Estácio - Mundo 3 - Missão Nível 4

Faculdade Estácio - Polo Centro – Belo Horizonte – MG

Curso: Desenvolvimento Full Stack.

Disciplina: Nível 4: Vamos integrar sistemas!

RPG0017.

Semestre Letivo: 3

Integrante: André Luiz Ferreira da Silva

Repositório: https://github.com/Andre-Luiz22/m3-n3

IDE: NetBeans

OBS.: As fotos do código foram tiradas no vscode com a extensão CodeSnap para facilitar, mais o codigo foi feito no netBeans como as dos resultados.

Título da Prática

Vamos integrar sistemas !

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

Objetivos da Prática

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

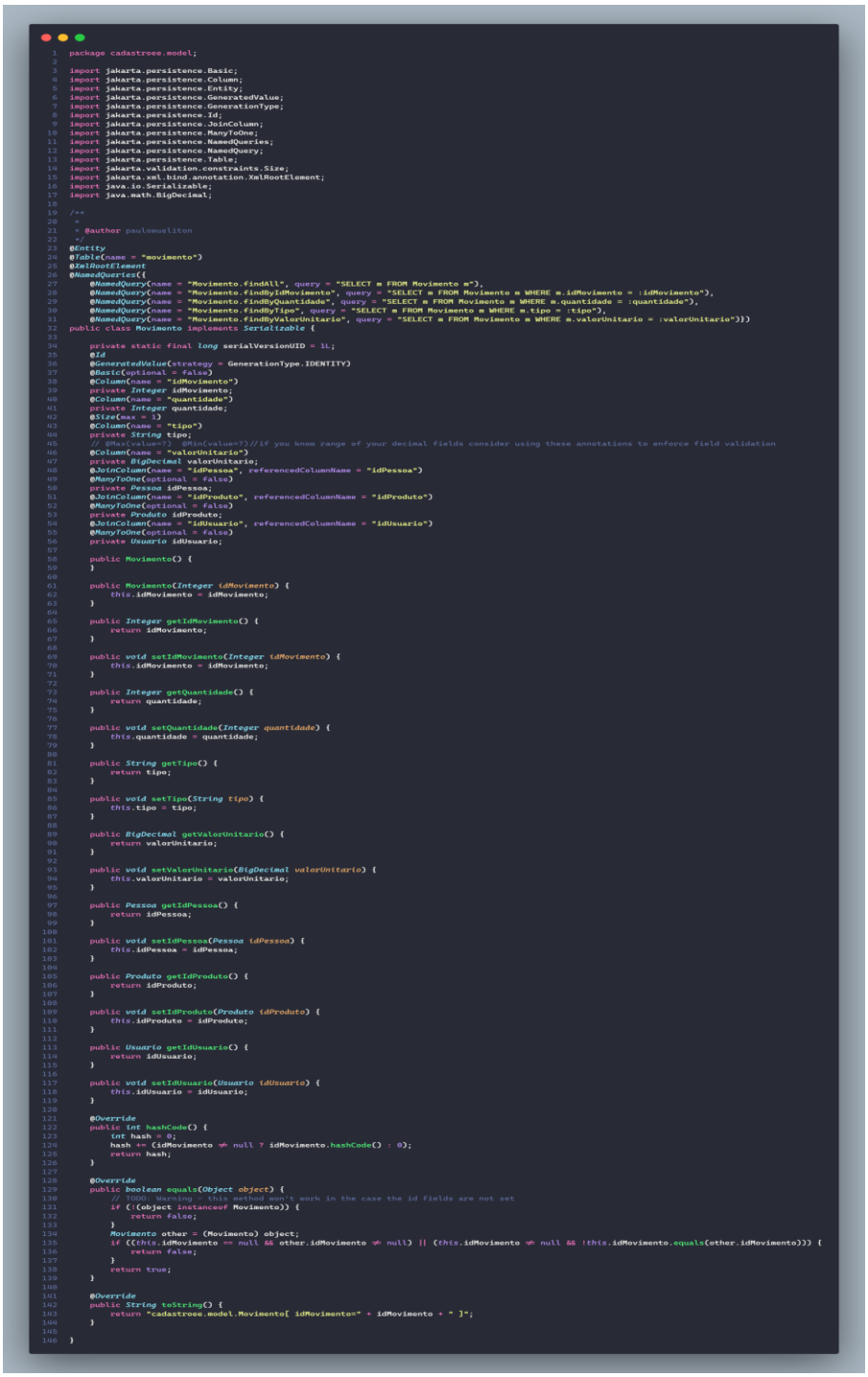
1º Procedimento | Camadas de Persistência e Controle

Arquivos do Projeto

CadastroEE-ejb

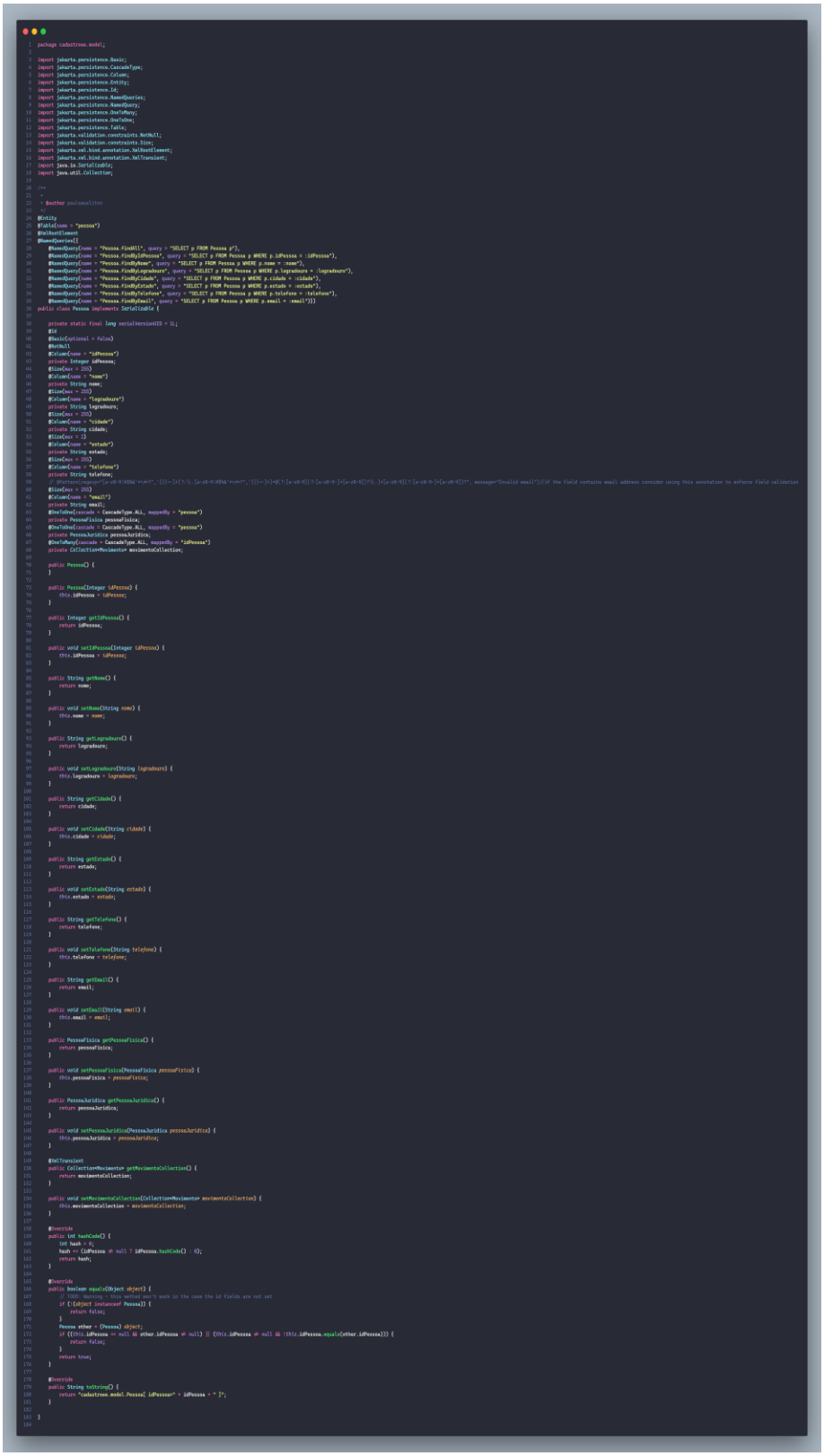
Cadastroee.model

Movimento



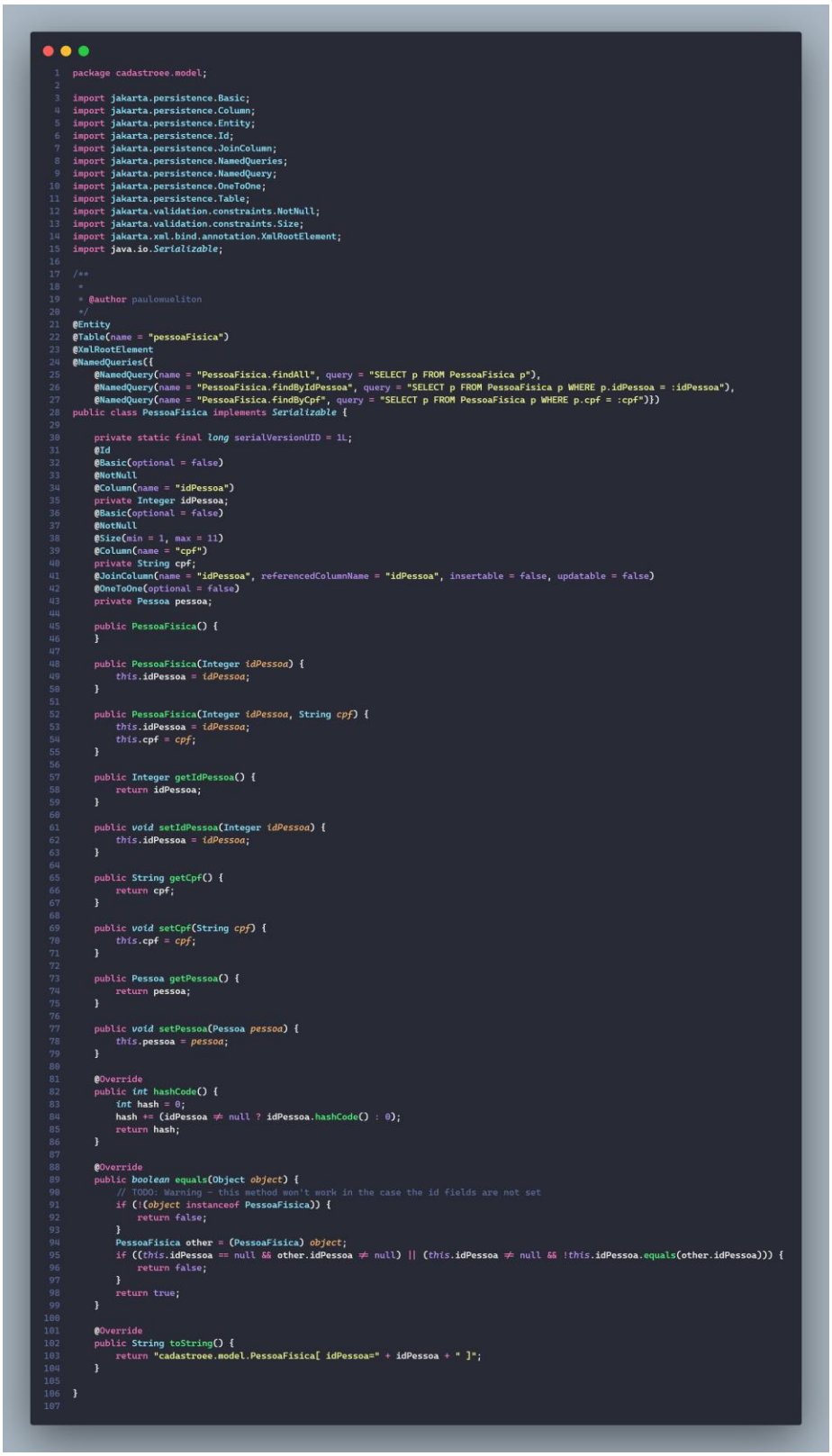
```java
package cadastroee.model;

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;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.Table;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;
import java.math.BigDecimal;

/**
 *
 * @author paulomueliton
 */
@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;
    @Size(max = 1)
    @Column(name = "tipo")
    private String 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;
    }

    public Integer getQuantidade() {
        return quantidade;
    }

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

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String 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 + " ]";
    }

}
```

Pessoa



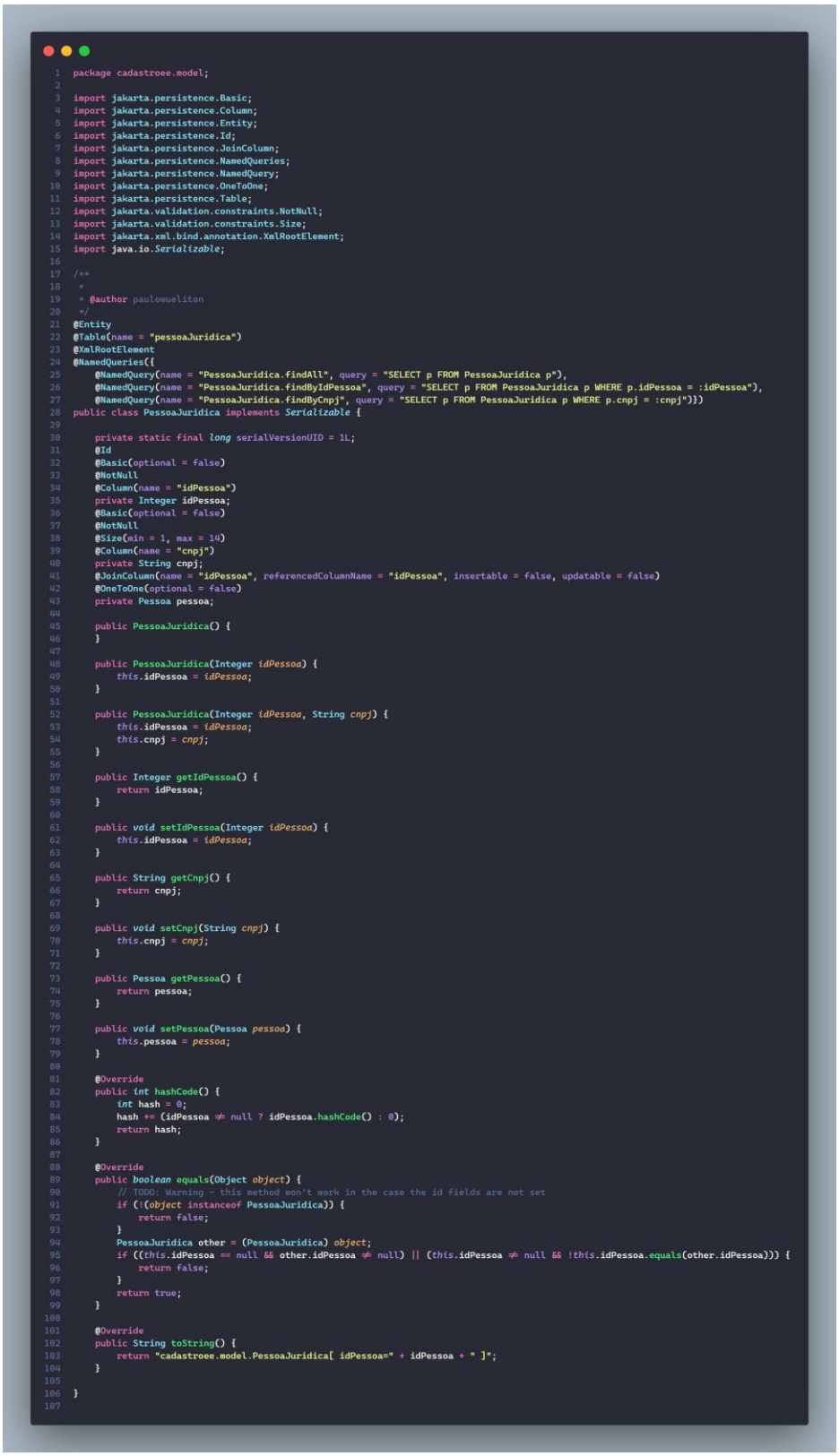
```java
package cadastroeo.model;

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.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;
import java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author paulonunlitos
 */
@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)
    @NotNull
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Size(max = 255)
    @Column(name = "nome")
    private String nome;
    @Size(max = 255)
    @Column(name = "logradouro")
    private String logradouro;
    @Size(max = 255)
    @Column(name = "cidade")
    private String cidade;
    @Size(max = 2)
    @Column(name = "estado")
    private String estado;
    @Size(max = 255)
    @Column(name = "telefone")
    private String telefone;
    // @Pattern(regexp="[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?", message="Invalid email")//if the field contains email address consider using this annotation to enforce field validation
    @Size(max = 255)
    @Column(name = "email")
    private String email;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaFisica pessoaFisica;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaJuridica pessoaJuridica;
    @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 PessoaFisica getPessoaFisica() {
        return pessoaFisica;
    }

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

    public PessoaJuridica getPessoaJuridica() {
        return pessoaJuridica;
    }

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

    @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) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        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 "cadastroeo.model.Pessoa[ idPessoa=" + idPessoa + " ]";
    }

}
```

Pessoa Fisica



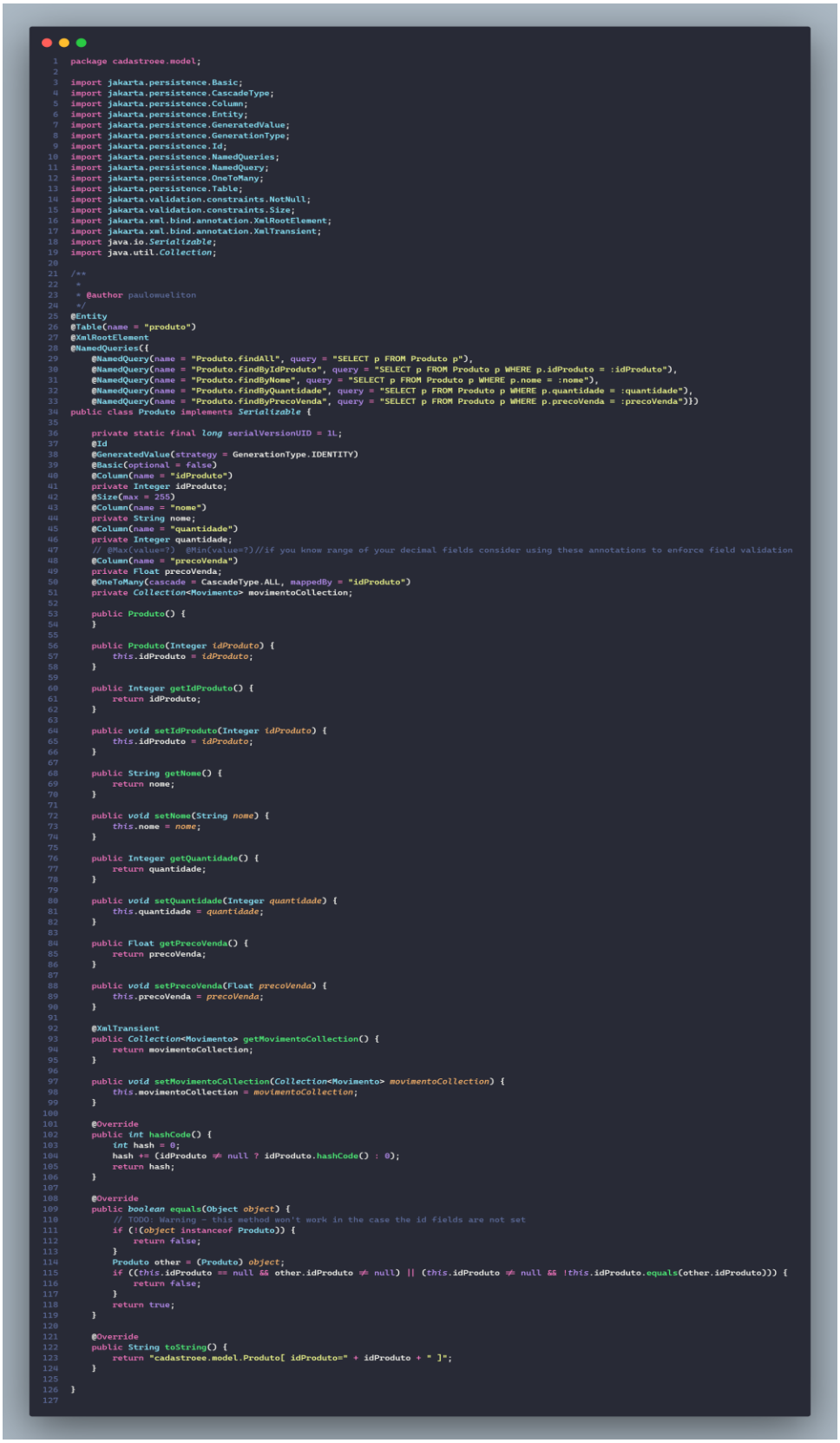
```java
package cadastroee.model;

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.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

/**
 *
 * @author paulomueliton
 */
@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)
    @NotNull
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 11)
    @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 PessoaFisica(Integer idPessoa, String cpf) {
        this.idPessoa = idPessoa;
        this.cpf = cpf;
    }

    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 + " ]";
    }
}

```

Pessoa Juridica



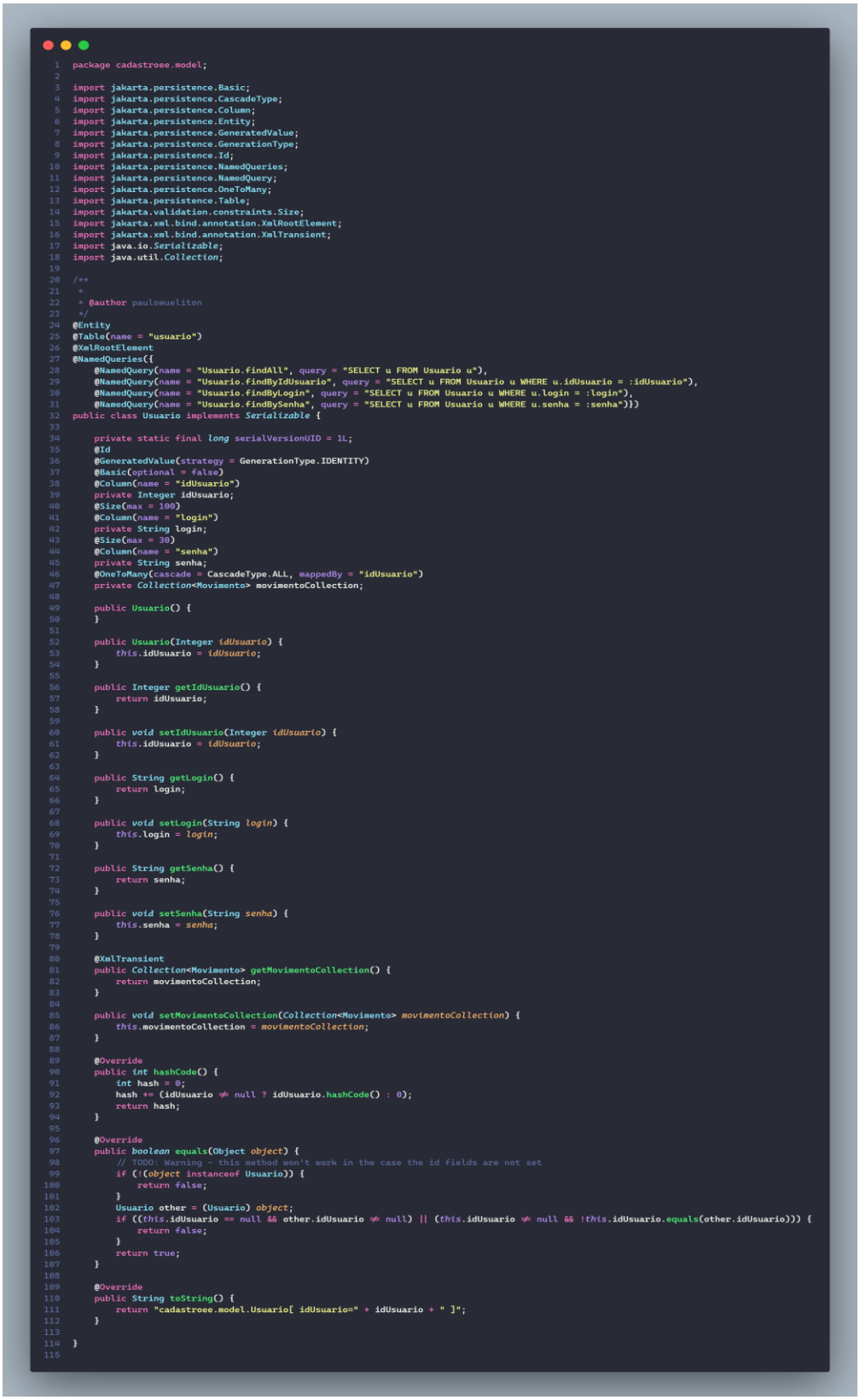
```java
package cadastroee.model;

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.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

/**
 *
 * @author paulomueliton
 */
@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)
    @NotNull
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 14)
    @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 PessoaJuridica(Integer idPessoa, String cnpj) {
        this.idPessoa = idPessoa;
        this.cnpj = cnpj;
    }

    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) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        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 + " ]";
    }

}
```

Produto



```java
package cadastroee.model;

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.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;
import java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author paulomueliton
 */
@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;
    @Size(max = 255)
    @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 validation
    @Column(name = "precoVenda")
    private Float precoVenda;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idProduto")
    private Collection<Movimento> movimentoCollection;

    public Produto() {
    }

    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) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        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 + " ]";
    }

}

```

Usuario



```java
package cadastroee.model;

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.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;
import java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author paulomueliton
 */
@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;
    @Size(max = 100)
    @Column(name = "login")
    private String login;
    @Size(max = 30)
    @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 + " ]";
    }

}

```

Persistence.xml

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

CadastroEE-war

Cadastroee.servlets

ServletProduto

```java
package cadastroee.servlets;

import cadastroee.controller.ProdutoFacadeLocal;
import jakarta.ejb.EJB;
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;

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

    @EJB
    ProdutoFacadeLocal 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("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("<ul>");
            this.facade.findAll().forEach(produto -> out.println("<li>"+ produto.getNome() +"</li>"));
            out.println("</ul>");
            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>

}
```

Execução

Analise e Conclusão

a) Como é organizado um projeto corporativo no NetBeans?

Um projeto corporativo é dividido em três camadas, utilizando o padrão MVC. Ao iniciar um projeto corporativo no NetBeans ele irá gerar três projetos, um projeto responsável pelo consumo e persistência de dados em um banco de dados, que seria a camada Model da aplicação, essa camada fornece todos os métodos para comunicação com o banco de dados; o segundo projeto EJB possui a responsabilidade de implementar as regras de negócios da aplicação, atuando como a camada Controller, além de fornecer e receber as solicitações da camada de visualização e por último a camada de View, que é responsável por fornecer toda a interface da aplicação para o cliente Web.

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

As duas tecnologias são cruciais para o desenvolvimento de aplicativos web no Java, o JPA permite abstrair a consulta e persistência dos dados em um banco de dados, permitindo o mapeamento objeto-relacional (ORM) em tabelas de bancos de dados relacionais, especificação de relacionamentos entre as tabelas com anotações Java e o consumo de diversos fornecedores de banco de dados com a mesma interface, e, o EJB fornece um conjunto pronto de componentes que agiliza o desenvolvimento de soluções, seu objetivo é fornecer todo o código de infraestrutura para que o desenvolvedor se concentre apenas no desenvolvimento das regras de negócios da aplicação.

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

O NetBeans fornece diversas ferramentas que agilizam e auxiliam na criação de projetos com tecnologias JPS e EJB, desde a criação do projeto, que possui uma estrutura que abstrai a infraestrutura necessária para o funcionamento dos componentes, além de fornecer ferramentas que permitem o mapeamento das entidades do banco de dados de forma automática, integração com diversos servidores que facilita o desenvolvimento, execução e depuração das aplicações. Todas essas ferramentas permitem que o desenvolvedor tenha foco apenas nas regras de negócios.

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

   Os Servlets são componentes que estendem as funcionalidades de servidores Web, ficam responsáveis por processar e responder as requisições HTTP. O NetBeans oferece suporte robusto para construção dos Servlets através de suas ferramentas. Na criação de um novo projeto é possível, através de seu assistente, criar e personalizar um servidor integrado ao projeto a partir de um modelo pré-definido, durante o processo de desenvolvimento ele fornece ferramentas de execução e depuração do servidor.

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

   A comunicação é realizada através da injeção de dependências, no Java EE os Session Beans podem ser injetados nos Servlets utilizando a anotação @EJB, desta forma é injetada uma referência ao Session Bean no momento da inicialização, que permite que o Servlet acesse os métodos do Session Bean diretamente, como se fossem métodos locais.

2º Procedimento | Interface cadastral com servlets e JSPs

CadastroEE-war

Cadastroee.servlets

ServletProdutoFC

```java
package cadastroee.servlets;

import cadastroee.controller.ProdutoFacadeLocal;
import cadastroee.model.Produto;
import jakarta.ejb.EJB;
import jakarta.servlet.RequestDispatcher;
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;

/**
 *
 * @author paulowueliton
 */
public class ServletProdutoFC extends HttpServlet {

    @EJB
    ProdutoFacadeLocal 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 {
        String acao = request.getParameter("acao");
        String destino = "ProdutoLista.jsp";

        if (acao == null) {
            acao = "listar";
        }

        switch (acao) {
            case "listar" -> {
                request.setAttribute("produtos", facade.findAll());
                break;
            }
            case "incluir" -> {
                Produto produto = this.getProduto(request);

                facade.create(produto);

                request.setAttribute("produtos", facade.findAll());

                break;
            }
            case "alterar" -> {
                Produto produto = this.getProduto(request);

                facade.edit(produto);

                request.setAttribute("produtos", facade.findAll());

                break;
            }
            case "excluir" -> {
                Integer id = Integer.valueOf(request.getParameter("id"));
                Produto produto = facade.find(id);
                facade.remove(produto);
                request.setAttribute("produtos", facade.findAll());
                break;
            }
            case "formIncluir" -> {
                destino = "ProdutoDados.jsp";
                break;

            }
            case "formAlterar" -> {
                destino = "ProdutoDados.jsp";
                Integer id;
                id = Integer.valueOf(request.getParameter("id"));
                request.setAttribute("produto", facade.find(id));
                break;
            }

        }

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

    private Produto getProduto(HttpServletRequest request) {
        String idParametro = request.getParameter("id");
        Integer id = Integer.valueOf(idParametro);
        String nome = request.getParameter("nome");
        Integer quantidade = Integer.valueOf(request.getParameter("quantidade"));
        Float precoVenda = Float.valueOf(request.getParameter("precoVenda"));

        Produto produto = new Produto();
        produto.setNome(nome);
        produto.setPrecoVenda(precoVenda);
        produto.setQuantidade(quantidade);

        if(idParametro != null) { produto.setIdProduto(id); }
        return produto;
    }

    // <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>
}
```

ProdutoLista.jsp

```jsp
<%--
    Document   : ProdutoLista
    Created on : 11 de mai. de 2024, 12:24:36
    Author     : andreLuiz
--%>

<%@page import="java.util.List"%>
<%@page import="cadastroee.model.Produto"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Listagem de Produtos</title>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>
    </head>
    <body class="container">
        <h1>Listagem de Produtos</h1>

        <a href="?acao=formIncluir" class="btn btn-primary m-2">Novo Produto</a>
        <table class="table table-striped">
            <thead class="table-dark">
                <th>#</th>
                <th>Nome</th>
                <th>Quantidade</th>
                <th>Preço de Venda</th>
                <th>Opções</th>
            </thead>
            <tbody>
                <% List<Produto> produtos = (List<Produto>) request.getAttribute("produtos"); %>
                <% for (Produto produto : produtos) { %>
                <tr>
                    <td><%= produto.getIdProduto() %></td>
                    <td><%= produto.getNome() %></td>
                    <td><%= produto.getQuantidade() %></td>
                    <td><%= produto.getPrecoVenda() %></td>
                    <td>
                        <a href="?acao=formAlterar&id=<%= produto.getIdProduto() %>" class="btn btn-primary btn-sm">Alterar</a>
                        <a href="?acao=excluir&id=<%= produto.getIdProduto() %>" class="btn btn-danger btn-sm">Excluir</a>
                    </td>
                </tr>
                <% } %>
            </tbody>
        </table>
    </body>
</html>

```

ProdutoDados.jsp

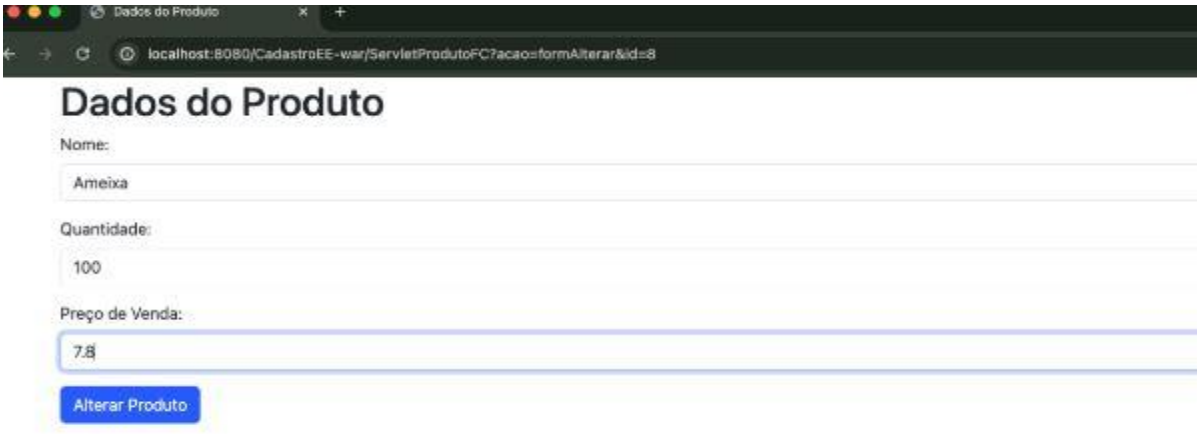
```jsp
<%--
    Document   : ProdutoDados
    Created on : 13 de mai. de 2024, 12:30:14
    Author     : andreLuiz
--%>

<%@page import="cadastroee.model.Produto"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<% String acao = "incluir"; %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title><%= acao == "incluir" ? "Adicionar" : "Editar" %> Produto</title>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>
    </head>
    <body class="container">
        <h1>Dados do Produto</h1>

        <% Produto produto = (Produto) request.getAttribute("produto"); %>

        <% String nome = ""; %>
        <% String quantidade = ""; %>
        <% String precoVenda = ""; %>

        <form action="ServletProdutoFC" method="POST" class="form">
            <% if(produto != null) { %>
                <% nome = produto.getNome(); %>
                <% quantidade = String.valueOf(produto.getQuantidade()); %>
                <% precoVenda = String.valueOf(produto.getPrecoVenda()); %>
                <% acao = "alterar"; %>
                <input type="hidden" name="id" value="<%= request.getParameter("id") %>" />
            <% } %>

            <div class="mb-3">
                <label for="nome" class="form-label">Nome: </label>
                <input id="nome" name="nome" value="<%= nome %>" class="form-control" autofocus />
            </div>

            <div class="mb-3">
                <label for="quantidade" class="form-label">Quantidade </label>
                <input type="number" min="1" id="quantidade" name="quantidade" value="<%= quantidade %>" class="form-control" />
            </div>

            <div class="mb-3">
                <label for="precoVenda" class="form-label">Preço de Venda </label>
                <input type="number" step="0.01" min="0" id="precoVenda" name="precoVenda" value="<%= precoVenda %>" class="form-control" />
            </div>

            <input type="hidden" name="acao" value="<%= acao %>" />

            <a href="ServletProdutoFC" class="btn btn-secondary">Voltar</a>
            <button type="submit" class="btn btn-primary"><%= acao == "incluir" ? "Adicionar" : "Editar" %> Produto</button>
        </form>
    </body>
</html>

```

Testes

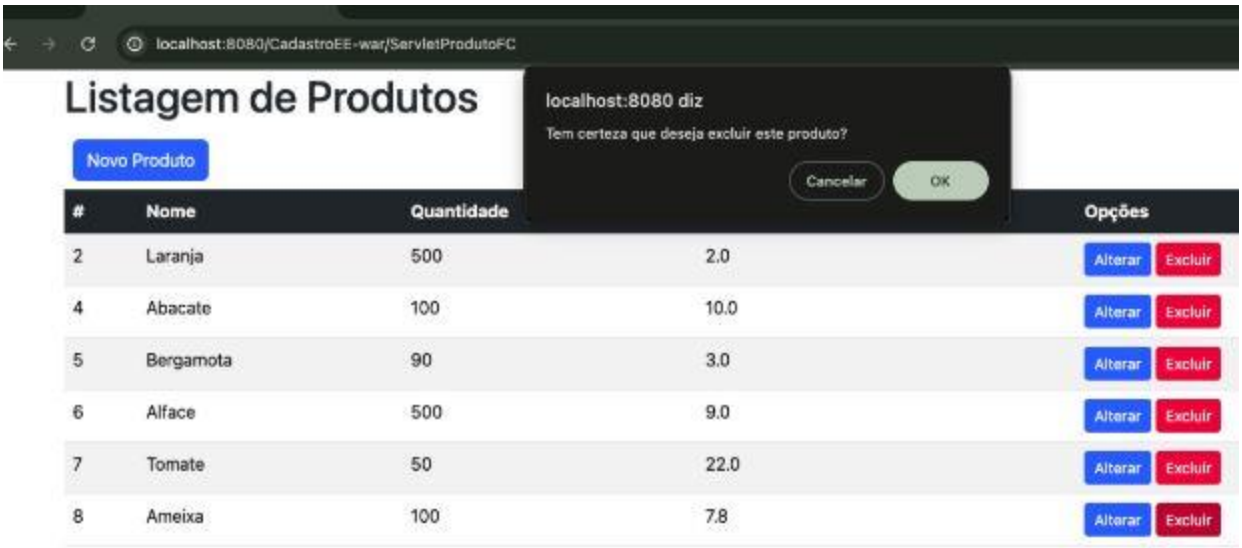
Listagem

Incluir

Alterar



Excluir



Análise e Conclusão

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

No padrão Front Controller, todas as requisições do cliente são direcionadas para um único ponto de entrada, que é o componente Front Controller, ele fica responsável por analisar cada requisição, determinar qual componente deve responder a solicitação e despachar a requisição para o componente, após o processamento, o Front Controller coordena a resposta à solicitação do cliente. Em um aplicativo Web Java, o Front Controller fica na camada View da aplicação, geralmente implementado como um Servlet que é mapeado para uma URL específica que atua como ponto de entrada central para todas as requisições do cliente, o Servlet fica responsável por analisar a requisição, determinar qual controller (EJB) deve processar a requisição e, após o processamento, deve responder a solicitação do cliente, seja com o redirecionamento para uma View específica ou enviando uma resposta em HTML/JSON/XML.

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

Os Servlets possuem um objetivo diferente dos JSPs, eles são responsáveis por ser o ponto central de recebimento e processamento das requisições de um servidor web, diferente dos JSPs, que são páginas que contém código HTML e Java responsável pela camada de apresentação (View) do conteúdo. Os dois possuem em comum a possibilidade de retornar código HTML como resposta a solicitação, assim como são executados do lado servidor, onde possuem acesso aos mesmos objetos de solicitação (HttpServletRequest) e de resposta (HttpServletResponse).

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?

   No redirecionamento simples, o Servlet responde a requisição com um redirecionamento que deverá ser executado pelo navegador ao receber a resposta, neste caso a nova solicitação não tem vínculo com a requisição original, no caso do método forward o redirecionamento é feito internamente pelo servidor, onde a solicitação pode ser enviada para um outro Servlet, uma página HTML ou JSP, esse segundo método permite o compartilhamento de informações por meio de parâmetros e/ou atributos. Os parâmetros e atributos nos objetos HttpRequest permitem que o Servlet adicione informações adicionais necessárias para o processamento e criação da View solicitada pelo usuário.

3º Procedimento | Melhorando o design da interface

OBS.: As imagens das interfaces com o bootstrap estão nos testes do Procedimento 2

Análise e Conclusão

a) Como o framework Bootstrap é utilizado?

O framework Boostrap é utilizado por meio de classes que possuem predefinições, essas classes podem ser aplicadas aos elementos HTML que irão definir a interface que será apresentada para o usuário que independem da estrutura do HTML, pois a sua maioria não depende de elementos específicos ou estruturas específicas para serem aplicadas. No projeto ela foi utilizada para tornar a interface mais amigável para o usuário, adicionando por meio de classes a responsividade da aplicação e caracterização dos elementos, como tabela, formulários e botões de ação de forma que os tornam mais fáceis de reconhecimento e entendimento das ações para o usuário da interface.

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

Suas classes de personalização, em sua maioria, não dependem de uma estrutura ou elemento HTML específico, onde uma classe de botão pode ser aplicada a qualquer elemento HTML, além de possuir uma grande quantidade de predefinições visuais responsivas, assim como o Grid responsivo, que permite desenvolver a interface com a reutilização de classes

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

O Boostrap é construído considerando o conceito de mobile first, que possui o foco em desenvolver interfaces primeiro para dispositivos móveis, e só então, a partir deles adaptar os elementos para dispositivos maiores, visto que hoje em dia mais de 60% dos dispositivos em uso são celulares. A maioria de seus componentes

predefinidos possuem um comportamento responsivo, que se adaptam ao tamanho disponível, mas também é possível usar o sistema de Grid e outras classes auxiliares do Bootstrap para definir como serão os comportamentos em cada tamanho de tela, chamados de Breakpoints; na maioria de suas classes é possível utilizar sufixos como sm, md, lg, etc... que irão determinar quando aquela personalização deverá ser aplicada de acordo com o tamanho de tela do dispositivo do usuário.