Campus: 1197 - POLO CENTRO - SÃO LOURENÇO DA MATA - PE
Curso: Desenvolvimento Full Stack - Graduação Tecnóloga
Disciplina: RPG0014 - Iniciando o caminho pelo Java
Turma: 9001 Semestre: 2024.1
Matrícula: 2023.01.53256-6 Aluno: Gilvan Pereira de Oliveira
Repositório GitHub: GilvanPOliveira/CadastroEE (github.com)

Relatório discente de acompanhamento

1º Procedimento - Criação das Entidades e Sistema de Persistência

Objetivo da prática:

Nesta etapa o objetivo foi criar um sistema de persistência, armazenar e recuperar, informações de entidades para um cadastro de pessoas físicas e jurídicas em Java, utilizando POO, herança, polimorfismo e manipulação de arquivos. O sistema permitirá operações de inserção, alteração, exclusão, recuperação e obtenção das entidades armazenadas, proporcionando uma maneira eficiente e segura de gerenciar e recuperar informações.

Prática:

- Criação das Entidades: Pessoa, PessoaFisica, PessoaJuridica;
- Criação dos gerenciadores: PessoaFisicaRepo, PessoaJuridicaRepo;
- Adição dos métodos: inserir, alterar, excluir, obter e obterTodos, além de pesistir e recuperar aos gerenciadores para armazenagem dos dados no disco;
- Alterar a classe principal (main) para testar os repositórios.
- Executar e verificar as funcionalidades implementadas e os arquivos gerados.

Códigos e resultados obtidos:

- Resultado da execução dos códigos:

- Códigos:

Utilizado o arquivo:

```
mssql-jdbc-12.2.0.jre8.jar
```

Conexão com o banco de dados SQL:

```
jdbc:sqlserver://localhost\6TNLO1P:1433;databaseName=Loja;encrypt=true;trustS
erverCertificate=true;

login: loja
```

senha: cadastrobd

Iniciando servidor GlassFish:

GlassFish_Server 6.2.1

Prompt de comando como administrador:

```
asadmin create-jdbc-connection-pool --
datasourceclassname=com.microsoft.sqlserver.jdbc.SQLServerDataSource --
restype=javax.sql.DataSource --
property="driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver:portNumber
=1433:password=cadastrobd:user=loja:serverName=localhost:databaseName=Loja:tr
ustServerCertificate=true:URL='jdbc:sqlserver://localhost:1433;databaseName=L
oja;encrypt=true;trustServerCertificate=true;'" SQLServerPool
```

Identificador do Pool:

```
asadmin create-jdbc-resource --connectionpoolid SQLServerPool jdbc/loja
```

Testar o pool de conexões, só foi possível via http://localhost:4848/, por terminal apresentou erro.

Criando o projeto CadastroEE, com o servidor GlassFish, além da plataforma Jakarta JEE 8:

No projeto CadastroEE-ejb, foi criado:

Um package denominado cadastroee.model, criado através do New Entitty Classes from Database, pegando classes já criadas no banco de dados solicitado (Loja), com as seguintes Classes:

Movimento.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
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 gilvan
 */
@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 = 10)
    @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 Pessoas idPessoa;
    @JoinColumn(name = "idProduto", referencedColumnName = "idProduto")
    @ManyToOne(optional = false)
    private Produtos idProduto;
    @JoinColumn(name = "idUsuario", referencedColumnName = "idUsuario")
    @ManyToOne(optional = false)
    private Usuarios 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 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 += (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 + "
]";
    }

}
```

PessoaFisica.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
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 gilvan
 */
@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 = 14)
    @Column(name = "cpf")
    private String cpf;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa",
insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoas pessoas;

    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 Pessoas getPessoas() {
        return pessoas;
    }

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

    @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 + " ]";
    }

}
```

PessoaJuridica.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
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 gilvan
 */
@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 = 18)
    @Column(name = "cnpj")
    private String cnpj;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa",
insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoas pessoas;

    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 Pessoas getPessoas() {
        return pessoas;
    }

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

    @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 + "
]";
    }

}
```

Pessoas.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
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.OneToOne;
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 gilvan
 */
@Entity
@Table(name = "pessoas")
```

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

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Size(max = 100)
    @Column(name = "nome")
    private String nome;
    @Size(max = 200)
    @Column(name = "logradouro")
    private String logradouro;
    @Size(max = 50)
    @Column(name = "cidade")
    private String cidade;
    @Size(max = 50)
    @Column(name = "estado")
    private String estado;
    @Size(max = 50)
    @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 = 50)
    @Column(name = "email")
    private String email;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoas")
    private PessoaFisica pessoaFisica;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoas")
    private PessoaJuridica pessoaJuridica;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idPessoa")
    private Collection<Movimento> movimentoCollection;

    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 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 Pessoas)) {
            return false;
        }
        Pessoas other = (Pessoas) 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.Pessoas[ idPessoa=" + idPessoa + " ]";
    }

}
```

Produtos.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
```

```
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 gilvan
 */
@Entity
@Table(name = "produtos")
@XmlRootElement
@NamedQueries({
    @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.findByPreco", query = "SELECT p FROM
Produtos p WHERE p.preco = :preco")})
public class Produtos implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idProduto")
    private Integer idProduto;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 100)
    @Column(name = "nome")
    private String nome;
    @Column(name = "quantidade")
    private Integer quantidade;
    @Column(name = "preco")
    private float preco;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idProduto")
    private Collection<Movimento> movimentoCollection;

    public Produtos() {
    }
```

```
    public Produtos(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public Produtos(Integer idProduto, String nome) {
        this.idProduto = idProduto;
        this.nome = nome;
    }

    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 getPreco() {
        return preco;
    }

    public void setPreco(float preco) {
        this.preco = preco;
    }

    @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 Produtos)) {
            return false;
        }
        Produtos other = (Produtos) 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.Produtos[ idProduto=" + idProduto + " ]";
    }

}
```

Usuarios.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
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 gilvan
 */
@Entity
@Table(name = "usuarios")
@XmlRootElement
@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.findByNome", query = "SELECT u FROM Usuarios
u WHERE u.nome = :nome"),
    @NamedQuery(name = "Usuarios.findBySenha", query = "SELECT u FROM
Usuarios u WHERE u.senha = :senha")})
public class Usuarios implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idUsuario")
    private Integer idUsuario;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 100)
    @Column(name = "nome")
    private String nome;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 50)
    @Column(name = "senha")
    private String senha;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idUsuario")
    private Collection<Movimento> movimentoCollection;

    public Usuarios() {
    }

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

    public Usuarios(Integer idUsuario, String nome, String senha) {
        this.idUsuario = idUsuario;
        this.nome = nome;
        this.senha = senha;
    }

    public Integer getIdUsuario() {
        return idUsuario;
    }

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

    public String getNome() {
        return nome;
    }

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

    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 Usuarios)) {
            return false;
        }
        Usuarios other = (Usuarios) 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.Usuarios[ idUsuario=" + idUsuario + " ]";
    }

}
```

Também foi necessário marcar a opção de criação de um arquivo denominado persistence.xml.

Em seguida, foi solicitado adicionar os componentes EJB ao projeto, através da opção New Session beans for Entity Classes, para gerar Session Beans com sufixo Facade, bem como interfaces, com o sufixo FacadeLocal, seguem os códigos das classes desse package novo denominado cadastroee.controller:

AbstractFacade.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
public abstract class AbstractFacade<T> {

    private Class<T> entityClass;

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

    protected abstract EntityManager getEntityManager();

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

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

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

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

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

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

    public int count() {
        jakarta.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
        jakarta.persistence.criteria.Root<T> rt = cq.from(entityClass);
        cq.select(getEntityManager().getCriteriaBuilder().count(rt));
```

```
        jakarta.persistence.Query q = getEntityManager().createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }

}
```

MovimentoFacade.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
@Stateless
public class MovimentoFacade extends AbstractFacade<Movimento> implements
MovimentoFacadeLocal {

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

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public MovimentoFacade() {
        super(Movimento.class);
    }

}
```

MovimentoFacadeLocal.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to
edit this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
```

```
 */
@Local
public interface MovimentoFacadeLocal {

    void create(Movimento movimento);

    void edit(Movimento movimento);

    void remove(Movimento movimento);

    Movimento find(Object id);

    List<Movimento> findAll();

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

    int count();

}
```

PessoaFisicaFacade.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
@Stateless
public class PessoaFisicaFacade extends AbstractFacade<PessoaFisica>
implements PessoaFisicaFacadeLocal {

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

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public PessoaFisicaFacade() {
        super(PessoaFisica.class);
    }

}
```

PessoaFisiscaFacadeLocal.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to
edit this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
@Local
public interface PessoaFisicaFacadeLocal {

    void create(PessoaFisica pessoaFisica);

    void edit(PessoaFisica pessoaFisica);

    void remove(PessoaFisica pessoaFisica);

    PessoaFisica find(Object id);

    List<PessoaFisica> findAll();

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

    int count();

}
```

PessoaJuridicaFacade.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
@Stateless
public class PessoaJuridicaFacade extends AbstractFacade<PessoaJuridica>
implements PessoaJuridicaFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
```

```
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public PessoaJuridicaFacade() {
        super(PessoaJuridica.class);
    }

}
```

PessoaJuridicaFacadeLocal.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to
edit this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
@Local
public interface PessoaJuridicaFacadeLocal {

    void create(PessoaJuridica pessoaJuridica);

    void edit(PessoaJuridica pessoaJuridica);

    void remove(PessoaJuridica pessoaJuridica);

    PessoaJuridica find(Object id);

    List<PessoaJuridica> findAll();

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

    int count();

}
```

PessoasFacade.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastroee.controller;
```

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

/**
 *
 * @author gilvan
 */
@Stateless
public class PessoasFacade extends AbstractFacade<Pessoas> implements
PessoasFacadeLocal {

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

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public PessoasFacade() {
        super(Pessoas.class);
    }

}
```

PessoasFacadeLocal.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to
edit this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
@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();

}
```

ProdutosFacade.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
@Stateless
public class ProdutosFacade extends AbstractFacade<Produtos> implements
ProdutosFacadeLocal {

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

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public ProdutosFacade() {
        super(Produtos.class);
    }

}
```

ProdutosFacadeLocal.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to
edit this template
 */
package cadastroee.controller;

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

/**
```

```
 *
 * @author gilvan
 */
@Local
public interface ProdutosFacadeLocal {

    void create(Produtos produtos);

    void edit(Produtos produtos);

    void remove(Produtos produtos);

    Produtos find(Object id);

    List<Produtos> findAll();

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

    int count();

}
```

UsuariosFacade.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
@Stateless
public class UsuariosFacade extends AbstractFacade<Usuarios> implements
UsuariosFacadeLocal {

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

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public UsuariosFacade() {
        super(Usuarios.class);
    }

}
```

UsuariosFacadeLocal.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to
edit this template
 */
package cadastroee.controller;

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

/**
 *
 * @author gilvan
 */
@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();

}
```

Após esse processo foi adicionado a biblioteca Jakarta EE 8 API ao projeto CadastroEE-ejb

Também foram modificadas todas as importações de pacotes javax para jakarta em todos os arquivos do projeto CadastroEE-ejb

Na entidade Produtos, foi necessário alterar o tipo do atributo Preco para Float, no lugar de BigDecimal, e foi necessário alterar o arquivo persistence.xml, com o seguinte código:

```
<?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>
```

Agora vem a criação de um Servlet de teste no projeto CadastroEE-war, com o nome ServletProduto dentro de um pacote denominado cadastroee.servlets, lembrando de deixar selecionado a opção Add information to deployment descriptor, pois é necessário devido a versão do GlassFish utilizada. Adicionar também a referência para a interface do EJB, através do código:

```
@EJB
ProdutoFacadeLocal facade;
```

Modificar a resposta do Servlet, utilizando o facade para recuperar os dados e apresenta-los na forma de lista HTML, além de adicionar a biblioteca Jakarta EE Web 8 API ao projeto CadastroEE-war e modificar todas as importações de pacotes javax para jakarta. Segue o código da página ServletProduto.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to
edit this template
 */
package cadastroee.servlets;

import cadastroee.controller.ProdutosFacadeLocal;
import cadastroee.model.Produtos;
import jakarta.ejb.EJB;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

/**
 *
 * @author gilvan
 */

public class ServletProduto extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
     * methods.
```

```
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @EJB
    ProdutosFacadeLocal facade;

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            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>");
                List<Produtos> listaDeProdutos = facade.findAll();
                for (Produtos produtos : listaDeProdutos) {
                    out.println("<li>" + produtos.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>

}
```

Modificar o arquivo web.xml para que a exibição seja feita:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">

 <servlet>

   <servlet-name>ServletProduto</servlet-name>

   <servlet-class>cadastroee.servlets.ServletProduto</servlet-class>

 </servlet>

 <servlet>

   <servlet-name>ServletProdutoFC</servlet-name>

   <servlet-class>cadastroee.servlets.ServletProdutoFC</servlet-class>

 </servlet>

 <session-config>

   <session-timeout>30</session-timeout>

 </session-config>

</web-app>
```

Também foi feita uma pequena modificação na página index.html, apenas para melhor acessar a página ServletProdutos.java, segue o código alterado da página index.html:

```
<!DOCTYPE html>
<!--
Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
```

```
Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Html.html to edit
this template
-->
<html>
    <head>
        <title>CadastroEE</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    </head>
    <body>
        <h1> <a href="./ServletProduto"> Servlets Produtos </a> </h1>
        <h1> <a href="./ServletProdutoFC"> Servlets Produtos FC </a> </h1>
    </body>
</html>
```

Executando o projeto a partir do endereço (http://localhost:8080/CadastroEE-war/ServletProduto) ou do link criado no index.html, obtemos o seguinte resultado:

Análise e Conclusão:

- Como é organizado um projeto corporativo no NetBeans?
Resposta: Um projeto organizado no NetBeans segue uma estrutura modular, baseada em componentes e módulos diferentes organizados através da interface do projeto.

- Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?
Resposta: A tecnologia JPA, permite o mapeamento, objeto-relacional, para persistência de entidades Java em banco de dados; já a tecnologia EJB, oferece um modelo construtivo para componentes de negócio transacionais, facilitando o desenvolvimento de aplicativos corporativos escaláveis.

- Como o NetBeans viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?
Resposta: O NetBeans integra e da suporte ao desenvolvimento com JPA e EJB, de todas as formas, autocompletando, com ferramentas visuais, depuração e geração de códigos para aumentar a produtividade.

- O que são Servlets, e como o NetBeans oferece suporte à construção desse tipo de componentes em um projeto Web?
Resposta: São componentes Java responsáveis por gerar conteúdo dinâmico em

servidores web, onde no NetBeans à uma simplificação a sua implementação e criação, pois ele oferece suporte à construção dos Servlets através de wizards e modelos de projetos para aplicações web.

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

Resposta: A comunicação ocorre via injeção de referências para os Session Beans nos Servlets, utilizando o @EJB por exemplo, ou através de serviços web.

2º Procedimento - Interface Cadastral com Servlet e JSPs

Foi solicitado criar um novo Servlet, no mesmo package, com o nome ServletProdutoFC em CadastroEE-war, utilizando o padrão Front Controller, adicionando uma referência para ProdutoFacadeLocal utilizando o nome facade para o atributo. Segue o código da página ServletProdutoFC.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to
edit this template
 */

package cadastroee.servlets;

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

/**
 *
 * @author gilvan
 */


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

    @EJB
    private ProdutosFacadeLocal facade;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String acao = request.getParameter("acao");

        if (acao == null || acao.isEmpty()) {
            acao = "listar";
        }

        switch (acao) {
            case "excluir":
                excluirProdutos(request, response);
                break;
            case "formAlterar":
                formAlterar(request, response);
```

```
                break;
            case "formIncluir":
                formIncluir(request, response);
                break;
            default:
                listarProdutos(request, response);
        }
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String acao = request.getParameter("acao");

        if (acao != null && !acao.isEmpty()) {
            switch (acao) {
                case "incluir":
                    incluirProdutos(request, response);
                    break;
                case "alterar":
                    alterarProdutos(request, response);
                    break;
            }
        } else {
            response.sendRedirect("ServletProdutoFC");
        }
    }

    private void listarProdutos(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        List<Produtos> produtos = facade.findAll();
        request.setAttribute("produtos", produtos);
        RequestDispatcher dispatcher =
request.getRequestDispatcher("ProdutoLista.jsp");
        dispatcher.forward(request, response);
    }

    private void excluirProdutos(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        try {
            int idExcluir = Integer.parseInt(request.getParameter("id"));
            facade.remove(facade.find(idExcluir));
            response.sendRedirect("ServletProdutoFC");
        } catch (NumberFormatException | NullPointerException e) {
            response.sendError(HttpServletResponse.SC_BAD_REQUEST);
        }
    }

    private void formAlterar(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try {
            int idAlterar = Integer.parseInt(request.getParameter("id"));
            Produtos produto = facade.find(idAlterar);
            request.setAttribute("produto", produto);
            RequestDispatcher dispatcher =
request.getRequestDispatcher("ProdutoDados.jsp");
            dispatcher.forward(request, response);
        } catch (NumberFormatException | NullPointerException e) {
            response.sendError(HttpServletResponse.SC_BAD_REQUEST);
        }
```

```
    }

    private void incluirProdutos(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        try {
            String nome = request.getParameter("nome");
            int quantidade =
Integer.parseInt(request.getParameter("quantidade"));
            Float preco = Float.valueOf(request.getParameter("preco"));

            Produtos produtoNovo = new Produtos();
            produtoNovo.setNome(nome);
            produtoNovo.setQuantidade(quantidade);
            produtoNovo.setPreco(preco);

            facade.create(produtoNovo);
            response.sendRedirect("ServletProdutoFC");
        } catch (NumberFormatException | NullPointerException e) {
            response.sendError(HttpServletResponse.SC_BAD_REQUEST);
    }
}


    private void alterarProdutos(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        try {
            int id = Integer.parseInt(request.getParameter("id"));
            String nome = request.getParameter("nome");
            int quantidade =
Integer.parseInt(request.getParameter("quantidade"));
            Float preco = Float.valueOf(request.getParameter("preco"));

            Produtos produtoAlterado = facade.find(id);
            produtoAlterado.setNome(nome);
            produtoAlterado.setQuantidade(quantidade);
            produtoAlterado.setPreco(preco);

            facade.edit(produtoAlterado);
            response.sendRedirect("ServletProdutoFC");
        } catch (NumberFormatException | NullPointerException e) {
            response.sendError(HttpServletResponse.SC_BAD_REQUEST);
        }
    }

    private void formIncluir(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        RequestDispatcher dispatcher =
request.getRequestDispatcher("ProdutoDados.jsp");
        dispatcher.forward(request, response);
    }
}
```

Após configurar os métodos no novo Servlet, foi criado uma página de consulta, denominada ProdutoLista.jsp, com o seguinte código:

```
<%--
    Document   : ProdutoLista
    Created on : 3 de mai. de 2024, 15:17:57
```

```
    Author     : gilvan
--%>

<%@ page import="cadastroee.model.Produtos" %>
<%@ page import="java.util.List" %>
<%@ page contentType="text/html" pageEncoding="UTF-8" %>

        <!DOCTYPE html>
        <html>

        <head>
          <meta charset="UTF-8">
          <title>JSP Page</title>
        </head>

        <body>
          <div>
            <h1>Listagem de Produtos</h1>
          </div>

          <div>
            <table>
              <thead>
                <tr>
                  <th>#</th>
                  <th>Nome</th>
                  <th>Quantidade</th>
                  <th>Preço de Venda</th>
                  <th>Opções</th>
                </tr>
              </thead>

              <tbody>

                <%
                    List<Produtos> produtos = (List<Produtos>)
request.getAttribute("produtos");
                    if (produtos != null && !produtos.isEmpty()) {
                        for (Produtos produto : produtos) {
                %>

                    <tr>
                        <td><%=produto.getIdProduto()%></td>
                        <td><%=produto.getNome()%></td>
                        <td><%=produto.getQuantidade()%></td>
                        <td><%=produto.getPreco()%></td>
                        <td>
                            <a
href="ServletProdutoFC?acao=formAlterar&id=<%=produto.getIdProduto()%>">
Alterar </a>
                            <a
href="ServletProdutoFC?acao=excluir&id=<%=produto.getIdProduto()%>"> Excluir
</a>
                        </td>
                    </tr>

                <%
                    }
                } else {
                %>
```

```
              </tbody>
            </table>

            <tr>
              <div>Nenhum produto encontrado!</div>
            </tr>

                <%
                    }
                %>
          </div>

          <div>
              <a href="ServletProdutoFC?acao=formIncluir"> Novo Produto </a>
          </div>
        </body>

        </html>
```

Após a criação da página de listar, foi gerado uma página para dados, assim contemplando inserir, alterar e excluir. Segue o código da página de dados ProdutoDados.jsp:

```
<%--
    Document   : ProdutoDados
    Created on : 3 de mai. de 2024, 15:34:03
    Author     : gilvan
--%>

<%@page import="cadastroee.model.Produtos"%>
<%@page import="java.util.List"%>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Cadastro de Produtos</title>
</head>
<body>
    <%
        Produtos produto = (Produtos) request.getAttribute("produto");
        String acao = produto != null ? "alterar" : "incluir";
    %>

    <div>
        <h1><%=acao.equals("alterar") ? "Dados" : "Inclusão"%> do
Produto</h1>
    </div>

    <div>
        <form action="ServletProdutoFC" method="post">
            <input type="hidden" name="acao" value="<%=acao%>">

            <% if (acao.equals("alterar")) { %>
                <input type="hidden" name="id"
value="<%=produto.getIdProduto()%>">
            <% } %>
```

```
            <div>
                <label for="nome">Nome</label>
                <input type="text" name="nome" value="<%=produto != null ?
produto.getNome() : ""%>" required>
            </div>
            <div>
                <label for="quantidade">Quantidade</label>
                <input type="text" name="quantidade" value="<%=produto !=
null ? produto.getQuantidade() : ""%>" required>
            </div>
            <div>
                <label for="preco">Preço de Venda</label>
                <input type="text" name="preco" value="<%=produto != null ?
produto.getPreco() : ""%>" required>
            </div>

            <div>
                <input type="submit" value="<%=acao.equals("incluir") ?
"Incluir" : "Alterar"%>">
            </div>
        </form>
    </div>
</body>
</html>
```

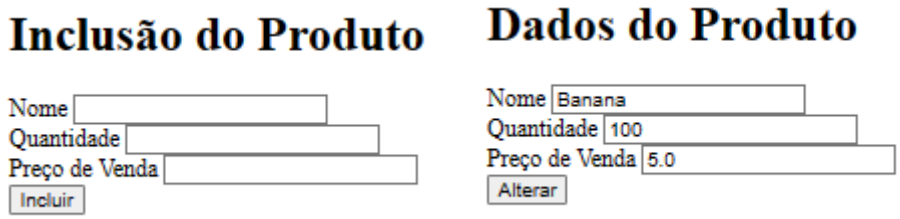
Listagem dos produtos pelo endereço:

```
http://localhost:8080/CadastroEE-war/ServletProdutoFC?acao=listar
```

Novamente foi modificado a página index.html para facilitar o acesso as demais páginas. Segue o código da página index.html:

```
<!DOCTYPE html>
<!--
Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Html.html to edit
this template
-->
<html>
    <head>
        <title>CadastroEE</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    </head>
    <body>
        <div>
            <h1>Servlets: </h1>
            <h2> <a href="./ServletProduto"> Servlets Produtos </a> </h2>
            <h2> <a href="./ServletProdutoFC"> Servlets Produtos FC </a>
</h2>
        </div>
    </body>
</html>
```

Obtendo assim o seguinte resultado nas telas:



Análise e Conclusão:

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

Resposta: É um design pattern que centraliza o procedimento de requisições em um único ponto de entrada em uma aplicação web. Implementado como um Servlet, na arquitetura MVM, que recebe todas as requisições do cliente decidindo qual controlador deve lidar com a requisição com base na URL e encaminha a requisição para esse controlador.

- Quais as diferenças e semelhanças entre Servlets e JSPs?

Resposta:

-Semelhanças: Ambos são usados em desenvolvimento web Java para criar páginas dinâmicas;

-Diferenças:

Servlet são classes Java que respondem a requisições HTTP

JSPs são arquivos de texto que misturam código Java com HTML, traduzidos em servlets pelo servidor.

- 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?

Resposta:

-Redirecionamento Simples: Ocorre quando o servidor envia uma resposta e uma nova URL para o navegador, e então o navegador faz uma nova requisição para essa URL;

-Método forward a partir do RequestDispatcher: Ocorre internamente no servidor, e não há uma nova requisição do navegador;

-Parâmentros e atributos no objeto HttpRequest: São informações enviadas do cliente para o servidor, geralmente através de formulários HTML ou pela URL.

3º Procedimento - Melhorando o Design da Interface

Inclusão do framework Bootstrap ao projeto, modificar as páginas ProdutoLista.jsp e ProdutoDados.jsp. Seguem os códigos e resultados obtidos:

ProdutoLista.jsp:

```
<%--
    Document   : ProdutoLista
    Created on : 3 de mai. de 2024, 15:17:57
    Author     : gilvan
--%>

<%@ page import="cadastroee.model.Produtos" %>
<%@ page import="java.util.List" %>
<%@ page contentType="text/html" pageEncoding="UTF-8" %>

        <!DOCTYPE html>
        <html>

        <head>
          <meta charset="UTF-8">
          <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css
" rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
          <title>JSP Page</title>
        </head>

        <body class="container">
          <div>
            <h1>Listagem de Produtos</h1>
          </div>

          <div>
            <table class="table table-striped table-bordered">

              <thead class="table-dark">
                <tr>
                  <th>#</th>
                  <th>Nome</th>
                  <th>Quantidade</th>
                  <th>Preço de Venda</th>
                  <th>Opções</th>
                </tr>
              </thead>

              <tbody>

                <%
                    List<Produtos> produtos = (List<Produtos>)
request.getAttribute("produtos");
                    if (produtos != null && !produtos.isEmpty()) {
                        for (Produtos produto : produtos) {
                %>
```

```
                    <tr>
                        <td><%=produto.getIdProduto()%></td>
                        <td><%=produto.getNome()%></td>
                        <td><%=produto.getQuantidade()%></td>
                        <td><%=produto.getPreco()%></td>
                        <td>
                            <a class="btn btn-primary btn-sm"
href="ServletProdutoFC?acao=formAlterar&id=<%=produto.getIdProduto()%>">
Alterar </a>
                            <a class="btn btn-danger btn-sm"
href="ServletProdutoFC?acao=excluir&id=<%=produto.getIdProduto()%>"> Excluir
</a>
                        </td>
                    </tr>

                <%
                    }
                } else {
                %>

              </tbody>
            </table>

            <tr>
              <div>Nenhum produto encontrado!</div>
            </tr>

                <%
                    }
                %>
          </div>

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

        </html>
```

Resultado obtido:

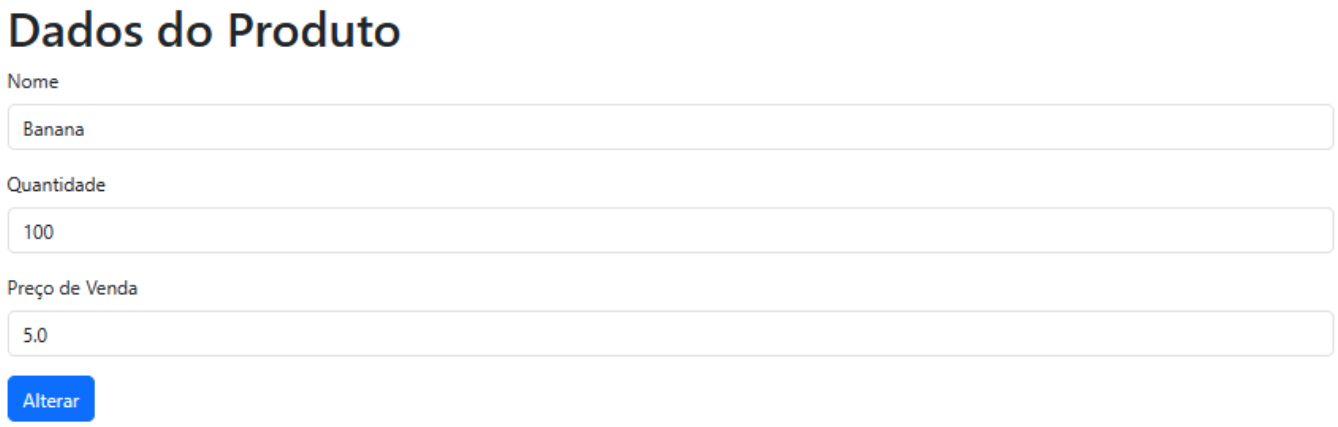
ProdutoDados.jsp:

```
<%--
    Document    : ProdutoDados
```

```
    Created on : 3 de mai. de 2024, 15:34:03
    Author     : gilvan
--%>

<%@page import="cadastroee.model.Produtos"%>
<%@page import="java.util.List"%>
<!DOCTYPE html>
<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="sha384-
4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9"
crossorigin="anonymous">
    <title>Cadastro de Produtos</title>
</head>
<body class="container">
    <%
        Produtos produto = (Produtos) request.getAttribute("produto");
        String acao = produto != null ? "alterar" : "incluir";
    %>

    <div>
        <h1><%=acao.equals("alterar") ? "Dados" : "Inclusão"%> do
Produto</h1>
    </div>

    <div>
        <form class="form" action="ServletProdutoFC" method="post">
            <input type="hidden" name="acao" value="<%=acao%>">

            <% if (acao.equals("alterar")) { %>
                <input type="hidden" name="id"
value="<%=produto.getIdProduto()%>">
            <% } %>

            <div class="mb-3">
                <label class="form-label" for="nome">Nome</label>
                <input class="form-control" type="text" name="nome"
value="<%=produto != null ? produto.getNome() : ""%>" required>
            </div>
            <div class="mb-3">
                <label class="form-label" for="quantidade">Quantidade</label>
                <input class="form-control" type="text" name="quantidade"
value="<%=produto != null ? produto.getQuantidade() : ""%>" required>
            </div>
            <div class="mb-3">
                <label class="form-label" for="preco">Preço de Venda</label>
                <input class="form-control" type="text" name="preco"
value="<%=produto != null ? produto.getPreco() : ""%>" required>
            </div>

            <div>
                <input class="btn btn-primary" type="submit"
value="<%=acao.equals("incluir") ? "Incluir" : "Alterar"%>">
            </div>
        </form>
    </div>
</body>
```

```
</html>
```

Resultado obtido:



Análise e Conclusão:

- Como o framework Bootstrap é utilizado?
Resposta: É utilizado para criar interfaces web responsivas e estilizadas de forma rápida e prática, fornecendo componentes e estilos pré-definidos.

- Por que o Bootstrap garante a independência estrutural do HTML?
Resposta: Porque ele separa a apresentação visual e o comportamento interativo, e foca totalmente na semântica do HTML, aplicando estilos e funcionalidades diretamente nas linhas de código HTML.

- Qual a relação entre o Boostrap e a responsividade da página?
Resposta: O Bootstrap possui um sistema de grid flexível e componentes adaptáveis que se ajustam ao tamanho da tela, garantindo a responsividade de forma simples e automática.