



Vamos integrar sistemas

Jonathan da Silva Araujo - 202205178111

Campus Nova Iguaçu

Vamos integrar sistemas (RPG0017) – 9001 – 2023.4

Objetivo da Prática

1. Implementar persistência com base em JPA.
2. Implementar regras de negócio na plataforma JEE, através de EJBs.
3. Implementar sistema cadastral Web com base em Servlets e JSPs.
4. Utilizar a biblioteca Bootstrap para melhoria do design.

1º Procedimento – Camadas de Persistência e Controle

Classe AbstractFacade:

```
package cadastroee.controller;
|
|
import jakarta.persistence.EntityManager;
import jakarta.persistence.Query;
import jakarta.persistence.criteria.CriteriaQuery;
import jakarta.persistence.criteria.Root;
import java.util.List;

6 usages 9 inheritors
public abstract class AbstractFacade<T> {
    no usages
    private Class<T> entityClass;

    no usages
    public AbstractFacade(Class<T> entityClass) { this.entityClass = entityClass; }

    12 usages 9 implementations
    protected abstract EntityManager getEntityManager();

    no usages
    public void create(T entity) { this.getEntityManager().persist(entity); }

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

    no usages
    public void remove(T entity) { this.getEntityManager().remove(this.getEntityManager().merge(entity)); }
```

```

no usages
public T find(Object id) { return this.getEntityManager().find(this.entityClass, id); }

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

no usages
public List<T> findRange(int[] range) {
    CriteriaQuery cq = this.getEntityManager().getCriteriaBuilder().createQuery();
    cq.select(cq.from(this.entityClass));
    Query q = this.getEntityManager().createQuery(cq);
    q.setMaxResults(range[1] - range[0] + 1);
    q.setFirstResult(range[0]);
    return q.getResultList();
}

no usages
public int count() {
    CriteriaQuery cq = this.getEntityManager().getCriteriaBuilder().createQuery();
    Root<T> rt = cq.from(this.entityClass);
    cq.select(this.getEntityManager().getCriteriaBuilder().count(rt));
    Query q = this.getEntityManager().createQuery(cq);
    return ((Long)q.getSingleResult()).intValue();
}
}

```

Classe MovimentosFacade:

```

6 package cadastroee.controller;
7
8 import cadastroee.model.Movimentos;
9 import jakarta.ejb.Stateless;
10 import jakarta.persistence.EntityManager;
11 import jakarta.persistence.PersistenceContext;
12
13 no usages
14 @Stateless
15 public class MovimentosFacade extends AbstractFacade<Movimentos> implements MovimentosFacadeLocal {
16     no usages
17     @PersistenceContext(
18         unitName = "CadastroEE-ejbPU"
19     )
20     private EntityManager em;
21
22     no usages
23     protected EntityManager getEntityManager() {
24         return this.em;
25     }
26
27     no usages
28     public MovimentosFacade() {
29         super(Movimentos.class);
30     }
31 }

```

Classe MovimentosFacadeLocal:

```

package cadastroee.controller;

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

1 usage 1 implementation
@Local
public interface MovimentosFacadeLocal {

    no usages 1 implementation
    void create(Movimentos movimentos);

    1 implementation
    void edit(Movimentos movimentos);

    no usages 1 implementation
    void remove(Movimentos movimentos);

    no usages 1 implementation
    Movimentos find(Object id);

    1 implementation
    List<Movimentos> findAll();

    no usages 1 implementation
    List<Movimentos> findRange(int[] range);

    no usages 1 implementation
    int count();
}

```

Classe PessoasFacade:

```

6 package cadastroee.controller;
7
8 import cadastroee.model.Pessoas;
9 import jakarta.ejb.Stateless;
10 import jakarta.persistence.EntityManager;
11 import jakarta.persistence.PersistenceContext;
12
13 no usages
14 @Stateless
15 public class PessoasFacade extends AbstractFacade<Pessoas> implements PessoasFacadeLocal {
16     no usages
17     @PersistenceContext(
18         unitName = "CadastroEE-ejbPU"
19     )
20     private EntityManager em;
21
22     no usages
23     protected EntityManager getEntityManager() {
24         return this.em;
25     }
26
27     no usages
28     public PessoasFacade() {
29         super(Pessoas.class);
30     }
31 }

```

Classe PessoasFacadeLocal:

```
6 package cadastroee.controller;
7
8 import cadastroee.model.Pessoas;
9 import jakarta.ejb.Local;
10 import java.util.List;
11
12 @Local
13 public interface PessoasFacadeLocal {
14     void create(Pessoas pessoas);
15
16     void edit(Pessoas pessoas);
17
18     void remove(Pessoas pessoas);
19
20     Pessoas find(Object id);
21
22     List<Pessoas> findAll();
23
24     List<Pessoas> findRange(int[] range);
25
26     int count();
27 }
```

Classe PessoasFisicasFacade:

```
package cadastroee.controller;

import ...

no usages
@Stateless
public class PessoasFisicasFacade extends AbstractFacade<PessoasFisicas> implements PessoasFisicasFacadeLocal {
    no usages
    @PersistenceContext(
        unitName = "CadastroEE-ejbPU"
    )
    private EntityManager em;

    no usages
    protected EntityManager getEntityManager() {
        return this.em;
    }

    no usages
    public PessoasFisicasFacade() {
        super(PessoasFisicas.class);
    }
}
```

Classe PessoasFisicasFacadeLocal:

```
package cadastroee.controller;

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

1 usage 2 implementations
@Local
public interface PessoasFisicasFacadeLocal {

    no usages 2 implementations
    void create(PessoasFisicas pessoasFisicas);

    2 implementations
    void edit(PessoasFisicas pessoasFisicas);

    no usages 2 implementations
    void remove(PessoasFisicas pessoasFisicas);

    no usages 2 implementations
    PessoasFisicas find(Object id);

    2 implementations
    List<PessoasFisicas> findAll();

    no usages 2 implementations
    List<PessoasFisicas> findRange(int[] range);

    no usages 2 implementations
    int count();
}
```

Classe PessoasJuridicasFacade:

```
package cadastroee.controller;

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

no usages
@Stateless
public class PessoasJuridicasFacade extends AbstractFacade<PessoasJuridicas> implements PessoasJuridicasFacadeLocal {

    no usages
    @PersistenceContext(
        unitName = "CadastroEE-ejbPU"
    )
    private EntityManager em;

    no usages
    protected EntityManager getEntityManager() {
        return this.em;
    }

    no usages
    public PessoasJuridicasFacade() {
        super(PessoasJuridicas.class);
    }
}
```

Classe PessoasJuridicasFacadeLocal:

```
6 package cadastroee.controller;
7
8 import cadastroee.model.PessoasJuridicas;
9 import jakarta.ejb.Local;
10 import java.util.List;
11
12 1 usage 1 implementation
13 @Local
14 public interface PessoasJuridicasFacadeLocal {
15     no usages 1 implementation
16     void create(PessoasJuridicas pessoasJuridicas);
17
18     1 implementation
19     void edit(PessoasJuridicas pessoasJuridicas);
20
21     no usages 1 implementation
22     void remove(PessoasJuridicas pessoasJuridicas);
23
24     no usages 1 implementation
25     PessoasJuridicas find(Object id);
26
27     1 implementation
28     List<PessoasJuridicas> findAll();
29
30     no usages 1 implementation
31     List<PessoasJuridicas> findRange(int[] range);
32
33     no usages 1 implementation
34     int count();
35 }
```

Classe ProdutosFacade:

```
6 package cadastroee.controller;
7
8 import cadastroee.model.Produtos;
9 import jakarta.ejb.Stateless;
10 import jakarta.persistence.EntityManager;
11 import jakarta.persistence.PersistenceContext;
12
13 no usages
14 @Stateless
15 public class ProdutosFacade extends AbstractFacade<Produtos> implements ProdutosFacadeLocal {
16     no usages
17     @PersistenceContext(
18         unitName = "CadastroEE-ejbPU"
19     )
20     private EntityManager em;
21
22     no usages
23     protected EntityManager getEntityManager() {
24         return this.em;
25     }
26
27     no usages
28     public ProdutosFacade() {
29         super(Produtos.class);
30     }
31 }
```

Classe ProdutosFacadeLocal:

```

6 package cadastroee.controller;
7
8 import cadastroee.model.Produtos;
9 import jakarta.ejb.Local;
10 import java.util.List;
11
12 5 usages 2 implementations
13 @Local
14 public interface ProdutosFacadeLocal {
15     1 usage 2 implementations
16     void create(Produtos produtos);
17
18     2 implementations
19     void edit(Produtos produtos);
20
21     1 usage 2 implementations
22     void remove(Produtos produtos);
23
24     3 usages 2 implementations
25     Produtos find(Object id);
26
27     2 implementations
28     List<Produtos> findAll();
29
30     no usages 2 implementations
31     List<Produtos> findRange(int[] range);
32
33     no usages 2 implementations
34     int count();
35 }

```

Classe UsuariosFacade:

```
package cadastroee.controller;

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

no usages
@Stateless
public class UsuariosFacade extends AbstractFacade<Usuarios> implements UsuariosFacadeLocal {
    no usages
    @PersistenceContext(
        unitName = "CadastroEE-ejbPU"
    )
    private EntityManager em;

    no usages
    protected EntityManager getEntityManager() {
        return this.em;
    }

    no usages
    public UsuariosFacade() {
        super(Usuarios.class);
    }
}
```

Classe UsuariosFacadeLocal:

```
package cadastroee.controller;

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

1 usage 1 implementation
@Local
public interface UsuariosFacadeLocal {

    no usages 1 implementation
    void create(Usuarios usuarios);

    1 implementation
    void edit(Usuarios usuarios);

    no usages 1 implementation
    void remove(Usuarios usuarios);

    no usages 1 implementation
    Usuarios find(Object id);

    1 implementation
    List<Usuarios> findAll();

    no usages 1 implementation
    List<Usuarios> findRange(int[] range);

    no usages 1 implementation
    int count();
}
```

Classe Movimentos:


```
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.ManyToOne;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.Table;
import java.io.Serializable;
import java.math.BigDecimal;
```

22 usages

```
@Entity
@Table(
    name = "Movimentos"
)
@NamedQueries({@NamedQuery(
    name = "Movimentos.findAll",
    query = "SELECT m FROM Movimentos m"
), @NamedQuery(
    name = "Movimentos.findByIdMovimentos",
    query = "SELECT m FROM Movimentos m WHERE m.idMovimentos = :idMovimentos"
), @NamedQuery(
    name = "Movimentos.findByQuantidade",
    query = "SELECT m FROM Movimentos m WHERE m.quantidade = :quantidade"
), @NamedQuery(
    name = "Movimentos.findByTipo",
    query = "SELECT m FROM Movimentos m WHERE m.tipo = :tipo"
), @NamedQuery(
    name = "Movimentos.findByPrecoUnitario",
    query = "SELECT m FROM Movimentos m WHERE m.precoUnitario = :precoUnitario"
)})
```

```
public class Movimentos implements Serializable {
```

no usages

```
private static final long serialVersionUID = 1L;
```

no usages

```
@Id
```

```
@Basic(
```

```
    optional = false
```

```
)
```

```
@Column(
```

```
    name = "idMovimentos"
```

```
)
```

```
private Integer idMovimentos;
```

no usages

```
@Column(
```

```
    name = "Quantidade"
```

```
)
```

```
private Integer quantidade;
```

no usages

```
@Column(
```

```
    name = "Tipo"
```

```
)
```

```
private Character tipo;
```

no usages

```
@Column(
```

```
    name = "PrecoUnitario"
```

```

    )
    private BigDecimal precoUnitario;
    no usages
    @JoinColumn(
        name = "idPessoa",
        referencedColumnName = "idPessoa"
    )
    @ManyToOne
    private Pessoas idPessoa;
    no usages
    @JoinColumn(
        name = "idProduto",
        referencedColumnName = "idProduto"
    )
    @ManyToOne
    private Produtos idProduto;
    no usages
    @JoinColumn(
        name = "idUserario",
        referencedColumnName = "idUserario"
    )
    @ManyToOne
    private Usuarios idUsuario;
    public Movimentos(Integer idMovimentos) {
        this.idMovimentos = idMovimentos;
    }

    no usages
    public Integer getIdMovimentos() {
        return this.idMovimentos;
    }

    no usages
    public void setIdMovimentos(Integer idMovimentos) {
        this.idMovimentos = idMovimentos;
    }

    no usages
    public Integer getQuantidade() {
        return this.quantidade;
    }

    no usages
    public void setQuantidade(Integer quantidade) {
        this.quantidade = quantidade;
    }

    no usages
    public Character getTipo() {
        return this.tipo;
    }
}

```

```

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

    no usages
    public BigDecimal getPrecoUnitario() {
        return this.precoUnitario;
    }

    no usages
    public void setPrecoUnitario(BigDecimal precoUnitario) {
        this.precoUnitario = precoUnitario;
    }

    no usages
    public Pessoas getIdPessoa() {
        return this.idPessoa;
    }

    no usages
    public void setIdPessoa(Pessoas idPessoa) {
        this.idPessoa = idPessoa;
    }

    no usages
    public Produtos getIdProduto() {
        return this.idProduto;
    }

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

    no usages
    public Usuarios getIdUsuario() {
        return this.idUsuario;
    }

    no usages
    public void setIdUsuario(Usuarios idUsuario) {
        this.idUsuario = idUsuario;
    }

    no usages
    public int hashCode() {
        int hash = 0;
        hash += this.idMovimentos != null ? this.idMovimentos.hashCode() : 0;
        return hash;
    }

```

```

    public boolean equals(Object object) {
        if (!(object instanceof Movimentos other)) {
            return false;
        } else {
            return (this.idMovimentos != null || other.idMovimentos == null) && (this.idMovimentos == null || this.idMovimentos.equals(other.idMovimentos));
        }
    }

    no usages
    public String toString() {
        return "cadastroee.model.Movimentos[ idMovimentos=" + this.idMovimentos + " ]";
    }
}

```

Classe Pessoas:

```

6 package cadastroee.model;
7
8 import jakarta.persistence.Basic;
9 import jakarta.persistence.CascadeType;
10 import jakarta.persistence.Column;
11 import jakarta.persistence.Entity;
12 import jakarta.persistence.Id;
13 import jakarta.persistence.NamedQueries;
14 import jakarta.persistence.NamedQuery;
15 import jakarta.persistence.OneToMany;
16 import jakarta.persistence.OneToOne;
17 import jakarta.persistence.Table;
18 import java.io.Serializable;
19 import java.util.Collection;
20
21 2 usages
22 @Entity
23 @Table(
24     name = "Pessoas"
25 )
26 @NamedQueries({@NamedQuery(
27     name = "Pessoas.findAll",
28     query = "SELECT p FROM Pessoas p"
29 ), @NamedQuery(
30     name = "Pessoas.findByIdPessoa",
31     query = "SELECT p FROM Pessoas p WHERE p.idPessoa = :idPessoa"
32 ), @NamedQuery(
33     name = "Pessoas.findByName",
34     query = "SELECT p FROM Pessoas p WHERE p.nome = :nome"
35 ), @NamedQuery(
36     name = "Pessoas.findByCidade",

```

```

36         query = "SELECT p FROM Pessoas p WHERE p.cidade = :cidade"
37     ), @NamedQuery(
38         name = "Pessoas.findByEstado",
39         query = "SELECT p FROM Pessoas p WHERE p.estado = :estado"
40     ), @NamedQuery(
41         name = "Pessoas.findByTelefone",
42         query = "SELECT p FROM Pessoas p WHERE p.telefone = :telefone"
43     ), @NamedQuery(
44         name = "Pessoas.findByEmail",
45         query = "SELECT p FROM Pessoas p WHERE p.email = :email"
46     ), @NamedQuery(
47         name = "Pessoas.findByLogradouro",
48         query = "SELECT p FROM Pessoas p WHERE p.logradouro = :logradouro"
49     )})
50     public class Pessoas implements Serializable {
51         no usages
52         private static final long serialVersionUID = 1L;
53         no usages
54         @Id
55         @Basic(
56             optional = false
57         )
58         @Column(
59             name = "idPessoa"
60         )
61         private Integer idPessoa;
62         no usages
63         @Column(
64             name = "nome"
65         )
66         private String nome;

```

```
64     @Column(  
65         name = "cidade"  
66     )  
67     private String cidade;  
68     no usages  
69     @Column(  
70         name = "estado"  
71     )  
72     private String estado;  
73     no usages  
74     @Column(  
75         name = "telefone"  
76     )  
77     private String telefone;  
78     no usages  
79     @Column(  
80         name = "email"  
81     )  
82     private String email;  
83     no usages  
84     @Column(  
85         name = "logradouro"  
86     )  
87     private String logradouro;  
88     no usages  
89     @OneToOne(  
90         cascade = {CascadeType.ALL},  
91         mappedBy = "pessoas"  
92     )  
93     private PessoasJuridicas pessoasJuridicas;
```

```

    @OneToMany(
        mappedBy = "idPessoa"
    )
    private Collection<Movimentos> movimentosCollection;
    no usages
    @OneToOne(
        cascade = {CascadeType.ALL},
        mappedBy = "pessoas"
    )
    private PessoasFisicas pessoasFisicas;

    no usages
    public Pessoas() {
    }

    no usages
    public Pessoas(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    no usages
    public Integer getIdPessoa() {
        return this.idPessoa;
    }

    no usages
    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

```

```
114 public String getNome() {
115     return this.nome;
116 }
117
118 no usages
118 public void setNome(String nome) {
119     this.nome = nome;
120 }
121
122 no usages
122 public String getCidade() {
123     return this.cidade;
124 }
125
126 no usages
126 public void setCidade(String cidade) {
127     this.cidade = cidade;
128 }
129
130 no usages
130 public String getEstado() {
131     return this.estado;
132 }
133
134 no usages
134 public void setEstado(String estado) {
135     this.estado = estado;
136 }
138 public String getTelefone() {
139     return this.telefone;
140 }
141
142 no usages
142 public void setTelefone(String telefone) {
143     this.telefone = telefone;
144 }
145
146 no usages
146 public String getEmail() {
147     return this.email;
148 }
149
150 no usages
150 public void setEmail(String email) {
151     this.email = email;
152 }
153
154 no usages
154 public String getLogradouro() {
155     return this.logradouro;
156 }
157
158 no usages
158 public void setLogradouro(String logradouro) {
159     this.logradouro = logradouro;
160 }
```



```

162     public PessoasJuridicas getPessoasJuridicas() {
163         return this.pessoasJuridicas;
164     }
165
166     no usages
167     public void setPessoasJuridicas(PessoasJuridicas pessoasJuridicas) {
168         this.pessoasJuridicas = pessoasJuridicas;
169     }
170
171     no usages
172     public Collection<Movimentos> getMovimentosCollection() {
173         return this.movimentosCollection;
174     }
175
176     no usages
177     public void setMovimentosCollection(Collection<Movimentos> movimentosCollection) {
178         this.movimentosCollection = movimentosCollection;
179     }
180
181     no usages
182     public PessoasFisicas getPessoasFisicas() {
183         return this.pessoasFisicas;
184     }
185
186     no usages
187     public void setPessoasFisicas(PessoasFisicas pessoasFisicas) {
188         this.pessoasFisicas = pessoasFisicas;
189     }
190
191     public int hashCode() {
192         int hash = 0;
193         hash += this.idPessoa != null ? this.idPessoa.hashCode() : 0;
194         return hash;
195     }
196
197     no usages
198     public boolean equals(Object object) {
199         if (!(object instanceof Pessoas other)) {
200             return false;
201         } else {
202             return (this.idPessoa != null || other.idPessoa == null) && (this.idPessoa == null || this.idPessoa.equals(other.idPessoa));
203         }
204     }
205
206     no usages
207     public String toString() { return "cadastroee.model.Pessoas[ idPessoa=" + this.idPessoa + " ]"; }
208 }

```

Classe PessoasFisicas:

```
6 package cadastroee.model;
7
8 import jakarta.persistence.Basic;
9 import jakarta.persistence.Column;
10 import jakarta.persistence.Entity;
11 import jakarta.persistence.Id;
12 import jakarta.persistence.JoinColumn;
13 import jakarta.persistence.NamedQueries;
14 import jakarta.persistence.NamedQuery;
15 import jakarta.persistence.OneToOne;
16 import jakarta.persistence.Table;
17 import java.io.Serializable;
18
19 @Entity
20 @Table(
21     name = "PessoasFisicas"
22 )
23 @NamedQueries({@NamedQuery(
24     name = "PessoasFisicas.findAll",
25     query = "SELECT p FROM PessoasFisicas p"
26 ), @NamedQuery(
27     name = "PessoasFisicas.findByIdPFisica",
28     query = "SELECT p FROM PessoasFisicas p WHERE p.idPFisica = :idPFisica"
29 ), @NamedQuery(
30     name = "PessoasFisicas.findByCpf",
31     query = "SELECT p FROM PessoasFisicas p WHERE p.cpf = :cpf"
32 )})
```

```

33 public class PessoasFisicas implements Serializable {
    no usages
34     private static final long serialVersionUID = 1L;
    no usages
35     @Id
36     @Basic(
37         optional = false
38     )
39     @Column(
40         name = "idPFisica"
41     )
42     private Integer idPFisica;
    no usages
43     @Column(
44         name = "cpf"
45     )
46     private String cpf;
    no usages
47     @JoinColumn(
48         name = "idPFisica",
49         referencedColumnName = "idPessoa",
50         insertable = false,
51         updatable = false
52     )
53     @OneToOne(
54         optional = false
55     )
56     private Pessoas pessoas;

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

    no usages
    public Integer getIdPFisica() {
        return this.idPFisica;
    }

    no usages
    public void setIdPFisica(Integer idPFisica) {
        this.idPFisica = idPFisica;
    }

    no usages
    public String getCpf() {
        return this.cpf;
    }

    no usages
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    no usages
    public Pessoas getPessoas() {
        return this.pessoas;
    }
}

```

```

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

    no usages
    public int hashCode() {
        int hash = 0;
        hash += this.idPFisica != null ? this.idPFisica.hashCode() : 0;
        return hash;
    }

    no usages
    public boolean equals(Object object) {
        if (!(object instanceof PessoasFisicas other)) {
            return false;
        } else {
            return (this.idPFisica != null || other.idPFisica == null) && (this.idPFisica == null || this.idPFisica.equals(other.idPFisica));
        }
    }

    no usages
    public String toString() { return "cadastroee.model.PessoasFisicas[ idPFisica=" + this.idPFisica + " ]"; }
}

```

Classe PessoasJuridicas:

```

6 package cadastroee.model;
7
8 import jakarta.persistence.Basic;
9 import jakarta.persistence.Column;
10 import jakarta.persistence.Entity;
11 import jakarta.persistence.Id;
12 import jakarta.persistence.JoinColumn;
13 import jakarta.persistence.NamedQueries;
14 import jakarta.persistence.NamedQuery;
15 import jakarta.persistence.OneToOne;
16 import jakarta.persistence.Table;
17 import java.io.Serializable;
18
19 @Entity
20 @Table(
21     name = "PessoasJuridicas"
22 )
23 @NamedQueries({@NamedQuery(
24     name = "PessoasJuridicas.findAll",
25     query = "SELECT p FROM PessoasJuridicas p"
26 ), @NamedQuery(
27     name = "PessoasJuridicas.findByIdPJuridica",
28     query = "SELECT p FROM PessoasJuridicas p WHERE p.idPJuridica = :idPJuridica"
29 ), @NamedQuery(
30     name = "PessoasJuridicas.findByCnpj",
31     query = "SELECT p FROM PessoasJuridicas p WHERE p.cnpj = :cnpj"
32 )})

```

```

33 public class PessoasJuridicas implements Serializable {
    no usages
34     private static final long serialVersionUID = 1L;
    no usages
35     @Id
36     @Basic(
37         optional = false
38     )
39     @Column(
40         name = "idPJuridica"
41     )
42     private Integer idPJuridica;
    no usages
43     @Column(
44         name = "cnpj"
45     )
46     private String cnpj;
    no usages
47     @JoinColumn(
48         name = "idPJuridica",
49         referencedColumnName = "idPessoa",
50         insertable = false,
51         updatable = false
52     )
53     @OneToOne(
54         optional = false
55     )
56     private Pessoas pessoas;
    public PessoasJuridicas(Integer idPJuridica) {
        this.idPJuridica = idPJuridica;
    }

    no usages
    public Integer getIdPJuridica() {
        return this.idPJuridica;
    }

    no usages
    public void setIdPJuridica(Integer idPJuridica) {
        this.idPJuridica = idPJuridica;
    }

    no usages
    public String getCnpj() {
        return this.cnpj;
    }

    no usages
    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    no usages
    public Pessoas getPessoas() {
        return this.pessoas;
    }
}

```

```

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

    no usages
    public int hashCode() {
        int hash = 0;
        hash += this.idPJuridica != null ? this.idPJuridica.hashCode() : 0;
        return hash;
    }

    no usages
    public boolean equals(Object object) {
        if (!(object instanceof PessoasJuridicas other)) {
            return false;
        } else {
            return (this.idPJuridica != null || other.idPJuridica == null) && (this.idPJuridica == null || this.idPJuridica.equals(other.idPJuridica));
        }
    }

    no usages
    public String toString() {
        return "cadastroee.model.PessoasJuridicas[ idPJuridica=" + this.idPJuridica + " ]";
    }
}

```

Classe Produtos:

```

package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import java.io.Serializable;
import java.util.Collection;

4 usages
@Entity
@Table(
    name = "Produtos"
)
@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.findByName",
    query = "SELECT p FROM Produtos p WHERE p.nome = :nome"

```

```
    ), @NamedQuery(  
        name = "Produtos.findByQuantidade",  
        query = "SELECT p FROM Produtos p WHERE p.quantidade = :quantidade"  
    ), @NamedQuery(  
        name = "Produtos.findByPrecoVenda",  
        query = "SELECT p FROM Produtos p WHERE p.precoVenda = :precoVenda"  
    ))  
public class Produtos implements Serializable {  
    no usages  
    private static final long serialVersionUID = 1L;  
    no usages  
    @Id  
    @Basic(  
        optional = false  
    )  
    @Column(  
        name = "idProduto"  
    )  
    private Integer idProduto;  
    no usages  
    @Column(  
        name = "nome"  
    )  
    private String nome;  
    no usages  
    @Column(  
        name = "quantidade"  
    )  
    private Integer quantidade;
```

```

@Column(
    name = "precoVenda"
)
private float precoVenda;

no usages
@OneToMany(
    mappedBy = "idProduto"
)
private Collection<Movimentos> movimentosCollection;

no usages
public Produtos() {
}

no usages
public Produtos(Integer idProduto) {
    this.idProduto = idProduto;
}

1 usage
public Integer getIdProduto() {
    return this.idProduto;
}

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

public String getNome() {
    return this.nome;
}

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

1 usage
public Integer getQuantidade() {
    return this.quantidade;
}

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

1 usage
public float getPrecoVenda() {
    return this.precoVenda;
}

2 usages
public void setPrecoVenda(float precoVenda) {
    this.precoVenda = precoVenda;
}

```



```

3   public Collection<Movimentos> getMovimentosCollection() {
4       return this.movimentosCollection;
5   }

no usages
3   public void setMovimentosCollection(Collection<Movimentos> movimentosCollection) {
4       this.movimentosCollection = movimentosCollection;
5   }

no usages
3   public int hashCode() {
4       int hash = 0;
5       hash += this.idProduto != null ? this.idProduto.hashCode() : 0;
6       return hash;
7   }

no usages
3   public boolean equals(Object object) {
4       if (!(object instanceof Produtos other)) {
5           return false;
6       } else {
7           return (this.idProduto != null || other.idProduto == null) && (this.idProduto == null || this.idProduto.equals(other.idProduto));
8       }
9   }

no usages
3   public String toString() { return "cadastroee.model.Produtos[ idProduto=" + this.idProduto + " ]"; }
}

```

Classe Usuarios:

```

package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import java.io.Serializable;
import java.util.Collection;

2 usages
@Entity
@Table(
    name = "Usuarios"
)
@NamedQueries({@NamedQuery(
    name = "Usuarios.findAll",
    query = "SELECT u FROM Usuarios u"
), @NamedQuery(
    name = "Usuarios.findByIdUsuario",
    query = "SELECT u FROM Usuarios u WHERE u.idUsuario = :idUsuario"
), @NamedQuery(
    name = "Usuarios.findByLogin",
    query = "SELECT u FROM Usuarios u WHERE u.login = :login"
), @NamedQuery(
    name = "Usuarios.findBySenha",
    query = "SELECT u FROM Usuarios u WHERE u.senha = :senha"
)})
}

```

```

public class Usuarios implements Serializable {
    no usages
    private static final long serialVersionUID = 1L;
    no usages
    @Id
    @Basic(
        optional = false
    )
    @Column(
        name = "idUseruario"
    )
    private Integer idUsuario;
    no usages
    @Column(
        name = "login"
    )
    private String login;
    no usages
    @Column(
        name = "senha"
    )
    private String senha;
    no usages
    @OneToMany(
        mappedBy = "idUseruario"
    )
    private Collection<Movimentos> movimentosCollection;
}

```

```

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

    no usages
    public Integer getIdUsuario() {
        return this.idUsuario;
    }

    no usages
    public void setIdUsuario(Integer idUsuario) {
        this.idUsuario = idUsuario;
    }

    no usages
    public String getLogin() {
        return this.login;
    }

    no usages
    public void setLogin(String login) {
        this.login = login;
    }

    no usages
    public String getSenha() {
        return this.senha;
    }

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

    no usages
    public Collection<Movimentos> getMovimentosCollection() {
        return this.movimentosCollection;
    }

    no usages
    public void setMovimentosCollection(Collection<Movimentos> movimentosCollection) {
        this.movimentosCollection = movimentosCollection;
    }

    no usages
    public int hashCode() {
        int hash = 0;
        hash += this.idUsuario != null ? this.idUsuario.hashCode() : 0;
        return hash;
    }

    no usages
    public boolean equals(Object object) {
        if (!(object instanceof Usuarios other)) {
            return false;
        } else {
            return (this.idUsuario != null || other.idUsuario == null) && (this.idUsuario == null || this.idUsuario.equals(other.idUsuario));
        }
    }

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

```

ServletProduto.java:

```
package cadastroee.servlets;

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

no usages
public class ServletProduto extends HttpServlet {
    no usages
    @EJB
    ProdutosFacadeLocal facade;

    9 usages
    public ServletProduto() {}

    2 usages
    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        try {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet ServletProduto</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet ServletProduto at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        } catch (Throwable var7) {
            if (out != null) {
                try {
                    out.close();
                } catch (Throwable var6) {
                    var7.addSuppressed(var6);
                }
            }

            throw var7;
        }

        if (out != null) {
            out.close();
        }
    }

    no usages
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        List<Produtos> produtos = this.facade.findAll();
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
    }
}
```

```

    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet ServletProduto</title>");
        out.println("<style>");
        out.println("ul { list-style: none; padding-left: 0; }");
        out.println("</style>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet ServletProduto at " + request.getContextPath() + "</h1>");
        out.println("<ul>");
        Iterator var5 = produtos.iterator();

        while(true) {
            if (!var5.hasNext()) {
                out.println("</ul>");
                out.println("</body>");
                out.println("</html>");
                break;
            }

            Produtos produto = (Produtos)var5.next();
            out.println("<li>" + produto.getNome() + "</li>");
        }
    } catch (Throwable var8) {
        if (out != null) {
            try {
                out.close();
            } catch (Throwable var7) {
                var8.addSuppressed(var7);
            }
        }
    }

    throw var8;
}

if (out != null) {
    out.close();
}

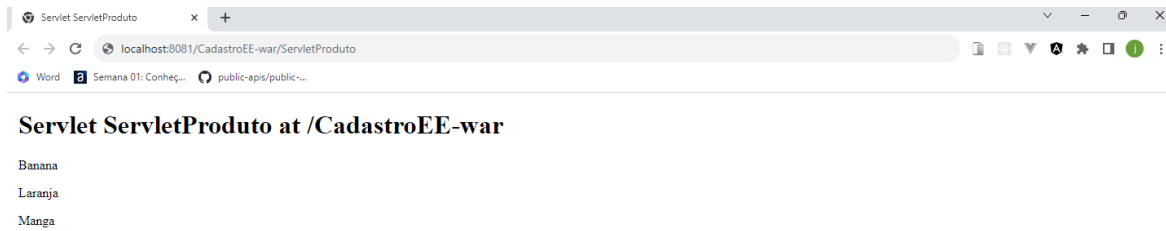
this.processRequest(request, response);
}

no usages
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    this.processRequest(request, response);
}

no usages
public String getServletInfo() {
    return "Short description";
}
}

```

Resultado da execução:



- a) Como é organizado um projeto corporativo no NetBeans?

Em um ambiente corporativo usando o NetBeans, é comum estruturar um projeto em módulos interdependentes que representam várias partes da aplicação. Tipicamente, existe um módulo central (EAR) que reúne outros módulos (EJBs, WARs) responsáveis por gerenciar aspectos como armazenamento de dados, lógica de negócios e interface web. Essa abordagem simplifica o processo de desenvolvimento baseado em módulos e a posterior implantação em servidores de aplicativos.

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

A utilização das tecnologias JPA (Java Persistence API) e EJB (Enterprise JavaBeans) é fundamental no processo de desenvolvimento de aplicações web em Java. O JPA estabelece um padrão que simplifica a tarefa de mapear objetos Java para tabelas de banco de dados, tornando mais fácil a persistência de dados. Por outro lado, o EJB é um componente que disponibiliza funcionalidades como gerenciamento de transações e acesso remoto, promovendo a escalabilidade e a reutilização de lógica empresarial em aplicações corporativas. Ambas essas tecnologias desempenham um papel significativo na construção eficaz e robusta de aplicações na plataforma web Java.

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

O NetBeans fornece suporte integrado e ferramentas visuais que tornam mais simples o desenvolvimento com tecnologias JPA e EJB, simplificando a criação, implementação e gerenciamento dessas funcionalidades. Seus recursos, incluindo a geração automática de código, o mapeamento visual de entidades e as capacidades de depuração avançada, aumentam a produtividade dos desenvolvedores ao lidar com desafios complexos relacionados à persistência de dados e à lógica de negócios em aplicações corporativas.

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

São componentes Java que lidam com solicitações e respostas HTTP, sendo essenciais para adicionar dinamismo a aplicativos web. O NetBeans simplifica o

processo de criação, configuração e implantação de Servlets em projetos web por meio de modelos e assistentes. Isso resulta em uma aceleração no desenvolvimento de interfaces interativas e dinâmicas para os usuários.

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

A comunicação entre Servlets e Session Beans do pool de EJBs é estabelecida usando injeção de dependência. O NetBeans simplifica essa integração por meio de anotações e assistentes que tornam acessíveis as interfaces dos Session Beans para os Servlets de maneira direta e eficaz. Isso permite que a lógica de negócios encapsulada nos EJBs seja executada a partir das ações desencadeadas pelos Servlets de forma fácil e eficiente.

2º Procedimento – Interface Cadastral com Servlet e JSPs

ServletProdutoFC.java:

```
5 package cadastroee.servlets;
6
7 import java.io.IOException;
8 import jakarta.servlet.ServletException;
9 import jakarta.servlet.annotation.WebServlet;
10 import jakarta.servlet.http.HttpServlet;
11 import jakarta.servlet.http.HttpServletRequest;
12 import jakarta.servlet.http.HttpServletResponse;
13 import jakarta.servlet.RequestDispatcher;
14 import jakarta.ejb.EJB;
15 import cadastroee.controller.ProdutosFacadeLocal;
16 import cadastroee.model.Produtos;
17 import java.util.List;
18
19 @WebServlet(name = "ServletProdutoFC", urlPatterns = {"/ServletProdutoFC"})
20 public class ServletProdutoFC extends HttpServlet {
21     @EJB
22     ProdutosFacadeLocal facade;
23
24     /**
25      * Processes requests for both HTTP GET and POST
26      * methods.
27      *
28      * @param request servlet request
29      * @param response servlet response
30      * @throws ServletException if a servlet-specific error occurs
31      * @throws IOException if an I/O error occurs
32      */
33     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
```

```

32         throws ServletException, IOException {
33     }
34     // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
35     /**
36      * Handles the HTTP <code>GET</code> method.
37      *
38      * @param request servlet request
39      * @param response servlet response
40      * @throws ServletException if a servlet-specific error occurs
41      * @throws IOException if an I/O error occurs
42      */
43     no usages
44     @Override
45     @
46     protected void doGet(HttpServletRequest request, HttpServletResponse response)
47         throws ServletException, IOException {
48         String acao = request.getParameter("acao");
49         String destino = null;
50
51         switch (acao) {
52             case "listar":
53                 List<Produtos> produtos = facade.findAll();
54                 request.setAttribute("produtos", produtos);
55                 destino = "ProdutoLista.jsp";
56                 break;
57             case "formIncluir":
58                 destino = "ProdutoDados.jsp";
59                 break;
60             case "formAlterar":
61                 int idAlterar = Integer.parseInt(request.getParameter("id"));
62                 Produtos produtoAlterar = facade.find(idAlterar);
63                 request.setAttribute("produto", produtoAlterar);
64                 destino = "ProdutoDados.jsp";
65                 break;
66             case "excluir":
67                 int idExcluir = Integer.parseInt(request.getParameter("id"));
68                 Produtos produtoExcluir = facade.find(idExcluir);
69                 facade.remove(produtoExcluir);
70                 List<Produtos> produtosExcluidos = facade.findAll();
71                 request.setAttribute("produtos", produtosExcluidos);
72                 destino = "ProdutoLista.jsp";
73                 break;
74             default:
75                 break;
76         }
77         RequestDispatcher rd = request.getRequestDispatcher(destino);
78         rd.forward(request, response);
79     }
80     /**
81      * Handles the HTTP <code>POST</code> method.
82      *
83      * @param request servlet request
84      * @param response servlet response
85      * @throws ServletException if a servlet-specific error occurs
86      * @throws IOException if an I/O error occurs
87      */
88     no usages
89     @Override
90     @
91     protected void doPost(HttpServletRequest request, HttpServletResponse response)
92         throws ServletException, IOException {
93         String acao = request.getParameter("acao");

```



```

90     List<Produtos> produtos = null;
91
92     switch (acao) {
93         case "incluir":
94             String nome = request.getParameter("nome");
95             int quantidade = Integer.parseInt(request.getParameter("quantidade"));
96             float preco = Float.parseFloat(request.getParameter("preco"));
97             produtos = facade.findAll();
98             int ultimoId = 0;
99             Produtos ultimoProduto = produtos.get(produtos.size() - 1);
100             ultimoId = ultimoProduto.getIdProduto();
101             int id = ultimoId + 1;
102
103             Produtos novoProduto = new Produtos();
104             novoProduto.setNome(nome);
105             novoProduto.setQuantidade(quantidade);
106             novoProduto.setPrecoVenda(preco);
107             novoProduto.setIdProduto(id);
108
109             facade.create(novoProduto);
110
111             produtos = facade.findAll();
112             request.setAttribute("produtos", produtos);
113
114             break;
115
116         case "alterar":
117             int idAlterar = Integer.parseInt(request.getParameter("id"));
118             Produtos produtoExistente = facade.find(idAlterar);
119
120             if (produtoExistente != null) {
121                 String novoNome = request.getParameter("nome");
122                 String novaQuantidadeStr = request.getParameter("quantidade");
123                 String novoPrecoStr = request.getParameter("preco");
124
125                 int novaQuantidade =
126                     (novaQuantidadeStr != null && !novaQuantidadeStr.isEmpty())
127                     ? Integer.parseInt(novaQuantidadeStr) : produtoExistente.getQuantidade();
128                 float novoPreco =
129                     (novoPrecoStr != null && !novoPrecoStr.isEmpty())
130                     ? Float.parseFloat(novoPrecoStr) : produtoExistente.getPrecoVenda();
131
132                 produtoExistente.setNome(novoNome);
133                 produtoExistente.setQuantidade(novaQuantidade);
134                 produtoExistente.setPrecoVenda(novoPreco);
135
136                 facade.edit(produtoExistente);
137                 produtos = facade.findAll();
138                 request.setAttribute("produtos", produtos);
139             } else {
140                 produtos = facade.findAll();
141                 request.setAttribute("produtos", produtos);
142             }
143             break;
144     }
145     RequestDispatcher rd = request.getRequestDispatcher("ProdutoLista.jsp");
146     rd.forward(request, response);
147
148     /**
149     * Returns a short description of the servlet.
150     *

```

```

151      * @return a String containing servlet description
152      */
      no usages
153      @Override
154      public String getServletInfo() {
155          return "Short description";
156      }// </editor-fold>
157  }

```

ProdutoLista.jsp:

```

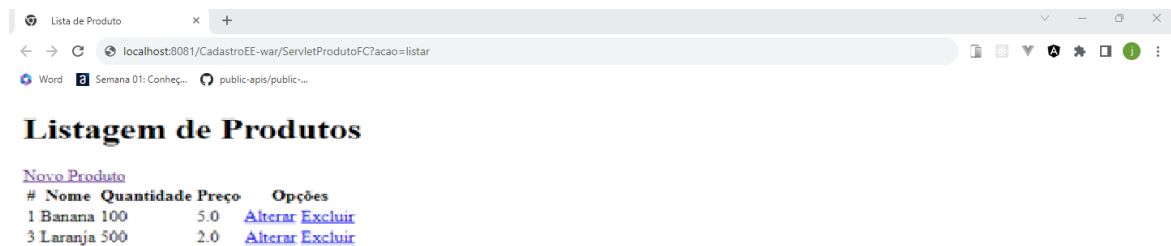
1  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
3  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4      "http://www.w3.org/TR/html4/loose.dtd">
5  <html>
6      <head>
7          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
8          <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet"
9              integrity="sha384-4bw+aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">
10         <title>Lista de Produtos</title>
11     </head>
12     <body class="container">
13         <h1>Listagem de Produtos</h1>
14         <a class="btn btn-primary m-2" href="ServletProdutoFC?acao=formIncluir">Novo Produto</a>
15         <table class="table table-striped" border="1">
16             <tr class="table-dark">
17                 <th>#</th>
18                 <th>Nome</th>
19                 <th>Quantidade</th>
20                 <th>Preço</th>
21                 <th>Opções</th>
22             </tr>
23             <!-- Aqui virá a lista de produtos -->
24             <!-- Loop para exibir a lista de produtos -->
25             <c:forEach items="${produtos}" var="produto">
26                 <tr>
27                     <td>${produto.idProduto}</td>
28                     <td>${produto.nome}</td>
29                     <td>${produto.quantidade}</td>
30                     <td>${produto.precoVenda}</td>
31                     <td>
32                         <a class="btn btn-primary btn-sm" href="ServletProdutoFC?acao=formAlterar&id=${produto.idProduto}">Alterar</a>
33                         <a class="btn btn-danger btn-sm" href="ServletProdutoFC?acao=excluir&id=${produto.idProduto}">Excluir</a>
34                     </td>
35                 </tr>
36             </c:forEach>
37         </table>
38     </body>
39 </html>

```

ProdutoDados.jsp:

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <meta charset="UTF-8">
7     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet"
8     integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRN+PtmoHDEsuppxvNDJzQIu9" crossorigin="anonymous">
9     <title>Cadastro de Produto</title>
10 </head>
11 <body class="container">
12     <h1>${empty produto ? 'Cadastro de Produto' : 'Dados do produto'}</h1>
13     <form class="form" action="ServletProdutoFC" method="post">
14         <input type="hidden" name="acao" value="${not empty produto ? 'alterar' : 'incluir'}">
15
16         <c:if test="${not empty produto}">
17             <input type="hidden" name="id" value="{produto.idProduto}">
18         </c:if>
19         <div class="mb-3">
20             <label class="form-label" for="nome">Nome:</label>
21             <input class="form-control" type="text" id="nome" name="nome" value="{not empty produto ? produto.nome : ''}">
22         </div>
23         <div class="mb-3">
24             <label class="form-label" for="quantidade">Quantidade:</label>
25             <input class="form-control" type="text" id="quantidade" name="quantidade" value="{not empty produto ? produto.quantidade : ''}">
26         </div>
27         <div class="mb-3">
28             <label class="form-label" for="preco">Preço:</label>
29             <input class="form-control" type="text" id="preco" name="preco" value="{not empty produto ? produto.precoVenda : ''}">
30         </div>
31         <br>
32     </form>
33 </body>
34 </html>
```

Resultados da execução:



#	Nome	Quantidade	Preço	Opções
1	Banana	100	5.0	Alterar Excluir
3	Laranja	500	2.0	Alterar Excluir

Novo produto:



Nome:

Quantidade:

Preço:

Alterar:



Nome:

Quantidade:

Preço:

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

Em uma aplicação web Java, o padrão Front Controller é adotado por meio de um servlet central que direciona as solicitações para controladores específicos, considerando mapeamentos de URLs. Isso contribui para uma organização mais eficiente das responsabilidades dentro de uma arquitetura MVC (Model-View-Controller).

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

Servlets e JSPs desempenham papéis cruciais no desenvolvimento de aplicações web em Java. Enquanto os Servlets são mais orientados à programação e ideais para a implementação da lógica de negócios, os JSPs simplificam a criação de interfaces de usuário dinâmicas, mantendo uma separação clara entre a lógica e o design. Ambos trabalham de maneira complementar para construir aplicativos web robustos.

- c) Qual a diferença entre um redirecionamento simples e o uso do método forward, a partir do RequestDispatcher? Para que servem parâmetros e atributos nos objetos HttpRequest?

O redirecionamento cria uma nova solicitação, o que pode resultar em um desempenho mais lento, enquanto o encaminhamento (forward) mantém a solicitação original, proporcionando uma execução mais rápida. Parâmetros são utilizados para transferir informações entre o cliente e o servidor, enquanto atributos são empregados para compartilhar dados entre componentes do servidor durante o processamento de uma única solicitação.

3º Procedimento – Melhorando o Desing da Interface

ProdutoLista.jsp:

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
3 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4 "http://www.w3.org/TR/html4/loose.dtd">
5 <html>
6 <head>
7 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
8 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet"
9 integrity="sha384-4bw+aepP/YC94hEpVNVgiZdgiC5+VKNBQNG6HeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">
10 <title>Lista de Produtos</title>
11 </head>
12 <body class="container">
13 <h1>Listagem de Produtos</h1>
14 <a class="btn btn-primary m-2" href="ServletProdutoFC?acao=formIncluir">Novo Produto</a>
15 <table class="table table-striped" border="1">
16 <tr class="table-dark">
17 <th>#</th>
18 <th>Nome</th>
19 <th>Quantidade</th>
20 <th>Preço</th>
21 <th>Opções</th>
22 </tr>
23 <!-- Aqui virá a lista de produtos -->
24 <!-- Loop para exibir a lista de produtos -->
25 <c:forEach items="{produtos}" var="produto">
26 <tr>
27 <td>${produto.idProduto}</td>
28 <td>${produto.nome}</td>
29 <td>${produto.quantidade}</td>
30 <td>${produto.precoVenda}</td>
31 <td>
32 <a class="btn btn-primary btn-sm" href="ServletProdutoFC?acao=formAlterar&id=${produto.idProduto}">Alterar</a>
33 <a class="btn btn-danger btn-sm" href="ServletProdutoFC?acao=excluir&id=${produto.idProduto}">Excluir</a>
34 </td>
35 </tr>
36 </c:forEach>
37 </table>
38 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"
39 integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>
40 </body>
41 </html>
42
```

ProdutoDados.jsp:

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet"
8 integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDzJQIu9" crossorigin="anonymous">
9 <title>Cadastro de Produto</title>
10 </head>
11 <body class="container">
12 <h1>${empty produto ? 'Cadastro de Produto' : 'Dados do produto'}</h1>
13 <form class="form" action="ServletProdutoFC" method="post">
14 <input type="hidden" name="acao" value="${not empty produto ? 'alterar' : 'incluir'}">
15
16 <c:if test="${not empty produto}">
17 <input type="hidden" name="id" value="${produto.idProduto}">
18 </c:if>
19 <div class="mb-3">
20 <label class="form-label" for="nome">Nome:</label>
21 <input class="form-control" type="text" id="nome" name="nome" value="${not empty produto ? produto.nome : ''}">
22 </div>
23 <div class="mb-3">
24 <label class="form-label" for="quantidade">Quantidade:</label>
25 <input class="form-control" type="text" id="quantidade" name="quantidade" value="${not empty produto ? produto.quantidade : ''}">
26 </div>
27 <div class="mb-3">
28 <label class="form-label" for="preco">Preço:</label>
29 <input class="form-control" type="text" id="preco" name="preco" value="${not empty produto ? produto.precoVenda : ''}">
30 </div>
31 <br>
32 <input class="btn btn-primary" type="submit" value="${not empty produto ? 'Alterar' : 'Incluir'}">
33 </form>
34 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"
35 integrity="sha384-HwwvtgBNo3bZJLlyd8oVXjrBZt8cqvSVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>
36 </body>
37 </html>
38

```

Resultados da execução:

Listagem de Produtos

[Novo Produto](#)

#	Nome	Quantidade	Preço	Opções
1	Banana	100	5.0	Alterar Excluir
3	Laranja	500	2.0	Alterar Excluir

Cadastro de Produto

Nome:

Quantidade:

Preço:

[Incluir](#)

a) Como o framework Bootstrap é utilizado?

Resumidamente, o Bootstrap é empregado em um projeto web ao integrar os seus recursos de estilo e scripts, ao atribuir classes CSS específicas aos elementos HTML e ao fazer uso dos componentes e funcionalidades predefinidos, adaptando-os às exigências de design e funcionalidade do projeto.

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

Em resumo, o Bootstrap promove a independência estrutural do HTML ao fornecer classes CSS semânticas, um sistema de grid responsivo, componentes prontos para uso e opções de personalização. Isso permite que você estilize e organize seu conteúdo de forma flexível, mantendo a estrutura do HTML intacta e preservando a semântica do documento.

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

Em resumo, o Bootstrap está intimamente relacionado à responsividade da página, uma vez que oferece ferramentas, classes e componentes projetados para criar layouts que se adaptam automaticamente a diferentes tamanhos de tela. Isso simplifica muito o desenvolvimento de páginas web responsivas e ajuda a proporcionar uma experiência consistente aos usuários em uma variedade de dispositivos.

Conclusão

Ao longo deste projeto, foi explorado a configuração e utilização das conexões com bancos de dados, a criação das camadas de persistência e controle por meio de EJBs e JPA, bem como a implementação da arquitetura MVC. Ao empregar o Front Controller, foi centralizado o gerenciamento das requisições, e as páginas JSP, enriquecidas com a estilização do Bootstrap, proporcionaram uma experiência de usuário amigável.