PRINTS


Listagem de Produtos
localhost:8080/CadastroEE-war/ServletProdutoFC?acao=excluir&id=1
Importar favoritos
Gmail
YouTube
Maps
Traduzir
Outlook
Chat
Listagem de Produtos
Novo Produto
Nome Quantidade Preço de Venda Opções
2 Banana 1000 5.0 Alterar | Excluir


Dados do Produto
localhost:8080/CadastroEE-war/ServletProdutoFC?acao=formAlterar&id=2
Importar favoritos
Gmail
YouTube
Maps
Traduzir
Outlook
Chat
Dados do Produto
Nome: Banana
Quantidade: 1000
Preço de Venda: 5.0
Alterar Produto

Conclusão

Esse trabalho mostrou como construir um sistema web corporativo em Java de forma organizada e funcional. A gente viu que é importante separar responsabilidades: os Servlets processam os pedidos do usuário, os JSPs cuidam da interface e os EJBs com JPA gerenciam a lógica de negócio e o acesso ao banco de dados. O Front Controller centraliza as ações, garantindo que tudo seja controlado de forma clara e mantendo o MVC funcionando corretamente.

Aprendemos também como os dados circulam entre camadas usando parâmetros e atributos, e como o NetBeans facilita a criação de entidades, facades e servlets, agilizando o desenvolvimento. No fim, o sistema funciona de forma prática, permitindo incluir, alterar, listar e excluir produtos, mostrando na prática como projetos corporativos Java Web são estruturados e mantidos de forma limpa e eficiente.