Vamos integrar sistemas

Walber do Carmo Farias, 202303752547

Polo de Abaetetuba (PA)
Nivel 4 – Vamos integrar sistemas – 2023.1 – 2024.1

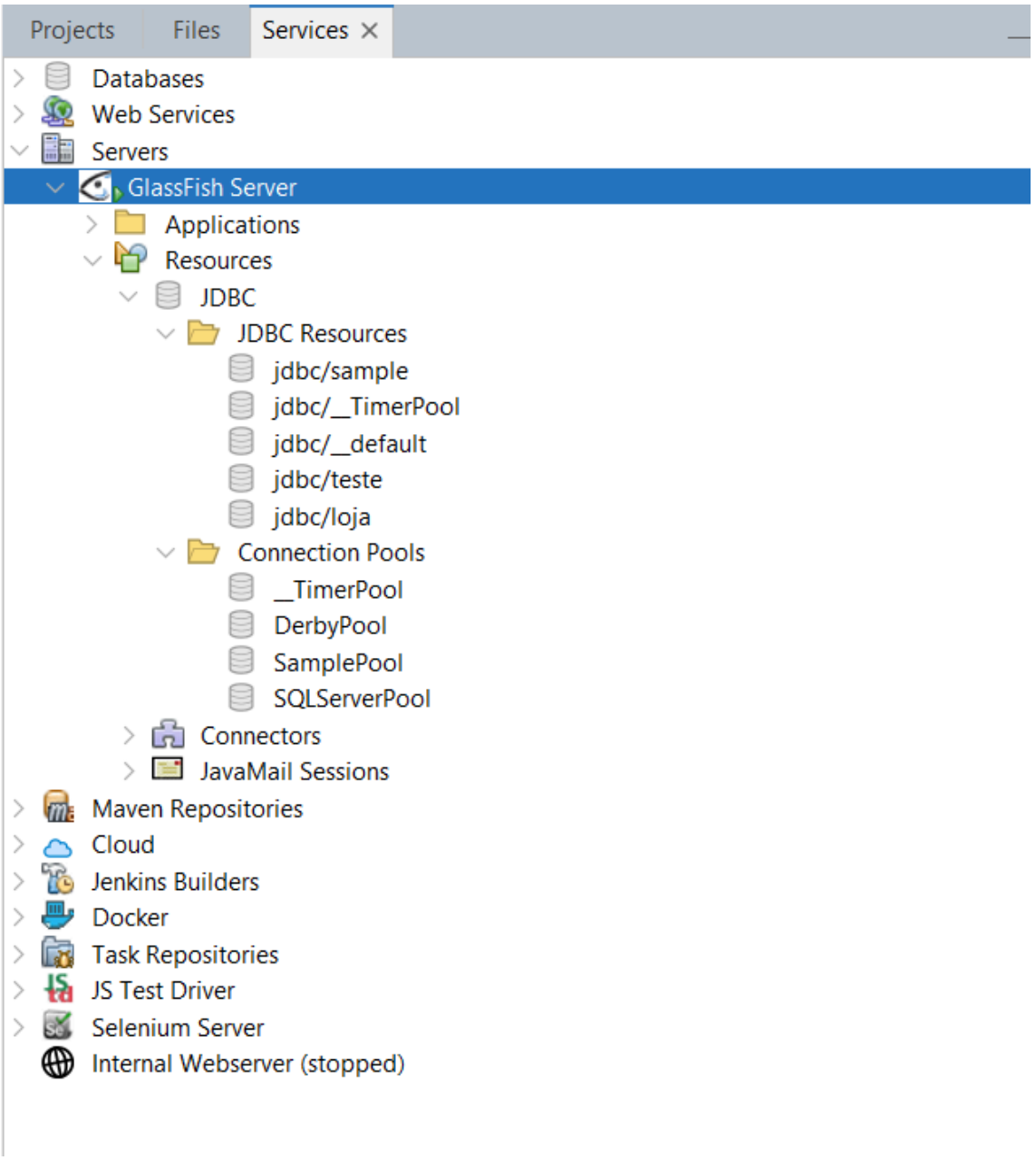
Objetivo da Prática

O objetivo desta missão foi implementar um sistema cadastral web em Java com uso de SQL Server e persistência dos dados utilizando middleware JDBC, além de ser feito o mapeamento objeto-relacional usando JPA para persistência dos dados, EJBs para regras de negócio, Servlets/JSPs para entrada e exibição de dados e a utilização de Bootstrap para melhoria do design das páginas.

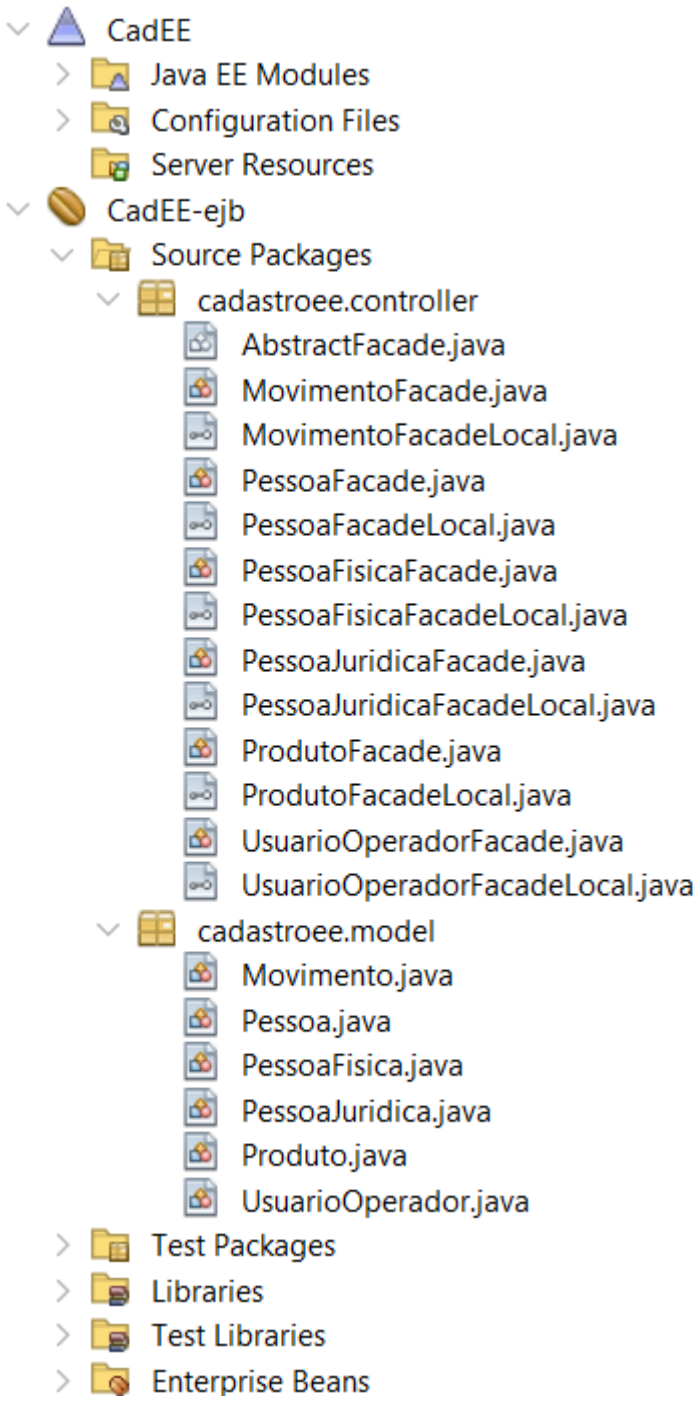
1º Procedimento | Camadas de Persistência e Controle

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 1º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

- Configurando o Pool de conexão entre o banco de dados SQL Server e o Servidor Glassfish por meio da aba de serviços do NetBeans:



- Criação das entidades mapeadas pelo JPA e dos componentes EJB, os quais foram gerados Sessions Beans com o sufixo Facade e FacadeLocal.



- Criação da Servlet ***ProdutoServlet***, adicionando a referência da interface do EJB *ProdutoFacade* com a anotação @EJB. Recuperando os dados do banco por meio da chamada do método *findAll()*, que retorna um objeto *List<Produto>*, o

qual foi feito uso de um for para iterar sobre esta lista e apresentar os dados em formato *HTML* na página *Web*.

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

    @EJB
    ProdutoFacadeLocal facade;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Produto</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Lista de Produtos</h1>");
            out.println("<ul>");


            List<Produto> produtos = facade.findAll();

            for (Produto produto : produtos) {
                out.println("<li>" + produto.getNome() + " - R$ " + produto.getPrecoVenda()
            }

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

- Executando e testando por meio do endereço http://localhost:8080/CadEE-war/ServletProduto

a) Como é organizado um projeto corporativo no NetBeans?

Resposta:

- É organizado em módulos, como módulos EJB e Web, com pastas para código-fonte, recursos e configurações.

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

Resposta:

- O JPA facilita a comunicação com bancos de dados, permitindo mapear objetos Java para tabelas e realizar operações de CRUD. O EJB fornece componentes para lógica de negócio e o gerenciamento de transações, facilitando a criação de aplicações com acesso a banco.

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

Resposta:

- O NetBeans melhora a produtividade com JPA e EJB oferecendo aba de criação de código para gerar entidades e EJBs, ferramenta de deploy e run integrados, além de um editor com auto-complete e validação em tempo real. Ele também facilita a configuração de persistência e integração com servidores de aplicação como o Glassfish, tornando o desenvolvimento mais produtivo.

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

Resposta:

- As Servlets são componentes Java que processam requisições e respostas HTTP em aplicações web. O NetBeans oferece suporte na construção de Servlets através da aba para a criação e configuração automática das Servlets, atribuindo a classe Java com a anotação *@WebServlet* e definindo a propriedade *urlPatters*, além da geração dos métodos *processRequest, doGet* e *doPost*.

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

Resposta:

- A Servlet injeta o Session Bean diretamente como uma dependência e pode chamar seus métodos para executar a lógica de negócio, permitindo que a Servlet use seus métodos de forma prática.

2º Procedimento | Interface cadastral com Servlet e JSPs Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 2º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

- Criação da Servlet **ServletProdutoFC** mapeada para o caminho *"/ProdutoFrontController"*, sendo feita a injeção de dependência do componente *EJB* para ser utilizado posteriormente no método *processRequest*. Realização da recuperação do parâmetro "acao" da requisição, sendo utilizado um *switch case* para ser feito o direcionamento do usuário para a JSP necessária.

```
@WebServlet(urlPatterns = {"/ProdutoFrontController"})
public class ServletProdutoFC 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");
        String destino = null;

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

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

            case "formIncluir":
                destino = "ProdutoDados.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 "excluir":
        int idExcluir = Integer.parseInt(request.getParameter("id"));
        Produto produtoExcluir = facade.find(idExcluir);
        facade.remove(produtoExcluir);
        request.setAttribute("produtos", facade.findAll());
        destino = "ProdutoLista.jsp";
        break;

    case "alterar":
        int idProdutoAlterar = Integer.parseInt(request.getParameter("id"));
        Produto produtoParaAlterar = facade.find(idProdutoAlterar);
        if (produtoParaAlterar != null) {
            produtoParaAlterar.setNome(request.getParameter("nome"));
            produtoParaAlterar.setPrecoVenda(Float.parseFloat(request.getParameter("precoVenda")));
            facade.edit(produtoParaAlterar);
        }
        request.setAttribute("produtos", facade.findAll());
        destino = "ProdutoLista.jsp";
        break;

    case "incluir":
        Produto novoProduto = new Produto();
        String nome = request.getParameter("nome");
        String precoStr = request.getParameter("precoVenda");
        String quantidadeStr = request.getParameter("quantidade");

        if (nome != null && !nome.isEmpty()) {
            novoProduto.setNome(nome);
        }

        facade.create(novoProduto);
        request.setAttribute("produtos", facade.findAll());
        destino = "ProdutoLista.jsp";
        break;


    default:
        List<Produto> produtosDefault = facade.findAll();
        request.setAttribute("produtos", produtosDefault);
        destino = "ProdutoLista.jsp";
        break;
}

RequestDispatcher rd = request.getRequestDispatcher(destino);
rd.forward(request, response);
```

- Criação do JSP **ProdutoLista.jsp** com o link direcionando para o caminho de inclusão de novo produto com o parâmetro *"acao"* valendo *"formIncluir"*, além de uma tabela com *thead* para o cabeçalho da tabela e um componente *c:forEach* da biblioteca JSTL para iterar sobre a lista de produtos e fazer sua exibição em um elemento *tr* da tabela.

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
    <title>Lista de Produtos</title>
    </head>
<body>
    <h1>Listagem de Produtos</h1>
    <a href="ProdutoFrontController?acao=formIncluir">Incluir Novo Produto</a>
    <table>
        <thead>
            <th>ID</th>
            <th>Nome</th>
            <th>Quantidade</th>
            <th>Preço</th>
            <th>Ações</th>
        </thead>
        <c:forEach var="produto" items="${produtos}">
            <tr>
                <td>${produto.idProduto}</td>
                <td>${produto.nome}</td>
                <td>${produto.quantidade}</td>
                <td>${produto.precoVenda}</td>
                <td>
                    <a href="ProdutoFrontController?acao=formAlterar&id=${produto.idProduto}">Alterar</a>
                    <a href="ProdutoFrontController?acao=excluir&id=${produto.idProduto}">Excluir</a>
                </td>
            </tr>
        </c:forEach>
    </table>

</body>
</html>
```

- Criação da JSP **DadosProduto.jsp** contendo um formulários com os inputs necessários para as ações incluir e alterar, que utilizará o mesmo formulário para cumprir suas funções. Além de um botão do tipo submit para enviar o formulário para a *ServletProdutoFC* que fará o tratamento dos dados enviados na requisição *POST* e a chamada dos métodos do componente EJB.

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Dados do Produto</title>

    </head>
<body>
    <h1>Dados do Produto</h1>
    <form action="ProdutoFrontController" method="post">
        <input type="hidden" name="acao" value="${param.acao eq 'formIncluir' ? 'incluir' : 'alterar'}">
        <input type="hidden" name="id" value="${produto != null ? produto.idProduto : ''}">
        <div>
            <label>Nome:</label>
            <input type="text" name="nome" value="${produto != null ? produto.nome : ''}">
        </div>
        <div>
            <label>Quantidade:</label>
            <input type="text" name="quantidade" value="${produto != null ? produto.quantidade : ''}">
        </div>
        <div>
            <label>Preço:</label>
            <input type="text" name="precoVenda" value="${produto != null ? produto.precoVenda : ''}">
        </div>

        <div>
            <button type="submit">${param.acao eq 'formIncluir' ? 'Incluir' : 'Alterar'}</button>
        </div>
    </form>

</body>
</html>
```

- Executando e testando por meio do endereço http://localhost:8080/CadEE-war/ProdutoFrontController

Lista de Produtos

ID	Nome	Quantidade	Preço	Ações
1	Banana	100	5.0	Alterar | Excluir
2	Laranja	500	2.0	Alterar | Excluir
3	Manga	800	4.0	Alterar | Excluir
4	abacate	1	3.4	Alterar | Excluir
5	abacateeditado	1	3.4	Alterar | Excluir

Incluir Novo Produto

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

Resposta:

- Em um aplicativo web Java com arquitetura *MVC*, o *Front Controller* recebe todas as requisições dos usuários e faz o roteamento para o controlador apropriado. O controlador manipula a requisição conforme necessário e seleciona a visualização apropriada para enviar de volta ao usuário.

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

Resposta:

- Servlets são como controladores que recebem e processam solicitações do navegador e que podem gerar respostas em *HTML* puro. Enquanto isso, *JSPs* são páginas *HTML* com fragmentos de código Java embutidos, tornando mais fácil misturar lógica de negócios com a apresentação da página.

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?

Resposta:

- Um redirecionamento simples leva o navegador para outra página, enquanto o método *forward()* envia a requisição do navegador

diretamente para outra página no mesmo servidor. Parâmetros e atributos no objeto *HttpRequest* são usados para enviar e receber informações entre páginas e *Servlets*, facilitando o compartilhamento de dados.

3º Procedimento | Melhorando o Design da Interface

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 2º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

- Adicionando a importação necessária para o Bootstrap e utilizando suas classes para melhoria do design.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Dados do Produto</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWT
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB
    </head>
    <body class="container">
    <h1>Dados do Produto</h1>
    <form class="form" action="ProdutoFrontController" method="post">
        <input type="hidden" name="acao" value="${param.acao eq 'formIncluir' ? 'incluir' : 'alterar'}">
        <input type="hidden" name="id" value="${produto != null ? produto.idProduto : ''}">
        <div class="mb-3">
            <label class="form-label">Nome:</label>
            <input class="form-control" type="text" name="nome" value="${produto != null ? produto.nome : ''}">
        </div>
        <div class="mb-3">
            <label class="form-label">Quantidade:</label>
            <input class="form-control" type="text" name="quantidade" value="${produto != null ? produto.quantidade : ''}">
        </div>
        <div class="mb-3">
            <label class="form-label">Preço:</label>
            <input class="form-control" type="text" name="precoVenda" value="${produto != null ? produto.precoVenda : ''}">
        </div>

        <div>
            <button class="btn btn-primary" type="submit">${param.acao eq 'formIncluir' ? 'Incluir' : 'Alterar'}</button>
        </div>
    </form>

</body>
</html>
```

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
    <title>Lista de Produtos</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPE
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmX
    </head>
<body class="container">
    <h1>Listagem de Produtos</h1>
    <a class="btn btn-primary m-2" href="ProdutoFrontController?acao=formIncluir">Incluir Novo Produto</a>
    <table class="table table-striped ">
        <thead class="table-dark">
            <th>ID</th>
            <th>Nome</th>
            <th>Quantidade</th>
            <th>Preço</th>
            <th>Ações</th>
        </thead>
        <c:forEach var="produto" items="${produtos}">
            <tr>
                <td>${produto.idProduto}</td>
                <td>${produto.nome}</td>
                <td>${produto.quantidade}</td>
                <td>${produto.precoVenda}</td>
                <td>
                    <a class=" btn btn-primary  btn-sm" href="ProdutoFrontController?acao=formAlterar&id=${produto.idProduto}">Alterar
                    <a class=" btn btn-danger btn-sm" href="ProdutoFrontController?acao=excluir&id=${produto.idProduto}">Excluir</a>
                </td>
            </tr>
        </c:forEach>
    </table>

</body>
```

- Executando e testando por meio do endereço http://localhost:8080/CadEE-war/ProdutoFrontController

a) Como o framework Bootstrap é utilizado?

Resposta:

- É usado para fazer websites e aplicativos web ficarem bonitos e fáceis de usar em diferentes tipos de dispositivos. Ele vem com muitos estilos e componentes prontos para usar, como botões e formulários, que ajudam os desenvolvedores a criar as páginas web.

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

Resposta:

- O Bootstrap garante a independência estrutural do HTML porque os estilos e componentes fornecidos pelo Bootstrap são aplicados através de classes CSS, e não através de elementos HTML específicos.

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

Resposta:

- O Booststrap fornece uma variedade de componentes responsivos que se adaptam automaticamente ao tamanho da tela do dispositivo em que a página está sendo visualizada

Conclusão

A implementação deste sistema não apenas solidificou os conhecimentos teóricos, mas também aprimorou as habilidades práticas na construção de aplicações Java Web. Ao utilizar JPA para persistência e EJBs para a lógica de negócio, foi possível criar uma base robusta para aplicações corporativas como esta. Além disso, a integração de Servlets e JSPs, em companhia da biblioteca Bootstrap para aprimorar o design, resultou em uma aplicação web dinâmica e visualmente atraente. Este projeto destacou a importância de seguir padrões de desenvolvimento e boas práticas de programação, oferecendo uma base sólida para os próximos projetos com Java Web.