UNIVERSIDADE ESTÁCIO DE SÁ

CAMPUS POLO CENTRO - MARICÁ - RJ

ALUNO: LUIZ CARLOS MARINHO JUNIOR

MATRÍCULA: 2023.11.17557-1

TURMA: 9001

SEMESTRE LETIVO: 2024.4

DISCIPLINA: NÍVEL 4: VAMOS INTEGRAR SISTEMAS

TÍTULO: MISSÃO PRÁTICA - NÍVEL 4

RIO DE JANEIRO

2024

LUIZ CARLOS MARINHO JUNIOR

MISSÃO PRÁTICA - NÍVEL 4

Trabalho prático para aprovação na disciplina de Nível 4: Vamos Integrar Sistemas.

RIO DE JANEIRO

2024

SUMÁRIO

1. 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.

2. INFORMAÇÕES COMPLEMENTARES

2.1. APRESENTAÇÃO

Esta missão prática está dividida em 3 procedimentos, o “1º Procedimento | Camadas de Persistência e Controle”, o “2º Procedimento | Interface Cadastral com Servlet e JSPs” e o “3º Procedimento | Melhorando o Design da Interface”. Todos os códigos que serão aqui apresentados, estão disponíveis de forma completa no repositório do Github.

O **link para o repositório do Github** se encontra logo abaixo:
https://github.com/luizmarinhojr/cadastro-ee

2.2. ORGANIZAÇÃO DO REPOSITÓRIO REMOTO

O repositório remoto do Github está dividida em quatro branchs:

- A Branch “main”, se refere ao desenvolvimento final, após os 3 procedimentos executados;
- A Branch “PrimeiroProcedimento”, se refere ao desenvolvimento até o 1º Procedimento, que envolve as Camadas de Persistência e Controle;
- A Branch “SegundoProcedimento”, se refere ao desenvolvimento do 1º ao 2º Procedimento, e inclui a Interface Cadastral com Servlet e JSPs;
- A Branch “TerceiroProcedimento”, se refere ao desenvolvimento do 1º ao 3º Procedimento, e adiciona a Melhoria do Design da Interface.

As seguintes branchs podem ser acessadas diretamente a partir dos seguintes links:

Branch main: https://github.com/luizmarinhojr/cadastro-ee

Branch PrimeiroProcedimento: https://github.com/luizmarinhojr/cadastro-ee/tree/PrimeiroProcedimento

Branch SegundoProcedimento: https://github.com/luizmarinhojr/cadastro-ee/tree/SegundoProcedimento

Branch TerceiroProcedimento: https://github.com/luizmarinhojr/cadastro-ee/tree/TerceiroProcedimento

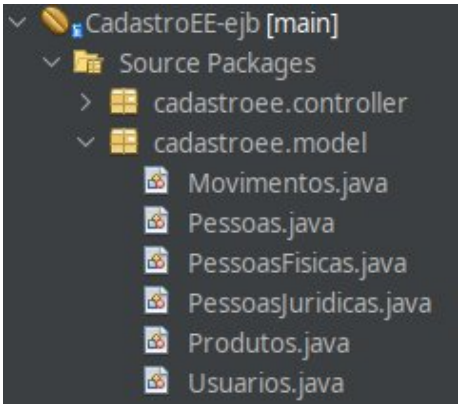
3. 1º PROCEDIMENTO | CAMADAS DE PERSISTÊNCIA E CONTROLE

3.1. CÓDIGOS SOLICITADOS

Os códigos serão demonstrados neste relatório de acordo com as instruções apresentadas no enunciado da missão prática. Somente serão demonstrados nesse relatório as principais instruções que são relevantes ao desenvolvimento do código.

3.2. PRINCIPAIS INSTRUÇÕES DO 1ª PROCEDIMENTO

1. Definir as camadas de persistência e controle no projeto CadastroEE-ejb.
   a. Criar as entidades JPA através de New Entity Classes from Database.
   b. Selecionar jdbc/loja como Data Source, e selecionar todas as tabelas.
   c. No passo seguinte, definir o pacote como cadastroee.model, além de marcar a opção para criação do arquivo persistence.xml.



Movimentos.java

```
@Entity
@Table(name = "Movimentos")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Movimentos.findAll", query = "SELECT m FROM Movimentos m"),
    @NamedQuery(name = "Movimentos.findByMovimentoID", query = "SELECT m FROM Movimentos m WHERE m.movimentoID = :movimentoID"),
    @NamedQuery(name = "Movimentos.findByQuantidade", query = "SELECT m FROM Movimentos m WHERE m.quantidade = :quantidade"),
    @NamedQuery(name = "Movimentos.findByTipo", query = "SELECT m FROM Movimentos m WHERE m.tipo = :tipo"),
    @NamedQuery(name = "Movimentos.findByValorUnitario", query = "SELECT m FROM Movimentos m WHERE m.valorUnitario = :valorUnitario")})
public class Movimentos implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "movimentoID")
    private Long movimentoID;
    @Basic(optional = false)
    @NotNull
    @Column(name = "quantidade")
    private int quantidade;
    @Basic(optional = false)
    @NotNull
    @Column(name = "tipo")
    private Character tipo;
```

```
   // @Max(value=?)  @Min(value=?)//if you know range of your decimal fields consider using these annotations to enforce field validation
    @Basic(optional = false)
    @NotNull
    @Column(name = "valorUnitario")
    private BigDecimal valorUnitario;
    @JoinColumn(name = "pessoaID", referencedColumnName = "pessoaID")
    @ManyToOne(optional = false)
    private Pessoas pessoaID;
    @JoinColumn(name = "produtoID", referencedColumnName = "produtoID")
    @ManyToOne(optional = false)
    private Produtos produtoID;
    @JoinColumn(name = "usuarioID", referencedColumnName = "usuarioID")
    @ManyToOne(optional = false)
    private Usuarios usuarioID;


    public Movimentos() {
    }
    public Movimentos(Long movimentoID) {
        this.movimentoID = movimentoID;
    }
    public Movimentos(Long movimentoID, int quantidade, Character tipo, BigDecimal valorUnitario) {
        this.movimentoID = movimentoID;
        this.quantidade = quantidade;
        this.tipo = tipo;
        this.valorUnitario = valorUnitario;
    }
...
```

Pessoas.java

```
@Entity
@Table(name = "Pessoas")
@XmlRootElement
@NamedQueries({
   @NamedQuery(name = "Pessoas.findAll", query = "SELECT p FROM Pessoas p"),
   @NamedQuery(name = "Pessoas.findByPessoaID", query = "SELECT p FROM Pessoas p WHERE p.pessoaID = :pessoaID"),
   @NamedQuery(name = "Pessoas.findByNome", query = "SELECT p FROM Pessoas p WHERE p.nome = :nome"),
   @NamedQuery(name = "Pessoas.findByLogradouro", query = "SELECT p FROM Pessoas p WHERE p.logradouro = :logradouro"),
   @NamedQuery(name = "Pessoas.findByCidade", query = "SELECT p FROM Pessoas p WHERE p.cidade = :cidade"),
   @NamedQuery(name = "Pessoas.findByEstado", query = "SELECT p FROM Pessoas p WHERE p.estado = :estado"),
   @NamedQuery(name = "Pessoas.findByTelefone", query = "SELECT p FROM Pessoas p WHERE p.telefone = :telefone"),
   @NamedQuery(name = "Pessoas.findByEmail", query = "SELECT p FROM Pessoas p WHERE p.email = :email")})
public class Pessoas implements Serializable {


   private static final long serialVersionUID = 1L;
   @Id
   @Basic(optional = false)
   @NotNull
   @Column(name = "pessoaID")
   private Long pessoaID;
   @Basic(optional = false)
   @NotNull
   @Size(min = 1, max = 255)
   @Column(name = "nome")
```

```
    private String nome;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
    @Column(name = "logradouro")
    private String logradouro;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
    @Column(name = "cidade")
    private String cidade;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 2)
    @Column(name = "estado")
    private String estado;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 11)
    @Column(name = "telefone")
    private String telefone;
    //                          @Pattern(regexp="[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-
9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]*[a-
z0-9])?", message="Invalid email")//if the field contains email address consider using
this annotation to enforce field validation
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
    @Column(name = "email")
    private String email;
```

```
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoas")
    private PessoasJuridicas pessoasJuridicas;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "pessoaID")
    private Collection<Movimentos> movimentosCollection;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoas")
    private PessoasFisicas pessoasFisicas;

    public Pessoas() {
    }

    public Pessoas(Long pessoaID) {
        this.pessoaID = pessoaID;
    }

   public Pessoas(Long pessoaID, String nome, String logradouro, String cidade, String
estado, String telefone, String email) {
        this.pessoaID = pessoaID;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }
...
```

PessoasFisicas.java

```
@Entity
@Table(name = "PessoasFisicas")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "PessoasFisicas.findAll", query = "SELECT p FROM PessoasFisicas p"),
    @NamedQuery(name = "PessoasFisicas.findByPessoaID", query = "SELECT p FROM PessoasFisicas p WHERE p.pessoaID = :pessoaID"),
    @NamedQuery(name = "PessoasFisicas.findByCpf", query = "SELECT p FROM PessoasFisicas p WHERE p.cpf = :cpf")})
public class PessoasFisicas implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "pessoaID")
    private Long pessoaID;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 11)
    @Column(name = "cpf")
    private String cpf;
    @JoinColumn(name = "pessoaID", referencedColumnName = "pessoaID", insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoas pessoas;

    public PessoasFisicas() {
    }
```

```
    public PessoasFisicas(Long pessoaID) {
        this.pessoaID = pessoaID;
    }

    public PessoasFisicas(Long pessoaID, String cpf) {
        this.pessoaID = pessoaID;
        this.cpf = cpf;
    }
...
```

PessoasJuridicas.java

```
@Entity
@Table(name = "PessoasJuridicas")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "PessoasJuridicas.findAll", query = "SELECT p FROM PessoasJuridicas p"),
    @NamedQuery(name = "PessoasJuridicas.findByPessoaID", query = "SELECT p FROM PessoasJuridicas p WHERE p.pessoaID = :pessoaID"),
   @NamedQuery(name = "PessoasJuridicas.findByCnpj", query = "SELECT p FROM PessoasJuridicas p WHERE p.cnpj = :cnpj")})
public class PessoasJuridicas implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "pessoaID")
```

```
    private Long pessoaID;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 14)
    @Column(name = "cnpj")
    private String cnpj;
    @JoinColumn(name = "pessoaID", referencedColumnName = "pessoaID",
insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoas pessoas;

    public PessoasJuridicas() {
    }
    public PessoasJuridicas(Long pessoaID) {
        this.pessoaID = pessoaID;
    }
    public PessoasJuridicas(Long pessoaID, String cnpj) {
        this.pessoaID = pessoaID;
        this.cnpj = cnpj;
    }
...
```

Produtos.java

```
@Entity
@Table(name = "Produtos")
@XmlRootElement
@NamedQueries({
   @NamedQuery(name = "Produtos.findAll", query = "SELECT p FROM Produtos p"),
```

```
    @NamedQuery(name = "Produtos.findByProdutoID", query = "SELECT p FROM Produtos p WHERE p.produtoID = :produtoID"),

    @NamedQuery(name = "Produtos.findByNome", query = "SELECT p FROM Produtos p WHERE p.nome = :nome"),

    @NamedQuery(name = "Produtos.findByQuantidade", query = "SELECT p FROM Produtos p WHERE p.quantidade = :quantidade"),

   @NamedQuery(name = "Produtos.findByPrecoVenda", query = "SELECT p FROM Produtos p WHERE p.precoVenda = :precoVenda")})

public class Produtos implements Serializable {



    private static final long serialVersionUID = 1L;

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Basic(optional = false)

    @Column(name = "produtoID")

    private Long produtoID;

    @Basic(optional = false)

    @NotNull

    @Size(min = 1, max = 255)

    @Column(name = "nome")

    private String nome;

    @Basic(optional = false)

    @NotNull

    @Column(name = "quantidade")

    private int quantidade;

   // @Max(value=?)  @Min(value=?)//if you know range of your decimal fields consider using these annotations to enforce field validation

    @Basic(optional = false)

    @NotNull

    @Column(name = "precoVenda")

    private Float precoVenda;
```

```
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "produtoID")
    private Collection<Movimentos> movimentosCollection;

    public Produtos() {
    }

    public Produtos(Long produtoID) {
        this.produtoID = produtoID;
    }

    public Produtos(Long produtoID, String nome, int quantidade, Float precoVenda) {
        this.produtoID = produtoID;
        this.nome = nome;
        this.quantidade = quantidade;
        this.precoVenda = precoVenda;
    }
...
```

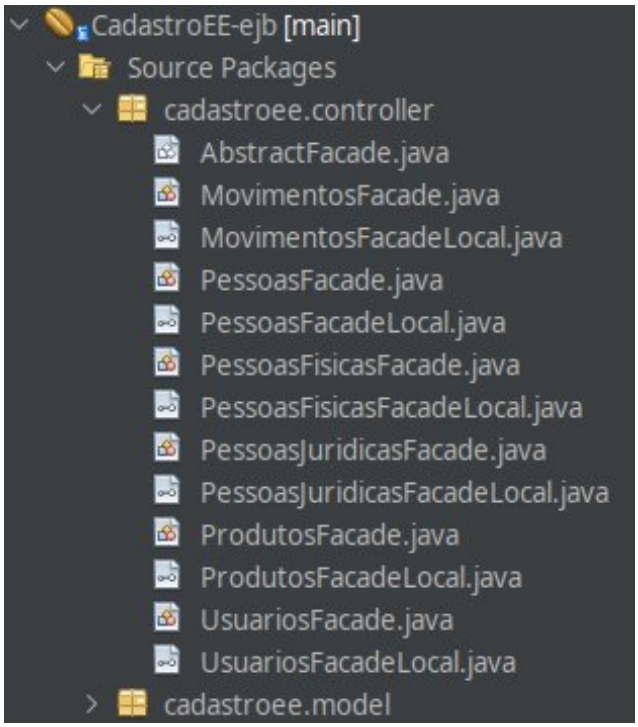
Usuarios.java

```
@Entity
@Table(name = "Usuarios")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Usuarios.findAll", query = "SELECT u FROM Usuarios u"),
    @NamedQuery(name = "Usuarios.findByUsuarioID", query = "SELECT u FROM Usuarios u WHERE u.usuarioID = :usuarioID"),
    @NamedQuery(name = "Usuarios.findByLogin", query = "SELECT u FROM Usuarios u WHERE u.login = :login"),
```

```
    @NamedQuery(name = "Usuarios.findBySenha", query = "SELECT u FROM Usuarios u WHERE u.senha = :senha")})
public class Usuarios implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "usuarioID")
    private Long usuarioID;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 60)
    @Column(name = "login")
    private String login;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 45)
    @Column(name = "senha")
    private String senha;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "usuarioID")
    private Collection<Movimentos> movimentosCollection;

    public Usuarios() {
    }

    public Usuarios(Long usuarioID) {
        this.usuarioID = usuarioID;
    }
```

```
public Usuarios(Long usuarioID, String login, String senha) {
    this.usuarioID = usuarioID;
    this.login = login;
    this.senha = senha;
}
```

d. Em seguida, adicionar os componentes EJB ao projeto, através da opção New Session Beans for Entity Classes.
e. Selecionar todas as entidades, marcar a geração da interface local, além de definir o nome do pacote como cadatroee.controller.
f. Serão gerados todos os Session Beans, com o sufixo Facade, bem como as interfaces, com o sufixo FacadeLocal.



AbstractFacade.java

```
public abstract class AbstractFacade<T> {

    private Class<T> entityClass;

    public AbstractFacade(Class<T> entityClass) {
        this.entityClass = entityClass;
    }

    protected abstract EntityManager getEntityManager();

    public void create(T entity) {
        getEntityManager().persist(entity);
    }

    public void edit(T entity) {
        getEntityManager().merge(entity);
    }

    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }

    public T find(Object id) {
        return getEntityManager().find(entityClass, id);
    }

    public List<T> findAll() {
        jakarta.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
```

```
        cq.select(cq.from(entityClass));
        return getEntityManager().createQuery(cq).getResultList();
    }

    public List<T> findRange(int[] range) {
        jakarta.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        jakarta.persistence.Query q = getEntityManager().createQuery(cq);
        q.setMaxResults(range[1] - range[0] + 1);
        q.setFirstResult(range[0]);
        return q.getResultList();
    }

    public int count() {
        jakarta.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
        jakarta.persistence.criteria.Root<T> rt = cq.from(entityClass);
        cq.select(getEntityManager().getCriteriaBuilder().count(rt));
        jakarta.persistence.Query q = getEntityManager().createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }
}
```

MovimentosFacade.java

```
@Stateless
```

```
public class MovimentosFacade extends AbstractFacade<Movimentos> implements MovimentosFacadeLocal {
    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public MovimentosFacade() {
        super(Movimentos.class);
    }
}
```

MovimentosFacadeLocal.java

```
@Local
public interface MovimentosFacadeLocal {
    void create(Movimentos movimentos);
    void edit(Movimentos movimentos);
    void remove(Movimentos movimentos);
    Movimentos find(Object id);
    List<Movimentos> findAll();
    List<Movimentos> findRange(int[] range);
    int count();
}
```

PessoasFacade.java

```
@Stateless
```

```
public class PessoasFacade extends AbstractFacade<Pessoas> implements PessoasFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public PessoasFacade() {
        super(Pessoas.class);
    }
}
```

PessoasFacadeLocal.java

```
@Local
public interface PessoasFacadeLocal {
    void create(Pessoas pessoas);
    void edit(Pessoas pessoas);
    void remove(Pessoas pessoas);
    Pessoas find(Object id);
    List<Pessoas> findAll();
    List<Pessoas> findRange(int[] range);
    int count();
}
```

PessoasFisicasFacade.java

```
@Stateless
```

```
public class PessoasFisicasFacade extends AbstractFacade<PessoasFisicas> implements PessoasFisicasFacadeLocal {
    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public PessoasFisicasFacade() {
        super(PessoasFisicas.class);
    }
}
```

PessoasFisicasFacadeLocal.java

```
@Local
public interface PessoasFisicasFacadeLocal {
    void create(PessoasFisicas pessoasFisicas);
    void edit(PessoasFisicas pessoasFisicas);
    void remove(PessoasFisicas pessoasFisicas);
    PessoasFisicas find(Object id);
    List<PessoasFisicas> findAll();
    List<PessoasFisicas> findRange(int[] range);
    int count();
}
```

PessoasJuridicasFacade.java

```
@Stateless
```

```
public class PessoasJuridicasFacade extends AbstractFacade<PessoasJuridicas>
implements PessoasJuridicasFacadeLocal {
    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public PessoasJuridicasFacade() {
        super(PessoasJuridicas.class);
    }
}
```

PessoasJuridicasFacadeLocal.java

```
@Local
public interface PessoasJuridicasFacadeLocal {
    void create(PessoasJuridicas pessoasJuridicas);
    void edit(PessoasJuridicas pessoasJuridicas);
    void remove(PessoasJuridicas pessoasJuridicas);
    PessoasJuridicas find(Object id);
    List<PessoasJuridicas> findAll();
    List<PessoasJuridicas> findRange(int[] range);
    int count();
}
```

ProdutosFacade.java

```
@Stateless
```

```
public class ProdutosFacade extends AbstractFacade<Produtos> implements ProdutosFacadeLocal {
    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public ProdutosFacade() {
        super(Produtos.class);
    }
}
```

ProdutosFacadeLocal.java

```
@Local
public interface ProdutosFacadeLocal {
    void create(Produtos produtos);
    void edit(Produtos produtos);
    void remove(Produtos produtos);
    Produtos find(Object id);
    List<Produtos> findAll();
    List<Produtos> findRange(int[] range);
    int count();
}
```

UsuariosFacade.java

```
@Stateless
```

```
public class UsuariosFacade extends AbstractFacade<Usuarios> implements UsuariosFacadeLocal {
    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public UsuariosFacade() {
        super(Usuarios.class);
    }
}
```

UsuariosFacadeLocal.java

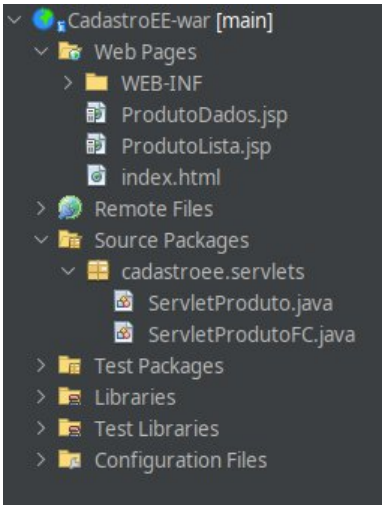
```
@Local
public interface UsuariosFacadeLocal {
    void create(Usuarios usuarios);
    void edit(Usuarios usuarios);
    void remove(Usuarios usuarios);
    Usuarios find(Object id);
    List<Usuarios> findAll();
    List<Usuarios> findRange(int[] range);
    int count();
}
```

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>
  </persistence-unit>
</persistence>
```

5. Criar um Servlet de teste no projeto CadastroEE-war.
   a. Utilizar o clique do botão direito e escolha da opção New..Servlet
   b. Definir o nome do Servlet como ServletProduto, e nome do pacote como cadastroee.servlets
   c. Marcar opção Add information to deployment descriptor, algo que ainda é necessário quando o GlassFish 6 é utilizado
   d. Adicionar, no código do Servlet, a referência para a interface do EJB
   e. Modificar a resposta do Servlet, utilizando o facade para recuperar os dados e apresentá-los na forma de lista HTML



ServletProduto.java

```
public class ServletProduto extends HttpServlet {
```

```
    @EJB
    ProdutosFacadeLocal facade;


    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet ServletProduto</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet ServletProduto at " + request.getContextPath() + "</h1>");
            var produtos = facade.findAll();
            out.println("<table>");
            out.println("<thead>\n" +
                "    <tr>\n" +
                "      <th scope=\"col\">Nome</th>\n" +
                "      <th scope=\"col\">Preço de venda</th>\n" +
                "      <th scope=\"col\">Quantidade</th>\n" +
                "    </tr>\n" +
                "  </thead>");
            out.println("<tbody>");
            for (Produtos produto: produtos) {
```

```
            out.println("<tr>");
            out.println("<td>" + produto.getNome() + "</td>");
            out.println("<td>" + produto.getPrecoVenda() + "</td>");
            out.println("<td>" + produto.getQuantidade() + "</td>");
            out.println("</tr>");
        }
        out.println("</tbody>");
        out.println("</table>");
        out.println("</body>");
        out.println("</html>");
    }
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

@Override
public String getServletInfo() {
    return "Short description";
}// </editor-fold>
```

```
}
```

6. Efetuar novos acertos no projeto, para uso do Jakarta:
   a. Adicionar a biblioteca Jakarta EE Web 8 API ao projeto CadatroEE-war
   b. Criado o Servlet e ajustadas as bibliotecas, o projeto deverá ficar como apresentado a seguir:
   c. Modificar TODAS as importações de pacotes javax para jakarta, em todos os arquivos do projeto CadastroEE-war
   d. Modificar o arquivo web.xml

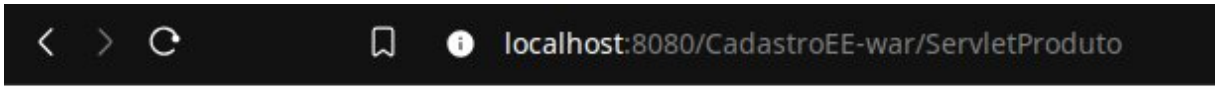
      web.xml

```
<?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-mapping>
      <servlet-name>ServletProduto</servlet-name>
      <url-pattern>/ServletProduto</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>ServletProdutoFC</servlet-name>
    <servlet-class>cadastroee.servlets.ServletProdutoFC</servlet-class>
  </servlet>
  <servlet-mapping>
     <servlet-name>ServletProdutoFC</servlet-name>
```

```
        <url-pattern>/ServletProdutoFC</url-pattern>
    </servlet-mapping>
    <session-config>
      <session-timeout>30</session-timeout>
    </session-config>
</web-app>
```

3.3. RESULTADO DOS CÓDIGOS

Ao acessar o endereço “http://localhost:8080/CadastroEE-war/ServletProduto”, há o seguinte resultado:



Servlet ServletProduto at /CadastroEE-war

Nome	Preço de venda	Quantidade
Banana	5.0	100
Laranja	2.0	500
Manga	4.0	800
Pera	21.0	30
Maça	14.0	60

3.4. ANÁLISE E CONCLUSÃO

3.4.1. a) COMO É ORGANIZADO UM PROJETO CORPORATIVO NO NETBEANS?

Um projeto corporativo no NetBeans é organizado dentro de um ambiente modular que separa a aplicação em diferentes camadas, como a de apresentação, lógica de negócios e persistência de dados. Geralmente, ele segue a arquitetura Java EE, utilizando projetos EAR, que agrupam módulos EJB para a lógica de negócios, WAR

para a interface web e JAR para bibliotecas compartilhadas, facilitando o desenvolvimento e a manutenção do sistema.

3.4.2. b) QUAL O PAPEL DAS TECNOLOGIAS JPA E EJB NA CONSTRUÇÃO DE UM APLICATIVO PARA A PLATAFORMA WEB NO AMBIENTE JAVA?

As tecnologias JPA e EJB desempenham um papel fundamental na construção de aplicativos para a plataforma web no ambiente Java. O JPA é responsável pelo mapeamento objeto-relacional, permitindo a persistência de dados de forma simplificada e padronizada. Já o EJB, gerencia a lógica de negócios, oferecendo serviços como transações, segurança e escalabilidade, além de facilitar a implementação de regras de negócio em um ambiente distribuído e seguro.

3.4.3. c) COMO O NETBEANS VIABILIZA A MELHORIA DE PRODUTIVIDADE AO LIDAR COM AS TECNOLOGIAS JPA E EJB?

O NetBeans melhora a produtividade ao lidar com JPA e EJB ao oferecer assistentes automáticos, geração de código, templates pré-configurados e integração nativa com servidores de aplicação, como GlassFish e Payara. Ele também permite a criação e configuração de entidades JPA e session beans de forma simplificada, gerando automaticamente classes e métodos padrões necessários para a persistência e gerenciamento de negócios, reduzindo o esforço do trabalho repetitivo no desenvolvimento da aplicação.

3.4.4. d) O QUE SÃO SERVLETS, E COMO O NETBEANS OFERECE SUPORTE À CONSTRUÇÃO DESSE TIPO DE COMPONENTES EM UM PROJETO WEB?

Os servlets são componentes Java utilizados para processar requisições HTTP em aplicações web, servindo como controladores que interagem com a lógica de negócios e a camada de persistência. O NetBeans oferece suporte à construção de servlets por meio de templates automáticos, geração de código padrão e integração com servidores de aplicação, permitindo a criação, configuração e teste desses componentes de maneira eficiente dentro do ambiente de desenvolvimento.

3.4.5. e) COMO É FEITA A COMUNICAÇÃO ENTRE OS SERLVETS E OS SESSION BEANS DO POOL DE EJBS?

A comunicação entre servlets e session beans do pool de EJBs é realizada por meio da injeção de dependência ou lookup JNDI. O servlet pode utilizar a anotação @EJB para obter uma instância gerenciada do EJB, garantindo que a lógica de negócios seja executada de maneira segura e eficiente. Quando a injeção direta não é possível, o servlet pode buscar o bean no contexto JNDI, permitindo a invocação remota ou local dos métodos disponíveis no session bean.

4. 2º PROCEDIMENTO | INTERFACE CADASTRAL COM SERVLET E JSPS

4.1. CÓDIGOS SOLICITADOS

Os códigos serão demonstrados neste relatório de acordo com as instruções apresentadas no enunciado da missão prática. Somente serão demonstrados nesse relatório as principais instruções que são relevantes ao desenvolvimento do código.

4.2. PRINCIPAIS INSTRUÇÕES DO 2º PROCEDIMENTO

1. Criar um Servlet com o nome ServletProdutoFC, no projeto CadastroEE-war:
   a. Utilizar o padrão Front Controller
   b. Adicionar uma referência para ProdutoFacadeLocal, utilizando o nome facade para o atributo
   c. Apagar o conteúdo interno do método processRequest, e efetuar nele as modificações seguintes
   d. Capturar o parâmetro acao a partir do request, o qual poderá assumir os valores listar, incluir, alterar, excluir, formIncluir e formAlterar
   e. Definir a variável destino, contendo o nome do JSP de apresentação, que terá os valores ProdutoDados.jsp, para acao valendo formAlterar ou formIncluir, ou ProdutoLista.jsp, para as demais opções

f. Para o valor listar, adicionar a listagem de produtos como atributo da requisição (request), com a consulta efetuada via facade
g. Para o valor formAlterar, capturar o id fornecido como parâmetro do request, consultar a entidade via facade, e adicioná-la como atributo da requisição (request)
h. Para o valor excluir, capturar o id fornecido como parâmetro do request, remover a entidade através do facade, e adicionar a listagem de produtos como atributo da requisição (request)
i. Para o valor alterar, capturar o id fornecido como parâmetro do request, consultar a entidade através do facade, preencher os demais campos com os valores fornecidos no request, alterar os dados via facade e adicionar a listagem de produtos como atributo da requisição (request)
j. Para o valor incluir, instanciar uma entidade do tipo Produto, preencher os campos com os valores fornecidos no request, inserir via facade e adicionar a listagem de produtos como atributo da requisição (request)
k. Ao final redirecionar para destino via RequestDispatcher, obtido a partir do objeto request.

ServletProdutoFC.java

```
public class ServletProdutoFC extends HttpServlet {

   @EJB
   private ProdutosFacadeLocal facade;

   protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
         throws ServletException, IOException {
      request.setCharacterEncoding("UTF-8");
      response.setContentType("text/html;charset=UTF-8");

      String acao = request.getParameter("acao");
```

```
        if (acao == null) {
            acao = "listar";
        }
        String destino = "ProdutoLista.jsp";
        switch(acao) {
            case "listar":
                List<Produtos> produtos = facade.findAll();
                request.setAttribute("produtos", produtos);
                break;
            case "incluir":
                Produtos produtoParaIncluir = new Produtos(
                        0L,
                        request.getParameter("nome"),
                        Integer.parseInt(request.getParameter("quantidade")),
                        Float.parseFloat(request.getParameter("preco"))
                );
                facade.create(produtoParaIncluir);
                request.setAttribute("produtos", facade.findAll());
                response.sendRedirect("ServletProdutoFC?acao=listar");
                return;
            case "alterar":
            Produtos produto = facade.find(Long.parseLong(request.getParameter("id")));
                produto.setNome(request.getParameter("nome"));
                produto.setPrecoVenda(Float.parseFloat(request.getParameter("preco")));

produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
                produto.setNome(request.getParameter("nome"));
                facade.edit(produto);
                request.setAttribute("produtos", facade.findAll());
                response.sendRedirect("ServletProdutoFC?acao=listar");
```

```
                return;
            case "excluir":
                facade.remove(facade.find(Long.parseLong(request.getParameter("id"))));
                request.setAttribute("produtos", facade.findAll());
                response.sendRedirect("ServletProdutoFC?acao=listar");
                return;
            case "formIncluir":

            case "formAlterar":
                destino = "ProdutoDados.jsp";
                if (acao.equals("formAlterar")) {
                    request.setAttribute("produto",
facade.find(Long.parseLong(request.getParameter("id"))));
                }
                break;
        }

        RequestDispatcher dispatcher = request.getRequestDispatcher(destino);
        dispatcher.forward(request, response);
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
```

```
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

2. Criar a página de consulta, com o nome ProdutoLista.jsp.
   a. Incluir um link para ServletProdutoFC, com acao formIncluir, voltado para a abertura do formulário de inclusão.
   b. Definir uma tabela para apresentação dos dados.
   c. Recuperar a lista de produtos enviada pelo Servlet.
   d. Para cada elemento da lista, apresentar id, nome, quantidade e preço como células da tabela.
   e. Criar, também, de forma dinâmica, links para alteração e exclusão, com a chamada para ServletProdutoFC, passando as ações corretas e o id do elemento corrente.

ProdutoLista.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="cadastroee.model.Produtos" %>
<%@ page import="java.util.List" %>
<!DOCTYPE html>
<html lang="pt-BR">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
        <title>Listagem de produtos</title>
    </head>
    <body>
        <h2>Listagem de Produtos</h2>
        <a href="ServletProdutoFC?acao=formIncluir">Novo Produto</a>

        <table>
            <thead>
                <tr>
                    <th scope="col">ID</th>
                    <th scope="col">Nome</th>
                    <th scope="col">Quantidade</th>
                    <th scope="col">Preço de venda</th>
                    <th scope="col">Opções</th>
                </tr>
            </thead>
            <tbody>
                <%
                    List<Produtos> produtos = (List<Produtos>) request.getAttribute("produtos");

                    for (Produtos produto: produtos) {
                        out.println("<tr>");
                        out.println("<td>" + produto.getProdutoID() + "</td>");
                        out.println("<td>" + produto.getNome() + "</td>");
                        out.println("<td>" + produto.getQuantidade() + "</td>");
                        out.println("<td>" + produto.getPrecoVenda() + "</td>");
                        out.println("<td>");
                        out.println("<a href=\"ServletProdutoFC?acao=formAlterar&id=" + produto.getProdutoID() + "\">Alterar</a>");
```

```
                    out.println("<a href=\"ServletProdutoFC?acao=excluir&id=" +
produto.getProdutoID() + "\">Excluir</a>");
                    out.println("</td>");
                    out.println("</tr>");
                }
            %>
        </tbody>
    </table>
</body>
</html>
```

3. Criar a página de cadastro, com o nome ProdutoDados.jsp.
   a. Definir um formulário com envio para ServletProdutoFC, modo post.
   b. Recuperar a entidade enviada pelo Servlet.
   c. Definir a variável acao, com valor incluir, para entidade nula, ou alterar, quando a entidade é fornecida.
   d. Incluir um campo do tipo hidden, para envio do valor de acao .
   e. Incluir um campo do tipo hidden, para envio do id, apenas quando o valor de acao for alterar.
   f. Incluir os campos para nome, quantidade e preço de venda, preenchendo os dados quando a entidade é fornecida.
   g. Concluir o formulário com um botão de envio, com o texto adequado para as situações de inclusão ou alteração de dados .

ProdutoDados.jsp

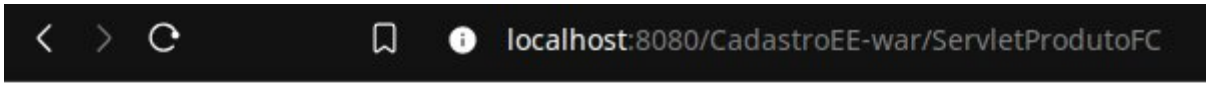
```
<%@page import="cadastroee.model.Produtos"%>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="pt-BR">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Dados do Produto</title>
    </head>
    <body>
        <h1>Dados do Produto</h1>


        <%
            Produtos produto = (Produtos) request.getAttribute("produto");
            String acao = produto != null ? "alterar" : "incluir";
        %>


        <form action="ServletProdutoFC" method="post">
            <input type="hidden" name="acao" value="<%= acao %>">
            <%
                if (acao.equals("alterar")) {
                    out.println("<input type=\"hidden\" name=\"id\" value=\"" +
produto.getProdutoID() + "\">");
                }
            %>
            <label>Nome:</label>
            <input type="text" name="nome" value="<%= acao == "alterar" ?
produto.getNome() : "" %>" required>


            <label>Quantidade:</label>
            <input type="text" name="quantidade" value="<%= acao == "alterar" ?
produto.getQuantidade() : "" %>" required>
```

```
        <label>Preço de venda:</label>
        <input type="text" name="preco" value="<%= acao == "alterar" ? produto.getPrecoVenda() : "" %>" required>

        <button type="submit"><%= acao == "alterar" ? "Alterar Produto" : "Incluir Produto" %></button>
    </form>
  </body>
</html>
```

4.3. RESULTADO DOS CÓDIGOS

Ao acessar o endereço “http://localhost:8080/CadastroEE-war/ServletProdutoFC”, há o seguinte resultado:

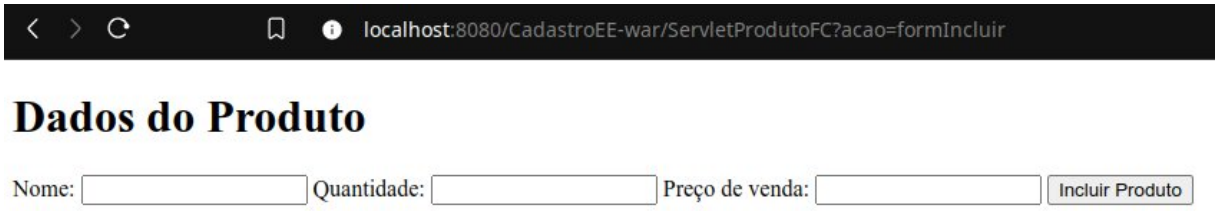


Listagem de Produtos

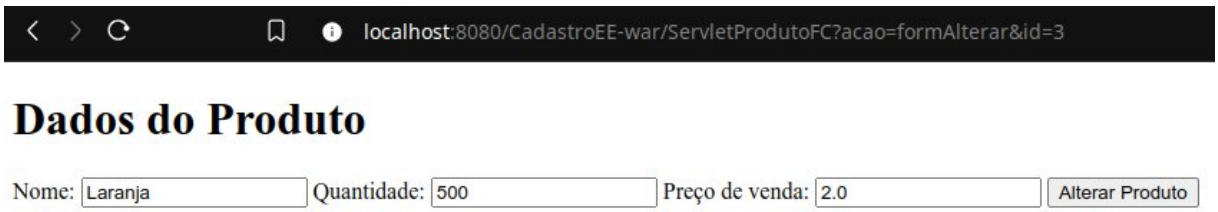
Novo Produto

ID	Nome	Quantidade	Preço de venda	Opções
1	Banana	100	5.0	Alterar Excluir
3	Laranja	500	2.0	Alterar Excluir
4	Manga	800	4.0	Alterar Excluir
11	Pera	30	21.0	Alterar Excluir
12	Maça	60	14.0	Alterar Excluir

Ao clicar no botão Novo Produto, há o seguinte resultado:



Ao clicar no botão alterar, há o seguinte resultado:



4.4. ANÁLISE E CONCLUSÃO

4.4.1. a) COMO FUNCIONA O PADRÃO FRONT CONTROLLER, E COMO ELE É IMPLEMENTADO EM UM APLICATIVO WEB JAVA, NA ARQUITETURA MVC?

O padrão Front Controller funciona como um ponto central de controle para todas as requisições de um aplicativo web, permitindo que a lógica de navegação e manipulação de requisições seja gerenciada em um único local. Em um aplicativo Java Web com a arquitetura MVC, ele é geralmente implementado por um servlet que recebe todas as requisições do usuário, processa os dados necessários, delega o processamento para a camada de negócios e encaminha a resposta para a camada de apresentação, que pode ser uma página JSP. Isso facilita a manutenção, organização do código e aplicação de regras de segurança e autorização.

4.4.2. b) QUAIS AS DIFERENÇAS E SEMELHANÇAS ENTRE SERVLETS E JSPS?

Os servlets e JSPs possuem semelhanças, pois ambos são componentes Java utilizados no desenvolvimento de aplicações web e executam no servidor para

processar requisições HTTP. No entanto, os servlets são escritos completamente em Java e lidam principalmente com a lógica da aplicação, enquanto os JSPs são mais voltados para a apresentação, permitindo a inclusão de código Java dentro de páginas HTML. A principal diferença é que os JSPs são convertidos em servlets pelo servidor de aplicação, tornando seu uso mais intuitivo para desenvolver interfaces gráficas, enquanto os servlets exigem a manipulação manual de saídas HTML através do código Java.

4.4.3. 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?

O redirecionamento simples, realizado com o método sendRedirect(), instrui o navegador do usuário a fazer uma nova requisição para outro recurso, o que resulta em uma nova solicitação HTTP e perda dos dados armazenados no escopo da requisição original. Já o método forward(), do RequestDispatcher, transfere o controle para outro recurso dentro do próprio servidor, sem gerar uma nova requisição HTTP, permitindo a reutilização dos dados armazenados na requisição original. Os parâmetros nos objetos HttpServletRequest servem para capturar dados enviados pelo usuário através de formulários ou URLs, enquanto os atributos permitem o armazenamento temporário de informações que podem ser compartilhadas entre diferentes componentes da aplicação durante o processamento da mesma requisição.

5. 3º PROCEDIMENTO | MELHORANDO O DESIGN DA INTERFACE

5.1. CÓDIGOS SOLICITADOS

Os códigos serão demonstrados neste relatório de acordo com as instruções apresentadas no enunciado da missão prática. Somente serão demonstrados nesse relatório as principais instruções que são relevantes ao desenvolvimento do código.

5.2. PRINCIPAIS INSTRUÇÕES DO 3º PROCEDIMENTO

1. Incluir as bibliotecas do framework Bootstrap nos arquivos ProdutoLista.jsp e ProdutoDados.jsp
   a. Visitar o site do BootStrap, no endereço https://getbootstrap.com/
   b. Rolar para baixo até encontrar a inclusão via CDN
   c. Clicar no botão para cópia do link CSS e colar na divisão head de cada uma das páginas JSP.
   d. Clicar no botão para cópia do link para a biblioteca Java Script e colar na divisão head de cada uma das páginas JSP

2. Modificar as características de ProdutoLista.jsp
   a. Adicionar a classe container ao body.
   b. Adicionar as classes btn, btn-primary e m-2 no link de inclusão.
   c. Adicionar as classes table e table-striped na tabela.
   d. Adicionar a classe table-dark ao thead.
   e. Adicionar as classes btn, btn-primary e btn-sm no link de alteração.
   f. Adicionar as classes btn, btn-danger e btn-sm no link de exclusão.

ProdutoLista.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="cadastroee.model.Produtos" %>
<%@ page import="java.util.List" %>
<!DOCTYPE html>
<html lang="pt-BR">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Listagem de produtos</title>
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
```

```
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>

    </head>

    <body class="container">

        <h1>Listagem de Produtos</h1>

        <a class="btn btn-primary m-2"
href="ServletProdutoFC?acao=formIncluir">Novo Produto</a>


        <table class="table table-striped">

            <thead class="table-dark">

                <tr>

                    <th scope="col">ID</th>

                    <th scope="col">Nome</th>

                    <th scope="col">Quantidade</th>

                    <th scope="col">Preço de venda</th>

                    <th scope="col">Opções</th>

                </tr>

            </thead>

            <tbody>

                <%

                    List<Produtos> produtos = (List<Produtos>)
request.getAttribute("produtos");


                    for (Produtos produto: produtos) {

                        out.println("<tr>");

                        out.println("<td>" + produto.getProdutoID() + "</td>");

                        out.println("<td>" + produto.getNome() + "</td>");

                        out.println("<td>" + produto.getQuantidade() + "</td>");

                        out.println("<td>" + produto.getPrecoVenda() + "</td>");
```

```
                    out.println("<td>");
                    out.println("<a class=\"btn btn-primary btn-sm\"
href=\"ServletProdutoFC?acao=formAlterar&id=" + produto.getProdutoID() +
"\">Alterar</a>");
                    out.println("<a class=\"btn btn-danger btn-sm\"
href=\"ServletProdutoFC?acao=excluir&id=" + produto.getProdutoID() +
"\">Excluir</a>");
                    out.println("</td>");
                    out.println("</tr>");
                }
            %>
        </tbody>
    </table>
  </body>
</html>
```

1. Modificar as características de ProdutoDados.jsp
   a. Adicionar a classe container ao body.
   b. Encapsule cada par label / input em div com classe mb-3.
   c. Adicionar a classe form ao formulário.
   d. Adicionar a classe form-label em cada label.
   e. Adicionar a classe form-control em cada input.
   f. Adicionar as classes btn e btn-primary ao botão de inclusão.
   g. Ajustar as características para obter o design apresentado a seguir.

ProdutoDados.jsp

```
<%@page import="cadastroee.model.Produtos"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
```

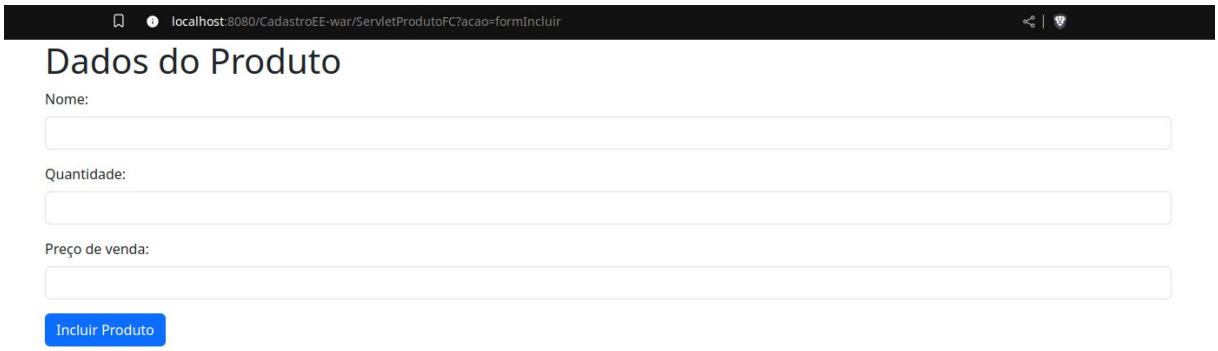
```
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Dados do Produto</title>
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
    </head>
    <body class="container">
        <h1>Dados do Produto</h1>


        <%
            Produtos produto = (Produtos) request.getAttribute("produto");
            String acao = produto != null ? "alterar" : "incluir";
        %>


        <form class="form" action="ServletProdutoFC" method="post">
            <input type="hidden" name="acao" value="<%= acao %>">
            <%
                if (acao.equals("alterar")) {
                    out.println("<input type=\"hidden\" name=\"id\" value=\"" +
produto.getProdutoID() + "\">");
                }
            %>
            <div class="mb-3">
                <label class="form-label">Nome:</label>
                <input class="form-control" type="text" name="nome" value="<%= acao ==
"alterar" ? produto.getNome() : "" %>" required>
```

```
        </div>


        <div class="mb-3">
            <label class="form-label">Quantidade:</label>
            <input class="form-control" type="text" name="quantidade" value="<%=
acao == "alterar" ? produto.getQuantidade() : "" %>" required>
        </div>


        <div class="mb-3">
            <label class="form-label">Preço de venda:</label>
            <input class="form-control" type="text" name="preco" value="<%= acao ==
"alterar" ? produto.getPrecoVenda() : "" %>" required>
        </div>


        <button class="btn btn-primary" type="submit"><%= acao == "alterar" ?
"Alterar Produto" : "Incluir Produto" %></button>
    </form>
  </body>
</html>
```

5.3. RESULTADO DOS CÓDIGOS

Ao acessar o endereço “http://localhost:8080/CadastroEE-war/ServletProdutoFC”, há o seguinte resultado:

Ao clicar no botão Novo Produto, há o seguinte resultado:



Ao clicar no botão alterar, há o seguinte resultado:

5.4. ANÁLISE E CONCLUSÃO

5.4.1. a) COMO O FRAMEWORK BOOTSTRAP É UTILIZADO?

O framework CSS Bootstrap é utilizado para facilitar o desenvolvimento de interfaces web modernas e responsivas, oferecendo um conjunto de estilos pré-definidos, componentes reutilizáveis e um sistema de grid flexível. Ele pode ser integrado ao projeto através de um link para sua CDN ou pela inclusão de seus arquivos locais, permitindo que os desenvolvedores utilizem suas classes CSS e componentes prontos para criar layouts bem estruturados sem a necessidade de escrever muito código manualmente.

5.4.2. b) POR QUE O BOOTSTRAP GARANTE A INDEPENDÊNCIA ESTRUTURAL DO HTML?

O Bootstrap garante a independência estrutural do HTML porque separa a lógica de apresentação da estrutura do conteúdo. Suas classes CSS podem ser aplicadas diretamente aos elementos HTML sem modificar a estrutura básica da página, permitindo que os desenvolvedores mantenham um código HTML limpo e sem a necessidade de adicionar estilos inline ou personalizações excessivas. Isso torna a manutenção do código mais fácil e permite que diferentes estilos e temas sejam aplicados sem alterar o HTML.

5.4.3. c) QUAL A RELAÇÃO ENTRE O BOOSTRAP E A RESPONSIVIDADE DA PÁGINA?

A relação entre o Bootstrap e a responsividade da página está no seu sistema de grid e nas classes utilitárias que adaptam os elementos de forma dinâmica a diferentes tamanhos de tela. O sistema de grid baseado em flexbox ou CSS grid permite organizar os elementos de maneira fluida, ajustando colunas e espaçamentos conforme a largura da tela. Além disso, o Bootstrap oferece classes específicas para tornar componentes e elementos adaptáveis a dispositivos móveis, garantindo uma boa experiência de uso em qualquer tipo de tela, desde celulares até monitores de alta resolução.

REFERÊNCIAS

ApacheNetbens, “Creating an Enterprise Application with EJB 3.1”. Disponível em: <https://netbeans.apache.org/tutorial/main/kb/docs/javaee/javaee-entapp-ejb/>. Acesso em 16 de fevereiro de 2025.

IBM, “Arquitetura Java Persistence API (JPA)”. Disponível em: <https://www.ibm.com/docs/pt-br/was/8.5.5?topic=SSEQTP_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/cejb_persistence.htm>. Acesso em 16 de fevereiro de 2025.

GeeksForGeeks, “Enterprise Java Beans (EJB)”. Disponível em: <https://www.geeksforgeeks.org/enterprise-java-beans-ejb/>. Acesso em 16 de fevereiro de 2025.

DevMedia, “Introdução a Servlets em Java”. Disponível em: <https://www.devmedia.com.br/introducao-a-servlets-em-java/25285>. Acesso em 16 de devereiro de 2025.

ApacheNetbeans, “Java EE & Java Web Learning Trail”. Disponível em <https://netbeans.apache.org/tutorial/main/kb/docs/java-ee/>. Acesso em 16 de fevereiro de 2025.

Oracle, “Tuning the EJB Pool”. Disponível em <https://docs.oracle.com/cd/E19159-01/819-3681/abeee/index.html>. Acesso em 16 de fevereiro de 2025.

Oracle Docs, “Administering JDBC Connection Pools”. Disponível em <https://docs.oracle.com/cd/E19798-01/821-1751/gharo/index.html#:~:text=A%20JDBC%20connection%20pool%20is,a%20pool%20of%20available%20connections.>. Acesso em 16 de fevereiro de 2025.

Oracle, “Core J2EE Patterns - Front Controller”. Disponível em <https://www.oracle.com/java/technologies/front-controller.html>. Acesso em 16 de fevereiro de 2025.

JDevTreinamento, “ARQUITETURA MVC – INTRODUÇÃO”. Disponível em <https://www.jdevtreinamento.com.br/arquitetura-mvc-introducao/#:~:text=A%20arquitetura%20MVC%20%C3%A9%20um,de%20neg%C3%B3cio%2C%20l%C3%B3gicas%20e%20fun%C3%A7%C3%B5es.>. Acesso em 16 de fevereiro de 2025.

DevMedia, “Introdução ao Java Server Pages - JSP”. Disponível em <https://www.devmedia.com.br/introducao-ao-java-server-pages-jsp/25602>. Acesso em 16 de fevereiro de 2025.

SmoothProgramming, “Differences between sendRedirect and forward Method Execution Flow”. Disponível em <https://smoothprogramming.com/java/differences-of-sendredirect-and-forward-method-execution-flow/#:~:text=SendRedirect%20tells%20the%20browser%20to,same%20request%20to%20next%20resource.>. Acesso em 16 de fevereiro de 2025.

GetBootstrap, “Get started with Bootstrap”. Disponível em <https://getbootstrap.com/docs/5.3/getting-started/introduction/>. Acesso em 16 de fevereiro de 2025.