



# **Estácio**

**FACULDADE ESTÁCIO**

**CÂMPUS MANAUS – AM**

**DESENVOLVIMENTO FULL STACK**

**DISCIPLINA – VAMOS INTEGRAR O SISTEMA**

**TURMA – 2023.2**

**SEMESTRE – 3**

**ALUNO – ALEX BARROSO PAZ**

**TUTOR – JHONATAN ALVES**

**GITHUB - <https://github.com/finntroll89/Vamos-Integrar-Sistema->**

## Implementação de sistema cadastral com interface web, baseado nas tecnologias de servlets, JPA e JEE

### 1. Título da Prática: “1º Procedimento | Camadas de Persistência e Controle”

### 2. Objetivo 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.

### 3. Códigos solicitados: estão no repositório <https://github.com/finntroll89/Vasmos-Integrar-Sistema->

### 4. Resultados da execução dos códigos

A configuração do ambiente e de todo o procedimento foram bem sucedidos, mas honestamente não é trivial, com alguns detalhes que requerem bastante atenção. Por exemplo, as strings de parâmetros de GlassFish contidas no enunciado da missão não estão totalmente corretas; foi necessário que o tutor da disciplina corrigisse no fórum.

Mesmo com o passo-a-passo detalhado, iniciantes que nunca entraram em contato com essas tecnologias (NetBeans, GlassFish, MS SQL Server Studio, JDBC) certamente terão dificuldades.

Contudo, a seguir seguem as figuras para fins de comprovação do sucesso da missão. Em especial, da correta configuração do GlassFish (fig. 1), da aplicação configurada via NetBeans e exibida na interface web do GlassFish (fig. 2), a configuração do arquivo persistence.xml (fig. 3) e a exibição da execução do ServletProduto (fig. 4), conforme solicitado.

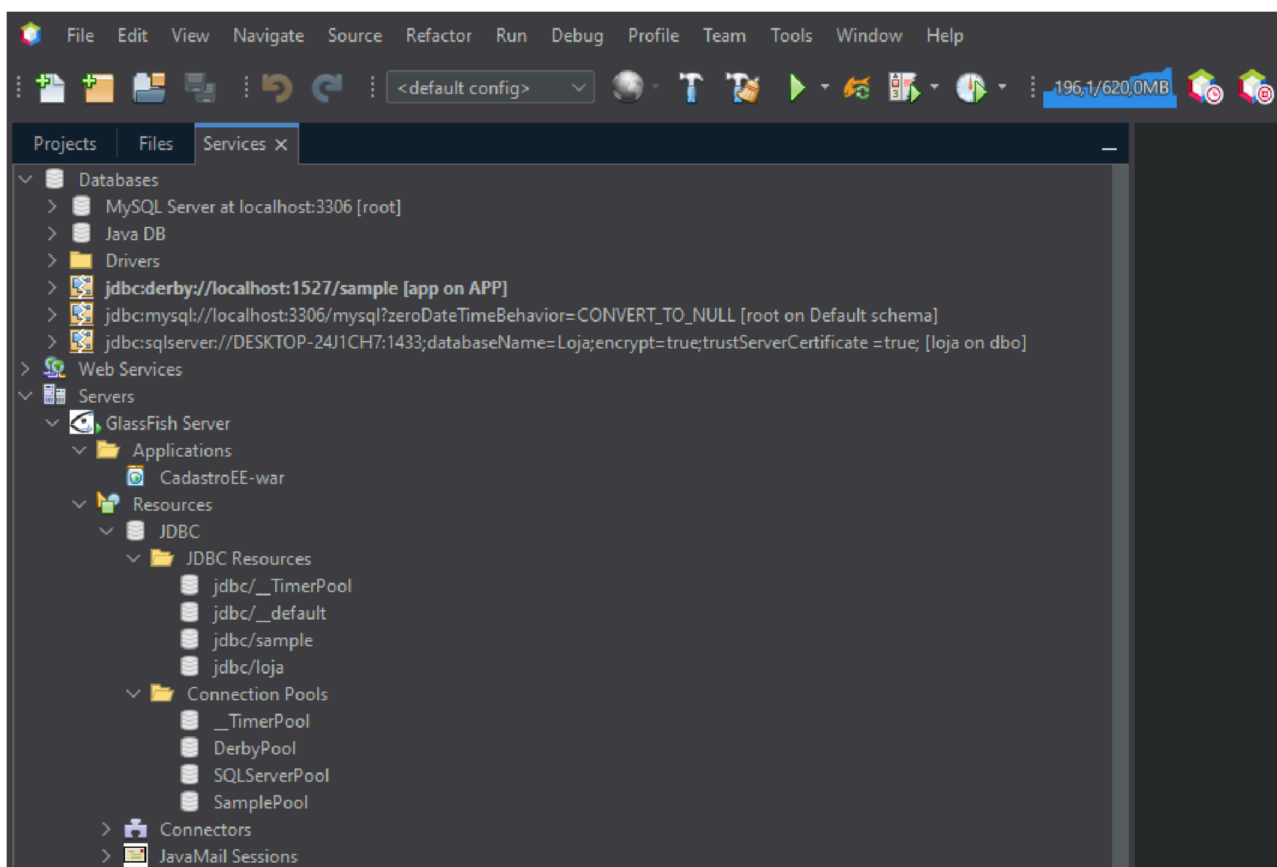


Figura 1. Serviços configurados no NetBeans: JDBC, GlassFish, jdbc/Loja, SQLServerPool.

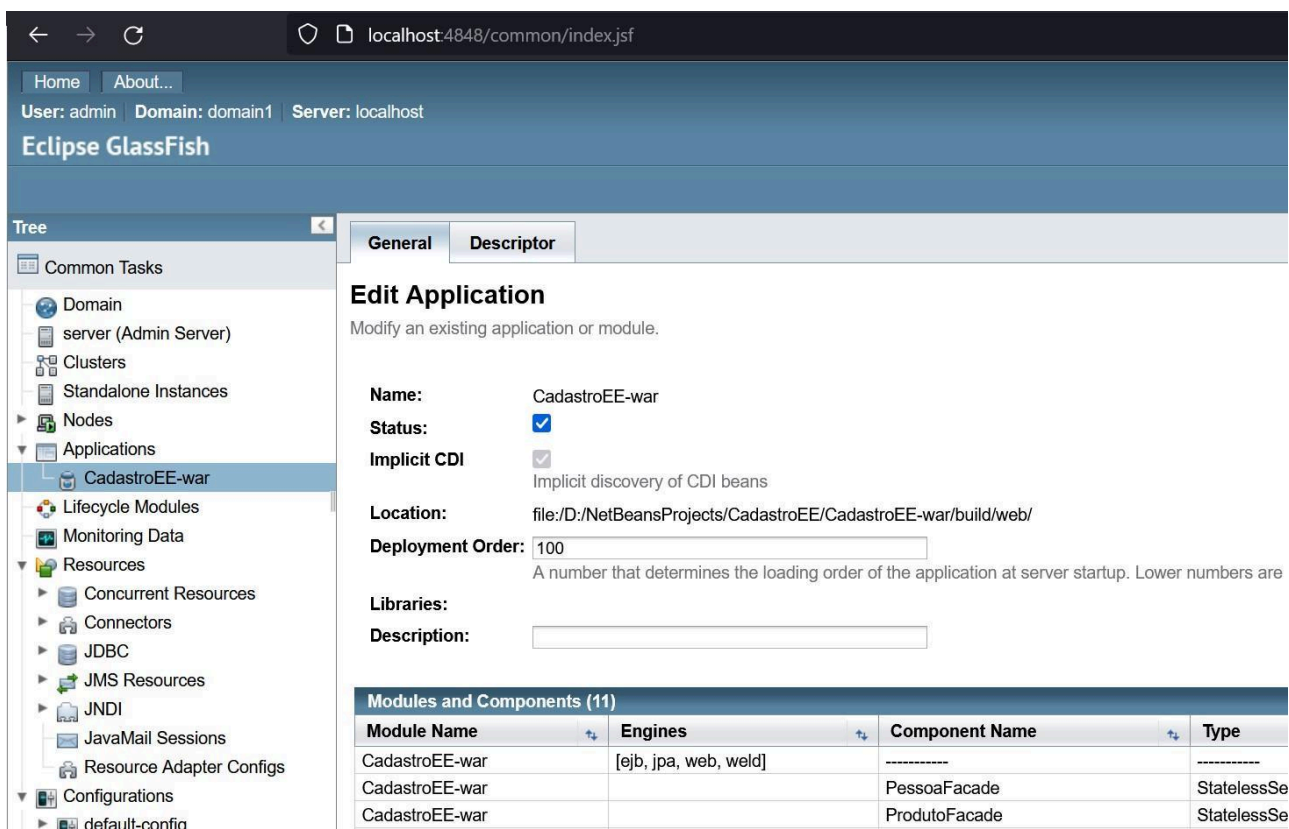
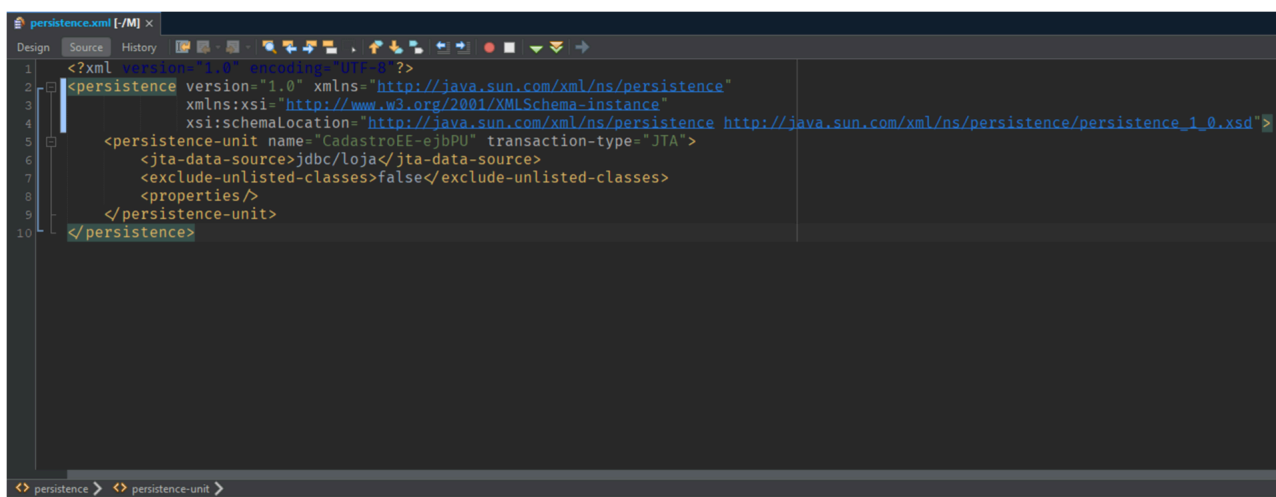
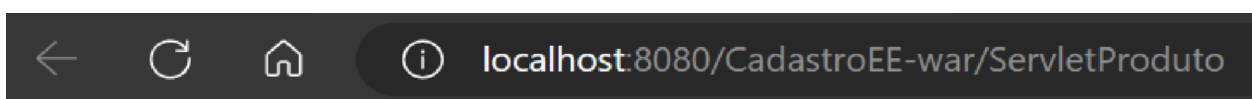


Figura 2: aplicação CadastroEE-war em execução no GlassFish.



**Figura 3:** Arquivo de configuração persistence.xml



## Lista de Produtos

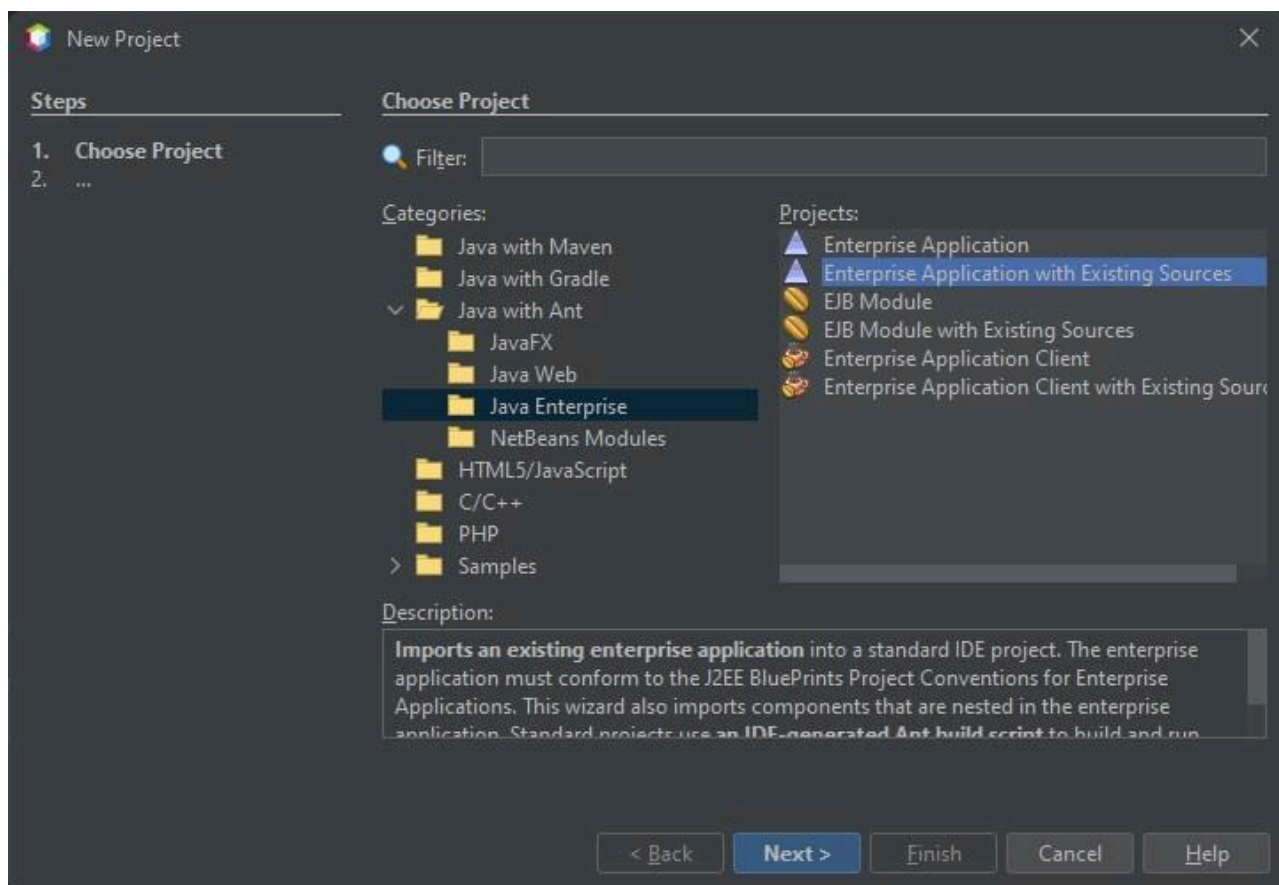
Nome do Produto	Preço
Banana	R\$ 5,00
Laranja	R\$ 2,00
Manga	R\$ 4,00
Tangerina	R\$ 7,00

**Figura 4.** ServletProduto em execução.

### 5. Análise e Conclusão

(a) Como é organizado um projeto corporativo no NetBeans?

Um projeto corporativo (ou Enterprise) no NetBeans segue uma estrutura modular hierárquica, claramente visível ao se criar um novo projeto: escolher uma categoria de gerenciador de pacotes, dentre os mais utilizados na atualidade: Maven, Gradle, Ant (fig. 5); em seguida, escolher o tipo de projeto corporativo mais adequado, conforme mostra a figura abaixo:



**Figura 5.** Criação de um novo projeto corporativo no NetBeans.

Projetos corporativos web no NetBeans, mesmo que sejam de diferentes categorias e tipos, seguem um padrão comum de pastas: código-fonte (source), bibliotecas ou dependências (libraries), configurações (configuration ou WEB-INF), páginas web ou servlets (web pages, war).

Nesta Missão Prática, foi escolhida a categoria “Java with Ant”, seguida da sub-categoria “Java Enterprise”, com tipo de projeto “Enterprise Application with Existing Sources”, o qual é gerado a partir de um banco de dados existente, como resultado: além da pasta principal Enterprise Application, são criados 2 módulos com diferentes funções: um módulo EJB (Cadastro-ejb) e um módulo de aplicação web (Cadastro-war), gerenciados separadamente e integrados para apresentar a total funcionalidade do sistema. O NetBeans facilita a organização desses módulos, com o uso de ferramentas e assistentes de configuração, construção e implantação de maneira eficiente.

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

As tecnologias JPA (*Jakarta Persistence API* [1], anteriormente *Java Persistence API*) e EJB (*Enterprise JavaBeans*) desempenham papéis importantes na construção de aplicativos para a plataforma web no ambiente Java.

O JPA é uma API (*Application Program Interface*) de persistência de dados padrão do Java, o qual permite aos desenvolvedores mapear objetos Java para tabelas em um banco de dados relacional. Com JPA, é possível escrever consultas em linguagem Java, as quais são traduzidas para consultas SQL pelo provedor de persistência. Isso simplifica o processo de interação com o banco de dados e torna o código mais portátil entre diferentes provedores de

banco de dados. O JPA é frequentemente usado para lidar com operações de banco de dados em aplicativos web Java.

O EJB é um componente de servidor do Java EE (*Enterprise Edition*) que simplifica o desenvolvimento de aplicativos corporativos. O EJB oferece uma maneira de encapsular a lógica de negócios em componentes reutilizáveis que podem ser implantados em um servidor de aplicativos Java EE, tais como o Eclipse GlassFish ou o Apache Tomcat.

Portanto, em uma aplicação web Java, pode-se utilizar o JPA para mapear e persistir objetos Java em um banco de dados relacional, e o EJB para implementar a lógica de negócios do aplicativo, tais como autenticação de usuário, processamento de requisições e assim por diante. Na prática, essas tecnologias combinadas podem ajudar a criar aplicativos web robustos e escaláveis no ambiente Java.

Um exemplo de uso: há diferentes tipos de EJBs, como EJBs de sessão, utilizados para implementar a lógica de negócios da aplicação, e EJBs de entidade, utilizados para representar entidades persistentes, geralmente mapeadas com JPA.

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

NetBeans possui **suporte integrado para JPA e EJB**, como a inclusão de assistentes para criar entidades JPA a partir de tabelas de banco de dados existentes, a geração automática de código para EJBs, a integração com os servidores de aplicativos Java EE para implantar e testar aplicativos que utilizam essas tecnologias.

NetBeans também possui **ferramentas de mapeamento de entidades JPA**, as quais permitem criar e facilmente editar os mapeamentos entre classes Java e as tabelas de banco de dados com uso do JPA. Isso simplifica o processo de definição de como objetos Java são persistidos no banco de dados, o que resulta em economia de tempo e redução de erros.

NetBeans conta com **geração automática de código para entidades JPA, sessões EJB** e outros componentes, com base em modelos predefinidos e nas configurações do projeto. Isso acelera o desenvolvimento, a fim de que desenvolvedores se concentrem mais na lógica de negócios, ao invés de concentrar esforços na escrita de código repetitivo de infraestrutura.

NetBeans é bastante hábil ao realizar **integração com ferramentas de bancos de dados**, as quais permitem aos desenvolvedores visualizar, modificar esquemas de banco de dados, executar consultas SQL e realizar outras tarefas relacionadas ao banco de dados diretamente na IDE. Isso simplifica o processo de desenvolvimento de aplicativos que utilizam JPA para acesso ao banco de dados.

Finalmente, o NetBeans possui **ferramentas avançadas de debug e testes**, essenciais para desenvolvedores identificarem e corrigirem problemas de desempenho em aplicativos que utilizam as tecnologias JPA e EJB. Isso pode ser especialmente útil ao otimizar consultas de banco de dados e tornar mais eficiente o uso de recursos do servidor.

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

Servlets são componentes em forma de classes Java, usadas para estender a capacidade de servidores web hospedar aplicativos web dinâmicos; fornecem uma maneira de responder a solicitações HTTP, dinamicamente, com base nos dados enviados pelo cliente, seja através de formulários (*form*) web ou parâmetros em URL. Servlets fazem parte da especificação Java EE e são comumente usados para lidar com a lógica de controle em aplicativos web Java.

NetBeans oferece um suporte robusto para a construção e o desenvolvimento de servlets em projetos web, como **assistentes e modelos**, os quais permitem montar automaticamente o esqueleto do servlet com os métodos doGet() e doPost(), ou então manualmente criar um servlet a partir da herança da classe javax.servlet.http.HttpServlet.

NetBeans também **gerencia automaticamente o arquivo de configurações web.xml**, usado para mapear servlets para URLs específicas e configurar outras propriedades do servlet.

NetBeans oferece suporte completo para **debug** de servlets, o qual permite adicionar *breakpoints* no código, executar a aplicação em modo de depuração para identificar e corrigir problemas facilmente.

NetBeans **integra-se facilmente com servidores de aplicações Java EE**, tais como Apache Tomcat e Eclipse GlassFish, o que permite implantar e testar servlets a partir da IDE.

NetBeans também fornece assistência ao desenvolvimento com **sugestões de código, realce de sintaxe, refatoração de código** e outras ferramentas que ajudam a aumentar a produtividade.

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

A comunicação entre Servlets e Session Beans em um pool de EJBs é realizada através de um mecanismo chamado **JNDI (Java Naming and Directory Interface)**. Basicamente, os Servlets utilizam JNDI para localizar e referenciar Session Beans disponíveis no pool; após a localização, os Servlets podem invocar métodos nos Session Beans como se realizassem chamadas a métodos em objetos locais. Essa abordagem permite que Servlets, que geralmente gerenciam solicitações de usuários via HTTP, interajam com lógicas de negócios complexas encapsuladas nos Session Beans, assim promovem uma eficiente separação entre camada de apresentação e lógica de negócios.

---

1. Título da Prática: “2º Procedimento | Interface Cadastral com Servlet e JSP”

2. Objetivo da Prática

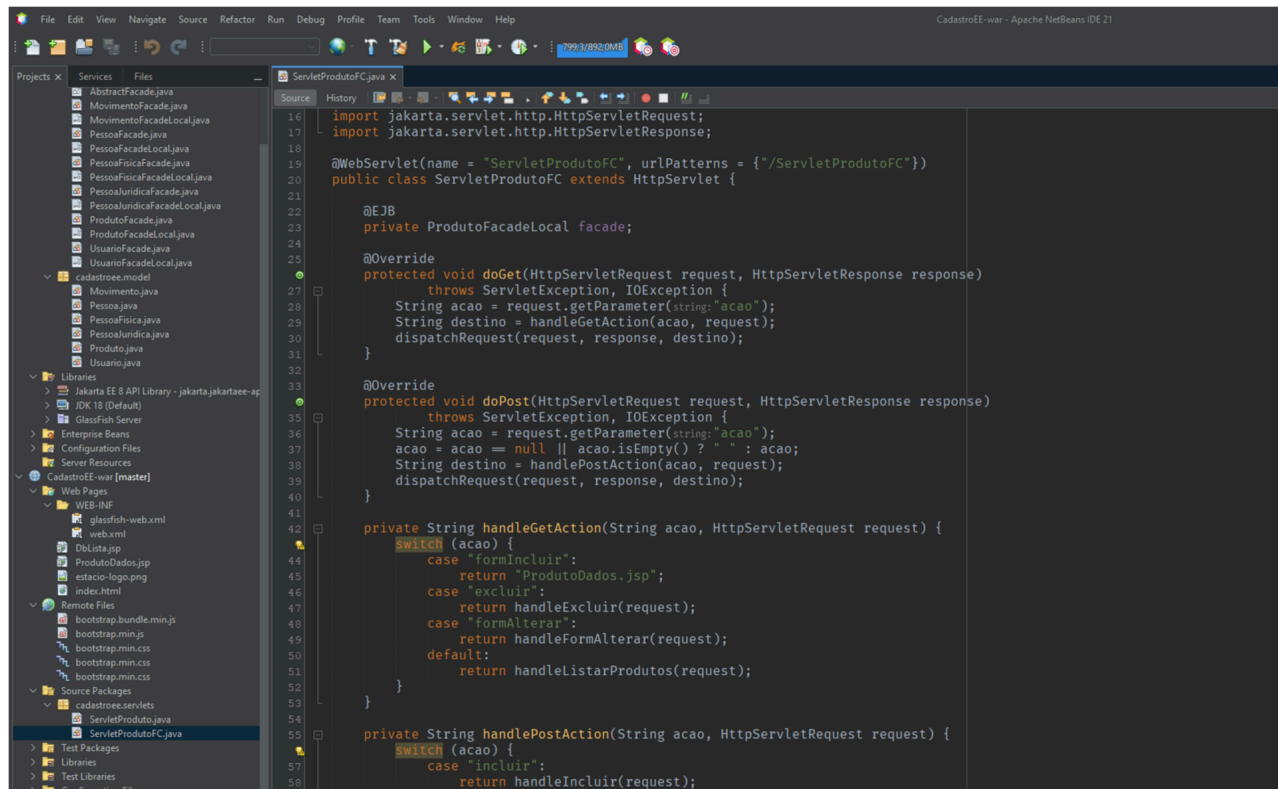
- Implementar persistência com base em JPA.
- Implementar regras de negócio na plataforma JEE, através de EJBs.
- Implementar sistema cadastral Web com base em Servlets e JSPs.
- Utilizar a biblioteca Bootstrap para melhoria do design.
- No final do exercício, o aluno terá criado todos os elementos necessários para exibição e entrada de dados na plataforma Java Web, tornando-se capacitado para lidar com contextos reais de aplicação.

3. Códigos solicitados: estão no repositório  
<https://github.com/finntroll89/Vasmos-Integrar-Sistema->

4. Resultados da execução dos códigos

O 2º procedimento é continuação do 1º procedimento, com a diferença que é criado um novo servlet denominado ServletProdutoFC (fig. 6), que utiliza o padrão FC (*Front Controller*), com a capacidade de exibir a lista de produtos, cadastrar, alterar e excluir produtos, armazenados em banco de dados, através de parâmetros de “ação” na URL (“incluir”, “alterar”, “excluir”).

As interfaces gráficas JSP (*Java Server Page*) (fig. 7), conforme solicitado neste 2º procedimento, utilizam basicamente HTML, sem uso de bibliotecas CSS, como mostrado na fig. 8. Essas interfaces JSP são templates escritos em HTML salvos em arquivos de extensão .jsp, com uso adequado e posicionado de tags `<% %>`, onde os dados da aplicação são corretamente inseridos e exibidos.



```
16 import jakarta.servlet.http.HttpServletRequest;
17 import jakarta.servlet.http.HttpServletResponse;
18
19 @WebServlet(name = "ServletProdutoFC", urlPatterns = {"/ServletProdutoFC"})
20 public class ServletProdutoFC extends HttpServlet {
21
22     @EJB
23     private ProdutoFacadeLocal facade;
24
25     @Override
26     protected void doGet(HttpServletRequest request, HttpServletResponse response)
27         throws ServletException, IOException {
28         String acao = request.getParameter(string:"acao");
29         String destino = handleGetAction(acao, request);
30         dispatchRequest(request, response, destino);
31     }
32
33     @Override
34     protected void doPost(HttpServletRequest request, HttpServletResponse response)
35         throws ServletException, IOException {
36         String acao = request.getParameter(string:"acao");
37         acao = acao == null || acao.isEmpty() ? " " : acao;
38         String destino = handlePostAction(acao, request);
39         dispatchRequest(request, response, destino);
40     }
41
42     private String handleGetAction(String acao, HttpServletRequest request) {
43         switch (acao) {
44             case "formIncluir":
45                 return "ProdutoDados.jsp";
46             case "excluir":
47                 return handleExcluir(request);
48             case "formAlterar":
49                 return handleFormAlterar(request);
50             default:
51                 return handleListarProdutos(request);
52         }
53     }
54
55     private String handlePostAction(String acao, HttpServletRequest request) {
56         switch (acao) {
57             case "incluir":
58                 return handleIncluir(request);
```

Figura 6. Trecho do código do ServletProdutoFC.



```

<table class="table table-striped table-bordered table-responsive">
  <thead>
    <tr class="table-dark">
      <th>ID</th>
      <th>Produto</th>
      <th>Quantidade</th>
      <th>Preço</th>
      <th>Ações</th>
    </tr>
  </thead>
  <tbody>
    <%
      DecimalFormat df = new DecimalFormat("#,##0.00");
      List<Produto> produtos = (List<Produto>) request.getAttribute("produtos");

      if (produtos != null && !produtos.isEmpty()) {
        for (Produto produto : produtos) {
    <tr>
      <td class="text-center"><%=produto.getIdProduto()%></td>
      <td class="text-center"><%=produto.getNome()%></td>
      <td class="text-center"><%=produto.getQuantidade()%></td>
      <td class="text-center">R$ <%=df.format(produto.getPrecoVenda())%></td>
      <td class="text-end">
        <a class="btn btn-primary btn-sm" href="ServletProdutoFC?acao=formAlterar&id=<%=produto.getIdProduto()%>">Alterar</a>
        <a class="btn btn-danger btn-sm" href="ServletProdutoFC?acao=excluir&id=<%=produto.getIdProduto()%>">Excluir</a>
      </td>
    </tr>
        }
      } else {
    <tr>
      <td colspan="5">Nenhum produto encontrado.</td>
    </tr>
      }
    <%>
  </tbody>
</table>

<div class="text-end mb-3">
  <a class="btn btn-primary" href="ServletProdutoFC?acao=formIncluir">Cadastrar Produto</a>
</div>

```

Figura 7. Trecho do código JSP responsável pela exibição da lista de produtos.

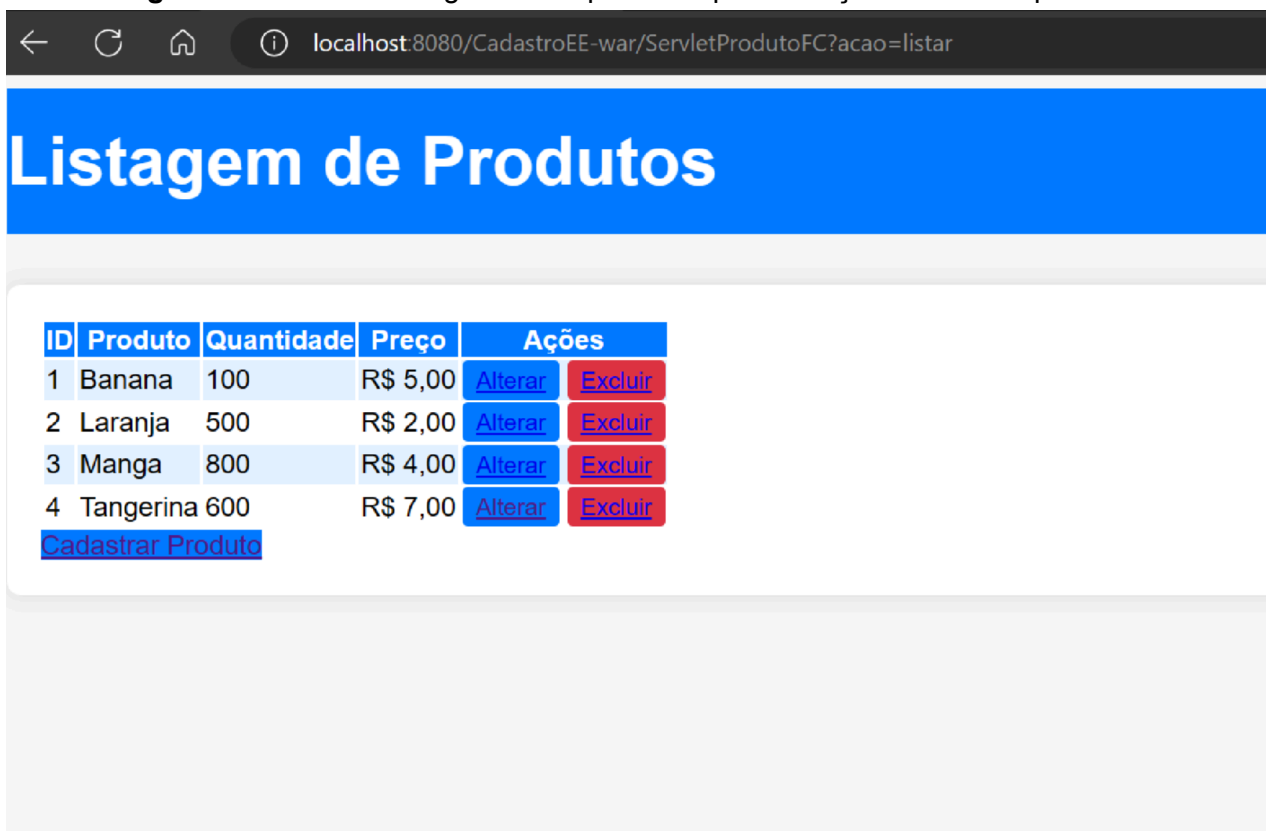


Figura 8. ServletProdutoFC em execução, sem uso de bibliotecas CSS, apenas CSS básico

## 5. Análise e Conclusão

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

O padrão Front Controller (FC) em aplicações web Java, especialmente na arquitetura MVC (Model-View-Controller), funciona como um controlador centralizado que gerencia todas as solicitações do cliente.

Ao invés de utilizar vários controladores para diferentes tipos de solicitações, o FC atua como um ponto único de entrada, que interpreta as solicitações, delega as tarefas apropriadas para modelos específicos (model); em seguida, seleciona a visualização correta (view) para responder ao cliente.

É comum esse fluxo de interações ser implementado através de um servlet, o qual intercepta todas as solicitações, realiza o processamento necessário ou lógica de negócios e, finalmente, encaminha a resposta para a página JSP ou outra tecnologia de visualização de renderização de páginas. Este padrão auxilia na manutenção e gerenciamento centralizado das solicitações, de modo a promover a reutilização de código em uma estrutura mais organizada.

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

Servlets e JSP são tecnologias usadas para criar aplicações web em Java, mas possuem abordagens bem distintas.

Servlets são classes Java que permitem gerar código HTML a partir de instruções e comandos em Java; assim, os servlets são mais adequados para a lógica de negócios e processamento de dados.

Por outro lado, JSP são páginas HTML com capacidade de incorporar código Java; logo são mais adequados para a apresentação da interface com o usuário.

Ambos, servlets e JSP, são executados no servidor de aplicações (GlassFish, Tomcat), com a possibilidade de interagir com bancos de dados, assim como outras tecnologias Java Enterprise.

Em resumo, enquanto servlets são mais voltados para controle, JSPs são mais adequados para a visualização e, frequentemente, ambos são usados em conjunto para separar a lógica de negócios da interface de usuário.

(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 HttpServletRequest?

O redirecionamento simples utiliza o método `response.sendRedirect(url)`, enquanto que o RequestDispatcher utiliza o método `forward`. A principal diferença entre eles está no tratamento da requisição. O redirecionamento simples envia uma resposta ao navegador, para indicar que este deve fazer uma nova requisição para outra URL. Já o `forward` encaminha a requisição atual para outro recurso no servidor sem informar ao cliente, e mantém a URL original.

Os parâmetros e atributos em objetos HttpServletRequest são fundamentais para passar informações entre cliente e servidor, ou entre diferentes partes do servidor. Parâmetros são tipicamente usados para enviar dados de formulários ou de solicitações de URL, enquanto atributos são usados para manter dados durante a vida útil de uma requisição ou sessão, de modo a permitir a comunicação entre diferentes componentes do servidor.

## 2. Objetivo 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.

3. Códigos solicitados: estão no repositório  
<https://github.com/finntroll89/N-vel-4-Vamos-Integrar-Sistemas.git>

## 4. Resultados da execução dos códigos

O 3º procedimento utiliza a biblioteca do framework Bootstrap, o que torna a interface gráfica muito mais bonita e agradável.

A fig. 9 mostra a mesma listagem de produtos do procedimento anterior, mas com uso de Bootstrap, conforme solicitado no enunciado da Missão.



ID	Produto	Quantidade	Preço	Ações
1	Banana	100	R\$ 5,00	<button>Alterar</button> <button>Excluir</button>
2	Laranja	500	R\$ 2,00	<button>Alterar</button> <button>Excluir</button>
3	Manga	800	R\$ 4,00	<button>Alterar</button> <button>Excluir</button>
4	Tangerina	600	R\$ 7,00	<button>Alterar</button> <button>Excluir</button>

Cadastrar Produto

**Figura 9.** ServletProdutoFC em execução, com uso de Bootstrap.

← ↻ 🏠 ⓘ localhost:8080/CadastroEE-war/ServletProdutoFC?acao=formIncluir 🔍 ☆ ⚙️ | 📄 ☆ 🔒 🔄 ...

## Cadastro de Produto

Voltar

**Nome**

**Quantidade**

**Preço de Venda**

Cadastrar

**Figura 10.** Tela de Cadastro de Produto.

Na fig. 10 é mostrada a tela de cadastro de produto, com uso de Bootstrap. Assim como na fig. 11 é mostrada a tela de alteração dos dados de produto cadastrado.

← ↻ 🏠 ⓘ localhost:8080/CadastroEE-war/ServletProdutoFC?acao=formAlterar&id... 🔍 ☆ ⚙️ | 📄 ☆ 🔒 🔄 ...

## Alteração de Produto

Voltar

**Nome**

**Quantidade**

**Preço de Venda**

Alterar

**Figura 11.** Tela de Alteração de Produto já cadastrado.

(a) Como o framework Bootstrap é utilizado?

O framework Bootstrap é amplamente utilizado para desenvolver interfaces de usuário responsivas, *mobile first*, para websites e aplicações web. Oferece um conjunto robusto de ferramentas baseadas em HTML, CSS e JavaScript, que incluem templates pré desenvolvidos para botões, formulários, navegação e outros elementos de interface, além de um sistema de grid flexível para layout. Isso permite aos desenvolvedores rapidamente

construir sites esteticamente agradáveis, funcionais, e que se adaptem automaticamente a diferentes tamanhos de tela e dispositivos, sem a necessidade de escrever código de estilização CSS desde o início.

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

O Bootstrap assegura a independência estrutural do HTML, ao fornecer um conjunto de classes CSS pré-definidas, assim como componentes de interface que podem ser facilmente integrados ao HTML. Em razão disso, ao invés de escrever e ajustar diferentes estilos CSS personalizados para cada elemento, os desenvolvedores podem simplesmente usar as classes do Bootstrap para alcançar um design consistente e responsivo.

Portanto, a estrutura do HTML permanece limpa e desacoplada de estilos específicos, o que facilita a manutenção e a escalabilidade do código.

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

O Bootstrap é um framework de desenvolvimento web que facilita a criação de páginas responsivas. Oferece um sistema de grid flexível, componentes pré-desenvolvidos e classes CSS que se ajustam automaticamente ao tamanho da tela do dispositivo, de maneira a garantir que o layout da página seja otimizado para desktops, tablets e smartphones. Isso simplifica o processo de design responsivo, o que permite que desenvolvedores criem sites que proporcionam uma experiência de usuário consistente em diversos dispositivos com menos esforço.

## Referências

Devmedia “**JDBC tutorial**”

Disponível em <https://www.devmedia.com.br/jdbc-tutorial/6638>

Acesso em 21 de agosto de 2024

Devmedia “**Introdução à JPA - Java Persistence API**”

Disponível em <https://www.devmedia.com.br/introducao-a-jpa-java-persistence-api/28173>

Acesso em 21 de agosto de 2024

Geeksforgeeks “**Enterprise Java Beans (EJB)**”

Disponível em <https://www.geeksforgeeks.org/enterprise-java-beans-ejb/>

Acesso em 21 de agosto de 2024

