Implementação de sistema cadastral com interface Web, baseado nas tecnologias de Servlets, JPA e JEE.

Leonardo Schaffer Mota – 2022.05.09098-1

Polo centro – Santo André – SP
Vamos integrar sistemas– 2023.2 – 2023.4

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.

1º Procedimento – Camadas de Persistência e Controle

Classe AbstractFacade:

```java
package cadastroee.controller;

import java.util.List;
import jakarta.persistence.EntityManager;

/**
 *
 * @author leosc
 */
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(o: entity);
    }

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

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

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

    public List<T> findAll() {
        jakarta.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(slctn: cq.from(type:entityClass));
        return getEntityManager().createQuery(cq).getResultList();
    }

    public List<T> findRange(int[] range) {
        jakarta.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(slctn: cq.from(type: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(type:entityClass);
            cq.select(slctn:getEntityManager().getCriteriaBuilder().count(exprsn:rt));
            jakarta.persistence.Query q = getEntityManager().createQuery(cq);
            return ((Long) q.getSingleResult()).intValue();
        }

    }
```

Classe MovimentosFacade:

```
package cadastroee.controller;

import cadastroee.model.Movimentos;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author leosc
 */
@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(entityClass:Movimentos.class);
    }

}
```

Classe MovimentosFacadeLocal:

```
package cadastroee.controller;

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

/**
 *
 * @author leosc
 */
@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();

}
```

Classe PessoasFacade:

```java
package cadastroee.controller;

import cadastroee.model.Pessoas;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author leosc
 */
@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(entityClass: Pessoas.class);
    }

}
```

Classe PessoasFacadeLocal:

```java
package cadastroee.controller;

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

/**
 *
 * @author leosc
 */
@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();

}
```

Classe PessoasFisicasFacade:

```java
package cadastroee.controller;

import cadastroee.model.PessoasFisicas;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author leosc
 */
@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(entityClass: PessoasFisicas.class);
    }

}
```

Classe PessoasFisicasFacadeLocal:

```
package cadastroee.controller;

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

/**
 *
 * @author leosc
 */
@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();

}
```

Classe PessoasJuridicasFacade:

```
package cadastroee.controller;

import cadastroee.model.PessoasJuridicas;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author leosc
 */
@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(entityClass: PessoasJuridicas.class);
    }

}
```

Classe PessoasJuridicasFacadeLocal:

```java
package cadastroee.controller;

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

/**
 *
 * @author leosc
 */
@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();

}
```

Classe ProdutosFacade:

```java
package cadastroee.controller;

import cadastroee.model.Produtos;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author leosc
 */
@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(entityClass: Produtos.class);
    }

}
```

Classe ProdutosFacadeLocal:

```java
package cadastroee.controller;

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

/**
 *
 * @author leosc
 */
@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();

}
```

Classe UsuariosFacade:

```java
package cadastroee.controller;

import cadastroee.model.Usuarios;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author leosc
 */
@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(entityClass: Usuarios.class);
    }

}
```

Classe UsuariosFacadeLocal:

```java
package cadastroee.controller;

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

/**
 *
 * @author leosc
 */
@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();

}
```

Classe Movimentos:

```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.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.Table;

/**
 *
 * @author leosc
 */
@Entity
@Table(name = "Movimentos")
@NamedQueries({
    @NamedQuery(name = "Movimentos.findAll", query = "SELECT m FROM Movimentos m"),
    @NamedQuery(name = "Movimentos.findByIdMovimentos", query = "SELECT m FROM Movimentos m WHERE m.idMovimentos = :idMovimentos"),
```

```java
@NamedQuery(name = "Movimentos.findByIdMovimentos", query = "SELECT m FROM Movimentos m WHERE m.idMovimentos = :idMovimentos"),
@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.findByPrecoUnitario", query = "SELECT m FROM Movimentos m WHERE m.precoUnitario = :precoUnitario")})
public class Movimentos implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "idMovimentos")
    private Integer idMovimentos;
    @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 validati
    @Column(name = "PrecoUnitario")
    private BigDecimal precoUnitario;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa")
    @ManyToOne
    private Pessoas idPessoa;
    @JoinColumn(name = "idProduto", referencedColumnName = "idProduto")
    @ManyToOne
    private Produtos idProduto;
    @JoinColumn(name = "idUsuario", referencedColumnName = "idUsuario")
    @ManyToOne
    private Usuarios idUsuario;

    public Movimentos() {
    }

    public Movimentos(Integer idMovimentos) {
        this.idMovimentos = idMovimentos;
    }

    public Integer getIdMovimentos() {
        return idMovimentos;
    }

    public void setIdMovimentos(Integer idMovimentos) {
        this.idMovimentos = idMovimentos;
    }

    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 getPrecoUnitario() {
        return precoUnitario;
    }

    public void setPrecoUnitario(BigDecimal precoUnitario) {
        this.precoUnitario = precoUnitario;
    }

    public Pessoas getIdPessoa() {
        return idPessoa;
    }

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

    public Produtos getIdProduto() {
        return idProduto;
    }

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

    public Usuarios getIdUsuario() {
        return idUsuario;
    }

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

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idMovimentos != null ? idMovimentos.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 Movimentos)) {
            return false;
        }
```

```
        Movimentos other = (Movimentos) object;
        if ((this.idMovimentos == null && other.idMovimentos != null) || (this.idMovimentos != null && !this.idMovimentos.equals(obj: otH
            return false;
        }
        return true;
    }

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

}
```

Classe Pessoas:

```
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;

/**
 *
 * @author leosc
 */
@Entity
@Table(name = "Pessoas")
@NamedQueries({
    @NamedQuery(name = "Pessoas.findAll", query = "SELECT p FROM Pessoas p"),
    @NamedQuery(name = "Pessoas.findByIdPessoa", query = "SELECT p FROM Pessoas p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "Pessoas.findByNome", query = "SELECT p FROM Pessoas p WHERE p.nome = :nome"),
    @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"),
    @NamedQuery(name = "Pessoas.findByLogradouro", query = "SELECT p FROM Pessoas p WHERE p.logradouro = :logradouro")})
public class Pessoas 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 = "cidade")
    private String cidade;
    @Column(name = "estado")
    private String estado;
    @Column(name = "telefone")
    private String telefone;
    @Column(name = "email")
    private String email;
    @Column(name = "logradouro")
    private String logradouro;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoas")
    private PessoasJuridicas pessoasJuridicas;
    @OneToMany(mappedBy = "idPessoa")
    private Collection<Movimentos> movimentosCollection;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoas")
    private PessoasFisicas pessoasFisicas;

    public Pessoas() {
    }

    public Pessoas(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 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 String getLogradouro() {
        return logradouro;
    }

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

    public PessoasJuridicas getPessoasJuridicas() {
        return pessoasJuridicas;
    }

    public void setPessoasJuridicas(PessoasJuridicas pessoasJuridicas) {
        this.pessoasJuridicas = pessoasJuridicas;
    }

    public Collection<Movimentos> getMovimentosCollection() {
        return movimentosCollection;
    }

    public void setMovimentosCollection(Collection<Movimentos> movimentosCollection) {
        this.movimentosCollection = movimentosCollection;
    }

    public PessoasFisicas getPessoasFisicas() {
        return pessoasFisicas;
    }

    public void setPessoasFisicas(PessoasFisicas pessoasFisicas) {
        this.pessoasFisicas = pessoasFisicas;
    }

    @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 Pessoas)) {
            return false;
        }
        Pessoas other = (Pessoas) object;
        if ((this.idPessoa == null && other.idPessoa != null) || (this.idPessoa != null && !this.idPessoa.equals(obj: other.idPessoa))) {
            return false;
        }
        return true;
    }

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

}
```

Classe PessoasFisicas:

```
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;

```

```java
/**
 *
 * @author leosc
 */
@Entity
@Table(name = "PessoasFisicas")
@NamedQueries({
    @NamedQuery(name = "PessoasFisicas.findAll", query = "SELECT p FROM PessoasFisicas p"),
    @NamedQuery(name = "PessoasFisicas.findByIdPFisica", query = "SELECT p FROM PessoasFisicas p WHERE p.idPFisica = :idPFisica"),
    @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)
    @Column(name = "idPFisica")
    private Integer idPFisica;
    @Column(name = "cpf")
    private String cpf;
    @JoinColumn(name = "idPFisica", referencedColumnName = "idPessoa", insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoas pessoas;

    public PessoasFisicas() {
    }

    public PessoasFisicas(Integer idPFisica) {
        this.idPFisica = idPFisica;
    }

    public Integer getIdPFisica() {
        return idPFisica;
    }

    public void setIdPFisica(Integer idPFisica) {
        this.idPFisica = idPFisica;
    }

    public String getCpf() {
        return cpf;
    }

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

    public Pessoas getPessoas() {
        return pessoas;
    }

    public void setPessoas(Pessoas pessoas) {
        this.pessoas = pessoas;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPFisica != null ? idPFisica.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 PessoasFisicas)) {
            return false;
        }
        PessoasFisicas other = (PessoasFisicas) object;
        if ((this.idPFisica == null && other.idPFisica != null) || (this.idPFisica != null && !this.idPFisica.equals(obj: other.idPFisica
            return false;
        }
        return true;
    }

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

}
```

Classe PessoasJuridicas:

```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;

/**
 *
 * @author leosc
 */
```

```java
@Entity
@Table(name = "PessoasJuridicas")
@NamedQueries({
    @NamedQuery(name = "PessoasJuridicas.findAll", query = "SELECT p FROM PessoasJuridicas p"),
    @NamedQuery(name = "PessoasJuridicas.findByIdPJuridica", query = "SELECT p FROM PessoasJuridicas p WHERE p.idPJuridica = :idPJuridica
    @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)
    @Column(name = "idPJuridica")
    private Integer idPJuridica;
    @Column(name = "cnpj")
    private String cnpj;
    @JoinColumn(name = "idPJuridica", referencedColumnName = "idPessoa", insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoas pessoas;

    public PessoasJuridicas() {
    }

    public PessoasJuridicas(Integer idPJuridica) {
        this.idPJuridica = idPJuridica;
    }

    public Integer getIdPJuridica() {
        return idPJuridica;
    }

    public void setIdPJuridica(Integer idPJuridica) {
        this.idPJuridica = idPJuridica;
    }

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    public Pessoas getPessoas() {
        return pessoas;
    }

    public void setPessoas(Pessoas pessoas) {
        this.pessoas = pessoas;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPJuridica != null ? idPJuridica.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 PessoasJuridicas)) {
            return false;
        }
        PessoasJuridicas other = (PessoasJuridicas) object;
        if ((this.idPJuridica == null && other.idPJuridica != null) || (this.idPJuridica != null && !this.idPJuridica.equals(obj: other.i
            return false;
        }
        return true;
    }

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

}
```

Classe Produtos:

```java
package cadastroee.model;

import java.io.Serializable;
import java.util.Collection;
import jakarta.persistence.Basic;
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.Table;

/**
 *
 * @author leosc
 */
@Entity
@Table(name = "Produtos")
@NamedQueries({
```

```java
@NamedQuery(name = "Produtos.findAll", query = "SELECT p FROM Produtos p"),
@NamedQuery(name = "Produtos.findByIdProduto", query = "SELECT p FROM Produtos p WHERE p.idProduto = :idProduto"),
@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
    @Basic(optional = false)
    @Column(name = "idProduto")
    private Integer idProduto;
    @Column(name = "nome")
    private String nome;
    @Column(name = "quantidade")
    private Integer quantidade;
    // @Max(value=?)  @Min(value=?)//if you know range of your decimal fields consider using these annotations to enforce field validati
    @Column(name = "precoVenda")
    private float precoVenda;
    @OneToMany(mappedBy = "idProduto")
    private Collection<Movimentos> movimentosCollection;

    public Produtos() {
    }

    public Produtos(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;
    }

    public Collection<Movimentos> getMovimentosCollection() {
        return movimentosCollection;
    }

    public void setMovimentosCollection(Collection<Movimentos> movimentosCollection) {
        this.movimentosCollection = movimentosCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idProduto != null ? idProduto.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 Produtos)) {
            return false;
        }
        Produtos other = (Produtos) object;
        if ((this.idProduto == null && other.idProduto != null) || (this.idProduto != null && !this.idProduto.equals(obj: other.idProduto
            return false;
        }
        return true;
    }

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

}
```

Classe Ususarios:

```java
package cadastroee.model;

import java.io.Serializable;
import java.util.Collection;
import jakarta.persistence.Basic;
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.Table;

/**
 *
 * @author leosc
 */
@Entity
@Table(name = "Usuarios")
@NamedQueries({
    @NamedQuery(name = "Usuarios.findAll", query = "SELECT u FROM Usuarios u"),
    @NamedQuery(name = "Usuarios.findByIdUsuario", query = "SELECT u FROM Usuarios u WHERE u.idUsuario = :idUsuario"),
    @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
    @Basic(optional = false)
    @Column(name = "idUsuario")
    private Integer idUsuario;
    @Column(name = "login")
    private String login;
    @Column(name = "senha")
    private String senha;
    @OneToMany(mappedBy = "idUsuario")
    private Collection<Movimentos> movimentosCollection;

    public Usuarios() {
    }

    public Usuarios(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;
    }

    public Collection<Movimentos> getMovimentosCollection() {
        return movimentosCollection;
    }

    public void setMovimentosCollection(Collection<Movimentos> movimentosCollection) {
        this.movimentosCollection = movimentosCollection;
    }

    @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 Usuarios)) {
            return false;
        }
        Usuarios other = (Usuarios) object;
        if ((this.idUsuario == null && other.idUsuario != null) || (this.idUsuario != null && !this.idUsuario.equals(obj: other.idUsuari
            return false;
        }
        return true;
    }

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

ServletProduto.java:

```java
package cadastroee.servlets;

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;
import jakarta.ejb.EJB;
import cadastroee.controller.ProdutosFacadeLocal;
import cadastroee.model.Produtos;
import java.util.List;

/**
 *
 * @author leosc
 */
public class ServletProduto extends HttpServlet {

    @EJB
    ProdutosFacadeLocal facade;
    /**
     * 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(string: "text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println(x: "<!DOCTYPE html>");
            out.println(x: "<html>");
            out.println(x: "<head>");
            out.println(x: "<title>Servlet ServletProduto</title>");
            out.println(x: "</head>");
            out.println(x: "<body>");
            out.println("<h1>Servlet ServletProduto at " + request.getContextPath() + "</h1>");
            out.println(x: "</body>");
            out.println(x: "</html>");
        }
    }
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit t
    /**
     * 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 {

            List<Produtos> produtos = facade.findAll();

            response.setContentType(string: "text/html;charset=UTF-8");
            try (PrintWriter out = response.getWriter()) {
                out.println(x: "<html>");
                out.println(x: "<head>");
                out.println(x: "<title>Servlet ServletProduto</title>");
                out.println(x: "<style>");
                out.println(x: "ul { list-style: none; padding-left: 0; }");
                out.println(x: "</style>");
                out.println(x: "</head>");
                out.println(x: "<body>");
                out.println("<h1>Servlet ServletProduto at " + request.getContextPath() + "</h1>");
                out.println(x: "<ul>");
                for (Produtos produto : produtos) {
                    out.println("<li>" + produto.getNome() + "</li>");
                }
                out.println(x: "</ul>");
                out.println(x: "</body>");
                out.println(x: "</html>");
            }
        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
     */
```

```java
    @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>

}
```

Resultado da execução:

a) Como é organizado um projeto corporativo no NetBeans?

Um projeto corporativo no NetBeans é organizado em módulos interconectados que representam diferentes componentes da aplicação. Geralmente, há um módulo principal (EAR) que agrupa outros módulos (EJBs, WARs) para camadas de persistência, lógica de negócios e interfaces web. Isso facilita o desenvolvimento modular e a implantação em servidores de aplicativos.

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 no desenvolvimento de aplicativos web Java. O JPA oferece um padrão para mapear objetos Java para tabelas de banco de dados, facilitando a persistência de dados. O EJB é um componente que oferece recursos como gerenciamento transacional e acesso remoto, promovendo a escalabilidade e a reutilização de lógica de negócios em aplicações corporativas. Ambas as tecnologias contribuem para a construção eficiente e robusta de aplicativos na plataforma web Java.

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

O NetBeans oferece suporte integrado e ferramentas visuais para desenvolvimento com tecnologias JPA e EJB, simplificando a criação, implementação e gerenciamento dessas funcionalidades. Seus recursos, como

geração de código automático, mapeamento visual de entidades e depuração avançada, otimizam a produtividade do desenvolvedor ao lidar com complexidades de persistência de dados e lógica de negócios em aplicativos corporativos.

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, usados para criar dinamismo em aplicativos web. O NetBeans oferece suporte à construção de Servlets por meio de modelos e assistentes, simplificando sua criação, configuração e implantação em projetos web. Isso agiliza o desenvolvimento de interfaces interativas e dinâmicas para os usuários.

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 de injeção de dependência. O NetBeans facilita essa integração por meio de anotações e assistentes que permitem que Servlets acessem as interfaces dos Session Beans de maneira simples e eficiente, permitindo a execução de lógica de negócios encapsulada nos EJBs a partir das ações desencadeadas pelos Servlets.

2º Procedimento – Interface Cadastral com Servlet e JSPs

ServletProdutoFC.java:

```java
package cadastroee.servlets;

import java.io.IOException;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.RequestDispatcher;
import jakarta.ejb.EJB;
import cadastroee.controller.ProdutosFacadeLocal;
import cadastroee.model.Produtos;
import java.util.List;
/**
 *
 * @author leosc
 */
@WebServlet(name = "ServletProdutoFC", urlPatterns = {"/ServletProdutoFC"})
public class ServletProdutoFC extends HttpServlet {
    @EJB
    ProdutosFacadeLocal facade;
    /**
     * 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
     */
```

```java
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    }
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit t
    /**
     * 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 {
        String acao = request.getParameter(string: "acao");
        String destino = null;

        switch (acao) {
            case "listar":
                List<Produtos> produtos = facade.findAll();
                request.setAttribute(string: "produtos", o: produtos);
                destino = "ProdutoLista.jsp";
                break;
            case "formIncluir":
                destino = "ProdutoDados.jsp";
                break;
            case "formAlterar":
                int idAlterar = Integer.parseInt(s: request.getParameter(string: "id"));
                Produtos produtoAlterar = facade.find(id: idAlterar);
                request.setAttribute(string: "produto", o: produtoAlterar);
                destino = "ProdutoDados.jsp";
                break;
            case "excluir":
                int idExcluir = Integer.parseInt(s: request.getParameter(string: "id"));
                Produtos produtoExcluir = facade.find(id: idExcluir);
                facade.remove(produtos: produtoExcluir);
                List<Produtos> produtosExcluidos = facade.findAll();
                request.setAttribute(string: "produtos", o: produtosExcluidos);
                destino = "ProdutoLista.jsp";
                break;
            default:
                break;
        }
        RequestDispatcher rd = request.getRequestDispatcher(string: destino);
        rd.forward(sr: request, sr1: 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 {
        String acao = request.getParameter(string: "acao");
        List<Produtos> produtos = null;

        switch (acao) {
            case "incluir":
                String nome = request.getParameter(string: "nome");
                int quantidade = Integer.parseInt(s: request.getParameter(string: "quantidade"));
                float preco = Float.parseFloat(s: request.getParameter(string: "preco"));
                produtos = facade.findAll();
                int ultimoId = 0;
                Produtos ultimoProduto = produtos.get(produtos.size() - 1);
                ultimoId = ultimoProduto.getIdProduto();
                int id = ultimoId + 1;

                Produtos novoProduto = new Produtos();
                novoProduto.setNome(nome);
                novoProduto.setQuantidade(quantidade);
                novoProduto.setPrecoVenda(precoVenda: preco);
                novoProduto.setIdProduto(idProduto: id);

                facade.create(produtos: novoProduto);

                produtos = facade.findAll();
                request.setAttribute(string: "produtos", o: produtos);

                break;

            case "alterar":
                int idAlterar = Integer.parseInt(s: request.getParameter(string: "id"));
                Produtos produtoExistente = facade.find(id: idAlterar);

                if (produtoExistente != null) {
                    String novoNome = request.getParameter(string: "nome");
                    String novaQuantidadeStr = request.getParameter(string: "quantidade");
                    String novoPrecoStr = request.getParameter(string: "preco");

                    int novaQuantidade =
                            (novaQuantidadeStr != null && !novaQuantidadeStr.isEmpty())
                            ? Integer.parseInt(s: novaQuantidadeStr) : produtoExistente.getQuantidade();
                    float novoPreco =
                            (novoPrecoStr != null && !novoPrecoStr.isEmpty())
                            ? Float.parseFloat(s: novoPrecoStr) : produtoExistente.getPrecoVenda();

                    produtoExistente.setNome(nome:novoNome);
                    produtoExistente.setQuantidade(quantidade:novaQuantidade);
                    produtoExistente.setPrecoVenda(precoVenda:novoPreco);
```

```
                    facade.edit(produtos: produtoExistente);
                    produtos = facade.findAll();
                    request.setAttribute(string: "produtos", o: produtos);
                } else {
                    produtos = facade.findAll();
                    request.setAttribute(string: "produtos", o: produtos);
                }
                break;
            }
            RequestDispatcher rd = request.getRequestDispatcher(string: "ProdutoLista.jsp");
            rd.forward(sr: request, sr1: response);
        }
    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

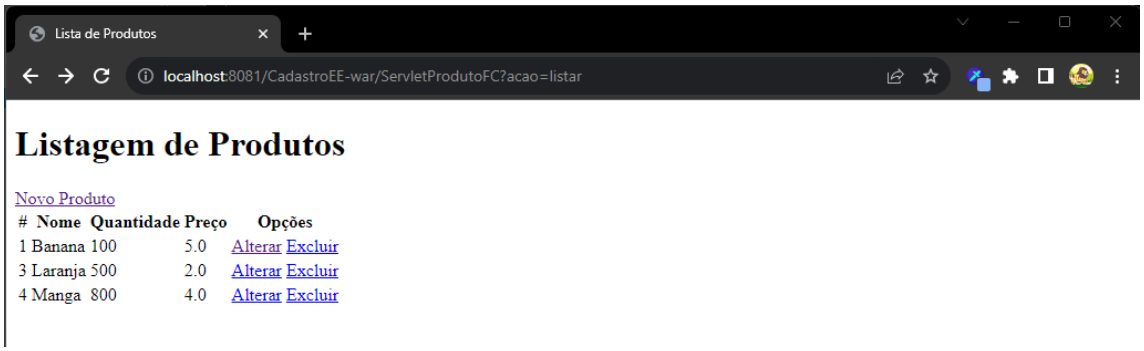
ProdutoList.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
            <title>Lista de Produtos</title>
        </head>
        <body class="">
            <h1>Listagem de Produtos</h1>
    <a class="" href="ServletProdutoFC?acao=formIncluir">Novo Produto</a>
    <table class="" >
        <tr class="">
            <th>#</th>
            <th>Nome</th>
            <th>Quantidade</th>
            <th>Preço</th>
            <th>Opções</th>
        </tr>
        <!-- Aqui virá a lista de produtos -->
        <%-- Loop para exibir a lista de produtos --%>
        <c:forEach items="${produtos}" var="produto">
    <tr>
        <td>${produto.idProduto}</td>
        <td>${produto.nome}</td>
        <td>${produto.quantidade}</td>
        <td>${produto.precoVenda}</td>
        <td>
            <a class="" href="ServletProdutoFC?acao=formAlterar&id=${produto.idProduto}">Alterar</a>
            <a class="" href="ServletProdutoFC?acao=excluir&id=${produto.idProduto}">Excluir</a>
        </td>
    </tr>
    </c:forEach>
  </table>
  </body>
</html>
```

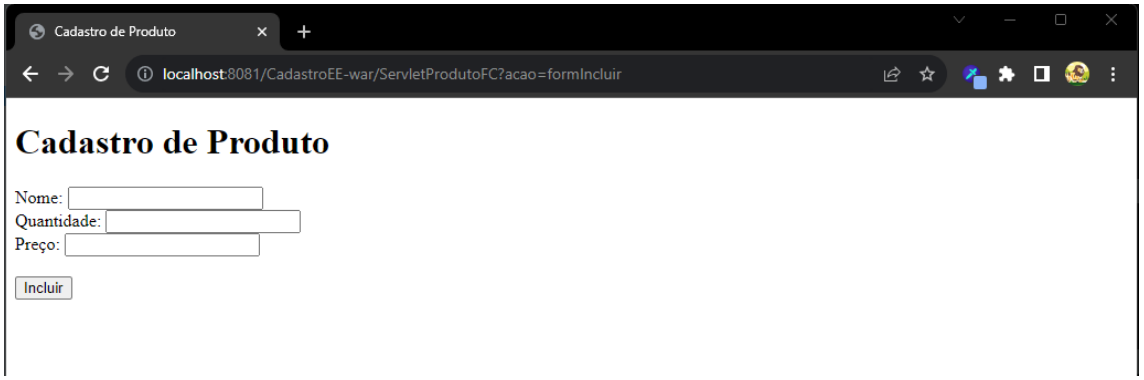
ProdutoDados.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Cadastro de Produto</title>
</head>
<body class="">
    <h1>${empty produto ? 'Cadastro de Produto' : 'Dados do produto'}</h1>
    <form class="" action="ServletProdutoFC" method="post">
        <input type="hidden" name="acao" value="${not empty produto ? 'alterar' : 'incluir'}">
        <c:if test="${not empty produto}">
            <input type="hidden" name="id" value="${produto.idProduto}">
        </c:if>
        <div class="">
            <label class="" for="nome">Nome:</label>
            <input class="" type="text" id="nome" name="nome" value="${not empty produto ? produto.nome : ''}">
        </div>
        <div class="">
            <label class="" for="quantidade">Quantidade:</label>
            <input class="" type="text" id="quantidade" name="quantidade" value="${not empty produto ? produto.quantidade : ''}"
        </div>
        <div class="">
            <label class="" for="preco">Preço:</label>
            <input class="" type="text" id="preco" name="preco" value="${not empty produto ? produto.precoVenda : ''}">
        </div>
            <br>
        <input class="" type="submit" value="${not empty produto ? 'Alterar' : 'Incluir'}">
    </form>
</body>
</html>
```

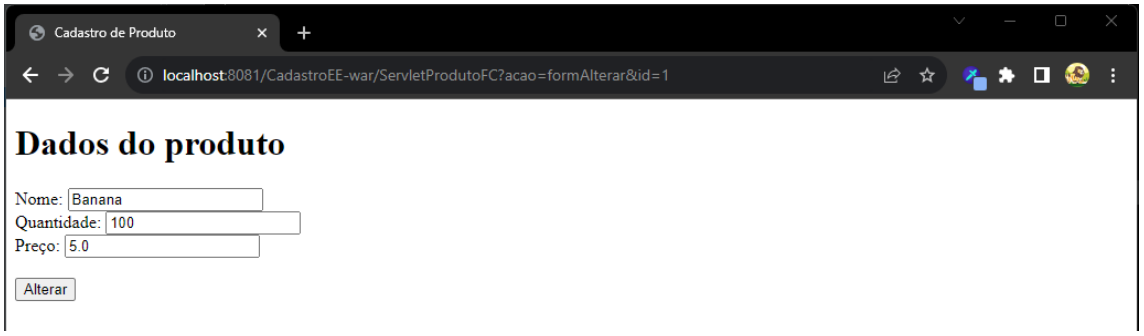
Resultados da execução:



Novo produto:



Alterar:



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

O padrão Front Controller é implementado em um aplicativo Web Java usando um servlet central que encaminha as requisições para controladores específicos com base em mapeamentos de URLs, permitindo uma melhor estruturação e organização das responsabilidades em uma arquitetura MVC.

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

Servlets e JSPs são componentes importantes no desenvolvimento web Java. Enquanto Servlets são mais programáticos e adequados para lógica de negócios, JSPs facilitam a criação de interfaces de usuário dinâmicas separando a lógica e

o design. Ambos desempenham papéis complementares na construção de aplicativos web robustos.

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?

Redirecionamento gera uma nova solicitação e pode ser mais lento, enquanto forward mantém a mesma solicitação e é mais rápido. Parâmetros passam dados entre cliente e servidor, e atributos compartilham dados entre componentes do servidor durante uma solicitação.

3º Procedimento – Melhorando o Design da Interface

ProdutoList.jsp:

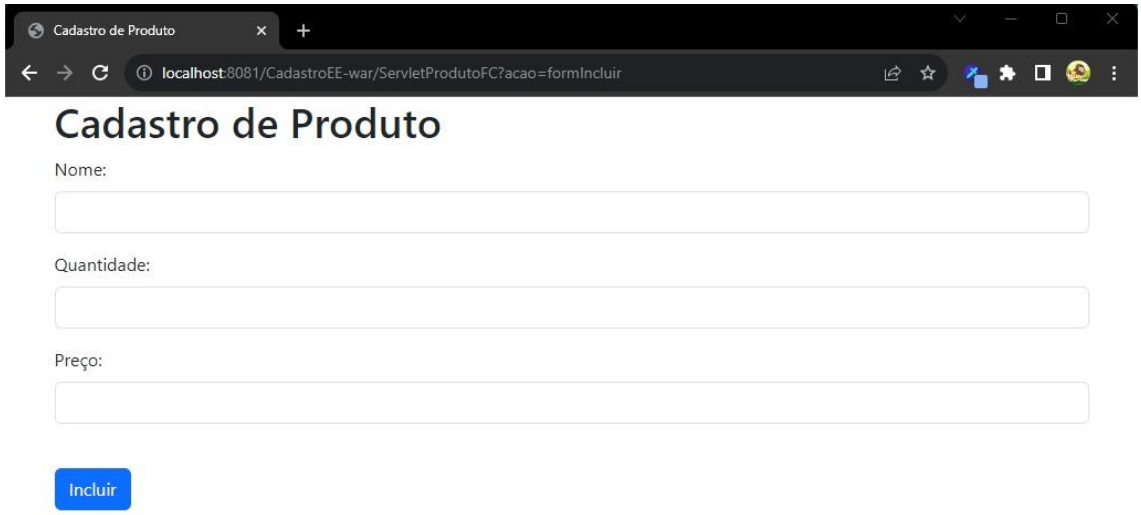
```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
            <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha
            <title>Lista de Produtos</title>
        </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" border="1">
        <tr class="table-dark">
            <th>#</th>
            <th>Nome</th>
            <th>Quantidade</th>
            <th>Preço</th>
            <th>Opções</th>
        </tr>
        <!-- Aqui virá a lista de produtos -->
        <%-- Loop para exibir a lista de produtos --%>
        <c:forEach items="${produtos}" var="produto">
    <tr>
        <td>${produto.idProduto}</td>
        <td>${produto.nome}</td>
        <td>${produto.quantidade}</td>
        <td>${produto.precoVenda}</td>
        <td>
            <a class="btn btn-primary  btn-sm" href="ServletProdutoFC?acao=formAlterar&id=${produto.idProduto}">Alterar</a>
            <a class="btn btn-danger  btn-sm" href="ServletProdutoFC?acao=excluir&id=${produto.idProduto}">Excluir</a>
        </td>
    </tr>
    </c:forEach>
    </table>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwvtgBNo3b
    </body>
</html>
```

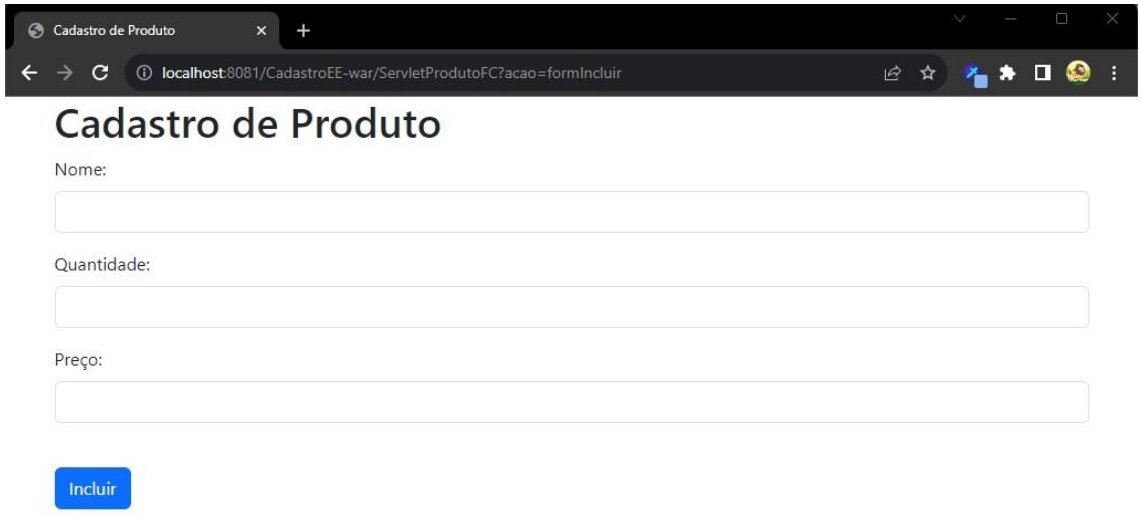
ProdutoDados.jsp:

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpV
    <title>Cadastro de Produto</title>
</head>
<body class="container">
    <h1>${empty produto ? 'Cadastro de Produto' : 'Dados do produto'}</h1>
    <form class="form" action="ServletProdutoFC" method="post">
        <input type="hidden" name="acao" value="${not empty produto ? 'alterar' : 'incluir'}">

        <c:if test="${not empty produto}">
            <input type="hidden" name="id" value="${produto.idProduto}">
        </c:if>
        <div class="mb-3">
            <label class="form-label" for="nome">Nome:</label>
            <input class="form-control" type="text" id="nome" name="nome" value="${not empty produto ? produto.nome : ''}">
        </div>
        <div class="mb-3">
            <label class="form-label" for="quantidade">Quantidade:</label>
            <input class="form-control" type="text" id="quantidade" name="quantidade" value="${not empty produto ? produto.quantidade : ''}">
        </div>
        <div class="mb-3">
            <label class="form-label" for="preco">Preço:</label>
            <input class="form-control" type="text" id="preco" name="preco" value="${not empty produto ? produto.precoVenda : ''}">
        </div>
            <br>
        <input class="btn  btn-primary" type="submit" value="${not empty produto ? 'Alterar' : 'Incluir'}">
    </form>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cq
</body>
</html>
```

Resultados da execução:





a) Como o framework Bootstrap é utilizado?

O framework Bootstrap é utilizado incluindo seus arquivos CSS e JavaScript nas páginas HTML do projeto. Isso é feito através de links ou referências a esses arquivos no cabeçalho do HTML. Com a estrutura básica do Bootstrap implementada, você pode aproveitar suas classes CSS para aplicar estilos predefinidos a elementos HTML, como botões, formulários e layouts de grade.

Além disso, o Bootstrap oferece componentes JavaScript interativos que podem ser usados para criar elementos como carrosséis, modais e abas. Com a combinação de suas classes e componentes, o Bootstrap simplifica o processo de desenvolvimento de interfaces de usuário modernas e responsivas.

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

O Bootstrap garante a independência estrutural do HTML ao separar a formatação visual do conteúdo estrutural. Ele utiliza classes CSS e componentes para estilizar elementos, permitindo que o HTML se concentre na estrutura e semântica. Isso resulta em códigos mais limpos, facilita a manutenção e possibilita alterações visuais sem afetar a estrutura do documento.

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

O Bootstrap é intimamente relacionado à responsividade da página, pois oferece uma grade responsiva e classes CSS que se adaptam a diferentes tamanhos de tela. Isso permite que os elementos se ajustem automaticamente, proporcionando uma experiência consistente e amigável em dispositivos variados, como desktops, tablets e smartphones. O Bootstrap facilita a criação de layouts flexíveis que respondem ao ambiente do usuário, garantindo uma experiência visualmente agradável em qualquer dispositivo.

Conclusão

Ao longo deste projeto, explorei e aplicação das configurações de conexão com bancos de dados, a criação de camadas de persistência e controle através de EJBs e JPA, bem como a implementação da arquitetura MVC. Ao utilizar o Front Controller, centralizei o controle das requisições, e as páginas JSP, estilizadas com Bootstrap, ofereceram uma experiência amigável.