



EAD - Polo Santa Luiza – Vitória – ES

DESENVOLVIMENTO FULL STACK

RELATÓRIO DE PRÁTICA

MUNDO 3 – NÍVEL 4 – MISSÃO PRÁTICA

RPG0017 - Vamos integrar sistemas

CARLOS ALTOMARE CATÃO

Matrícula: 202403460912

Semestre Letivo: 2025/1 - 3º Período

Data: 2025/05

Objetivos da Prática

A Missão Prática do Mundo 3 Nível 4 se compõe por 3 procedimentos que de maneira evolutiva permitem navegar por várias tecnologias, conforme o grau de sofisticação dos recursos utilizados.

Basicamente trabalha-se o desenvolvimento de uma aplicação web que possibilita trabalhar com camadas de persistência e controle; utilizando um projeto configurado na **IDE NetBeans**, utilizando as tecnologias **Jakarta EE**, com foco na construção de **entidades JPA**, criação de **Session Beans (EJB)** para operações **CRUD**, além da configuração de **conexão com banco de dados SQL Server** via *persistence.xml*.

Será utilizado um servidor *GlassFish*, apoiado por um *Servlet*, encarregado por responder a requisições feitas por clientes, geralmente encaminhadas via **protocolo HTTP**.

Na questão de estilização será trabalhada a utilização de interfaces como o **Bootstrap**, de maneiras a proporcionar uma melhor experiência visual ao usuário.

A proposta busca exercitar conhecimentos de desenvolvimento Full Stack utilizando tecnologias padrão do ecossistema Java corporativo, promovendo a integração entre camadas de apresentação, controle e persistência de dados.

Requisitos

- ✓ Jakarta EE
- ✓ Servlets
- ✓ EJB (Enterprise JavaBeans)
- ✓ JPA (Jakarta Persistence API)
- ✓ SQL Server
- ✓ JDBC Driver para SQL Server (configurado no GlassFish)
- ✓ NetBeans
- ✓ GlassFish
- ✓ Git
- ✓ Java
- ✓ JSP
- ✓ HTML
- ✓ Ant (sistema de build)

- ✓ JDK 17
- ✓ Google Chrome
- ✓ Bootstrap

Procedimento 1 - Camadas de Persistência e Controle

Objetivos

O objetivo do Procedimento 1 foi o desenvolvimento de uma aplicação web baseada em **Jakarta EE**, denominada **CadastroEE**, possibilitando trabalhar com as camadas de persistência e controle; para tanto foi configurado todo um projeto na **IDE NetBeans**, utilizando as tecnologias **Jakarta EE**, com foco na construção de **entidades JPA**, criação de **Session Beans (EJB)** para operações **CRUD**, além da configuração de **conexão com banco de dados SQL Server** via *persistence.xml*.

Foi utilizado um servidor *GlassFish*, apoiado por um *Servlet*, encarregado por responder a requisições feitas por clientes, geralmente encaminhadas via **protocolo HTTP**.

A prática também incluiu a migração de código para o **namespace jakarta**, conforme as atualizações das especificações Jakarta EE 10.

Desta forma, a aplicação serve como base para sistemas empresariais que exigem persistência de dados, organização em camadas e acesso via interface web.

Tecnologias Utilizadas

  Jakarta EE

 Servlets

 EJB (Enterprise JavaBeans)

 JPA (Jakarta Persistence API)

 SQL Server

 JDBC Driver para SQL Server (configurado no GlassFish)

 NetBeans

 GlassFish

 Git

 Java

 JSP

 HTML

 Ant (sistema de build)

 JDK 17

Estrutura do Projeto

O projeto obedeceu a uma estrutura organizada de forma a separar as responsabilidades em diferentes **módulos** ou **camadas**.

1. **Projeto Web (WAR)**: Contém os **Servlets**, **JSPs**, **recursos estáticos** (imagens, CSS, JS), e o **arquivo de configuração web.xml** .

Função:

- a Controlar requisições e respostas HTTP através de Servlets.
- b **Exibir páginas ao usuário**, sendo a camada de apresentação da aplicação.
- c **Delegar as ações** para o EJB, que executa a lógica de negócio.

2. **Projeto EJB (JAR)**: Contém **Session Beans**, as **entidades JPA** para persistência e as classes que compõem a **lógica de negócio**.

Função:

- a Implementar as regras de negócio, isolando a lógica da camada web.
- b Efetuar a persistência em banco de dados usando Jakarta Persistence (**JPA**).
- c Expor funcionalidades via **Session Beans**, possibilitando escalabilidade e reutilização de código.

3. **Configurações e Bibliotecas:**

- a *persistence.xml*: Define a **unidade de persistência**, incluindo o datasource (ex: jdbc/loja) e as propriedades do provedor **JPA**.
- b *web.xml*: Arquivo de configuração do **módulo web**, onde são definidos os mapeamentos de **Servlets**, filtros, **listeners**, tempo de sessão, etc.

c **Bibliotecas:**

- **Jakarta EE 10 API** → adicionada ao projeto para suportar **Servlets**, **EJB**, **JPA**.

- **GlassFish Server** → Servidor de aplicação para execução e testes.

Funcionalidades

A aplicação oferece as seguintes funcionalidades:

- ✓ Conexão Segura com o Banco de Dados.
- 📁 Modelagem e Mapeamento de Entidades
- 💰 Ajuste no Tipo de Dados
- ⚙️ Geração Automática de Session Beans
- 📖 Inclusão da Biblioteca Jakarta EE 10
- 🔄 Atualização das Importações
- 📝 Configuração da Persistência
- 🔄 Integração entre os módulos

[Client Browser]



[Servlet]



[Session Bean]



[JPA / DB]

Atividades Realizadas

- ✓ Configuração da conexão com SQL Server via JDBC.
- ✓ Ajuste do parâmetro `trustServerCertificate=true` para permitir conexões seguras.
- ✓ Criação das entidades de modelo, com destaque para Produto.
- ✓ Alteração do tipo do atributo `precoVenda` de *BigDecimal* para *Float*.
- ✓ Geração automática de *Session Beans* (EJB) com interfaces locais através do assistente "New → Session Beans for Entity Classes" no NetBeans.

- ✓ Inclusão da biblioteca **Jakarta EE 10** ao projeto.
- ✓ Atualização de todas as importações de *javax* para *jakarta*.
- ✓ Ajuste do arquivo *persistence.xml* conforme especificado.
- ✓ Modelagem da entidade Produto com anotações JPA;
- ✓ Criação da camada de persistência (ProdutoFacade) com uso do padrão Facade;
- ✓ Configuração do persistence.xml com datasource JTA para conexão com banco de dados;
- ✓ Desenvolvimento do ServletProduto para apresentar uma listagem dinâmica de produtos em HTML;
- ✓ Empacotamento dos módulos como .jar (EJB), .war (web) e .ear (distribuição final);
- ✓ Deploy e testes no servidor GlassFish.

Desafios Enfrentados

- ! Erro de conexão com o SQL Server
- ! Alteração de imports javax para jakarta
- ! Geração dos Session Beans (EJB) com o assistente.
- ! Erro no XML.
- ! Bloqueio de arquivos JAR.
- ! Configuração de datasource JDBC no servidor e ajuste de permissões.
- ! Necessidade de ajustes de compatibilidade entre versões do JDK e configurações do Ant.

Códigos

Os códigos foram desenvolvidos com a IDE NetBeans e se encontram em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_4_Missao_Pratica/tree/main/Procedimento-1/CadastroEE.

Testes

Os testes foram realizados de forma manual, validando os seguintes pontos:

- Execução do servlet ServletProduto via navegador;
- Correta renderização da tabela HTML com dados do banco;
- Conectividade com o banco de dados via JPA;
- Testes com entidades reais no banco;
- Verificação dos logs do GlassFish em tempo real para identificação de erros.

Resultados

Os resultados da execução dos códigos se encontram ilustrados no arquivo *Resultados.pdf* que se encontra em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_4_Missao_Pratica/blob/main/Procedimento-1/RESULTADOS.pdf.

Conclusão

Esta prática proporcionou experiência na **configuração de ambientes corporativos** utilizando **Jakarta EE**, com atenção especial às mudanças na especificação.

O projeto "CadastroEE" foi concebido com uma arquitetura modular - foi implantado no servidor como um arquivo **.ear**, que inclui **.war** (web) e **.jar** (EJB) – o que facilita nas **manutenções futuras** e na **escalabilidade**.

O uso de tecnologias modernas e práticas como **EJB**, **JPA** e **Servlets** permitiu a separação clara entre camadas e responsabilidade de cada módulo, e facilitaram os testes.

Apesar dos desafios técnicos, a aplicação demonstrou robustez e eficiência.

As dificuldades enfrentadas foram superadas com pesquisa, entendimento da documentação oficial e apoio das ferramentas do próprio NetBeans.

A prática proporcionou uma visão completa sobre o ciclo de desenvolvimento, desde a modelagem até a implantação em um servidor de aplicações.

O projeto finalizou com sucesso, possibilitando a realização de operações **CRUD**, com persistência em banco de dados.

O exercício permitiu consolidar conhecimentos técnicos importantes, como:

- ✓ **Programação Java** aplicada a sistemas corporativos.
- ✓ **Criação e configuração de Servlets** para manipular requisições HTTP.
- ✓ **Desenvolvimento de componentes EJB** para encapsular regras de negócio.
- ✓ **Mapeamento de entidades JPA** e persistência de dados em **SQL Server**.
- ✓ **Configuração de JDBC Driver** para estabelecer conexões com o banco de dados.
- ✓ **Integração de módulos** em um ambiente **Jakarta EE 10**, utilizando **NetBeans**.
- ✓ **Configuração de servidores de aplicação** como **GlassFish**.

? Questionamentos

▶ Como é organizado um projeto corporativo no NetBeans?

Um **projeto corporativo** no NetBeans, especialmente quando voltado para **Java EE/Jakarta EE**, é geralmente estruturado de forma **modular**, permitindo separação de responsabilidades. Os principais módulos são:

- **Módulo EJB (Enterprise JavaBeans):** Contém a lógica de negócios. Inclui Session Beans, entidades JPA, etc.
- **Módulo Web:** Responsável pela interface com o usuário (Servlets, JSP, JSF).
- **Módulo EAR (Enterprise Archive):** Empacota os módulos EJB e Web em um único arquivo .ear para implantação em servidores como GlassFish ou WildFly.
- **Bibliotecas compartilhadas:** Utilizadas por múltiplos módulos, como arquivos JAR de dependências externas.

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

- **JPA (Java Persistence API):**
 - Responsável pela **persistência de dados** (interação com banco de dados).
 - Mapeia objetos Java para tabelas do banco de dados (ORM).
 - Facilita operações CRUD sem a necessidade de SQL explícito.
- **EJB (Enterprise JavaBeans):**
 - Fornece uma camada de negócios transacional, segura e escalável.
 - Os **Session Beans** encapsulam lógica de negócio e são gerenciados pelo container EJB.
 - Permite transações distribuídas, segurança declarativa, e injeção de dependência.

Juntos, EJB e JPA formam a base do *backend* de um sistema corporativo Java EE.

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

O NetBeans melhora a produtividade através de várias facilidades incorporadas, tais como:

- Assistentes para criação de entidades **JPA** a partir de tabelas do banco de dados.
- Geração automática de facades **EJB** para entidades JPA (métodos CRUD prontos).
- Refatoração assistida, completamento de código, e validação em tempo real.
- Gerenciamento visual de persistência e EJBs.
- Deploy automático em servidores integrados.
- Suporte a **anotações JPA e EJB**, com sugestões e validações automáticas.

▶ 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 tratam requisições HTTP. Fazem parte da especificação Java EE e são utilizados para controlar o fluxo de dados entre a camada de apresentação (navegador) e a lógica de negócios.

O **NetBeans** oferece suporte a Servlets por meio de:

- Assistente de criação de **Servlet** com geração de código básico.
- Suporte ao mapeamento de URLs via anotação (**@WebServlet**) ou via *web.xml*.
- Execução e debug automático em servidores integrados.
- Auto-complete e validação para métodos como *doGet*, *doPost*, etc.

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

A comunicação entre **Servlets (Web layer)** e **Session Beans (EJBs - Business layer)** é feita por meio de *injeção de dependência*, utilizando a anotação `@EJB` dentro do Servlet.

Essa injeção permite que o container EJB entregue automaticamente uma instância gerenciada do Session Bean ao Servlet. Isso garante:

- Gerenciamento de transações.
- Controle de concorrência.
- *Pooling* eficiente de recursos.

Procedimento 2 - Interface Cadastral com Servlet e JSPs

Objetivos

O principal objetivo deste procedimento foi desenvolver uma aplicação web funcional baseada na plataforma Java EE, com foco na implementação de um CRUD (Create, Read, Update, Delete) para o gerenciamento completo da entidade **Produto**. A proposta buscou exercitar conhecimentos de desenvolvimento Full Stack utilizando tecnologias padrão do ecossistema Java corporativo, promovendo a integração entre camadas de apresentação, controle e persistência de dados.

O foco foi explorar a integração entre Servlets, JSP, EJB e JPA em um ambiente web.


Tecnologias Utilizadas

 Jakarta EE

 Servlets

 EJB (Enterprise JavaBeans)

 JPA (Jakarta Persistence API)

 SQL Server

 JDBC Driver para SQL Server (configurado no GlassFish)

 NetBeans

 GlassFish

 Git


 Java

 JSP

 HTML

 Ant (sistema de build)

 JDK 17

 Google Chrome

Estrutura do Projeto

O projeto foi dividido nas seguintes camadas:

- modelo/Produto.java → Classe entidade representando a tabela produto, anotada com JPA para mapeamento objeto-relacional.
- modelo/ProdutoDAO.java → Classe de acesso a dados usando JPA.
- Interface ProdutoFacadeLocal encapsulando o acesso aos dados com EJB.
- controle/ServletProdutoFC.java → *Servlet* controlador responsável por gerenciar as requisições CRUD para inclusão, listagem, alteração e exclusão e encaminhar para as páginas JSP.
- ProdutoDados.jsp → Formulário para inclusão e edição de produtos.
- ProdutoLista.jsp → Página para listagem e gerenciamento de produtos com opções de alteração e exclusão.

Funcionalidades

- Cadastro de novos produtos com nome, quantidade e preço.
- Listagem de todos os produtos em formato de tabela.
- Edição dos dados de um produto existente.
- Exclusão de produtos com atualização automática da lista.
- Links dinâmicos para ações no ServletProdutoFC.
- Encaminhamento de requisições por RequestDispatcher.

Atividades Realizadas

- ✓ Criação da estrutura de diretórios Java EE.
- ✓ Modelagem da entidade Produto.
- ✓ Implementação da camada de persistência com JPA.
- ✓ Desenvolvimento do servlet controlador com tratamento de múltiplas ações (acao).
- ✓ Criação das páginas JSP para interação com o usuário.
- ✓ Testes funcionais de cada operação do CRUD.
- ✓ Tratamento de conversão de tipos (ex: Float.parseFloat para preços).
- ✓ Organização de navegação entre páginas e envio de dados.

Desafios Enfrentados

- ! Conversão correta de valores numéricos no `request.getParameter` para tipos primitivos como `float` e `int`.
- ! Controle de redirecionamento e envio de parâmetros entre páginas JSP e Servlets.
- ! Configuração do ambiente de servidor (GlassFish/WildFly) e persistência com JPA.
- ! Tratamento de requisições com múltiplas ações no servlet usando `if-else`.
- ! Evitar erros de `NullPointerException` ao acessar objetos ou listas não inicializadas.

Códigos

Os códigos foram desenvolvidos com a IDE NetBeans e se encontram em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_4_Missao_Pratica/tree/main/Procedimento-2/CadastroEE.

Testes

O processo de testes é essencial para garantir a robustez do sistema e identificar pontos de melhoria tanto na interface quanto na lógica de negócios. Através dessas iterações, o sistema evolui para uma versão estável e funcional.

Testes Realizados

1. **Teste de Inclusão**

- Foram testados cenários com campos válidos e inválidos. Verificou-se a adição correta do produto na lista e a persistência dos dados no banco.

2. **Teste de Alteração**

- Verificou-se se os dados do produto eram carregados corretamente ao clicar em "Alterar", e se, após a edição, os dados eram atualizados corretamente na listagem.

3. Teste de Exclusão

- Testou-se a exclusão de produtos com e sem confirmação. Após confirmar a exclusão, foi verificado se o produto desaparecia da lista.

4. Teste de Validação

- Foram realizados testes para envio do formulário com campos em branco, valores negativos e preço com formato inválido.

5. Teste de Navegação

- Todos os fluxos de navegação foram testados: incluir, alterar, excluir, voltar à lista. A consistência do estado da aplicação foi verificada após cada ação.

Resultados

Os resultados da execução dos códigos se encontram ilustrados no arquivo *Resultados.pdf* que se encontra em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_4_Missao_Pratica/blob/main/Procedimento-2/RESULTADOS.pdf.

Conclusão

O desenvolvimento deste procedimento permitiu aplicar os principais conceitos da arquitetura Java EE para web, incluindo a separação de responsabilidades entre camada de apresentação, controle e persistência. A integração entre Servlets e JSP foi consolidada, assim como a utilização de JPA para abstração do banco de dados.

O uso de ServletProdutoFC como controlador central demonstrou ser uma abordagem eficaz para controlar a lógica do sistema e a navegação entre as telas.

Como resultado final, obteve-se uma aplicação funcional, organizada e preparada para futuras extensões, como autenticação de usuários ou filtros por categorias.

O projeto reforçou a importância de boas práticas no desenvolvimento Full Stack, como reutilização de código, clareza na organização e validação de dados.

? Questionamentos

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

O padrão **Front Controller** é um padrão de projeto onde todas as requisições passam por um único ponto de entrada, que centraliza o controle da navegação e do fluxo da aplicação.

- Recebe todas as requisições HTTP.
- Analisa a requisição (parâmetros, URL, ações).
- Encaminha a requisição para o componente apropriado (geralmente um Controller específico ou diretamente para a View).
- É o coração da camada **Controller** na arquitetura MVC.

Implementação em Java:

Em aplicações Java Web, o Front Controller, geralmente, é implementado como um **Servlet** ou com frameworks como **Spring MVC**, que automatizam isso com controllers anotados.

▶ Quais as diferenças e semelhanças entre Servlets e JSPs?

Diferenças:

Característica	Servlet	JSP (JavaServer Pages)
Objetivo principal	Lógica de controle (Controller)	Apresentação (View)
Sintaxe	Código Java puro	Mistura HTML com Java (scripts ou EL)
Foco	Processamento de requisições	Renderização da resposta (HTML)

Característica	Servlet	JSP (JavaServer Pages)
Facilidade	Menos legível para interface	Mais intuitivo para designers
Compilação	Já compilado como classe Java	Traduzido em Servlet na primeira execução

Semelhanças:

- Ambos são executados no servidor e geram conteúdo dinâmico.
- Fazem parte da especificação Java EE.
- Podem acessar `HttpServletRequest`, `HttpServletResponse`, sessões e contexto da aplicação.
- Podem se comunicar entre si.

Finalizando, pode-se afirmar que **Servlets** controlam o fluxo e **JSPs** exibem a interface.

► 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 `HttpServletRequest`?

Redirecionamento Simples (`response.sendRedirect`):

- Envia uma nova resposta HTTP com código 302 (redirecionamento).
- O navegador faz nova requisição para a nova URL.
- Perde os dados do *request* original.
- A URL no navegador é atualizada.
- Útil para redirecionar para outra página após um **POST**, por exemplo.

Encaminhamento (`RequestDispatcher.forward`):

- Ocorre internamente no servidor, sem o navegador saber.
- A requisição não é reiniciada, os objetos *request* e *response* são mantidos.
- A URL no navegador permanece a mesma.
- Útil para MVC: *controller* processa e encaminha para JSP exibir.

Parâmetros vs. Atributos no HttpServletRequest:

Tipo	Método	Escopo	Exemplo de uso
Parâmetro	request.getParameter()	Requisição HTTP (query/form)	Dados enviados pelo usuário
Atributo	request.setAttribute()	Durante o ciclo da requisição	Dados entre Servlet e JSP

Procedimento 3 - Melhorando o Design da Interface

Objetivos

O objetivo deste procedimento foi implementar e aprimorar um sistema de cadastro de produtos, permitindo as operações de **inclusão, alteração, listagem e exclusão** de dados de produtos de forma prática, eficiente e visualmente agradável.

Foram aplicadas boas práticas de desenvolvimento utilizando **design patterns** como **Facade** e estilização por meio de interfaces com o **Bootstrap**, de maneiras a proporcionar uma melhor experiência visual ao usuário.


Tecnologias Utilizadas

  Jakarta EE

 Servlets

 EJB (Enterprise JavaBeans)

 JPA (Jakarta Persistence API)

 SQL Server

 JDBC Driver para SQL Server (configurado no GlassFish)

 NetBeans

 GlassFish

 Git


 Java

 JSP

 HTML

 Ant (sistema de build)

 JDK 17

 Google Chrome

 Bootstrap

Estrutura do Projeto

O projeto foi dividido nas seguintes camadas:

- modelo/Produto.java → Classe entidade representando a tabela produto, anotada com JPA para mapeamento objeto-relacional.
- modelo/ProdutoDAO.java → Classe de acesso a dados usando JPA.
- Interface ProdutoFacadeLocal encapsulando o acesso aos dados com EJB.
- controle/ServletProdutoFC.java → *Servlet* controlador responsável por gerenciar as requisições CRUD para inclusão, listagem, alteração e exclusão e encaminhar para as páginas JSP.
- ProdutoDados.jsp → Formulário para inclusão e edição de produtos.
- ProdutoLista.jsp → Página para listagem e gerenciamento de produtos com opções de alteração e exclusão.

Funcionalidades

- Cadastro de novos produtos com nome, quantidade e preço.
- Listagem de todos os produtos em formato de tabela.
- Edição dos dados de um produto existente.
- Exclusão de produtos com atualização automática da lista.
- Links dinâmicos para ações no ServletProdutoFC.
- Encaminhamento de requisições por RequestDispatcher.

Atividades Realizadas

- ✓ Criação do ProdutoFacade encapsulando a lógica de validação e interação com o Facade.
- ✓ Desenvolvimento do ServletProdutoFC responsável por receber requisições HTTP e delegar ações ao ProdutoFacade.
- ✓ Implementação e estilização das páginas JSP:
 - ProdutoLista.jsp: aplicação das classes do Bootstrap para estilização da tabela e botões.
 - ProdutoDados.jsp: encapsulamento de campos em div com classes Bootstrap e ajustes para responsividade.

- ✓ Validações nos campos garantindo que não sejam aceitos valores nulos ou inválidos.
- ✓ Formatação correta dos campos, como a exibição do preço de venda com duas casas decimais.
- ✓ Configuração do ambiente incluindo GlassFish, SQL Server, JDBC e integração com o NetBeans.

Desafios Enfrentados

- ! Compatibilidade de versões: ajustar a versão do Bootstrap para garantir compatibilidade com Jakarta EE 10 e o ambiente atual.
- ! Conversão de dados: tratamento adequado de formatos numéricos, especialmente na exibição de preços com duas casas decimais.
- ! Integração frontend-backend: garantir que os valores enviados e recebidos entre JSP, Servlet e banco de dados estivessem corretos.
- ! Validação server-side e client-side: implementar validações robustas tanto no frontend quanto no backend, prevenindo falhas e inconsistências.

Códigos

Os códigos foram desenvolvidos com a IDE NetBeans e se encontram em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_4_Missao_Pratica/tree/main/Procedimento-3/CadastroEE.

Testes

O processo de testes é essencial para garantir a robustez do sistema e identificar pontos de melhoria tanto na interface quanto na lógica de negócios. Através dessas iterações, o sistema evolui para uma versão estável e funcional.

Testes Realizados

6. Teste de Inclusão

- Foram testados cenários com campos válidos e inválidos. Verificou-se a adição correta do produto na lista e a persistência dos dados no banco.

7. Teste de Alteração

- Verificou-se se os dados do produto eram carregados corretamente ao clicar em "Alterar", e se, após a edição, os dados eram atualizados corretamente na listagem.

8. Teste de Exclusão

- Testou-se a exclusão de produtos com e sem confirmação. Após confirmar a exclusão, foi verificado se o produto desaparecia da lista.

9. Teste de Validação

- Foram realizados testes para envio do formulário com campos em branco, valores negativos e preço com formato inválido.

10. Teste de Navegação

- Todos os fluxos de navegação foram testados: incluir, alterar, excluir, voltar à lista. A consistência do estado da aplicação foi verificada após cada ação.

Resultados

Os resultados da execução dos códigos se encontram ilustrados no arquivo *Resultados.pdf* que se encontra em repositório no GitHub onde podem ser acessados pelo link: https://github.com/CarlosCatao/Mundo_3_Nivel_4_Missao_Pratica/blob/main/Procedimento-3/RESULTADOS.pdf.

Conclusão

A realização deste procedimento permitiu consolidar conhecimentos essenciais em desenvolvimento de sistemas web com Java e Jakarta EE, além de reforçar a importância do uso de frameworks de estilização como o Bootstrap para proporcionar interfaces modernas e responsivas. A aplicação do padrão

Facade trouxe maior organização e manutenibilidade ao sistema, centralizando regras de negócio.

Apesar dos desafios técnicos encontrados, especialmente relacionados à compatibilidade e formatação de dados, as soluções implementadas demonstraram eficácia e alinhamento com boas práticas de desenvolvimento. O sistema resultante é funcional, organizado e com uma interface intuitiva para o usuário final.

? Questionamentos

▶ Como o framework Bootstrap é utilizado?

O **Bootstrap** é um framework front-end baseado em **HTML**, **CSS** e **JavaScript** que facilita a criação de interfaces modernas, responsivas e compatíveis com diversos navegadores e dispositivos.

Para usar o **CDN do Bootstrap**, basta incluir as tags de *link* e *script* no seu arquivo HTML. O **Bootstrap** oferece os arquivos necessários para **CSS** e **JavaScript** diretamente através de um link de **CDN (Content Delivery Network)**, o que facilita a implementação sem precisar baixar nada no seu servidor.

▶ Por que o Bootstrap garante a independência estrutural do HTML?

O **Bootstrap** separa a estrutura (**HTML**) da aparência e comportamento, promovendo independência estrutural, porque:

- HTML mantém apenas a marcação sem estilo personalizado embutido.
- Toda a estilização é feita por meio de classes CSS padronizadas.
- Esta padronização permite que se mude o layout visual sem reescrever o HTML, apenas alterando ou combinando as classes.
- Reutilização de código
- Padronização visual
- Facilidade de manutenção

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

A responsividade é uma característica que permite a uma página web se adaptar automaticamente a diferentes tamanhos de tela (desktop, tablet, celular).

A responsividade no Bootstrap é possível graças a um conjunto de grade flexível (grid system), classes utilitárias e componentes interativos, tudo projetado para se ajustar automaticamente às diferentes resoluções de tela.

Desta forma, por meio de várias ferramentas e classes integradas, o Bootstrap habilita a criação de layouts responsivos, sem precisar escrever media queries manuais, o que economiza tempo e reduz erros.

Conclusão Final da Prática

A realização da Missão Prática do Mundo 3 – Nível 4 proporcionou uma imersão completa no desenvolvimento de aplicações corporativas utilizando o ecossistema Jakarta EE, reforçando conceitos fundamentais de arquitetura em múltiplas camadas, integração de tecnologias e boas práticas de desenvolvimento Full Stack.

Ao longo dos três procedimentos, foi possível experimentar a construção de um sistema completo, desde a modelagem das entidades e configuração da persistência com JPA, passando pela implementação de regras de negócio com EJB, até a criação de uma interface web funcional e responsiva com Servlets, JSP e estilização com Bootstrap.

Os desafios enfrentados, especialmente relacionados à configuração de ambientes, integração de módulos e adequação às atualizações das especificações Jakarta EE, foram superados com pesquisa, consulta à documentação oficial e aplicação de soluções técnicas adequadas. Esses obstáculos foram essenciais para o aprimoramento das habilidades práticas e para a consolidação do conhecimento teórico.

O projeto resultou em uma aplicação web robusta, modular, escalável e preparada para evoluções futuras, como autenticação de usuários, filtros avançados e integração com APIs externas. A aplicação do padrão MVC e do design pattern Facade conferiu clareza, organização e facilidade de manutenção ao código.

O uso do Bootstrap elevou a qualidade da interface, proporcionando uma experiência mais agradável e intuitiva ao usuário final, demonstrando a importância da preocupação com a camada de apresentação.

De maneira geral, a prática consolidou competências essenciais para o desenvolvedor Full Stack, como:

- ✅ Desenvolvimento e integração de sistemas baseados em Jakarta EE;
- ✅ Aplicação de conceitos de arquitetura em camadas;
- ✅ Utilização de EJBs para encapsulamento de regras de negócio;
- ✅ Persistência de dados com JPA e SQL Server;
- ✅ Construção de interfaces web dinâmicas com Servlets e JSP;

- ✓ Estilização responsiva com Bootstrap;
- ✓ Deploy e testes em servidores corporativos como GlassFish.

A prática foi concluída com êxito, representando uma importante etapa no processo de formação profissional, ampliando a visão sobre o ciclo de vida completo de desenvolvimento de sistemas corporativos e fortalecendo a capacidade de lidar com tecnologias amplamente utilizadas no mercado.