Missão Pratica - Vamos integrar sistemas

Andrey Haertel Aires - Matricula: 2021.07.22851-2

Polo Centro - Palhoça – SC
Nível 4: Vamos integrar sistemas – T 9001 – 3º Semestre Letivo

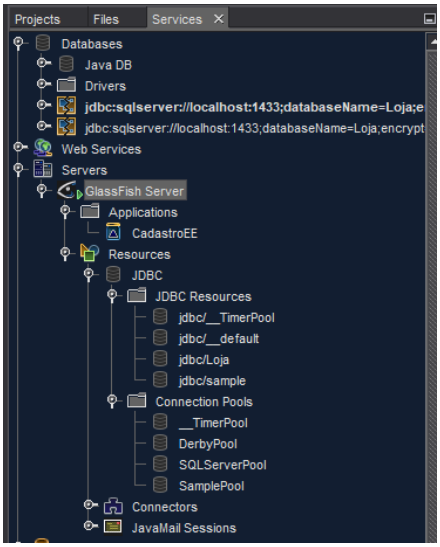
Objetivo da Prática

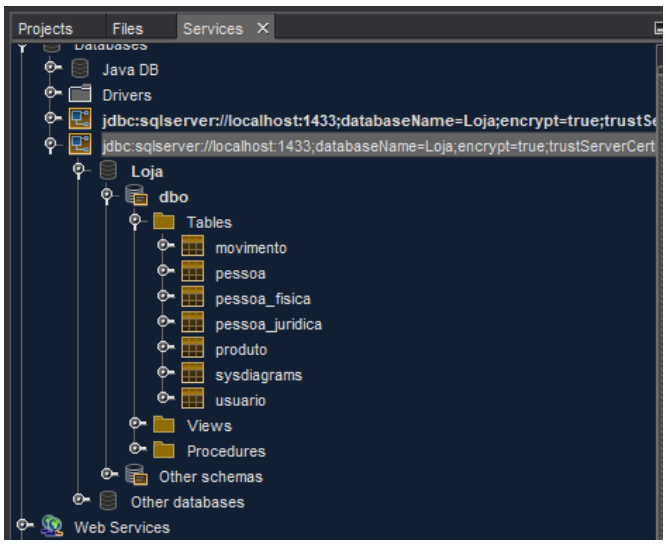
- Implementar persistência com base em JPA.
- Implementar regras de negócio na plataforma JEE, através de EJBs.
- Implementar sistema cadastral Web com base em Servlets e JSPs.
- Utilizar a biblioteca Bootstrap para melhoria do design.
- No final do exercício, o aluno terá criado todos os elementos necessários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

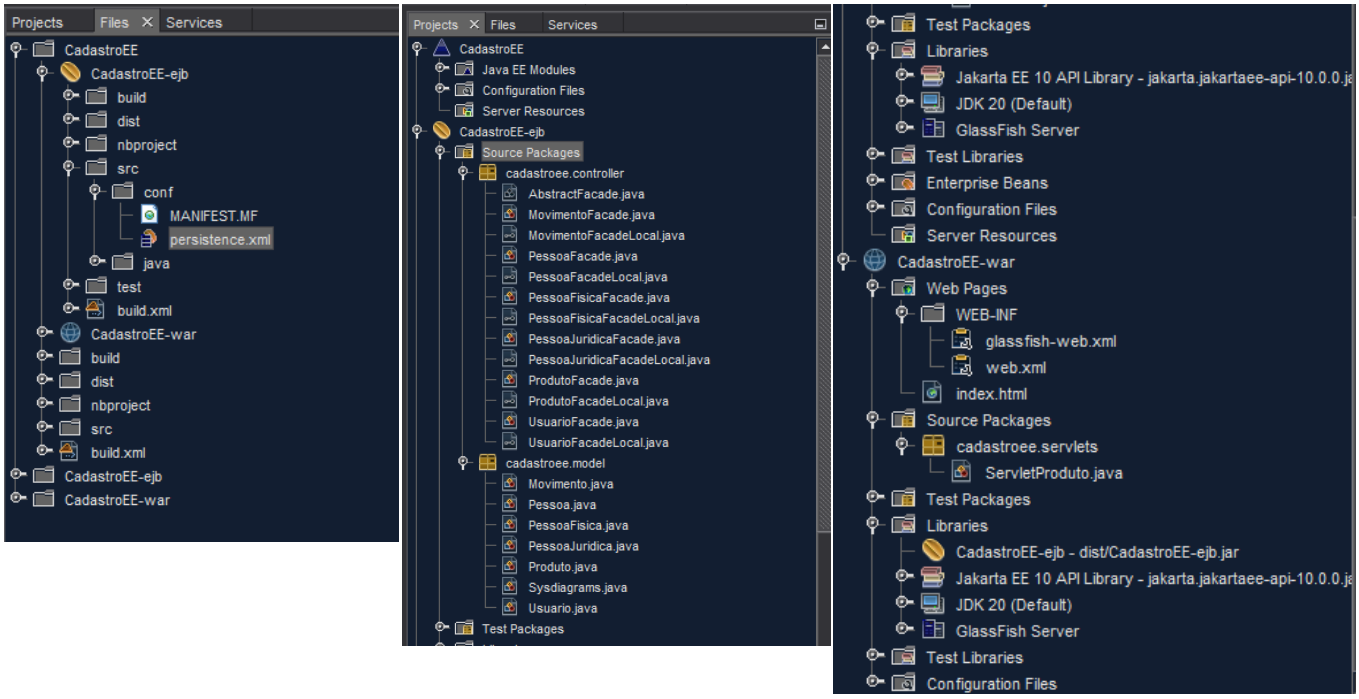
1º Procedimento | Camadas de Persistência e Controle

- Configurar a conexão com SQL Server via NetBeans e o pool de conexões no GlassFish Server.
- Criar o aplicativo corporativo no NetBeans.
- Definir as camadas de persistência e controle no projeto CadastroEE-ejb.
- Efetuar pequenos acertos no projeto, para uso do Jakarta.
- Criar um Servlet de teste no projeto CadastroEE-war.
- Executar o projeto.

Configurações de conexões no NETBEANS:







Códigos solicitados neste roteiro de aula:

Pacote: cadastroee.model

Movimento.java

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java para
editar este modelo
 */
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 java.io.Serializable;
import java.math.BigDecimal;

/**
 * Entidade que representa um Movimento no sistema.
 * Esta classe mapeia a tabela 'movimento' no banco de dados e fornece
consultas nomeadas para operações específicas.
 * @author Andrey
 */
@Entity
@Table(name = "movimento")
@NamedQueries({
```

```java
    @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.findByQuantidadeMovimento", query = "SELECT m
FROM Movimento m WHERE m.quantidadeMovimento = :quantidadeMovimento"),
    @NamedQuery(name = "Movimento.findByTipoMovimento", query = "SELECT m FROM
Movimento m WHERE m.tipoMovimento = :tipoMovimento"),
    @NamedQuery(name = "Movimento.findByValorUnitarioMov", query = "SELECT m
FROM Movimento m WHERE m.valorUnitarioMov = :valorUnitarioMov")})
public class Movimento implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_movimento")
    private Integer idMovimento;
    @Column(name = "quantidade_movimento")
    private Integer quantidadeMovimento;
    @Column(name = "tipo_movimento")
    private String tipoMovimento;
    // @Max(value=?)  @Min(value=?)//if you know range of your decimal fields
consider using these annotations to enforce field validation
    @Column(name = "valor_unitario_mov")
    private BigDecimal valorUnitarioMov;
    @JoinColumn(name = "id_pessoa", referencedColumnName = "id_pessoa")
    @ManyToOne(optional = false)
    private Pessoa idPessoa;
    @JoinColumn(name = "id_produto", referencedColumnName = "id_produto")
    @ManyToOne(optional = false)
    private Produto idProduto;
    @JoinColumn(name = "id_usuario", referencedColumnName = "id_usuario")
    @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 getQuantidadeMovimento() {
        return quantidadeMovimento;
    }

    public void setQuantidadeMovimento(Integer quantidadeMovimento) {
        this.quantidadeMovimento = quantidadeMovimento;
    }

    public String getTipoMovimento() {
        return tipoMovimento;
    }

    public void setTipoMovimento(String tipoMovimento) {
        this.tipoMovimento = tipoMovimento;
```

```
    }

    public BigDecimal getValorUnitarioMov() {
        return valorUnitarioMov;
    }

    public void setValorUnitarioMov(BigDecimal valorUnitarioMov) {
        this.valorUnitarioMov = valorUnitarioMov;
    }

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

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

}
```

Pessoa.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 java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author Andrey
 */
@Entity
@Table(name = "pessoa")
@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.findByNomePessoa", query = "SELECT p FROM Pessoa
p WHERE p.nomePessoa = :nomePessoa"),
    @NamedQuery(name = "Pessoa.findByLogradouroPessoa", query = "SELECT p FROM
Pessoa p WHERE p.logradouroPessoa = :logradouroPessoa"),
    @NamedQuery(name = "Pessoa.findByCidadePessoa", query = "SELECT p FROM
Pessoa p WHERE p.cidadePessoa = :cidadePessoa"),
    @NamedQuery(name = "Pessoa.findByEstadoPessoa", query = "SELECT p FROM
Pessoa p WHERE p.estadoPessoa = :estadoPessoa"),
    @NamedQuery(name = "Pessoa.findByTelefonePessoa", query = "SELECT p FROM
Pessoa p WHERE p.telefonePessoa = :telefonePessoa"),
    @NamedQuery(name = "Pessoa.findByEmailPessoa", query = "SELECT p FROM
Pessoa p WHERE p.emailPessoa = :emailPessoa")})
public class Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_pessoa")
    private Integer idPessoa;
    @Basic(optional = false)
    @Column(name = "nome_pessoa")
    private String nomePessoa;
    @Column(name = "logradouro_pessoa")
    private String logradouroPessoa;
    @Column(name = "cidade_pessoa")
    private String cidadePessoa;
    @Column(name = "estado_pessoa")
    private String estadoPessoa;
    @Column(name = "telefone_pessoa")
```

```
    private String telefonePessoa;
    @Column(name = "email_pessoa")
    private String emailPessoa;
    @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 Pessoa(Integer idPessoa, String nomePessoa) {
        this.idPessoa = idPessoa;
        this.nomePessoa = nomePessoa;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

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

    public String getNomePessoa() {
        return nomePessoa;
    }

    public void setNomePessoa(String nomePessoa) {
        this.nomePessoa = nomePessoa;
    }

    public String getLogradouroPessoa() {
        return logradouroPessoa;
    }

    public void setLogradouroPessoa(String logradouroPessoa) {
        this.logradouroPessoa = logradouroPessoa;
    }

    public String getCidadePessoa() {
        return cidadePessoa;
    }

    public void setCidadePessoa(String cidadePessoa) {
        this.cidadePessoa = cidadePessoa;
    }

    public String getEstadoPessoa() {
        return estadoPessoa;
    }

    public void setEstadoPessoa(String estadoPessoa) {
        this.estadoPessoa = estadoPessoa;
    }

    public String getTelefonePessoa() {
        return telefonePessoa;
    }
```

```
    public void setTelefonePessoa(String telefonePessoa) {
        this.telefonePessoa = telefonePessoa;
    }

    public String getEmailPessoa() {
        return emailPessoa;
    }

    public void setEmailPessoa(String emailPessoa) {
        this.emailPessoa = emailPessoa;
    }

    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;
    }

    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 "cadastroee.model.Pessoa[ idPessoa=" + idPessoa + " ]";
    }
    }
```

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 java.io.Serializable;

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

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

    public PessoaFisica() {
    }

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

    public Integer getIdPessoa() {
        return idPessoa;
    }

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

    public String getCpf() {
        return cpf;
```

```
    }

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

    public Pessoa getPessoa() {
        return pessoa;
    }

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

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

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

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

}
```

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 java.io.Serializable;

/**
```

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

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

    public PessoaJuridica() {
    }

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

    public Integer getIdPessoa() {
        return idPessoa;
    }

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

    public String getCnpj() {
        return cnpj;
    }

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

    public Pessoa getPessoa() {
        return pessoa;
    }

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

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

    @Override
```

```
    public boolean equals(Object object) {
        // 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

```
/*
 * 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.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author Andrey
 */
@Entity
@Table(name = "produto")
@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.findByNomeProduto", query = "SELECT p FROM Produto p WHERE p.nomeProduto = :nomeProduto"),
    @NamedQuery(name = "Produto.findByQuantidadeProduto", query = "SELECT p FROM Produto p WHERE p.quantidadeProduto = :quantidadeProduto"),
    @NamedQuery(name = "Produto.findByPrecoVendaProduto", query = "SELECT p FROM Produto p WHERE p.precoVendaProduto = :precoVendaProduto")})
public class Produto implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
```

```
    @Column(name = "id_produto")
    private Integer idProduto;
    @Column(name = "nome_produto")
    private String nomeProduto;
    @Column(name = "quantidade_produto")
    private Integer quantidadeProduto;
    // @Max(value=?)  @Min(value=?)//if you know range of your decimal fields
consider using these annotations to enforce field validation
    @Column(name = "preco_venda_produto")
    private Float precoVendaProduto;
    @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 getNomeProduto() {
        return nomeProduto;
    }

    public void setNomeProduto(String nomeProduto) {
        this.nomeProduto = nomeProduto;
    }

    public Integer getQuantidadeProduto() {
        return quantidadeProduto;
    }

    public void setQuantidadeProduto(Integer quantidadeProduto) {
        this.quantidadeProduto = quantidadeProduto;
    }

    public Float getPrecoVendaProduto() {
        return precoVendaProduto;
    }

    public void setPrecoVendaProduto(Float precoVendaProduto) {
        this.precoVendaProduto = precoVendaProduto;
    }

    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 + " ]";
    }

}
```

Sysdiagrams.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.Lob;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.Table;
import java.io.Serializable;

/**
 *
 * @author Andrey
 */
@Entity
@Table(name = "sysdiagrams")
@NamedQueries({
    @NamedQuery(name = "Sysdiagrams.findAll", query = "SELECT s FROM
Sysdiagrams s"),
    @NamedQuery(name = "Sysdiagrams.findByName", query = "SELECT s FROM
Sysdiagrams s WHERE s.name = :name"),
    @NamedQuery(name = "Sysdiagrams.findByPrincipalId", query = "SELECT s FROM
Sysdiagrams s WHERE s.principalId = :principalId"),
    @NamedQuery(name = "Sysdiagrams.findByDiagramId", query = "SELECT s FROM
Sysdiagrams s WHERE s.diagramId = :diagramId"),
    @NamedQuery(name = "Sysdiagrams.findByVersion", query = "SELECT s FROM
Sysdiagrams s WHERE s.version = :version")})
```

```java
public class Sysdiagrams implements Serializable {

    private static final long serialVersionUID = 1L;
    @Basic(optional = false)
    @Column(name = "name")
    private String name;
    @Basic(optional = false)
    @Column(name = "principal_id")
    private int principalId;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "diagram_id")
    private Integer diagramId;
    @Column(name = "version")
    private Integer version;
    @Lob
    @Column(name = "definition")
    private byte[] definition;

    public Sysdiagrams() {
    }

    public Sysdiagrams(Integer diagramId) {
        this.diagramId = diagramId;
    }

    public Sysdiagrams(Integer diagramId, String name, int principalId) {
        this.diagramId = diagramId;
        this.name = name;
        this.principalId = principalId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getPrincipalId() {
        return principalId;
    }

    public void setPrincipalId(int principalId) {
        this.principalId = principalId;
    }

    public Integer getDiagramId() {
        return diagramId;
    }

    public void setDiagramId(Integer diagramId) {
        this.diagramId = diagramId;
    }

    public Integer getVersion() {
        return version;
    }

    public void setVersion(Integer version) {
        this.version = version;
    }
```

```
    public byte[] getDefinition() {
        return definition;
    }

    public void setDefinition(byte[] definition) {
        this.definition = definition;
    }

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

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

}
```

Usuario.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 java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author Andrey
```

```
 */
@Entity
@Table(name = "usuario")
@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.findByLoginUsuario", query = "SELECT u FROM
Usuario u WHERE u.loginUsuario = :loginUsuario"),
    @NamedQuery(name = "Usuario.findBySenhaUsuario", query = "SELECT u FROM
Usuario u WHERE u.senhaUsuario = :senhaUsuario")})
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_usuario")
    private Integer idUsuario;
    @Column(name = "login_usuario")
    private String loginUsuario;
    @Column(name = "senha_usuario")
    private String senhaUsuario;
    @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 getLoginUsuario() {
        return loginUsuario;
    }

    public void setLoginUsuario(String loginUsuario) {
        this.loginUsuario = loginUsuario;
    }

    public String getSenhaUsuario() {
        return senhaUsuario;
    }

    public void setSenhaUsuario(String senhaUsuario) {
        this.senhaUsuario = senhaUsuario;
    }

    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 + " ]";
    }

}
```

Pacote cadastroee.controller

AbstractFacade.java

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java para
editar este modelo
 */
package cadastroee.controller;

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

/**
 * Classe abstrata que fornece métodos comuns para operações CRUD.
 * @param <T> Tipo da entidade
 * @author Andrey
 */
public abstract class AbstractFacade<T> {

    private final Class<T> entityClass;

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

    protected abstract EntityManager getEntityManager();

    /**
     * Cria uma nova entidade no banco de dados.
     * @param entity A entidade a ser criada
     */
```

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

    /**
     * Atualiza uma entidade no banco de dados.
     * @param entity A entidade a ser atualizada
     */
    public void edit(T entity) {
        getEntityManager().merge(entity);
    }

    /**
     * Remove uma entidade do banco de dados.
     * @param entity A entidade a ser removida
     */
    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }

    /**
     * Retorna uma entidade com base no ID fornecido.
     * @param id O ID da entidade
     * @return A entidade encontrada ou null se não encontrada
     */
    public T find(Object id) {
        return getEntityManager().find(entityClass, id);
    }

    /**
     * Retorna todas as entidades do banco de dados.
     * @return Uma lista de todas as entidades
     */
    public List<T> findAll() {
        jakarta.persistence.criteria.CriteriaQuery<T> cq =
getEntityManager().getCriteriaBuilder().createQuery(entityClass);
        cq.select(cq.from(entityClass));
        return getEntityManager().createQuery(cq).getResultList();
    }

    /**
     * Retorna uma lista de entidades dentro de um intervalo especificado.
     * @param range Um array de dois elementos representando o intervalo
     * @return Uma lista de entidades no intervalo especificado
     */
    public List<T> findRange(int[] range) {
        jakarta.persistence.criteria.CriteriaQuery<T> cq =
getEntityManager().getCriteriaBuilder().createQuery(entityClass);
        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();
    }

    /**
     * Retorna o número total de entidades no banco de dados.
     * @return O número total de entidades
     */
    public int count() {
        jakarta.persistence.criteria.CriteriaQuery<Long> cq =
getEntityManager().getCriteriaBuilder().createQuery(Long.class);
        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

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java para
editar este modelo
 */
package cadastroee.controller;

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

/**
 * Facade para a entidade Movimento, fornece métodos específicos além das
operações CRUD genéricas.
 * @author Andrey
 */
@jakarta.ejb.Stateless
public class MovimentoFacade extends AbstractFacade<Movimento> implements
MovimentoFacadeLocal {

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

    /**
     * Obtém o gerenciador de entidades associado a esta classe.
     * @return O gerenciador de entidades
     */
    @Override
    protected EntityManager getEntityManager() {
        return entityManager;
    }

    /**
     * Construtor padrão que define a classe da entidade para Movimento.
     */
    public MovimentoFacade() {
        super(Movimento.class);
    }

}
```

MovimentoFacadeLocal.java

```
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java
para editar este modelo
 */
package cadastroee.controller;

import cadastroee.model.Movimento;
import java.util.List;

/**
 * Interface que define os métodos para manipulação da entidade Movimento.
```

```
 * Essa interface é utilizada pelas classes que implementam operações
específicas para a entidade Movimento.
 * @author Andrey
 */
@jakarta.ejb.Local
public interface MovimentoFacadeLocal {

    /**
     * Cria um novo registro de movimento.
     * @param movimento O movimento a ser criado
     */
    void create(Movimento movimento);

    /**
     * Atualiza um registro de movimento existente.
     * @param movimento O movimento a ser atualizado
     */
    void edit(Movimento movimento);

    /**
     * Remove um registro de movimento.
     * @param movimento O movimento a ser removido
     */
    void remove(Movimento movimento);

    /**
     * Busca um movimento pelo ID.
     * @param id O ID do movimento a ser encontrado
     * @return O movimento encontrado, ou null se não existir
     */
    Movimento find(Object id);

    /**
     * Obtém todos os registros de movimento.
     * @return Uma lista contendo todos os movimentos
     */
    List<Movimento> findAll();

    /**
     * Obtém uma faixa específica de registros de movimento.
     * @param range Um array de inteiros representando a faixa desejada
     * @return Uma lista contendo os movimentos na faixa especificada
     */
    List<Movimento> findRange(int[] range);

    /**
     * Conta o número total de registros de movimento.
     * @return O número total de movimentos
     */
    int count();

}
```

PessoaFacade.java

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java para
editar este modelo
 */
package cadastroee.controller;
```

```java
import cadastroee.model.Pessoa;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 * Classe responsável por operações específicas relacionadas à entidade Pessoa.
 * Esta classe interage com o banco de dados para executar operações
relacionadas a Pessoa.
 * @author Andrey
 */
@jakarta.ejb.Stateless
public class PessoaFacade extends AbstractFacade<Pessoa> implements
PessoaFacadeLocal {

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

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

    /**
     * Construtor padrão da classe PessoaFacade.
     * Inicializa a classe base com a entidade Pessoa.
     */
    public PessoaFacade() {
        super(Pessoa.class);
    }

}
```

PessoaFacadeLocal.java

```java
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java
para editar este modelo
 */
package cadastroee.controller;

import cadastroee.model.Pessoa;
import java.util.List;

/**
 * Interface responsável por definir métodos para operações relacionadas à
entidade Pessoa.
 * Esta interface é utilizada para garantir a implementação padrão dos métodos
em classes que realizam operações com a entidade Pessoa.
 * @author Andrey
 */
@jakarta.ejb.Local
public interface PessoaFacadeLocal {

    /**
     * Cria uma nova entidade Pessoa no banco de dados.
     * @param pessoa A entidade Pessoa a ser criada.
     */
    void create(Pessoa pessoa);

    /**
     * Edita uma entidade Pessoa existente no banco de dados.
```

```
     * @param pessoa A entidade Pessoa a ser editada.
     */
    void edit(Pessoa pessoa);

    /**
     * Remove uma entidade Pessoa do banco de dados.
     * @param pessoa A entidade Pessoa a ser removida.
     */
    void remove(Pessoa pessoa);

    /**
     * Encontra uma entidade Pessoa no banco de dados com base no ID.
     * @param id O ID da entidade Pessoa a ser encontrada.
     * @return A entidade Pessoa encontrada, ou null se não encontrada.
     */
    Pessoa find(Object id);

    /**
     * Obtém todas as entidades Pessoa do banco de dados.
     * @return Uma lista contendo todas as entidades Pessoa.
     */
    List<Pessoa> findAll();

    /**
     * Obtém uma lista de entidades Pessoa dentro de uma faixa específica no
banco de dados.
     * @param range Um array de inteiros representando a faixa de resultados
desejada.
     * @return Uma lista contendo entidades Pessoa dentro da faixa
especificada.
     */
    List<Pessoa> findRange(int[] range);

    /**
     * Obtém o número total de entidades Pessoa no banco de dados.
     * @return O número total de entidades Pessoa.
     */
    int count();

}
```

PessoaFisicaFacade.java

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java para
editar este modelo
 */
package cadastroee.controller;

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

/**
 * Classe responsável por operações específicas para a entidade Pessoa Física.
 * Esta classe interage com o banco de dados para realizar operações
relacionadas à Pessoa Física.
 * @author Andrey
 */
@jakarta.ejb.Stateless
public class PessoaFisicaFacade extends AbstractFacade<PessoaFisica> implements
PessoaFisicaFacadeLocal {
```

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

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

    /**
     * Construtor padrão da classe PessoaFisicaFacade.
     * Inicializa a classe base com a entidade Pessoa Física.
     */
    public PessoaFisicaFacade() {
        super(PessoaFisica.class);
    }

}
```

PessoaFisicaFacadeLocal.java

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java
para editar este modelo
 */
package cadastroee.controller;

import cadastroee.model.PessoaFisica;
import java.util.List;

/**
 * Interface que define operações específicas para a entidade Pessoa Física.
 * Esta interface descreve métodos para interagir com o banco de dados e
realizar operações relacionadas à Pessoa Física.
 * @author Andrey
 */
@jakarta.ejb.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

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java para
editar este modelo
 */
package cadastroee.controller;

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

/**
 * Classe que realiza operações específicas para a entidade Pessoa Jurídica.
 * Esta classe é responsável por interagir com o banco de dados para realizar
operações relacionadas à Pessoa Jurídica.
 * @author Andrey
 */
@jakarta.ejb.Stateless
public class PessoaJuridicaFacade extends AbstractFacade<PessoaJuridica>
implements PessoaJuridicaFacadeLocal {

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

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

    /**
     * Construtor padrão da classe PessoaJuridicaFacade.
     * Inicializa a classe base com a entidade Pessoa Jurídica.
     */
    public PessoaJuridicaFacade() {
        super(PessoaJuridica.class);
    }

}
```

PessoaJuridicaFacadeLocal.java

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java
para editar este modelo
 */
package cadastroee.controller;

import cadastroee.model.PessoaJuridica;
import java.util.List;

/**
 * Interface que define operações específicas para a entidade Pessoa Jurídica.
 * Esta interface é utilizada para declarar métodos relacionados a Pessoa
Jurídica.
 * @author Andrey
 */
@jakarta.ejb.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();

}
```

ProdutoFacade.java

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java para
editar este modelo
 */
package cadastroee.controller;

import cadastroee.model.Produto;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 * Classe que realiza operações específicas para a entidade Produto.
 * Esta classe é responsável por interagir com o banco de dados para realizar
operações relacionadas a Produto.
 * @author Mari
 */
@jakarta.ejb.Stateless
public class ProdutoFacade extends AbstractFacade<Produto> implements
ProdutoFacadeLocal {

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

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

    /**
     * Construtor padrão da classe ProdutoFacade.
     * Inicializa a classe base com a entidade Produto.
     */
    public ProdutoFacade() {
        super(Produto.class);
    }

}
```

ProdutoFacadeLocal.java

```java
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java para editar este modelo
 */
package cadastroee.controller;

import cadastroee.model.Produto;
import java.util.List;

/**
 * Interface que define as operações disponíveis para a entidade Produto.
 * Essa interface lista os métodos CRUD (Create, Read, Update, Delete) para a entidade Produto.
 * @author Andrey
 */
@jakarta.ejb.Local
public interface ProdutoFacadeLocal {

    void create(Produto produto);

    void edit(Produto produto);

    void remove(Produto produto);

    Produto find(Object id);

    List<Produto> findAll();

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

    int count();

}
```

UsuarioFacade.java

```java
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java para editar este modelo
 */
package cadastroee.controller;

import cadastroee.model.Usuario;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 * Classe que realiza operações específicas para a entidade Usuario.
 * Esta classe é responsável por interagir com o banco de dados para realizar operações relacionadas a Usuario.
 * @author Andrey
 */
@jakarta.ejb.Stateless
public class UsuarioFacade extends AbstractFacade<Usuario> implements UsuarioFacadeLocal {

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

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

    /**
     * Construtor padrão da classe UsuarioFacade.
     * Inicializa a classe base com a entidade Usuario.
     */
    public UsuarioFacade() {
        super(Usuario.class);
    }

}
```

UsuarioFacadeLocal.java

```
/*
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt para alterar esta licença
 * Clique em nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java
para editar este modelo
 */
package cadastroee.controller;

import cadastroee.model.Usuario;
import java.util.List;

/**
 * Interface que define as operações disponíveis para a entidade Usuario.
 * Esta interface lista os métodos para criar, editar, remover, encontrar,
listar todos, listar em um intervalo e contar Usuarios.
 * @author Andrey
 */
@jakarta.ejb.Local
public interface UsuarioFacadeLocal {

    void create(Usuario usuario);

    void edit(Usuario usuario);

    void remove(Usuario usuario);

    Usuario find(Object id);

    List<Usuario> findAll();

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

    int count();

}
```

Persistence.xml

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

ServletProduto.java

```
package cadastroee.servlets;

import cadastroee.model.Produto;
import cadastroee.controller.ProdutoFacadeLocal;

import jakarta.ejb.EJB;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

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

    @EJB
    private ProdutoFacadeLocal facade;

    @Override
    protected void doGet(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 /CadastroEE-war</h1>");
            out.println("<p>Banana</p>");
            // Use o facade para obter a lista de produtos
            List<Produto> produtos = facade.findAll();

            // Apresente os produtos na forma de lista HTML
            out.println("<ul>");
            for (Produto produto : produtos) {
                out.println("<li>" + formatProduto(produto) + "</li>");
            }
            out.println("</ul>");

            out.println("</body>");
```

```
            out.println("</html>");
        }
    }

    // Método para formatar as informações do produto
    private String formatProduto(Produto produto) {
        // Ajuste conforme os métodos reais na classe Produto
        return produto.getNomeProduto() + " - R$" +
produto.getPrecoVendaProduto();
    }

    // Remova o método processRequest, pois não está sendo usado
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        // Se necessário, adicione lógica para processar solicitações POST
    }

    @Override
    public String getServletInfo() {
        return "Servlet para listar produtos";
    }
}
```

Resultado:
Execução da url: http://localhost:8080/CadastroEE-war/ServletProduto

Análise e Conclusão:

Como é organizado um projeto corporativo no NetBeans?

Um projeto corporativo no NetBeans é organizado em camadas, incluindo configuração de conexões, estruturação com projeto principal e subprojetos (EJB e web), camadas de persistência e controle (EJB), desenvolvimento web com Servlets, execução no GlassFish, e ajustes para a plataforma Jakarta EE 8. A organização segue princípios de modularidade e separação de camadas.

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

JPA (Java Persistence API): Gerencia a persistência de dados em um banco relacional, mapeando objetos Java para entidades do banco.

EJB (Enterprise JavaBeans): Facilita o desenvolvimento de componentes empresariais em Java, incluindo lógica de negócios, transações e segurança, para aplicações corporativas web

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

O NetBeans melhora a produtividade ao lidar com JPA e EJB oferecendo:

- Ferramentas visuais e assistência de código para mapeamento JPA.
- Navegação facilitada entre entidades e consultas.
- Geração automatizada de código para EJBs.
- Suporte integrado para depuração, teste e servidores de aplicação.
- Gerenciamento eficiente de dependências e integração com sistemas de controle de versão.

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

O NetBeans facilita a construção de Servlets, componentes Java para processamento de requisições web, oferecendo assistência de código, mapeamento simplificado, integração com ferramentas de depuração, suporte ao ciclo de vida do Servlet, e integração com outras tecnologias web. Isso simplifica o desenvolvimento, permitindo que os desenvolvedores se concentrem na lógica da aplicação.

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

A comunicação entre Servlets e Session Beans do pool de EJBs ocorre por meio da injeção de dependência. O Servlet usa anotações como @EJB para declarar a dependência, e o container Java EE gerencia a injeção, permitindo que o Servlet acesse os métodos do Session Bean diretamente. Essa abordagem promove uma separação clara de responsabilidades e facilita a manutenção da aplicação.