UNIVERSIDADE ESTÁCIO DE SÁ

PROGRAMAÇÃO FULL STACK

PÓLO NORTE SHOPPING

CLEYDSON ASSIS COELHO

MISSÃO PRÁTICA 3 MUNDO 3
PROCEDIMENTO 4

RPG0017 - Vamos integrar sistemas

RIO DE JANEIRO

2025

Objetivos da prática

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

servletProduto.java

```
package cadastroee.servlets;


import cadastroee.controller.ProdutoFacadeLocal;
import cadastroee.model.Produto;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import jakarta.ejb.EJB;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class ServletProduto extends HttpServlet {
   private static final long serialVersionUID = 1L;

   @EJB
   private ProdutoFacadeLocal facade;

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      // Define o tipo de resposta como HTML
      response.setContentType("text/html;charset=UTF-8");

      // Recupera todos produtos via EJB
      List<Produto> produtos = facade.findAll();

      try (PrintWriter out = response.getWriter()) {
         out.println("<!DOCTYPE html>");
         out.println("<html><head><title>Lista de Produtos</title></head><body>");
         out.println("<h1>Produtos Cadastrados</h1>");
         out.println("<ul>");
         for (Produto p : produtos) {
            out.printf("<li>%s — Preço: R$ %.2f</li>%n",
```

```
                    p.getNome(),
                    p.getPrecoVenda() != null ? p.getPrecoVenda() : 0f);
            }
            out.println("</ul>");
            out.println("</body></html>");
        }
    }

    // Opcional: encaminhar POST para GET
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

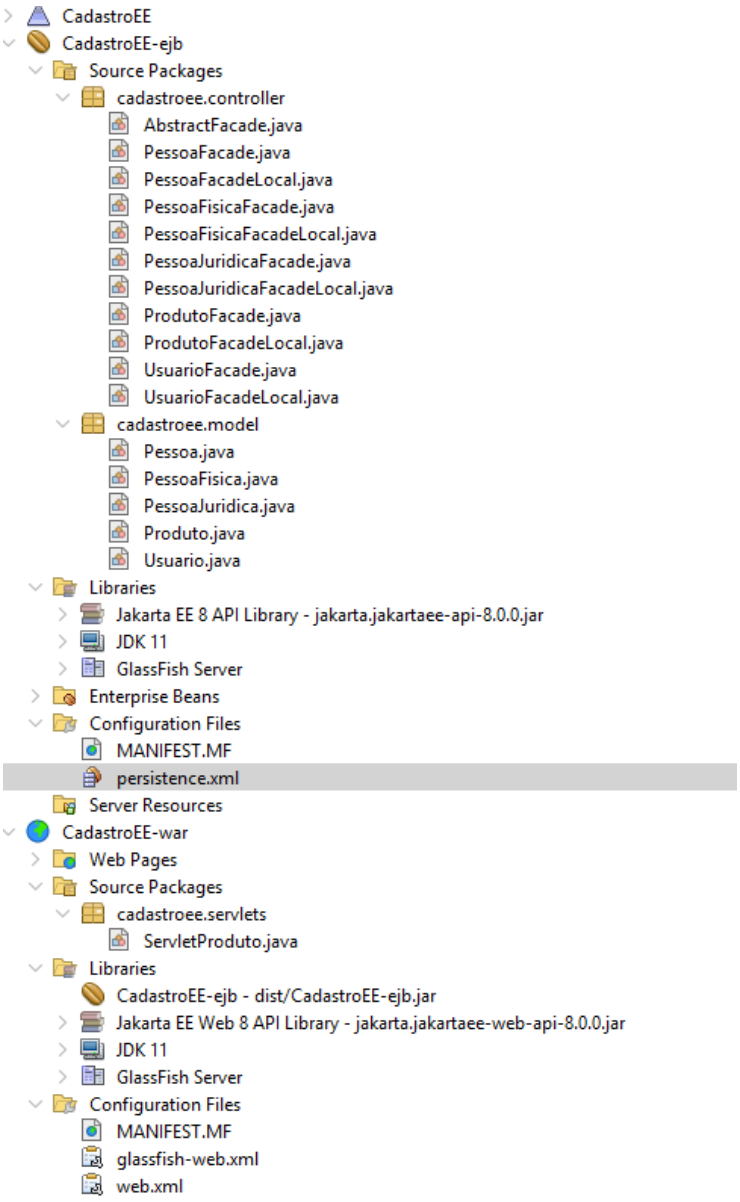
web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">
<servlet>
<servlet-name>ServletProduto</servlet-name>
<servlet-class>cadastroee.servlets.ServletProduto</servlet-class>
</servlet>
<servlet>
<servlet-name>ServletProdutoFC</servlet-name>
<servlet-class>cadastroee.servlets.ServletProdutoFC
</servlet-class>
</servlet>
<session-config>
<session-timeout>30</session-timeout>
</session-config>
</web-app>
```

persistence.xml

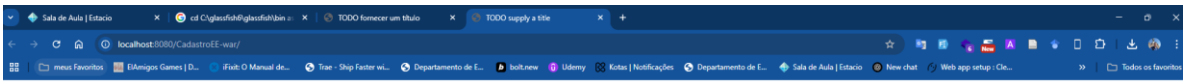
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0"
xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
<persistence-unit name="CadastroEE-ejbPU" transaction-type="JTA">
<jta-data-source>jdbc/loja</jta-data-source>
<exclude-unlisted-classes>false</exclude-unlisted-classes>
```

<properties/>
</persistence-unit>
</persistence>



Resultados:

Ao dar run em CadastroEE a pagina index.html e funciona perfeitamente, sem erros.



Funcionando perfeitamente

Análise e Conclusão – Procedimento 1

Como é organizado um projeto corporativo no NetBeans?
Um projeto corporativo no NetBeans, seguindo o modelo de Enterprise Application, é estruturado em três módulos principais:

Módulo EAR, que empacota e orquestra toda a aplicação corporativa;

Módulo EJB (CadastroEE-ejb), que contém as entidades JPA, os Session Beans e toda a lógica de negócio;

Módulo Web (CadastroEE-war), que reúne os Servlets, JSPs e recursos estáticos para a camada de apresentação.
Essa divisão em camadas promove separação de responsabilidades e facilita o deploy conjunto no servidor de aplicação .

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 responsável pela **persistência de dados**: ela mapeia as tabelas do banco a classes Java (@Entity), gerencia o ciclo de vida dessas entidades e abstrai detalhes de SQL por meio de consultas JPQL.

EJB (Enterprise JavaBeans) oferece a **camada de negócio**, permitindo implementar regras transacionais e serviços compartilhados (por exemplo, façades para entidades) por meio de beans stateless ou stateful. Os EJBs são gerenciados pelo contêiner, o que garante injeção de dependência, controle de transações e segurança declarativa .

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

O NetBeans automatiza tarefas repetitivas, como:

Geração de classes @Entity a partir de um DataSource configurado ("Entity Classes from Database");

Criação de Session Beans para cada entidade, com interfaces locais e métodos CRUD ("Session Beans for Entity Classes");

Atualização automática de dependências Jakarta EE e configuração de persistence.xml. Essas ferramentas de code-generation e os templates integrados reduzem erros manuais e agilizam o desenvolvimento .

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

Servlets são classes Java que processam requisições HTTP e geram respostas dinâmicas, funcionando como controladores na arquitetura Web.

No NetBeans, a opção **New → Servlet** gera automaticamente o esqueleto da classe, registra-a no web.xml (quando necessário) e já inclui métodos doGet/doPost, além de sugerir mapeamentos de URL. Dessa forma, o desenvolvedor foca apenas na lógica de tratamento de requisição e resposta .

Como é feita a comunicação entre os Servlets e os Session Beans do pool de EJBs?
A comunicação ocorre via **injeção de dependência**: basta declarar no Servlet um atributo anotado com @EJB, apontando para a interface local do bean (ProdutoFacadeLocal facade;). O contêiner do GlassFish injeta a instância apropriada, permitindo ao Servlet chamar métodos como facade.findAll() diretamente, sem necessidade de lookup manual, garantindo controle de transações e pooling oferecidos pelo EJB