 **POLO COHABINAL - PARNAMIRIM - RN**

CURSO: **DESENVOLVIMENTO FULL STACK**

DISCIPLINA: **VAMOS INTEGRAR SISTEMAS**

TURMA: **2024.1** SEMESTRE: **3°**

ALUNO: **ENILDO ARAÚJO DE OLIVEIRA JÚNIOR** MAT.: **202402159101**

GIT: **https://github.com/Enildo/javaee**

**Missão Prática - Nível 4**

Objetivos 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.

**CÓDIGOS JAVA**

No GIT **https://github.com/Enildo/javaee**

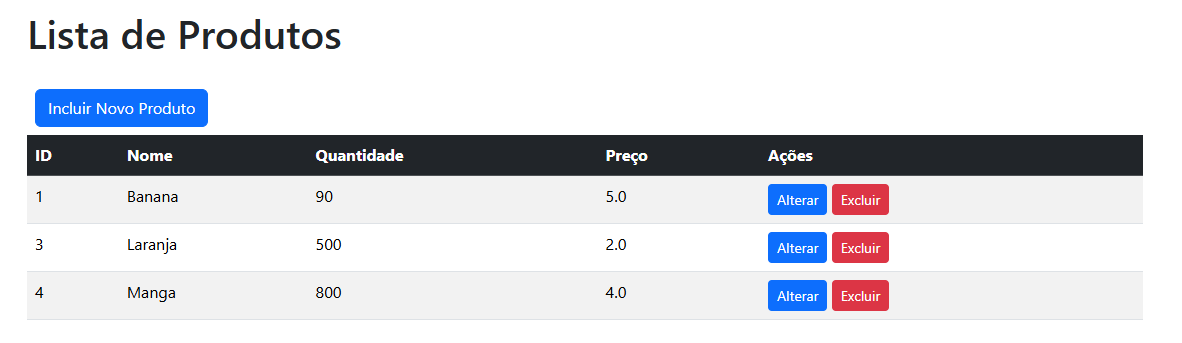
**Resultados**

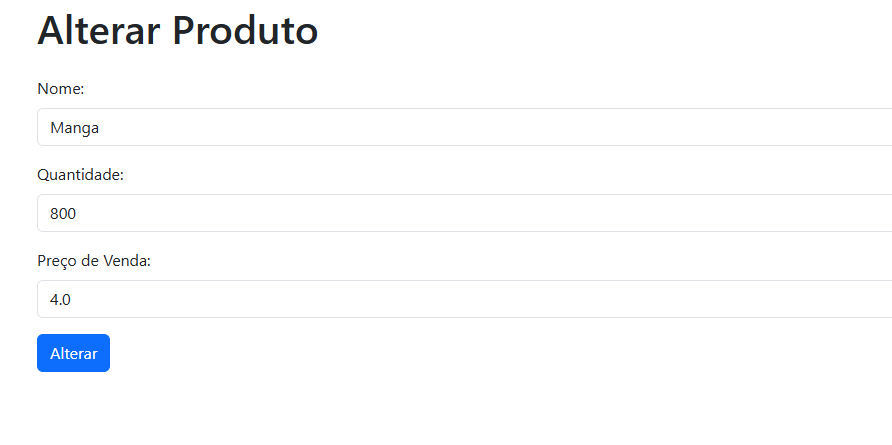




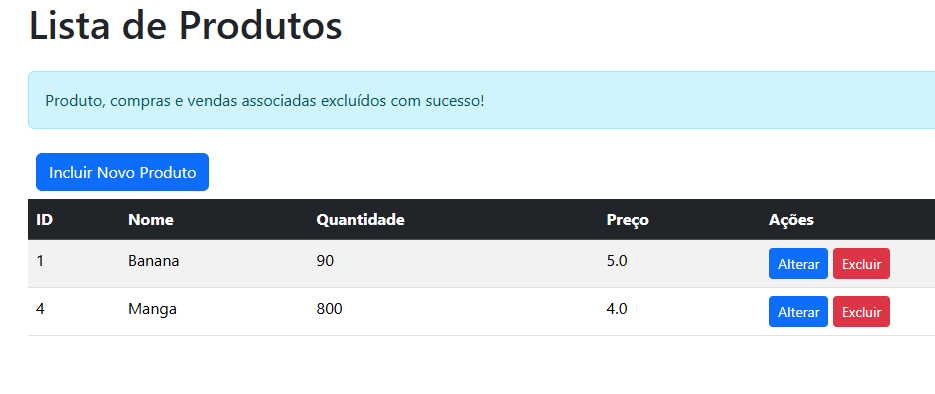












## ****ANÁLISE E CONCLUSÃO****

**✅ 1. Como é organizado um projeto corporativo no NetBeans?**

➡️ **Organização por módulos**:

* **EJB Module (CadastroEE-ejb)**: Contém regras de negócio, entidades JPA e Session Beans.
* **Web Module (CadastroEE-war)**: Contém servlets, JSPs, arquivos estáticos e web.xml.
* **EAR Project (CadastroEE)**: Empacota os módulos EJB e WAR para deploy conjunto.

➡️ **Padrão corporativo** → separação clara entre **camada de negócios** e **camada de apresentação**.

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

➡️ **JPA (Java Persistence API):**

* Gerencia o **mapeamento objeto-relacional (ORM)** entre classes Java e tabelas no banco de dados.
* Facilita operações CRUD via **EntityManager**.

➡️ **EJB (Enterprise Java Beans):**

* Implementa a **lógica de negócios** com componentes como **Session Beans**.
* Garante recursos como **transações, segurança e injeção de dependências**.

✅ Ambas são fundamentais para criar aplicações **modulares, reutilizáveis e escaláveis**.

**✅ 3. Como o NetBeans viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?**

➡️ NetBeans oferece:  
✅ **Assistentes automáticos** para gerar **Entidades JPA** a partir do banco de dados.  
✅ Geradores de **Session Beans (Facade)** para encapsular operações CRUD.  
✅ Integração com servidores (GlassFish, Payara) para **deploy direto** e **teste fácil**.  
✅ Autocompletar e validação para anotações JPA/EJB (@Entity, @Stateless).

**✅ 4. O que são Servlets e como o NetBeans oferece suporte à construção desse tipo de componente em um projeto Web?**

➡️ **Servlets** são componentes Java que processam **requisições HTTP** e geram **respostas** dinâmicas.

➡️ O NetBeans facilita:  
✅ Criando automaticamente a estrutura do servlet e configurando no web.xml.  
✅ Suporte à anotação @WebServlet.  
✅ Geração automática de métodos como doGet() e doPost().

**✅ 5. Como é feita a comunicação entre os Servlets e os Session Beans do pool de EJBs?**

➡️ Via **injeção de dependência** usando @EJB.

Exemplo:

java

CopiarEditar

@EJB

private ProdutoFacadeLocal facade;

✅ O **container EJB** gerencia o **ciclo de vida** e entrega a instância para o servlet.  
✅ O servlet usa o Session Bean para acessar a **lógica de negócios** e a **persistência**.

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

➡️ O **Front Controller** é um padrão que centraliza o **controle das requisições** em um único componente.

➡️ Em Java Web:  
✅ Implementado com um **Servlet** que atua como **ponto único de entrada**.  
✅ Baseado na arquitetura **MVC**:

* **Model** → entidades e EJBs.
* **View** → JSPs.
* **Controller** → Servlets, especialmente o **Front Controller**.

➡️ O Front Controller decide para qual componente enviar a requisição.

**✅ 7. Quais as diferenças e semelhanças entre Servlets e JSPs?**

| **✅** | **Servlet** | **JSP** |
| --- | --- | --- |
| ✔️ | Código Java que gera HTML. | Código HTML que pode conter Java embutido. |
| ✔️ | Mais indicado para **lógica de controle**. | Mais indicado para **apresentação** (View). |
| ✔️ | Mais difícil para criar páginas complexas. | Mais fácil e intuitivo para criar interfaces. |
| ✔️ | Manipula HttpServletRequest diretamente. | Usa **EL (Expression Language)** e **JSTL**. |

➡️ **Semelhança**: ambos processam requisições HTTP e geram respostas dinâmicas.

**✅ 8. 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 (response.sendRedirect)**:  
✅ Manda uma nova requisição para o navegador → muda a URL → ocorre **nova transação HTTP**.

➡️ **Forward (RequestDispatcher.forward)**:  
✅ Encaminha a mesma requisição para outro recurso → **não muda a URL** → mantém os **parâmetros e atributos**.

➡️ **Parâmetros**:  
✅ Vêm da URL ou do formulário (request.getParameter("nome")).

➡️ **Atributos**:  
✅ São objetos anexados pelo servidor para **compartilhar dados** entre componentes:  
request.setAttribute("produto", produto);

**✅ 9. Como o framework Bootstrap é utilizado?**

➡️ **Bootstrap** é um **framework CSS e JS** para criar **interfaces modernas e responsivas**.

➡️ Utilização:  
✅ Importar os arquivos CSS/JS do Bootstrap no HTML ou JSP:

html

CopiarEditar

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

✅ Usar as **classes Bootstrap**:

html

CopiarEditar

<div class="container">

<h1 class="text-center">Título</h1>

</div>

**✅ 10. Por que o Bootstrap garante a independência estrutural do HTML?**

➡️ Porque a **estrutura** da página é criada com **HTML puro**, enquanto o **estilo** e o **layout responsivo** são controlados apenas pelas **classes CSS do Bootstrap**.

✅ Assim, pode-se alterar totalmente o visual da página **sem mudar o HTML** → separação clara de **conteúdo e apresentação**.

**✅ 11. Qual a relação entre o Bootstrap e a responsividade da página?**

➡️ O Bootstrap fornece uma **grid system** baseada em **breakpoints**, que ajusta automaticamente o layout para:  
✅ Desktop  
✅ Tablet  
✅ Smartphone

➡️ Assim, o desenvolvedor cria páginas que **se adaptam** a diferentes tamanhos de tela **sem esforço extra**.

✅ Exemplo:

html

CopiarEditar

<div class="row">

<div class="col-md-6">Coluna 1</div>

<div class="col-md-6">Coluna 2</div>

</div>

➡️ Essa estrutura será exibida em 2 colunas no **desktop** e empilhada no **mobile**.

**✅ Resumo final:**

✅ NetBeans → facilita desenvolvimento corporativo com **módulos separados** e **assistentes**.  
✅ JPA → gerencia **persistência**.  
✅ EJB → gerencia **lógica de negócios** e **transações**.  
✅ Servlets → atuam como **controladores** na arquitetura MVC.  
✅ Bootstrap → facilita a criação de interfaces **responsivas** e separadas do conteúdo.

Ponto de Entrada: O método *main* é o ponto de entrada da aplicação Java. A Máquina Virtual Java (JVM) o executa antes que qualquer objeto da classe principal seja instanciado.

Natureza do *static*: Método *static* pertencem à classe em si, e não a uma instância (objeto) específica. Eles podem ser chamados usando o nome da classe (NomeDaClasse.metodoEstatico()) sem a necessidade de criar um objeto.

Chamada pelo *main*: Como o *main* é executado em um contexto onde ainda não existem objetos da sua classe, ele próprio deve ser static. Consequentemente, se o *main* (ou qualquer outro método estático) precisa invocar outro método da mesma classe diretamente (sem criar um objeto dela), esse outro método também precisa ser static para ser acessível nesse contexto "pré-objeto".

Em resumo: *main* é static porque é o início, executado antes da criação de objetos. Métodos chamados diretamente por ele, sem a intermediação de um objeto, também devem ser *static* para serem acessíveis nesse estágio inicial.