Nível 4: Vamos Integrar Sistemas (RPG0017)

GILSON MIRANDA NETO 202204437562

Campus Pintangueiras
Vamos integrar sistemas **– 2023.2 – 3º semestre do curso**

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.

1º Procedimento – Camadas de Persistência e Controle

CÓDIGOS SOLICITADOS NO ROTEIRO DE AULA:

Arquivo: MovimentoFacade java

```
package cadastroee.controller;

import cadastroee.model.Movimento;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import jakarta.persistence.TypedQuery;
import java.util.List;


@Stateless
public class MovimentoFacade implements MovimentoFacadeLocal {

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

    @Override
    public void criarMovimento(Movimento movimento) {
        em.persist(movimento);
    }

    @Override
    public void atualizarMovimento(Movimento movimento) {
        em.merge(movimento);
    }

    @Override
    public void excluirMovimento(int idMovimento) {
        Movimento movimento = em.find(Movimento.class, idMovimento);
        if (movimento != null) {
            em.remove(movimento);
        }
    }

    @Override
    public Movimento encontrarMovimentoPorId(int idMovimento) {
        return em.find(Movimento.class, idMovimento);
    }

    @Override
    public List<Movimento> listarTodosMovimentos() {
        TypedQuery<Movimento> query = em.createQuery("SELECT m FROM
Movimento m", Movimento.class);
        return query.getResultList();
    }
}
```

Arquivo: MovimentoFacadeLocal.java

```java
package cadastroee.controller;

import cadastroee.model.Movimento;
import jakarta.ejb.Local;
import java.util.List;


@Local
public interface MovimentoFacadeLocal {

    void criarMovimento(Movimento movimento);

    void atualizarMovimento(Movimento movimento);

    void excluirMovimento(int idMovimento);

    Movimento encontrarMovimentoPorId(int idMovimento);

    List<Movimento> listarTodosMovimentos();
}
```

Arquivo: PessoaFacade.java

```java
package cadastroee.controller;

import cadastroee.model.Pessoa;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import java.util.List;


@Stateless
public class PessoaFacade implements PessoaFacadeLocal {

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

    @Override
    public void criar(Pessoa pessoa) {
        em.persist(pessoa);
    }

    @Override
    public void atualizar(Pessoa pessoa) {
        em.merge(pessoa);
    }

    @Override
    public void remover(Pessoa pessoa) {
        em.remove(em.merge(pessoa));
    }

    @Override
    public Pessoa encontrar(Object id) {
        return em.find(Pessoa.class, id);
    }
```

```
    @Override
    public List<Pessoa> encontrarTodos() {
        return em.createNamedQuery("Pessoa.findAll").getResultList();
    }

    @Override
    public List<Pessoa> encontrarRange(int[] range) {
        jakarta.persistence.criteria.CriteriaQuery cq =
em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(Pessoa.class));
        jakarta.persistence.Query q = em.createQuery(cq);
        q.setMaxResults(range[1] - range[0] + 1);
        q.setFirstResult(range[0]);
        return q.getResultList();
    }

    @Override
    public int contar() {
        jakarta.persistence.criteria.CriteriaQuery cq =
em.getCriteriaBuilder().createQuery();
        jakarta.persistence.criteria.Root<Pessoa> rt =
cq.from(Pessoa.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        jakarta.persistence.Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }
}

    public static ResultSet getSelect(String sql) throws SQLException
{
        return getPrepared(sql).executeQuery();
    }

    public static void close(PreparedStatement statement, ResultSet
resultSet, Connection connection) {
        try {
            if (resultSet != null) {
                resultSet.close();
            }
            if (statement != null) {
                statement.close();
            }
            if (connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Arquivo: PessoaFacadeLocal.java

```
package cadastroee.controller;

import cadastroee.model.Pessoa;
import jakarta.ejb.Local;
import java.util.List;
```

```java
@Local
public interface PessoaFacadeLocal {

    void criar(Pessoa pessoa);

    void atualizar(Pessoa pessoa);

    void remover(Pessoa pessoa);

    Pessoa encontrar(Object id);

    List<Pessoa> encontrarTodos();

    List<Pessoa> encontrarRange(int[] range);

    int contar();
}
```

Arquivo: PessoaFisicaFacade.java

```java
package cadastroee.controller;

import cadastroee.model.PessoaFisica;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import java.util.List;


@Stateless
public class PessoaFisicaFacade implements PessoaFisicaFacadeLocal {

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

    @Override
    public void create(PessoaFisica pessoaFisica) {
        em.persist(pessoaFisica);
    }

    @Override
    public void edit(PessoaFisica pessoaFisica) {
        em.merge(pessoaFisica);
    }

    @Override
    public void remove(PessoaFisica pessoaFisica) {
        em.remove(em.merge(pessoaFisica));
    }

    @Override
    public PessoaFisica find(Object id) {
        return em.find(PessoaFisica.class, id);
    }

    @Override
    public List<PessoaFisica> findAll() {
        return
```

```
em.createNamedQuery("PessoaFisica.findAll").getResultList();
    }

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

    @Override
    public int count() {
        jakarta.persistence.criteria.CriteriaQuery cq =
em.getCriteriaBuilder().createQuery();
        jakarta.persistence.criteria.Root<PessoaFisica> rt =
cq.from(PessoaFisica.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        jakarta.persistence.Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }
}
```

Arquivo: PessoaFisicaFacadeLocal.java

```
package cadastroee.controller;

import cadastroee.model.PessoaFisica;
import java.util.List;
import jakarta.ejb.Local;

@Local
public interface PessoaFisicaFacadeLocal {

    void create(PessoaFisica pessoaFisica);

    void edit(PessoaFisica pessoaFisica);

    void remove(PessoaFisica pessoaFisica);

    PessoaFisica find(Object id);

    List<PessoaFisica> findAll();

    List<PessoaFisica> findRange(int[] range);

    int count();
}
```

Arquivo: PessoaJuridicaFacade.java

```
package cadastroee.controller;

import jakarta.ejb.Stateless;
```

```java
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import java.util.List;
import cadastroee.model.PessoaJuridica;


@Stateless
public class PessoaJuridicaFacade implements PessoaJuridicaFacadeLocal
{

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

    @Override
    public void create(PessoaJuridica pessoaJuridica) {
        em.persist(pessoaJuridica);
    }

    @Override
    public void edit(PessoaJuridica pessoaJuridica) {
        em.merge(pessoaJuridica);
    }

    @Override
    public void remove(PessoaJuridica pessoaJuridica) {
        em.remove(em.merge(pessoaJuridica));
    }

    @Override
    public PessoaJuridica find(Object id) {
        return em.find(PessoaJuridica.class, id);
    }

    @Override
    public List<PessoaJuridica> findAll() {
        return em.createQuery("select object(o) from PessoaJuridica as
o").getResultList();
    }

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

    @Override
    public int count() {
        jakarta.persistence.criteria.CriteriaQuery cq =
em.getCriteriaBuilder().createQuery();
        jakarta.persistence.criteria.Root<PessoaJuridica> rt =
cq.from(PessoaJuridica.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        jakarta.persistence.Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }
}
```

Arquivo: PessoaJuridicaFacadeLocal.java

```java
package cadastroee.controller;

import java.util.List;
import cadastroee.model.PessoaJuridica;
import jakarta.ejb.Local;


@Local
public interface PessoaJuridicaFacadeLocal {

    void create(PessoaJuridica pessoaJuridica);

    void edit(PessoaJuridica pessoaJuridica);

    void remove(PessoaJuridica pessoaJuridica);

    PessoaJuridica find(Object id);

    List<PessoaJuridica> findAll();

    List<PessoaJuridica> findRange(int[] range);

    int count();
}
```

Arquivo: ProdutoFacade.java

```java
package cadastroee.controller;

import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import cadastroee.model.Produto;
import java.util.List;


@Stateless
public class ProdutoFacade implements ProdutoFacadeLocal {

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

    @Override
    public void create(Produto produto) {
        em.persist(produto);
    }

    @Override
    public void edit(Produto produto) {
        em.merge(produto);
    }

    @Override
    public void remove(Produto produto) {
```

```
        em.remove(em.merge(produto));
    }

    @Override
    public Produto find(Object idProduto) {
        return em.find(Produto.class, idProduto);
    }

    @Override
    public List<Produto> findAll() {
        return em.createNamedQuery("Produto.findAll",
Produto.class).getResultList();
    }


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

    @Override
    public int count() {
        jakarta.persistence.criteria.CriteriaQuery cq =
em.getCriteriaBuilder().createQuery();
        jakarta.persistence.criteria.Root<Produto> rt =
cq.from(Produto.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        jakarta.persistence.Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }
}
```

Arquivo: ProdutoFacadeLocal.java

```
package cadastroee.controller;

import java.util.List;
import jakarta.ejb.Local;
import cadastroee.model.Produto;


@Local
public interface ProdutoFacadeLocal {

    void create(Produto produto);

    void edit(Produto produto);

    void remove(Produto produto);

    Produto find(Object idProduto);

    List<Produto> findAll();
```

```
    List<Produto> findRange(int[] range);

    int count();
}
```

Arquivo: UsuarioFacade.java

```
package cadastroee.controller;

import cadastroee.model.Usuario;
import jakarta.ejb.Stateless;
import java.util.List;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;


@Stateless
public class UsuarioFacade implements UsuarioFacadeLocal {

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

    @Override
    public void create(Usuario usuario) {
        em.persist(usuario);
    }

    @Override
    public void edit(Usuario usuario) {
        em.merge(usuario);
    }

    @Override
    public void remove(Usuario usuario) {
        em.remove(em.merge(usuario));
    }

    @Override
    public Usuario find(Object id) {
        return em.find(Usuario.class, id);
    }

    @Override
    public List<Usuario> findAll() {
        jakarta.persistence.criteria.CriteriaQuery cq =
em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(Usuario.class));
        return em.createQuery(cq).getResultList();
    }

    @Override
    public List<Usuario> findRange(int[] range) {
        jakarta.persistence.criteria.CriteriaQuery cq =
em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(Usuario.class));
        jakarta.persistence.Query q = em.createQuery(cq);
        q.setMaxResults(range[1] - range[0] + 1);
        q.setFirstResult(range[0]);
```

```
        return q.getResultList();
    }

    @Override
    public int count() {
        jakarta.persistence.criteria.CriteriaQuery cq =
em.getCriteriaBuilder().createQuery();
        jakarta.persistence.criteria.Root<Usuario> rt =
cq.from(Usuario.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        jakarta.persistence.Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }
}
```

Arquivo: UsuarioFacadeLocal.java

```
package cadastroee.controller;

import cadastroee.model.Usuario;
import java.util.List;
import jakarta.ejb.Local;


@Local
public interface UsuarioFacadeLocal {

    void create(Usuario usuario);

    void edit(Usuario usuario);

    void remove(Usuario usuario);

    Usuario find(Object id);

    List<Usuario> findAll();

    List<Usuario> findRange(int[] range);

    int count();
}
```

Arquivo: Movimento.java

```
package cadastroee.model;

import java.io.Serializable;
import java.math.BigDecimal;
import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.NamedQueries;
```

```java
import jakarta.persistence.NamedQuery;
import jakarta.persistence.Table;
import jakarta.xml.bind.annotation.XmlRootElement;


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

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idMovimento")
    private Integer idMovimento;
    @Column(name = "quantidade")
    private Integer quantidade;
    @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
    @Column(name = "valorUnitario")
    private BigDecimal valorUnitario;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa")
    @ManyToOne(optional = false)
    private Pessoa idPessoa;
    @JoinColumn(name = "idProduto", referencedColumnName =
"idProduto")
    @ManyToOne(optional = false)
    private Produto idProduto;
    @JoinColumn(name = "idUsuario", referencedColumnName =
"idUsuario")
    @ManyToOne(optional = false)
    private Usuario idUsuario;

    public Movimento() {
    }

    public Movimento(Integer idMovimento) {
        this.idMovimento = idMovimento;
    }

    public Integer getIdMovimento() {
        return idMovimento;
    }

    public void setIdMovimento(Integer idMovimento) {
        this.idMovimento = idMovimento;
    }
```

```java
    public Integer getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(Integer quantidade) {
        this.quantidade = quantidade;
    }

    public Character getTipo() {
        return tipo;
    }

    public void setTipo(Character tipo) {
        this.tipo = tipo;
    }

    public BigDecimal getValorUnitario() {
        return valorUnitario;
    }

    public void setValorUnitario(BigDecimal valorUnitario) {
        this.valorUnitario = valorUnitario;
    }

    public Pessoa getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Pessoa idPessoa) {
        this.idPessoa = idPessoa;
    }

    public Produto getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Produto idProduto) {
        this.idProduto = idProduto;
    }

    public Usuario getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(Usuario idUsuario) {
        this.idUsuario = idUsuario;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idMovimento != null ? idMovimento.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id
fields are not set
        if (!(object instanceof Movimento)) {
            return false;
        }
```

```
        Movimento other = (Movimento) object;
        if ((this.idMovimento == null && other.idMovimento != null) ||
(this.idMovimento != null &&
!this.idMovimento.equals(other.idMovimento))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Movimento[ idMovimento=" +
idMovimento + " ]";
    }

}
```

Arquivo: Pessoa.java

```
package cadastroee.model;

import java.io.Serializable;
import java.util.Collection;
import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;


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

```
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Column(name = "nome")
    private String nome;
    @Column(name = "logradouro")
    private String logradouro;
    @Column(name = "cidade")
    private String cidade;
    @Column(name = "estado")
    private String estado;
    @Column(name = "telefone")
    private String telefone;
    @Column(name = "email")
    private String email;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaJuridica pessoaJuridica;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaFisica pessoaFisica;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idPessoa")
    private Collection<Movimento> movimentoCollection;

    public Pessoa() {
    }

    public Pessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }
```

```
    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public PessoaJuridica getPessoaJuridica() {
        return pessoaJuridica;
    }

    public void setPessoaJuridica(PessoaJuridica pessoaJuridica) {
        this.pessoaJuridica = pessoaJuridica;
    }

    public PessoaFisica getPessoaFisica() {
        return pessoaFisica;
    }

    public void setPessoaFisica(PessoaFisica pessoaFisica) {
        this.pessoaFisica = pessoaFisica;
    }

    @XmlTransient
    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento>
movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPessoa != null ? idPessoa.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {

        if (!(object instanceof Pessoa)) {
```

```
            return false;
        }
        Pessoa other = (Pessoa) object;
        if ((this.idPessoa == null && other.idPessoa != null) ||
(this.idPessoa != null && !this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Pessoa[ idPessoa=" + idPessoa + " ]";
    }

}
```

Arquivo: PessoaFisica.java

```
package cadastroee.model;

import java.io.Serializable;
import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import jakarta.xml.bind.annotation.XmlRootElement;


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

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Column(name = "cpf")
    private String cpf;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa",
insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoa pessoa;

    public PessoaFisica() {
```

```
    }

    public PessoaFisica(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public Pessoa getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoa pessoa) {
        this.pessoa = pessoa;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPessoa != null ? idPessoa.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id
fields are not set
        if (!(object instanceof PessoaFisica)) {
            return false;
        }
        PessoaFisica other = (PessoaFisica) object;
        if ((this.idPessoa == null && other.idPessoa != null) ||
(this.idPessoa != null && !this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.PessoaFisica[ idPessoa=" + idPessoa +
" ]";
    }

}
```

Arquivo: PessoaJuridica.java

```
package cadastroee.model;

import java.io.Serializable;
import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import jakarta.xml.bind.annotation.XmlRootElement;


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

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Column(name = "cnpj")
    private String cnpj;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa",
insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoa pessoa;

    public PessoaJuridica() {
    }

    public PessoaJuridica(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
```

```
        this.cnpj = cnpj;
    }

    public Pessoa getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoa pessoa) {
        this.pessoa = pessoa;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPessoa != null ? idPessoa.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {

        if (!(object instanceof PessoaJuridica)) {
            return false;
        }
        PessoaJuridica other = (PessoaJuridica) object;
        if ((this.idPessoa == null && other.idPessoa != null) ||
(this.idPessoa != null && !this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.PessoaJuridica[ idPessoa=" + idPessoa
+ " ]";
    }

}
```

Arquivo: Produto.java

```
package cadastroee.model;

import java.io.Serializable;
import java.util.Collection;
import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;
```

```java
@Entity
@Table(name = "Produto")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Produto.findAll", query = "SELECT p FROM
Produto p"),
    @NamedQuery(name = "Produto.findByIdProduto", query = "SELECT p
FROM Produto p WHERE p.idProduto = :idProduto"),
    @NamedQuery(name = "Produto.findByNome", query = "SELECT p FROM
Produto p WHERE p.nome = :nome"),
    @NamedQuery(name = "Produto.findByQuantidade", query = "SELECT p
FROM Produto p WHERE p.quantidade = :quantidade"),
    @NamedQuery(name = "Produto.findByPrecoVenda", query = "SELECT p
FROM Produto p WHERE p.precoVenda = :precoVenda")})
public class Produto implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idProduto")
    private Integer idProduto;
    @Column(name = "nome")
    private String nome;
    @Column(name = "quantidade")
    private Integer quantidade;
    @Column(name = "precoVenda")
    private Float precoVenda;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idProduto")
    private Collection<Movimento> movimentoCollection;

    public Produto() {
    }

    public Produto(Integer idProduto, String nome, Integer quantidade,
Float precoVenda) {
    this.idProduto = idProduto;
    this.nome = nome;
    this.quantidade = quantidade;
    this.precoVenda = precoVenda;
    }

    public Produto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public Integer getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
```

```
    }

    public Integer getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(Integer quantidade) {
        this.quantidade = quantidade;
    }

    public Float getPrecoVenda() {
        return precoVenda;
    }

    public void setPrecoVenda(Float precoVenda) {
        this.precoVenda = precoVenda;
    }

    @XmlTransient
    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento>
movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idProduto != null ? idProduto.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {

        if (!(object instanceof Produto)) {
            return false;
        }
        Produto other = (Produto) object;
        if ((this.idProduto == null && other.idProduto != null) ||
(this.idProduto != null && !this.idProduto.equals(other.idProduto))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Produto[ idProduto=" + idProduto + "
]";
    }

}
```

Arquivo: Usuario.java

```java
package cadastroee.model;

import java.io.Serializable;
import java.util.Collection;
import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;


@Entity
@Table(name = "Usuario")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM
Usuario u"),
    @NamedQuery(name = "Usuario.findByIdUsuario", query = "SELECT u
FROM Usuario u WHERE u.idUsuario = :idUsuario"),
    @NamedQuery(name = "Usuario.findByLogin", query = "SELECT u FROM
Usuario u WHERE u.login = :login"),
    @NamedQuery(name = "Usuario.findBySenha", query = "SELECT u FROM
Usuario u WHERE u.senha = :senha")})
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idUsuario")
    private Integer idUsuario;
    @Column(name = "login")
    private String login;
    @Column(name = "senha")
    private String senha;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idUsuario")
    private Collection<Movimento> movimentoCollection;

    public Usuario() {
    }

    public Usuario(Integer idUsuario) {
        this.idUsuario = idUsuario;
    }

    public Integer getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(Integer idUsuario) {
        this.idUsuario = idUsuario;
    }
```

```
    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
        this.senha = senha;
    }

    @XmlTransient
    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento>
movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idUsuario != null ? idUsuario.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id
fields are not set
        if (!(object instanceof Usuario)) {
            return false;
        }
        Usuario other = (Usuario) object;
        if ((this.idUsuario == null && other.idUsuario != null) ||
(this.idUsuario != null && !this.idUsuario.equals(other.idUsuario))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Usuario[ idUsuario=" + idUsuario + "
]";
    }

}
```

Arquivo: ServletProduto.java

```
package cadastroee.servlets;
```

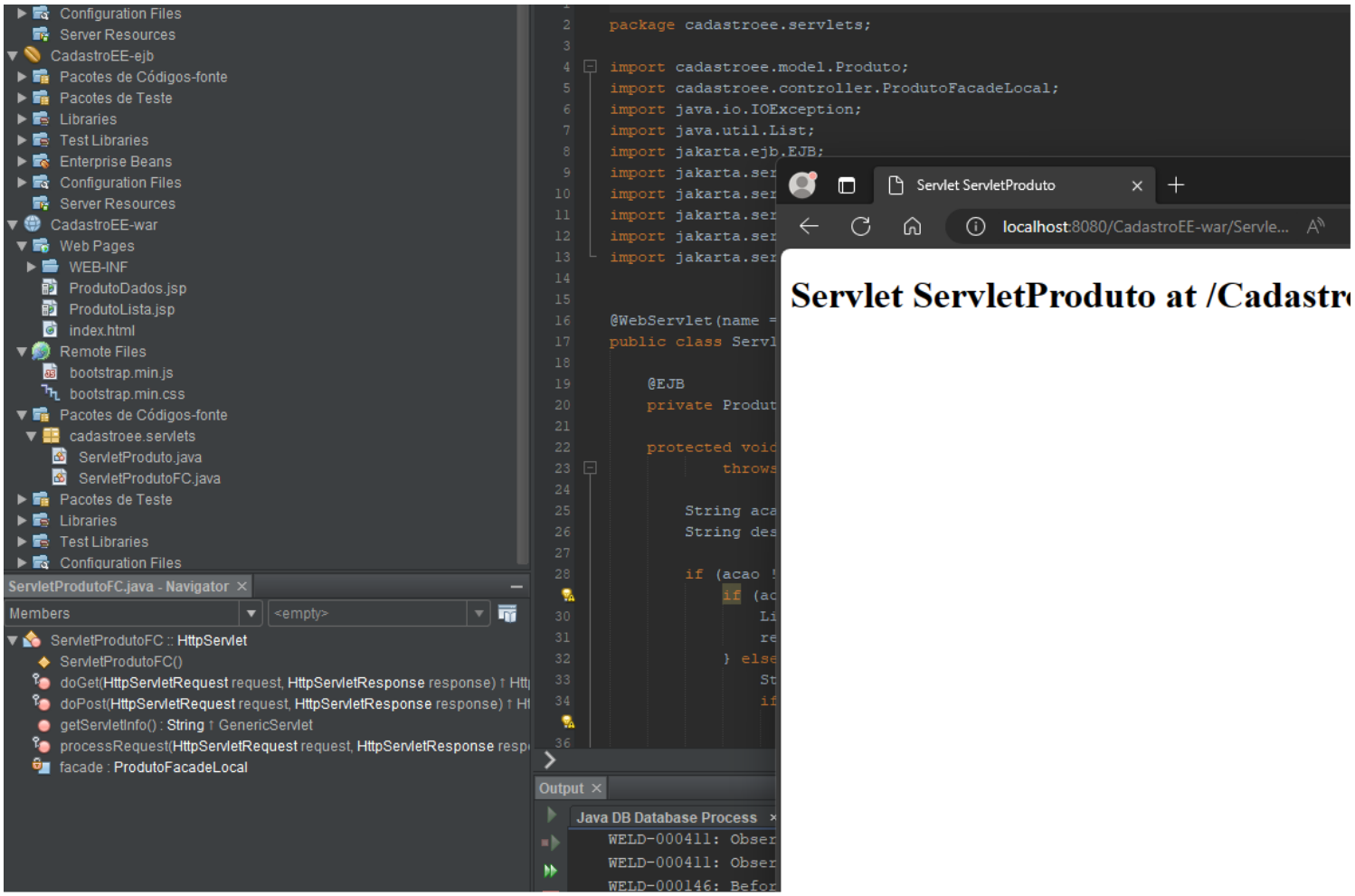
```java
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;


public class ServletProduto extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        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>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet
methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
```

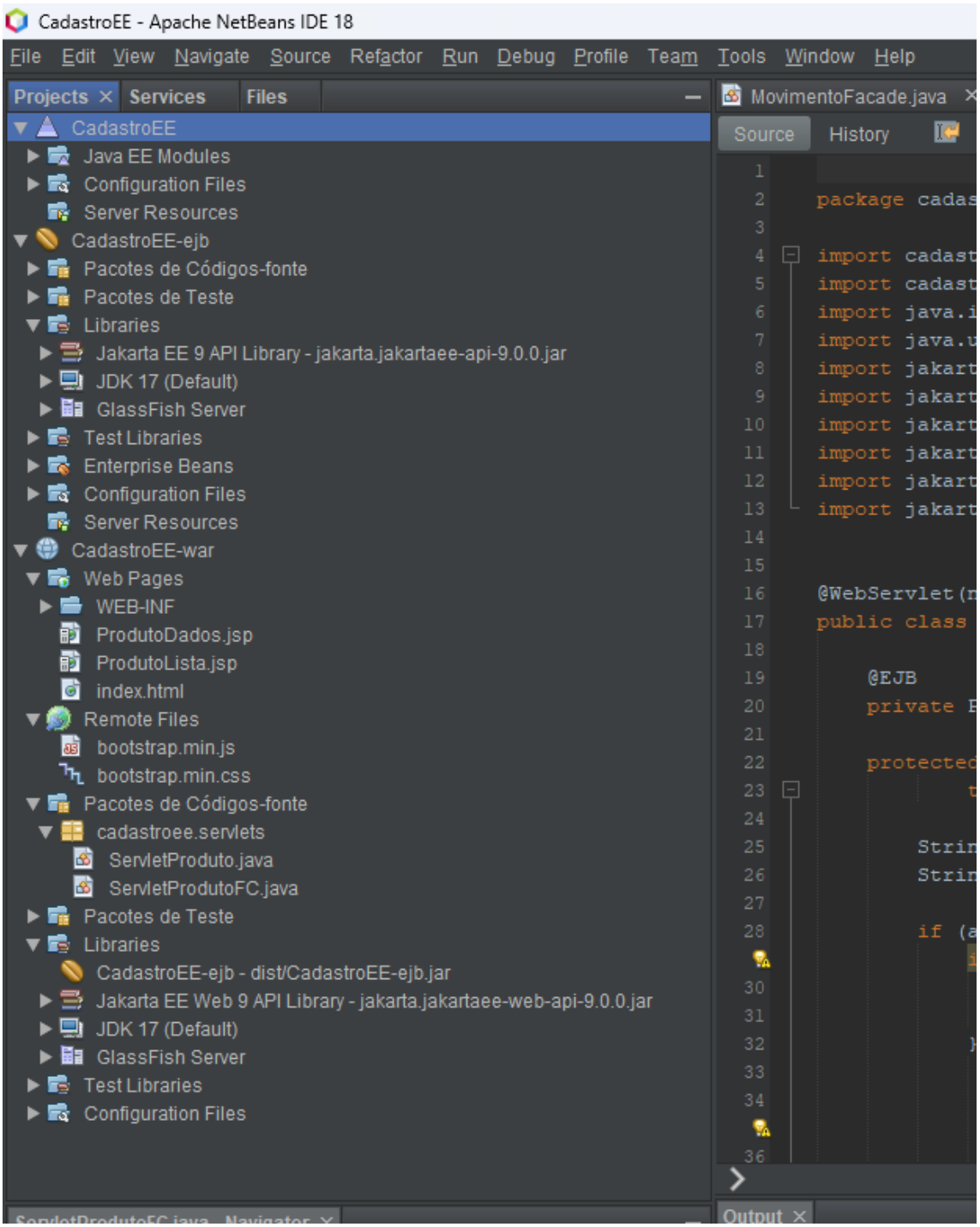
```
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

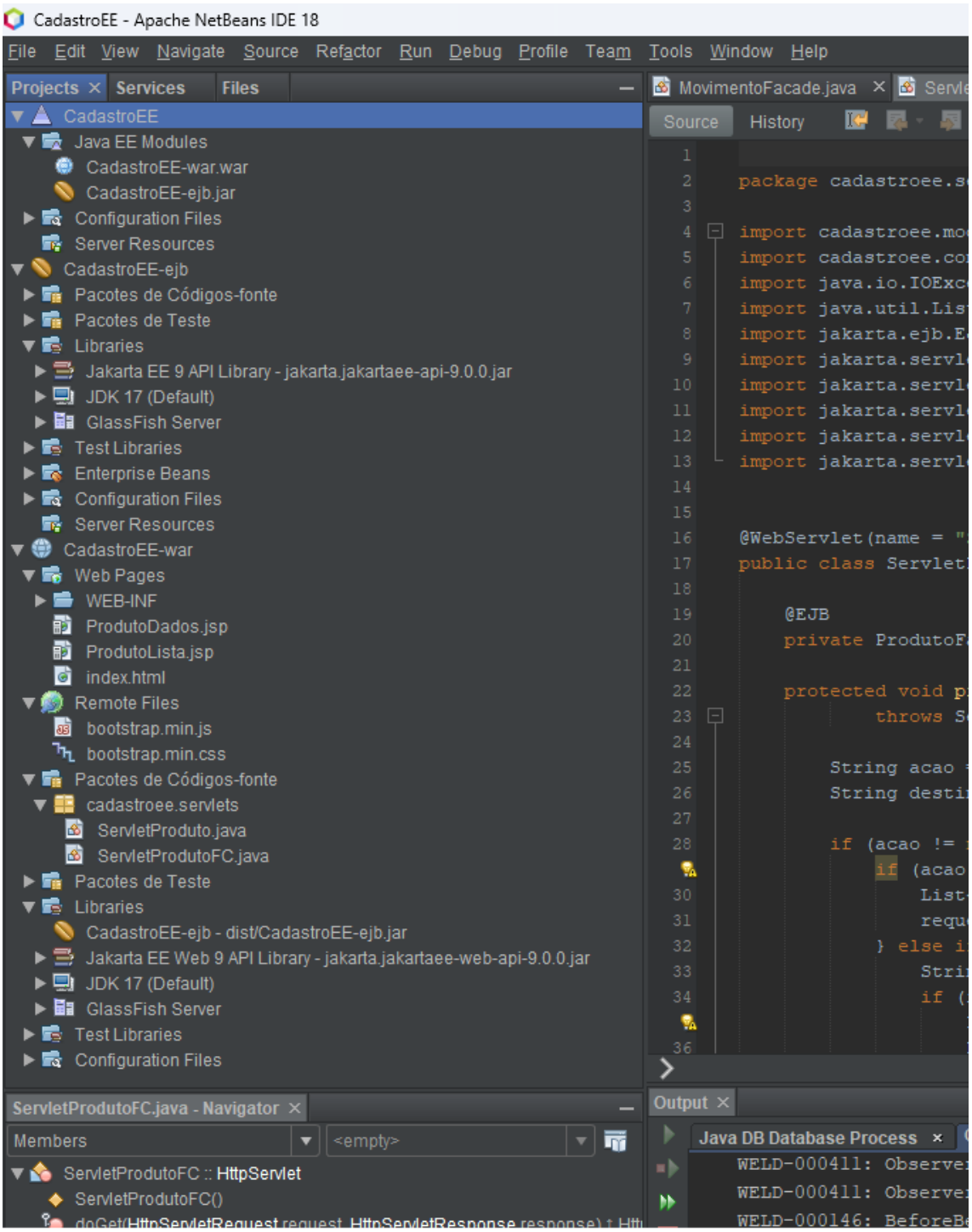
Arquivo: persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="3.0"
  xmlns="https://jakarta.ee/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence
https://jakarta.ee/xml/ns/persistence/persistence_3_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 DA EXECUÇÃO DOS CÓDIGOS:




CadastroEE - Apache NetBeans IDE 18
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Services
Files
CadastroEE
Java EE Modules
Configuration Files
Server Resources
CadastroEE-ejb
Pacotes de Códigos-fonte
Pacotes de Teste
Libraries
Jakarta EE 9 API Library - jakarta.jakartaee-api-9.0.0.jar
JDK 17 (Default)
GlassFish Server
Test Libraries
Enterprise Beans
Configuration Files
Server Resources
CadastroEE-war
Web Pages
WEB-INF
ProdutoDados.jsp
ProdutoLista.jsp
index.html
Remote Files
bootstrap.min.js
bootstrap.min.css
Pacotes de Códigos-fonte
cadastroee.servlets
ServletProduto.java
ServletProdutoFC.java
Pacotes de Teste
Libraries
CadastroEE-ejb - dist/CadastroEE-ejb.jar
Jakarta EE Web 9 API Library - jakarta.jakartaee-web-api-9.0.0.jar
JDK 17 (Default)
GlassFish Server
Test Libraries
Configuration Files
MovimentoFacade.java
Source
History
package cadas
import cadast
import cadast
import java.i
import java.u
import jakart
import jakart
import jakart
import jakart
import jakart
import jakart
@WebServlet(n
public class
@EJB
private
protected
Strin
Strin
if (a
Output


CadastroEE - Apache NetBeans IDE 18
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files
CadastroEE
Java EE Modules
CadastroEE-war.war
CadastroEE-ejb.jar
Configuration Files
Server Resources
CadastroEE-ejb
Pacotes de Códigos-fonte
Pacotes de Teste
Libraries
Jakarta EE 9 API Library - jakarta.jakartaee-api-9.0.0.jar
JDK 17 (Default)
GlassFish Server
Test Libraries
Enterprise Beans
Configuration Files
Server Resources
CadastroEE-war
Web Pages
WEB-INF
ProdutoDados.jsp
ProdutoLista.jsp
index.html
Remote Files
bootstrap.min.js
bootstrap.min.css
Pacotes de Códigos-fonte
cadastroee.servlets
ServletProduto.java
ServletProdutoFC.java
Pacotes de Teste
Libraries
CadastroEE-ejb - dist/CadastroEE-ejb.jar
Jakarta EE Web 9 API Library - jakarta.jakartaee-web-api-9.0.0.jar
JDK 17 (Default)
GlassFish Server
Test Libraries
Configuration Files
MovimentoFacade.java
Source History
package cadastroee.s
import cadastroee.mo
import cadastroee.co
import java.io.IOExc
import java.util.Lis
import jakarta.ejb.E
import jakarta.servl
import jakarta.servl
import jakarta.servl
import jakarta.servl
import jakarta.servl
@WebServlet(name = "
public class Servlet
@EJB
private ProdutoF
protected void p
throws S
String acao
String desti
if (acao !=
if (acao
List
requ
} else i
Stri
if (
ServletProdutoFC.java - Navigator
Members
<empty>
ServletProdutoFC :: HttpServlet
ServletProdutoFC()
Output
Java DB Database Process
WELD-000411: Observe
WELD-000411: Observe
WELD-000146: BeforeB

ANÁLISE E CONCLUSÃO:

a. Como é organizado um projeto corporativo no NetBeans?

Um projeto corporativo Java EE é uma estrutura organizacional que abrange diversos módulos para construir aplicativos empresariais de alta qualidade.

1 - Módulo EJB: Contém componentes que gerenciam a lógica de negócios, permitindo transações, acesso a dados e operações complexas.

2 - Projeto WEB: Envolve recursos de interface de usuário, como páginas web, para interação com os usuários em aplicativos Java EE.

3 - Módulo JAR: Armazena classes reutilizáveis, compartilhadas entre partes do aplicativo ou outros aplicativos Java EE.

4 - Diretórios de Configuração: Mantém arquivos de configuração específicos, como "web.xml" e "META-INF", para configurar comportamentos do aplicativo.

5 - Servidores de Aplicativos Java EE: Plataformas que hospedam e gerenciam aplicativos Java EE, fornecendo recursos como segurança e transações.

6 - Gerenciamento de Dependências: Organiza bibliotecas externas e frameworks essenciais para garantir que as versões corretas estejam disponíveis durante o desenvolvimento e implantação.

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

As tecnologias JPA (Java Persistence API) e EJB (Enterprise JavaBeans) desempenham papéis essenciais na construção de aplicativos para a plataforma web no ambiente Java, principalmente em aplicativos corporativos.

1 - JPA (Java Persistence API): Abstrai o acesso a bancos de dados relacionais, permitindo que objetos Java sejam mapeados para tabelas de banco de dados, simplificando a persistência de dados em aplicativos web.

2 - EJB (Enterprise JavaBeans): Fornece componentes Java reutilizáveis para implementar lógica de negócios em aplicativos corporativos. Oferece recursos críticos, como transações distribuídas e segurança, facilitando o desenvolvimento de aplicativos

web empresariais.

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

O NetBeans, como IDE para desenvolvimento Java EE, impulsiona a produtividade com JPA e EJB:

1 - Integração completa e ambiente unificado para JPA e EJB simplifica o desenvolvimento.

2 - Assistentes e geração de código automático aceleram a criação de entidades JPA e EJBs.

3 - Depuração integrada facilita a identificação de problemas em aplicativos Java EE.

4 - Gerenciamento de implantação simplificado permite visualizar alterações instantaneamente.

5 - Suporte a bancos de dados e mapeamento de entidades simplificam a persistência de dados.

6 - Construção e gerenciamento de projetos Java EE são otimizados.

7 - Integração com servidores de aplicativos facilita a implantação e teste.

8 - Ferramentas de persistência auxiliam na modelagem visual de dados.

9 - Melhoria da eficiência do desenvolvimento Java EE.

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

Servlets são componentes Java que processam solicitações e respostas HTTP em aplicativos web. Eles executam lógica de negócios, interagem com bancos de dados e geram conteúdo dinâmico.

O NetBeans oferece suporte à construção de Servlets em projetos web de várias maneiras:

1 - Assistentes de Criação: Fornecem modelos para criar Servlets rapidamente.

2 - Mapeamento de URL: Permite configurar como os Servlets respondem a URLs.

3 - Depuração Integrada: Facilita a identificação e correção de problemas nos Servlets.

4 - Integração com Servidores: Suporta implantação direta em servidores Java EE.

5 - Gerenciamento de Dependências: Simplifica o gerenciamento de bibliotecas Servlet.

6 - Editor de Código: Fornece recursos avançados para edição de código Servlet.

7 - Modelagem Visual: Possibilita criar interfaces web visualmente para interagir com Servlets.

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 em Java EE é essencial para a construção de aplicativos empresariais. Isso é facilitado por meio de injeção de dependência, onde os Servlets podem acessar Session Beans por meio de anotações. Os Session Beans podem fornecer interfaces locais para otimizar o desempenho, e o contêiner EJB gerencia o ciclo de vida desses beans, tornando-os reutilizáveis. Os Servlets podem chamar diretamente os métodos dos Session Beans para executar operações de negócios, e a comunicação ocorre de forma eficiente e dentro do contexto transacional, garantindo a consistência dos dados. Isso simplifica o desenvolvimento de aplicativos Java EE e permite que Servlets e Session Beans trabalhem juntos para oferecer funcionalidades empresariais robustas.

2º Procedimento – Alimentando a Base

CÓDIGOS SOLICITADOS NO ROTEIRO DE AULA:

Arquivo: ServetProdutoFC.java

```
package cadastroee.servlets;

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


@WebServlet(name = "ServletProdutoFC", urlPatterns =
{"/ServletProdutoFC"})
public class ServletProdutoFC extends HttpServlet {

    @EJB
    private ProdutoFacadeLocal facade;

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {

        String acao = request.getParameter("acao");
        String destino = "ProdutoLista.jsp"; // Página padrão de
destino

        if (acao != null) {
            if (acao.equals("listar")) {
                List<Produto> produtos = facade.findAll(); //
Recuperando todos os produtos
                request.setAttribute("produtos", produtos);
            } else if (acao.equals("formAlterar")) {
                String idProdutoStr =
request.getParameter("idProduto");
                if (idProdutoStr != null) {
                    Integer idProduto =
Integer.parseInt(idProdutoStr);
                    Produto produto = facade.find(idProduto); //
Recuperando um produto pelo ID
                    request.setAttribute("produto", produto);
                    destino = "ProdutoDados.jsp";
                }
            } else if (acao.equals("formIncluir")) {
                // Ação para exibir o formulário de inclusão
                destino = "ProdutoDados.jsp";
            } else if (acao.equals("excluir")) {
                String idProdutoStr =
```

```
request.getParameter("idProduto");
                if (idProdutoStr != null) {
                    Integer idProduto =
Integer.parseInt(idProdutoStr);
                    Produto produto = facade.find(idProduto); //
Recuperando um produto pelo ID
                    if (produto != null) {
                        facade.remove(produto);
                    }
                    // Recarregando a lista após a exclusão
                    List<Produto> produtos = facade.findAll(); //
Recuperando todos os produtos
                    request.setAttribute("produtos", produtos);
                }
            } else if (acao.equals("alterar") ||
acao.equals("incluir")) {
                // Recuperando os parâmetros do formulário
                String idProdutoStr =
request.getParameter("idProduto");
                String nome = request.getParameter("nome");
                String quantidadeStr =
request.getParameter("quantidade");
                String precoVendaStr =
request.getParameter("precoVenda");

                // Valores padrão
                Integer idProduto = null;
                Integer quantidade = null;
                Float precoVenda = null;

                // Convertendo valores se estiverem disponíveis
                if (idProdutoStr != null && !idProdutoStr.isEmpty()) {
                    idProduto = Integer.parseInt(idProdutoStr);
                }
                if (quantidadeStr != null && !quantidadeStr.isEmpty())
{
                    quantidade = Integer.parseInt(quantidadeStr);
                }
                if (precoVendaStr != null && !precoVendaStr.isEmpty())
{
                    precoVenda = Float.parseFloat(precoVendaStr);
                }

                // Criando ou atualizando o produto
                Produto produto;
                if (acao.equals("alterar")) {
                    produto = facade.find(idProduto); // Recuperando
um produto pelo ID
                    produto.setNome(nome);
                    produto.setQuantidade(quantidade);
                    produto.setPrecoVenda(precoVenda);
                } else {
                    produto = new Produto(idProduto, nome, quantidade,
precoVenda);
                }
                facade.edit(produto); // Inserindo ou atualizando um
produto

                // Recarregando a lista após a inclusão ou atualização
                List<Produto> produtos = facade.findAll(); //
Recuperando todos os produtos
                request.setAttribute("produtos", produtos);
```

```
            }
        }

        // Redirecionamento para o destino apropriado
        request.getRequestDispatcher(destino).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 "ServletProdutoFC";
    }
}
```

Arquivo: ProdutoDados.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Cadastro de Produto</title>
    <!-- Inclua o link para o Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap
.min.css">

    <!-- Inclua o link para o Bootstrap JavaScript (copiado do
Bootstrap CDN) -->
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.m
in.js"></script>
</head>
<body class="container"> <!-- Adicione a classe 'container' ao body --
>
    <h1>Cadastro de Produto</h1>

    <form action="ServletProdutoFC" method="post" class="form"> <!--
Adicione a classe 'form' ao formulário -->
        <input type="hidden" name="acao" value="${empty produto ?
'incluir' : 'alterar'}">
        <c:if test="${not empty produto}">
            <input type="hidden" name="idProduto"
value="${produto.idProduto}">
```

```
        </c:if>

        <div class="mb-3"> <!-- Encapsule cada par label / input em
div com classe 'mb-3' -->
            <label for="nome" class="form-label">Nome:</label> <!--
Adicione a classe 'form-label' em cada label -->
            <input type="text" id="nome" name="nome" value="${empty
produto ? '' : produto.nome}" required class="form-control"> <!--
Adicione a classe 'form-control' em cada input -->
        </div>

        <div class="mb-3"> <!-- Encapsule cada par label / input em
div com classe 'mb-3' -->
            <label for="quantidade" class="form-
label">Quantidade:</label> <!-- Adicione a classe 'form-label' em cada
label -->
            <input type="number" id="quantidade" name="quantidade"
value="${empty produto ? '' : produto.quantidade}" required
class="form-control"> <!-- Adicione a classe 'form-control' em cada
input -->
        </div>

        <div class="mb-3"> <!-- Encapsule cada par label / input em
div com classe 'mb-3' -->
            <label for="precoVenda" class="form-label">Preço de
Venda:</label> <!-- Adicione a classe 'form-label' em cada label -->
            <input type="number" id="precoVenda" name="precoVenda"
value="${empty produto ? '' : produto.precoVenda}" required
class="form-control"> <!-- Adicione a classe 'form-control' em cada
input -->
        </div>

        <input type="submit" value="${empty produto ? 'Incluir' :
'Alterar'} Produto" class="btn btn-primary"> <!-- Adicione as classes
'btn' e 'btn-primary' ao botão -->
    </form>
</body>
</html>
```

Arquivo: ProdutoLista.jsp

```
<%--
    Document   : ProdutoLista
    Created on : 8 de set. de 2023
    Author     :
--%>

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%@ page import="java.util.List" %>
<%@ page import="cadastroee.model.Produto" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title class="h3">Lista de Produtos</title>
    <!-- Inclua o link para o Bootstrap CSS -->
    <link rel="stylesheet"
```

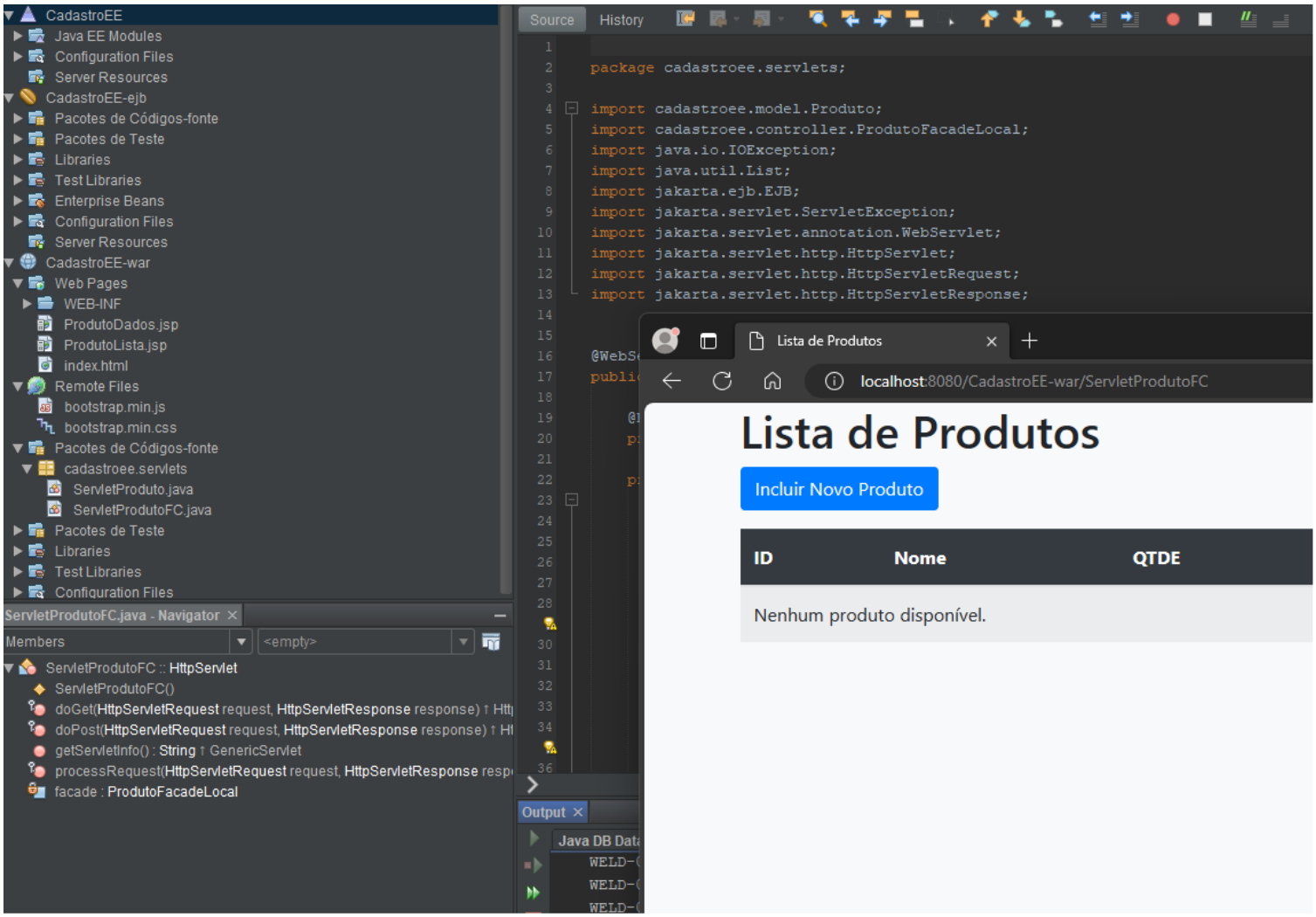
```
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap
.min.css">

    <!-- Inclua o link para o Bootstrap JavaScript (copiado do
Bootstrap CDN) -->
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.m
in.js"></script>
</head>
<body class="bg-light"> <!-- Adicione a classe 'container' ao body -->
    <div class="container">
    <h1>Lista de Produtos</h1>

    <!-- Link para abrir o formulário de inclusão -->
    <a href="ServletProdutoFC?acao=formIncluir" class="btn btn-primary
mb-3">Incluir Novo Produto</a> <!-- Adicione as classes 'btn', 'btn-
primary' e 'm-2' -->

    <table class="table table-striped "> <!-- Adicione as classes
'table' e 'table-striped' à tabela -->
        <thead class="table-dark"> <!-- Adicione a classe 'table-dark'
ao thead -->
            <tr>
                <th>ID</th>
                <th>Nome</th>
                <th>QTDE</th>
                <th>Preço</th>
                <th>Ações</th>
            </tr>
        </thead>
        <tbody>
        <%
            List<Produto> produtos = (List<Produto>)
request.getAttribute("produtos");
            if (produtos != null && !produtos.isEmpty()) {
                for (Produto produto : produtos) {
        %>
        <tr>
            <td><%= produto.getIdProduto() %></td>
            <td><%= produto.getNome() %></td>
            <td><%= produto.getQuantidade() %></td>
            <td><%= produto.getPrecoVenda() %></td>
            <td>
                <!-- Link para ação de alteração -->
                <a
href="ServletProdutoFC?acao=formAlterar&idProduto=<%=
produto.getIdProduto() %>" class="btn btn-primary btn-sm">Alterar</a>
<!-- Adicione as classes 'btn', 'btn-primary' e 'btn-sm' -->
                <!-- Link para ação de exclusão -->
                <a href="ServletProdutoFC?acao=excluir&idProduto=<%=
produto.getIdProduto() %>" class="btn btn-danger btn-sm">Excluir</a>
<!-- Adicione as classes 'btn', 'btn-danger' e 'btn-sm' -->
            </td>
        </tr>
        <%
                }
            } else {
        %>
        <tr>
            <td colspan="5">Nenhum produto disponível.</td>
        </tr>
        <%
```

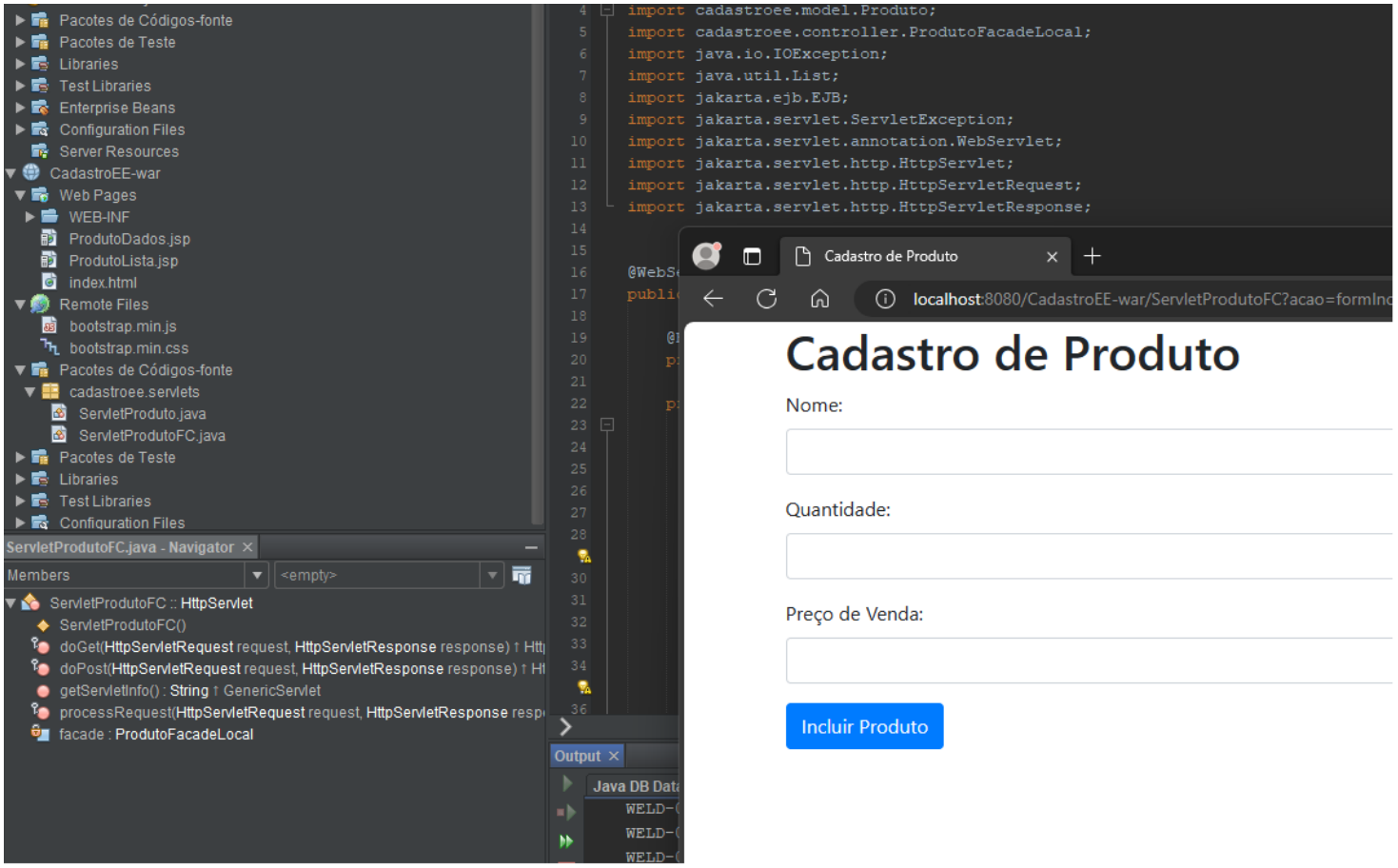
```
            }
        %>
        </tbody>
    </table>
</div>
</body>
</html>
```

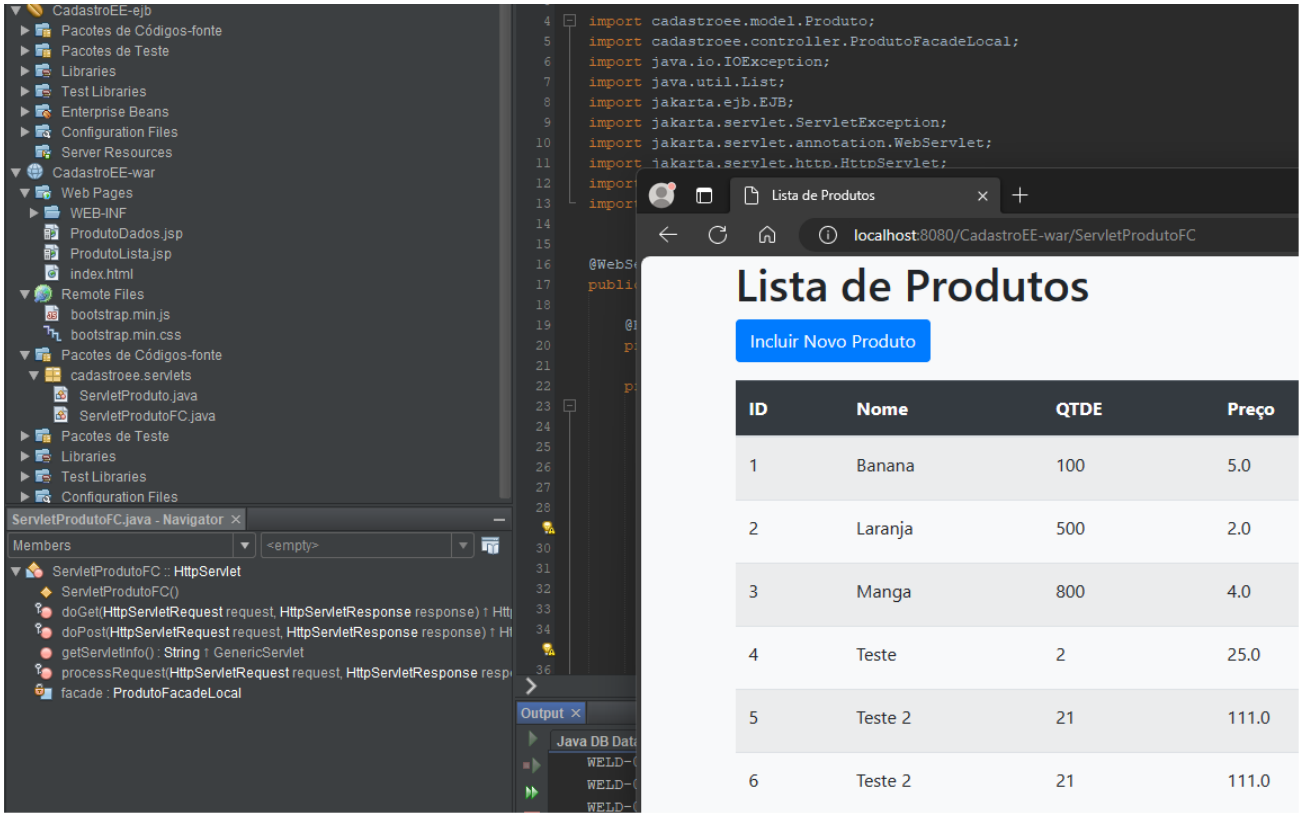
Arquivo: index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Minha Página</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
<body>
    <div>
        <p><a href="http://localhost:8080/CadastroEE-
war/ServletProdutoFC?acao=listar">Visualizar a Lista de
Produtos</a></p>
    </div>
</body>
</html>
```

RESULTADOS DA EXECUÇÃO DOS CÓDIGOS:




Pacotes de Códigos-fonte
Pacotes de Teste
Libraries
Test Libraries
Enterprise Beans
Configuration Files
Server Resources
CadastroEE-war
Web Pages
WEB-INF
ProdutoDados.jsp
ProdutoLista.jsp
index.html
Remote Files
bootstrap.min.js
bootstrap.min.css
Pacotes de Códigos-fonte
cadastroee.servlets
ServletProduto.java
ServletProdutoFC.java
Pacotes de Teste
Libraries
Test Libraries
Configuration Files
ServletProdutoFC.java - Navigator
Members
<empty>
ServletProdutoFC :: HttpServlet
ServletProdutoFC()
doGet(HttpServletRequest request, HttpServletResponse response) ↑ Htt
doPost(HttpServletRequest request, HttpServletResponse response) ↑ Ht
getServletInfo() : String ↑ GenericServlet
processRequest(HttpServletRequest request, HttpServletResponse resp
facade : ProdutoFacadeLocal
import cadastroee.model.Produto;
import cadastroee.controller.ProdutoFacadeLocal;
import java.io.IOException;
import java.util.List;
import jakarta.ejb.EJB;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
Output
Java DB Data
WELD-(
WELD-(
WELD-(
Cadastro de Produto
localhost:8080/CadastroEE-war/ServletProdutoFC?acao=formInc
Cadastro de Produto
Nome:
Quantidade:
Preço de Venda:
Incluir Produto



ANÁLISE E CONCLUSÃO:

a. Como o framework Bootstrap é utilizado?

O Bootstrap é um framework amplamente utilizado para estilizar e criar interfaces web responsivas. Ele é incorporado em projetos web adicionando os arquivos CSS e JavaScript do Bootstrap ao código-fonte. Através do uso de classes CSS específicas fornecidas pelo Bootstrap, você pode aplicar estilos aos elementos HTML do seu site. O Bootstrap oferece um sistema de grade flexível que ajuda a organizar o layout da página, permitindo que você crie designs personalizados. Além disso, o framework disponibiliza componentes prontos para uso, como botões, formulários e barras de navegação, que podem ser facilmente adicionados ao seu código HTML com o atributo "class". Isso simplifica o desenvolvimento de páginas web interativas e atraentes, enquanto garante que o site seja responsivo e compatível com uma variedade de dispositivos e navegadores.

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

o Bootstrap adota uma abordagem de design que permite que você mantenha a estrutura semântica do HTML enquanto aplica estilos e layouts por meio de classes CSS. Isso promove a independência estrutural, facilita a manutenção e personalização, e ajuda a criar sites acessíveis e eficientes.

c. Qual a relação entre o Boostrap e a responsividade da página?

O Bootstrap desempenha um papel crucial na responsividade das páginas web. Ele oferece um sistema de grade flexível e classes CSS responsivas que permitem criar layouts que se adaptam automaticamente a diferentes tamanhos de tela, como dispositivos móveis, tablets e desktops. Além disso, os componentes prontos para uso do Bootstrap são projetados para serem responsivos, garantindo que o conteúdo seja exibido de forma agradável em qualquer dispositivo.

CONCLUSÃO FINAL

Em conclusão, um projeto corporativo Java EE representa uma estrutura robusta para o desenvolvimento de aplicativos empresariais escaláveis e seguros. O uso de Servlets como componentes web permite a criação de interfaces dinâmicas e a manipulação de solicitações HTTP.

A IDE NetBeans oferece um ambiente integrado para simplificar o desenvolvimento, incluindo suporte para Servlets e Session Beans, componentes fundamentais para a lógica de negócios e a interação com bancos de dados.

Os Session Beans do EJB enriquecem o Java EE com componentes de negócios que facilitam a criação de aplicativos corporativos, oferecendo transações distribuídas e segurança. Além disso, o Bootstrap é uma ferramenta valiosa para criar interfaces web responsivas e atraentes, garantindo que a experiência do usuário seja consistente em diversos dispositivos.

Ao integrar esses elementos em um projeto Java EE, os desenvolvedores podem construir aplicativos empresariais de alta qualidade, alinhados com as melhores práticas e preparados para enfrentar desafios empresariais complexos, garantindo um desenvolvimento eficiente e uma experiência do usuário de alta qualidade.