



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

Leonardo Schaffer Mota – 2022.05.09098-1

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

Objetivo da Prática

Implementar persistência com base em JPA.

Implementar regras de negócio na plataforma JEE, através de EJBs.

Implementar sistema cadastral Web com base em Servlets e JSPs.

Utilizar a biblioteca Bootstrap para melhoria do design.

1º Procedimento – Camadas de Persistência e Controle

Classe AbstractFacade:

```
5 package cadastroee.controller;
6
7 import java.util.List;
8 import jakarta.persistence.EntityManager;
9
10 /**
11  *
12  * @author leosc
13  */
14 public abstract class AbstractFacade<T> {
15
16     private Class<T> entityClass;
17
18     public AbstractFacade(Class<T> entityClass) {
19         this.entityClass = entityClass;
20     }
21
22     protected abstract EntityManager getEntityManager();
23
24     public void create(T entity) {
25         getEntityManager().persist(o: entity);
26     }
27
28     public void edit(T entity) {
29         getEntityManager().merge(t: entity);
30     }
31
32     public void remove(T entity) {
33         getEntityManager().remove(o: getEntityManager().merge(t: entity));
34     }
35
36     public T find(Object id) {
37         return getEntityManager().find(type:entityClass, o: id);
38     }
39
40     public List<T> findAll() {
41         jakarta.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
42         cq.select(slctn: cq.from(type:entityClass));
43         return getEntityManager().createQuery(cq).getResultList();
44     }
45
46     public List<T> findRange(int[] range) {
47         jakarta.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
48         cq.select(slctn: cq.from(type:entityClass));
49         jakarta.persistence.Query q = getEntityManager().createQuery(cq);
```

```

50         q.setMaxResults(range[1] - range[0] + 1);
51         q.setFirstResult(range[0]);
52         return q.getResultList();
53     }
54
55     public int count() {
56         jakarta.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
57         jakarta.persistence.criteria.Root<T> rt = cq.from<T>(type: entityClass);
58         cq.select(select: getEntityManager().getCriteriaBuilder().count(expression: rt));
59         jakarta.persistence.Query q = getEntityManager().createQuery(cq);
60         return ((Long) q.getSingleResult()).intValue();
61     }
62
63 }

```

Classe MovimentosFacade:

```

5 package cadastroee.controller;
6
7 import cadastroee.model.Movimentos;
8 import jakarta.ejb.Stateless;
9 import jakarta.persistence.EntityManager;
10 import jakarta.persistence.PersistenceContext;
11
12 /**
13  *
14  * @author leosc
15  */
16 @Stateless
17 public class MovimentosFacade extends AbstractFacade<Movimentos> implements MovimentosFacadeLocal {
18
19     @PersistenceContext(unitName = "CadastroEE-ejbPU")
20     private EntityManager em;
21
22     @Override
23     protected EntityManager getEntityManager() {
24         return em;
25     }
26
27     public MovimentosFacade() {
28         super(entityClass: Movimentos.class);
29     }
30
31 }

```

Classe MovimentosFacadeLocal:

```

5 package cadastroee.controller;
6
7 import cadastroee.model.Movimentos;
8 import java.util.List;
9 import jakarta.ejb.Local;
10
11 /**
12  *
13  * @author leosc
14  */
15 @Local
16 public interface MovimentosFacadeLocal {
17
18     void create(Movimentos movimentos);
19
20     void edit(Movimentos movimentos);
21
22     void remove(Movimentos movimentos);
23
24     Movimentos find(Object id);
25
26     List<Movimentos> findAll();
27
28     List<Movimentos> findRange(int[] range);
29
30     int count();
31
32 }

```

Classe PessoasFacade:

```
5 package cadastroee.controller;
6
7 import cadastroee.model.Pessoas;
8 import jakarta.ejb.Stateless;
9 import jakarta.persistence.EntityManager;
10 import jakarta.persistence.PersistenceContext;
11
12 /**
13  *
14  * @author leosc
15  */
16 @Stateless
17 public class PessoasFacade extends AbstractFacade<Pessoas> implements PessoasFacadeLocal {
18
19     @PersistenceContext(unitName = "CadastroEE-ejbPU")
20     private EntityManager em;
21
22     @Override
23     protected EntityManager getEntityManager() {
24         return em;
25     }
26
27     public PessoasFacade() {
28         super(entityClass: Pessoas.class);
29     }
30
31 }
32
```

Classe PessoasFacadeLocal:

```
5 package cadastroee.controller;
6
7 import cadastroee.model.Pessoas;
8 import java.util.List;
9 import jakarta.ejb.Local;
10
11 /**
12  *
13  * @author leosc
14  */
15 @Local
16 public interface PessoasFacadeLocal {
17
18     void create(Pessoas pessoas);
19
20     void edit(Pessoas pessoas);
21
22     void remove(Pessoas pessoas);
23
24     Pessoas find(Object id);
25
26     List<Pessoas> findAll();
27
28     List<Pessoas> findRange(int[] range);
29
30     int count();
31
32 }
```

Classe PessoasFisicasFacade:

```
5 package cadastroee.controller;
6
7 import cadastroee.model.PessoasFisicas;
8 import jakarta.ejb.Stateless;
9 import jakarta.persistence.EntityManager;
10 import jakarta.persistence.PersistenceContext;
11
12 /**
13  *
14  * @author leosc
15  */
16 @Stateless
17 public class PessoasFisicasFacade extends AbstractFacade<PessoasFisicas> implements PessoasFisicasFacadeLocal {
18
19     @PersistenceContext(unitName = "CadastroEE-ejbPU")
20     private EntityManager em;
21
22 }
```

```

22     @Override
23     protected EntityManager getEntityManager() {
24         return em;
25     }
26
27     public PessoasFisicasFacade() {
28         super(entityClass: PessoasFisicas.class);
29     }
30
31 }

```

Classe PessoasFisicasFacadeLocal:

```

5 package cadastroee.controller;
6
7 import cadastroee.model.PessoasFisicas;
8 import java.util.List;
9 import jakarta.ejb.Local;
10
11 /**
12  *
13  * @author leosc
14  */
15 @Local
16 public interface PessoasFisicasFacadeLocal {
17
18     void create(PessoasFisicas pessoasFisicas);
19
20     void edit(PessoasFisicas pessoasFisicas);
21
22     void remove(PessoasFisicas pessoasFisicas);
23
24     PessoasFisicas find(Object id);
25
26     List<PessoasFisicas> findAll();
27
28     List<PessoasFisicas> findRange(int[] range);
29
30     int count();
31
32 }

```

Classe PessoasJuridicasFacade:

```

5 package cadastroee.controller;
6
7 import cadastroee.model.PessoasJuridicas;
8 import jakarta.ejb.Stateless;
9 import jakarta.persistence.EntityManager;
10 import jakarta.persistence.PersistenceContext;
11
12 /**
13  *
14  * @author leosc
15  */
16 @Stateless
17 public class PessoasJuridicasFacade extends AbstractFacade<PessoasJuridicas> implements PessoasJuridicasFacadeLocal {
18
19     @PersistenceContext(unitName = "CadastroEE-ejbPU")
20     private EntityManager em;
21
22     @Override
23     protected EntityManager getEntityManager() {
24         return em;
25     }
26
27     public PessoasJuridicasFacade() {
28         super(entityClass: PessoasJuridicas.class);
29     }
30
31 }

```

Classe PessoasJuridicasFacadeLocal:

```
5 package cadastroee.controller;
6
7 import cadastroee.model.PessoasJuridicas;
8 import java.util.List;
9 import jakarta.ejb.Local;
10
11 /**
12  *
13  * @author leosc
14  */
15 @Local
16 public interface PessoasJuridicasFacadeLocal {
17
18     void create(PessoasJuridicas pessoasJuridicas);
19
20     void edit(PessoasJuridicas pessoasJuridicas);
21
22     void remove(PessoasJuridicas pessoasJuridicas);
23
24     PessoasJuridicas find(Object id);
25
26     List<PessoasJuridicas> findAll();
27
28     List<PessoasJuridicas> findRange(int[] range);
29
30     int count();
31
32 }
```

Classe ProdutosFacade:

```
5 package cadastroee.controller;
6
7 import cadastroee.model.Produtos;
8 import jakarta.ejb.Stateless;
9 import jakarta.persistence.EntityManager;
10 import jakarta.persistence.PersistenceContext;
11
12 /**
13  *
14  * @author leosc
15  */
16 @Stateless
17 public class ProdutosFacade extends AbstractFacade<Produtos> implements ProdutosFacadeLocal {
18
19     @PersistenceContext(unitName = "CadastroEE-ejbPU")
20     private EntityManager em;
21
22     @Override
23     protected EntityManager getEntityManager() {
24         return em;
25     }
26
27     public ProdutosFacade() {
28         super(entityClass: Produtos.class);
29     }
30
31 }
```

Classe ProdutosFacadeLocal:

```
5 package cadastroee.controller;
6
7 import cadastroee.model.Produtos;
8 import java.util.List;
9 import jakarta.ejb.Local;
10
11 /**
12  *
13  * @author leosc
14  */
15 @Local
16 public interface ProdutosFacadeLocal {
17
18     void create(Produtos produtos);
19
20     void edit(Produtos produtos);
21
22     void remove(Produtos produtos);
23
24     Produtos find(Object id);
25
26     List<Produtos> findAll();
27
28     List<Produtos> findRange(int[] range);
29
30     int count();
31
32 }
```

Classe UsuariosFacade:

```
5 package cadastroee.controller;
6
7 import cadastroee.model.Usuarios;
8 import jakarta.ejb.Stateless;
9 import jakarta.persistence.EntityManager;
10 import jakarta.persistence.PersistenceContext;
11
12 /**
13  *
14  * @author leosc
15  */
16 @Stateless
17 public class UsuariosFacade extends AbstractFacade<Usuarios> implements UsuariosFacadeLocal {
18
19     @PersistenceContext(unitName = "CadastroEE-ejbPU")
20     private EntityManager em;
21
22     @Override
23     protected EntityManager getEntityManager() {
24         return em;
25     }
26
27     public UsuariosFacade() {
28         super(entityClass: Usuarios.class);
29     }
30
31 }
```

Classe UsuariosFacadeLocal:

```
5 package cadastroee.controller;
6
7 import cadastroee.model.Usuarios;
8 import java.util.List;
9 import jakarta.ejb.Local;
10
11 /**
12  *
13  * @author leosc
14  */
15 @Local
16 public interface UsuariosFacadeLocal {
17
18     void create(Usuarios usuarios);
19
20     void edit(Usuarios usuarios);
21
22     void remove(Usuarios usuarios);
23
24     Usuarios find(Object id);
25
26     List<Usuarios> findAll();
27
28     List<Usuarios> findRange(int[] range);
29
30     int count();
31
32 }
```

Classe Movimentos:

```
5 package cadastroee.model;
6
7 import java.io.Serializable;
8 import java.math.BigDecimal;
9 import jakarta.persistence.Basic;
10 import jakarta.persistence.Column;
11 import jakarta.persistence.Entity;
12 import jakarta.persistence.Id;
13 import jakarta.persistence.JoinColumn;
14 import jakarta.persistence.ManyToOne;
15 import jakarta.persistence.NamedQueries;
16 import jakarta.persistence.NamedQuery;
17 import jakarta.persistence.Table;
18
19 /**
20  *
21  * @author leosc
22  */
23 @Entity
24 @Table(name = "Movimentos")
25 @NamedQueries({
26     @NamedQuery(name = "Movimentos.findAll", query = "SELECT m FROM Movimentos m"),
27     @NamedQuery(name = "Movimentos.findByIdMovimentos", query = "SELECT m FROM Movimentos m WHERE m.idMovimentos = :idMovimentos"),
28 })
```

```

27 @NamedQuery(name = "Movimentos.findByMovimentos", query = "SELECT m FROM Movimentos m WHERE m.idMovimentos = :idMovimentos"),
28 @NamedQuery(name = "Movimentos.findByQuantidade", query = "SELECT m FROM Movimentos m WHERE m.quantidade = :quantidade"),
29 @NamedQuery(name = "Movimentos.findByTipo", query = "SELECT m FROM Movimentos m WHERE m.tipo = :tipo"),
30 @NamedQuery(name = "Movimentos.findByPrecoUnitario", query = "SELECT m FROM Movimentos m WHERE m.precoUnitario = :precoUnitario"))
31 public class Movimentos implements Serializable {
32
33     private static final long serialVersionUID = 1L;
34     @Id
35     @Basic(optional = false)
36     @Column(name = "idMovimentos")
37     private Integer idMovimentos;
38     @Column(name = "Quantidade")
39     private Integer quantidade;
40     @Column(name = "Tipo")
41     private Character tipo;
42     // @Max(value=?) @Min(value=?)//if you know range of your decimal fields consider using these annotations to enforce field validation
43     @Column(name = "PrecoUnitario")
44     private BigDecimal precoUnitario;
45     @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa")
46     @ManyToOne
47     private Pessoas idPessoa;
48     @JoinColumn(name = "idProduto", referencedColumnName = "idProduto")
49     @ManyToOne
50     private Produtos idProduto;
51     @JoinColumn(name = "idUserario", referencedColumnName = "idUserario")
52     @ManyToOne
53     private Usuarios idUsuario;
54
55     public Movimentos() {
56     }
57
58     public Movimentos(Integer idMovimentos) {
59         this.idMovimentos = idMovimentos;
60     }
61
62     public Integer getIdMovimentos() {
63         return idMovimentos;
64     }
65
66     public void setIdMovimentos(Integer idMovimentos) {
67         this.idMovimentos = idMovimentos;
68     }
69
70     public Integer getQuantidade() {
71         return quantidade;
72     }
73
74     public void setQuantidade(Integer quantidade) {
75         this.quantidade = quantidade;
76     }
77
78     public Character getTipo() {
79         return tipo;
80     }
81
82     public void setTipo(Character tipo) {
83         this.tipo = tipo;
84     }
85
86     public BigDecimal getPrecoUnitario() {
87         return precoUnitario;
88     }
89
90     public void setPrecoUnitario(BigDecimal precoUnitario) {
91         this.precoUnitario = precoUnitario;
92     }
93
94     public Pessoas getIdPessoa() {
95         return idPessoa;
96     }
97
98     public void setIdPessoa(Pessoas idPessoa) {
99         this.idPessoa = idPessoa;
100     }
101
102     public Produtos getIdProduto() {
103         return idProduto;
104     }
105
106     public void setIdProduto(Produtos idProduto) {
107         this.idProduto = idProduto;
108     }
109
110     public Usuarios getIdUsuario() {
111         return idUsuario;
112     }
113
114     public void setIdUsuario(Usuarios idUsuario) {
115         this.idUsuario = idUsuario;
116     }
117
118     @Override
119     public int hashCode() {
120         int hash = 0;
121         hash += (idMovimentos != null ? idMovimentos.hashCode() : 0);
122         return hash;
123     }
124
125     @Override
126     public boolean equals(Object object) {
127         // TODO: Warning - this method won't work in the case the id fields are not set
128         if (!(object instanceof Movimentos)) {
129             return false;
130         }

```

```

131 Movimentos other = (Movimentos) object;
132 if ((this.idMovimentos == null && other.idMovimentos != null) || (this.idMovimentos != null && !this.idMovimentos.equals(other.idMovimentos)))
133     return false;
134 }
135 return true;
136 }
137
138 @Override
139 public String toString() {
140     return "cadastroee.model.Movimentos[ idMovimentos=" + idMovimentos + " ]";
141 }
142 }
143 }

```

Classe Pessoas:

```

5 package cadastroee.model;
6
7 import java.io.Serializable;
8 import java.util.Collection;
9 import jakarta.persistence.Basic;
10 import jakarta.persistence.CascadeType;
11 import jakarta.persistence.Column;
12 import jakarta.persistence.Entity;
13 import jakarta.persistence.Id;
14 import jakarta.persistence.NamedQueries;
15 import jakarta.persistence.NamedQuery;
16 import jakarta.persistence.OneToMany;
17 import jakarta.persistence.OneToOne;
18 import jakarta.persistence.Table;
19
20 /**
21  *
22  * @author leosc
23  */
24 @Entity
25 @Table(name = "Pessoas")
26 @NamedQueries({
27     @NamedQuery(name = "Pessoas.findAll", query = "SELECT p FROM Pessoas p"),
28     @NamedQuery(name = "Pessoas.findByIdPessoa", query = "SELECT p FROM Pessoas p WHERE p.idPessoa = :idPessoa"),
29     @NamedQuery(name = "Pessoas.findByName", query = "SELECT p FROM Pessoas p WHERE p.nome = :nome"),
30     @NamedQuery(name = "Pessoas.findByCidade", query = "SELECT p FROM Pessoas p WHERE p.cidade = :cidade"),
31     @NamedQuery(name = "Pessoas.findByEstado", query = "SELECT p FROM Pessoas p WHERE p.estado = :estado"),
32     @NamedQuery(name = "Pessoas.findByTelefone", query = "SELECT p FROM Pessoas p WHERE p.telefone = :telefone"),
33     @NamedQuery(name = "Pessoas.findByEmail", query = "SELECT p FROM Pessoas p WHERE p.email = :email"),
34     @NamedQuery(name = "Pessoas.findByLogradouro", query = "SELECT p FROM Pessoas p WHERE p.logradouro = :logradouro")
35 })
36 public class Pessoas implements Serializable {
37     private static final long serialVersionUID = 1L;
38     @Id
39     @Basic(optional = false)
40     @Column(name = "idPessoa")
41     private Integer idPessoa;
42     @Column(name = "nome")
43     private String nome;
44     @Column(name = "cidade")
45     private String cidade;
46     @Column(name = "estado")
47     private String estado;
48     @Column(name = "telefone")
49     private String telefone;
50     @Column(name = "email")
51     private String email;
52     @Column(name = "logradouro")
53     private String logradouro;
54     @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoas")
55     private PessoasJuridicas pessoasJuridicas;
56     @OneToMany(mappedBy = "idPessoa")
57     private Collection<Movimentos> movimentosCollection;
58     @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoas")
59     private PessoasFisicas pessoasFisicas;
60
61     public Pessoas() {
62     }
63
64     public Pessoas(Integer idPessoa) {
65         this.idPessoa = idPessoa;
66     }
67
68     public Integer getIdPessoa() {
69         return idPessoa;
70     }
71
72     public void setIdPessoa(Integer idPessoa) {
73         this.idPessoa = idPessoa;
74     }
75
76     public String getNome() {
77         return nome;
78     }
79
80     public void setNome(String nome) {
81         this.nome = nome;
82     }
83
84     public String getCidade() {
85         return cidade;
86     }
87
88     public void setCidade(String cidade) {
89         this.cidade = cidade;
90     }

```



```

91
92 public String getEstado() {
93     return estado;
94 }
95
96 public void setEstado(String estado) {
97     this.estado = estado;
98 }
99
100 public String getTelefone() {
101     return telefone;
102 }
103
104 public void setTelefone(String telefone) {
105     this.telefone = telefone;
106 }
107
108 public String getEmail() {
109     return email;
110 }
111
112 public void setEmail(String email) {
113     this.email = email;
114 }
115
116 public String getLogradouro() {
117     return logradouro;
118 }
119
120 public void setLogradouro(String logradouro) {
121     this.logradouro = logradouro;
122 }
123
124 public PessoasJuridicas getPessoasJuridicas() {
125     return pessoasJuridicas;
126 }
127
128 public void setPessoasJuridicas(PessoasJuridicas pessoasJuridicas) {
129     this.pessoasJuridicas = pessoasJuridicas;
130 }
131
132 public Collection<Movimentos> getMovimentosCollection() {
133     return movimentosCollection;
134 }
135
136 public void setMovimentosCollection(Collection<Movimentos> movimentosCollection) {
137     this.movimentosCollection = movimentosCollection;
138 }
139
140 public PessoasFisicas getPessoasFisicas() {
141     return pessoasFisicas;
142 }
143
144 public void setPessoasFisicas(PessoasFisicas pessoasFisicas) {
145     this.pessoasFisicas = pessoasFisicas;
146 }
147
148 @Override
149 public int hashCode() {
150     int hash = 0;
151     hash += (idPessoa != null ? idPessoa.hashCode() : 0);
152     return hash;
153 }
154
155 @Override
156 public boolean equals(Object object) {
157     // TODO: Warning - this method won't work in the case the id fields are not set
158     if (!(object instanceof Pessoas)) {
159         return false;
160     }
161     Pessoas other = (Pessoas) object;
162     if ((this.idPessoa == null && other.idPessoa != null) || (this.idPessoa != null && !this.idPessoa.equals(obj: other.idPessoa))) {
163         return false;
164     }
165     return true;
166 }
167
168 @Override
169 public String toString() {
170     return "cadastroee.model.Pessoas[ idPessoa=" + idPessoa + " ]";
171 }
172
173 }

```

Classe PessoasFisicas:

```

5 package cadastroee.model;
6
7 import java.io.Serializable;
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

```

```

18  /**
19   *
20   * @author leosc
21   */
22  @Entity
23  @Table(name = "PessoasFisicas")
24  @NamedQueries({
25      @NamedQuery(name = "PessoasFisicas.findAll", query = "SELECT p FROM PessoasFisicas p"),
26      @NamedQuery(name = "PessoasFisicas.findByIdPFisica", query = "SELECT p FROM PessoasFisicas p WHERE p.idPFisica = :idPFisica"),
27      @NamedQuery(name = "PessoasFisicas.findByCpf", query = "SELECT p FROM PessoasFisicas p WHERE p.cpf = :cpf"))
28  }
29  public class PessoasFisicas implements Serializable {
30
31      private static final long serialVersionUID = 1L;
32      @Id
33      @Basic(optional = false)
34      @Column(name = "idPFisica")
35      private Integer idPFisica;
36      @Column(name = "cpf")
37      private String cpf;
38      @JoinColumn(name = "idPFisica", referencedColumnName = "idPessoa", insertable = false, updatable = false)
39      @OneToOne(optional = false)
40      private Pessoas pessoas;
41
42      public PessoasFisicas() {
43      }
44
45      public PessoasFisicas(Integer idPFisica) {
46          this.idPFisica = idPFisica;
47      }
48
49      public Integer getIdPFisica() {
50          return idPFisica;
51      }
52
53      public void setIdPFisica(Integer idPFisica) {
54          this.idPFisica = idPFisica;
55      }
56
57      public String getCpf() {
58          return cpf;
59      }
60
61      public void setCpf(String cpf) {
62          this.cpf = cpf;
63      }
64
65      public Pessoas getPessoas() {
66          return pessoas;
67      }
68
69      public void setPessoas(Pessoas pessoas) {
70          this.pessoas = pessoas;
71      }
72
73      @Override
74      public int hashCode() {
75          int hash = 0;
76          hash += (idPFisica != null ? idPFisica.hashCode() : 0);
77          return hash;
78      }
79
80      @Override
81      public boolean equals(Object object) {
82          // TODO: Warning - this method won't work in the case the id fields are not set
83          if (!(object instanceof PessoasFisicas)) {
84              return false;
85          }
86          PessoasFisicas other = (PessoasFisicas) object;
87          if (((this.idPFisica == null && other.idPFisica != null) || (this.idPFisica != null && !this.idPFisica.equals(obj) : other.idPFisica
88              return false;
89          }
90          return true;
91      }
92
93      @Override
94      public String toString() {
95          return "cadastroee.model.PessoasFisicas[ idPFisica=" + idPFisica + " ]";
96      }
97  }

```

Classe PessoasJuridicas:

```

5  package cadastroee.model;
6
7  import java.io.Serializable;
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
18 /**
19  *
20  * @author leosc
21  */

```

```

22 @Entity
23 @Table(name = "PessoasJuridicas")
24 @NamedQueries({
25     @NamedQuery(name = "PessoasJuridicas.findAll", query = "SELECT p FROM PessoasJuridicas p"),
26     @NamedQuery(name = "PessoasJuridicas.findByIdPJuridica", query = "SELECT p FROM PessoasJuridicas p WHERE p.idPJuridica = :idPJuridica"),
27     @NamedQuery(name = "PessoasJuridicas.findByCnpj", query = "SELECT p FROM PessoasJuridicas p WHERE p.cnpj = :cnpj"))
28 public class PessoasJuridicas implements Serializable {
29
30     private static final long serialVersionUID = 1L;
31     @Id
32     @Basic(optional = false)
33     @Column(name = "idPJuridica")
34     private Integer idPJuridica;
35     @Column(name = "cnpj")
36     private String cnpj;
37     @JoinColumn(name = "idPJuridica", referencedColumnName = "idPessoa", insertable = false, updatable = false)
38     @OneToOne(optional = false)
39     private Pessoas pessoas;
40
41     public PessoasJuridicas() {
42     }
43
44     public PessoasJuridicas(Integer idPJuridica) {
45         this.idPJuridica = idPJuridica;
46     }
47
48     public Integer getIdPJuridica() {
49         return idPJuridica;
50     }
51
52     public void setIdPJuridica(Integer idPJuridica) {
53         this.idPJuridica = idPJuridica;
54     }
55
56     public String getCnpj() {
57         return cnpj;
58     }
59
60     public void setCnpj(String cnpj) {
61         this.cnpj = cnpj;
62     }
63
64     public Pessoas getPessoas() {
65         return pessoas;
66     }
67
68     public void setPessoas(Pessoas pessoas) {
69         this.pessoas = pessoas;
70     }
71
72     @Override
73     public int hashCode() {
74         int hash = 0;
75         hash += (idPJuridica != null ? idPJuridica.hashCode() : 0);
76         return hash;
77     }
78
79     @Override
80     public boolean equals(Object object) {
81         // TODO: Warning - this method won't work in the case the id fields are not set
82         if (!(object instanceof PessoasJuridicas)) {
83             return false;
84         }
85         PessoasJuridicas other = (PessoasJuridicas) object;
86         return ((this.idPJuridica == null && other.idPJuridica != null) || (this.idPJuridica != null && !this.idPJuridica.equals(obj : other.idPJuridica)))
87             || (this.cnpj == null && other.cnpj != null) || (this.cnpj != null && !this.cnpj.equals(obj : other.cnpj));
88     }
89
90     @Override
91     public String toString() {
92         return "cadastroee.model.PessoasJuridicas[ idPJuridica=" + idPJuridica + " ]";
93     }
94 }

```

Classe Produtos:

```

5 package cadastroee.model;
6
7 import java.io.Serializable;
8 import java.util.Collection;
9 import jakarta.persistence.Basic;
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.Table;
17
18 /**
19  *
20  * @author leosc
21  */
22 @Entity
23 @Table(name = "Produtos")
24 @NamedQueries({

```

```

25 @NamedQuery(name = "Produtos.findAll", query = "SELECT p FROM Produtos p"),
26 @NamedQuery(name = "Produtos.findByIdProduto", query = "SELECT p FROM Produtos p WHERE p.idProduto = :idProduto"),
27 @NamedQuery(name = "Produtos.findByName", query = "SELECT p FROM Produtos p WHERE p.nome = :nome"),
28 @NamedQuery(name = "Produtos.findByQuantidade", query = "SELECT p FROM Produtos p WHERE p.quantidade = :quantidade"),
29 @NamedQuery(name = "Produtos.findByPrecoVenda", query = "SELECT p FROM Produtos p WHERE p.precoVenda = :precoVenda"))}
30 public class Produtos implements Serializable {
31
32     private static final long serialVersionUID = 1L;
33     @Id
34     @Basic(optional = false)
35     @Column(name = "idProduto")
36     private Integer idProduto;
37     @Column(name = "nome")
38     private String nome;
39     @Column(name = "quantidade")
40     private Integer quantidade;
41     // @Max(value=?) @Min(value=?)//if you know range of your decimal fields consider using these annotations to enforce field validation
42     @Column(name = "precoVenda")
43     private float precoVenda;
44     @OneToMany(mappedBy = "idProduto")
45     private Collection<Movimentos> movimentosCollection;
46
47     public Produtos() {
48     }
49
50     public Produtos(Integer idProduto) {
51         this.idProduto = idProduto;
52     }
53
54     public Integer getIdProduto() {
55         return idProduto;
56     }
57
58     public void setIdProduto(Integer idProduto) {
59         this.idProduto = idProduto;
60     }
61
62     public String getNome() {
63         return nome;
64     }
65
66     public void setNome(String nome) {
67         this.nome = nome;
68     }
69
70     public Integer getQuantidade() {
71         return quantidade;
72     }
73
74     public void setQuantidade(Integer quantidade) {
75         this.quantidade = quantidade;
76     }
77
78     public float getPrecoVenda() {
79         return precoVenda;
80     }
81
82     public void setPrecoVenda(float precoVenda) {
83         this.precoVenda = precoVenda;
84     }
85
86     public Collection<Movimentos> getMovimentosCollection() {
87         return movimentosCollection;
88     }
89
90     public void setMovimentosCollection(Collection<Movimentos> movimentosCollection) {
91         this.movimentosCollection = movimentosCollection;
92     }
93
94     @Override
95     public int hashCode() {
96         int hash = 0;
97         hash += (idProduto != null ? idProduto.hashCode() : 0);
98         return hash;
99     }
100
101     @Override
102     public boolean equals(Object object) {
103         // TODO: Warning - this method won't work in the case the id fields are not set
104         if (!(object instanceof Produtos)) {
105             return false;
106         }
107         Produtos other = (Produtos) object;
108         if ((this.idProduto == null && other.idProduto != null) || (this.idProduto != null && !this.idProduto.equals(obj: other.idProduto))) {
109             return false;
110         }
111         return true;
112     }
113
114     @Override
115     public String toString() {
116         return "cadastroee.model.Produtos[ idProduto=" + idProduto + " ]";
117     }
118
119 }

```

Classe Usuarios:

```
5 package cadastroee.model;
6
7 import java.io.Serializable;
8 import java.util.Collection;
9 import jakarta.persistence.Basic;
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.Table;
17
18 /**
19  *
20  * @author leosc
21  */
22 @Entity
23 @Table(name = "Usuarios")
24 @NamedQueries({
25     @NamedQuery(name = "Usuarios.findAll", query = "SELECT u FROM Usuarios u"),
26     @NamedQuery(name = "Usuarios.findByIdUsuario", query = "SELECT u FROM Usuarios u WHERE u.idUsuario = :idUsuario"),
27     @NamedQuery(name = "Usuarios.findByLogin", query = "SELECT u FROM Usuarios u WHERE u.login = :login"),
28     @NamedQuery(name = "Usuarios.findBySenha", query = "SELECT u FROM Usuarios u WHERE u.senha = :senha"))
29 public class Usuarios implements Serializable {
30
31     private static final long serialVersionUID = 1L;
32     @Id
33     @Basic(optional = false)
34     @Column(name = "idUsuario")
35     private Integer idUsuario;
36     @Column(name = "login")
37     private String login;
38     @Column(name = "senha")
39     private String senha;
40     @OneToMany(mappedBy = "idUsuario")
41     private Collection<Movimentos> movimentosCollection;
42
43     public Usuarios() {
44     }
45
46     public Usuarios(Integer idUsuario) {
47         this.idUsuario = idUsuario;
48     }
49
50     public Integer getIdUsuario() {
51         return idUsuario;
52     }
53
54     public void setIdUsuario(Integer idUsuario) {
55         this.idUsuario = idUsuario;
56     }
57
58     public String getLogin() {
59         return login;
60     }
61
62     public void setLogin(String login) {
63         this.login = login;
64     }
65
66     public String getSenha() {
67         return senha;
68     }
69
70     public void setSenha(String senha) {
71         this.senha = senha;
72     }
73
74     public Collection<Movimentos> getMovimentosCollection() {
75         return movimentosCollection;
76     }
77
78     public void setMovimentosCollection(Collection<Movimentos> movimentosCollection) {
79         this.movimentosCollection = movimentosCollection;
80     }
81
82     @Override
83     public int hashCode() {
84         int hash = 0;
85         hash += (idUsuario != null ? idUsuario.hashCode() : 0);
86         return hash;
87     }
88
89     @Override
90     public boolean equals(Object object) {
91         // TODO: Warning - this method won't work in the case the id fields are not set
92         if (!(object instanceof Usuarios)) {
93             return false;
94         }
95         Usuarios other = (Usuarios) object;
96         return ((this.idUsuario == null && other.idUsuario != null) || (this.idUsuario != null && !this.idUsuario.equals(obj: other.idUsuari
97         return false;
98     }
99     return true;
100 }
101
102 @Override
103 public String toString() {
104     return "cadastroee.model.Usuarios[ idUsuario=" + idUsuario + " ]";
105 }
```

ServletProduto.java:

```
5 package cadastroee.servlets;
6
7 import java.io.IOException;
8 import java.io.PrintWriter;
9 import jakarta.servlet.ServletException;
10 import jakarta.servlet.http.HttpServlet;
11 import jakarta.servlet.http.HttpServletRequest;
12 import jakarta.servlet.http.HttpServletResponse;
13 import jakarta.ejb.EJB;
14 import cadastroee.controller.ProdutosFacadeLocal;
15 import cadastroee.model.Produtos;
16 import java.util.List;
17
18 /**
19  *
20  * @author leosc
21  */
22 public class ServletProduto extends HttpServlet {
23
24     @EJB
25     ProdutosFacadeLocal facade;
26
27     /**
28      * Processes requests for both HTTP GET and POST
29      * methods.
30      *
31      * @param request servlet request
32      * @param response servlet response
33      * @throws ServletException if a servlet-specific error occurs
34      * @throws IOException if an I/O error occurs
35      */
36     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
37         throws ServletException, IOException {
38         response.setContentType("text/html;charset=UTF-8");
39         try (PrintWriter out = response.getWriter()) {
40             /* TODO output your page here. You may use following sample code. */
41             out.println("<!DOCTYPE html>");
42             out.println("<html>");
43             out.println("<head>");
44             out.println("<title>Servlet ServletProduto</title>");
45             out.println("</head>");
46             out.println("<body>");
47             out.println("<h1>Servlet ServletProduto at " + request.getContextPath() + "</h1>");
48             out.println("</body>");
49             out.println("</html>");
50         }
51     }
52
53     /**
54      * Handles the HTTP GET method.
55      *
56      * @param request servlet request
57      * @param response servlet response
58      * @throws ServletException if a servlet-specific error occurs
59      * @throws IOException if an I/O error occurs
60      */
61     @Override
62     protected void doGet(HttpServletRequest request, HttpServletResponse response)
63         throws ServletException, IOException {
64
65         List<Produtos> produtos = facade.findAll();
66
67         response.setContentType("text/html;charset=UTF-8");
68         try (PrintWriter out = response.getWriter()) {
69             out.println("<html>");
70             out.println("<head>");
71             out.println("<title>Servlet ServletProduto</title>");
72             out.println("<style>");
73             out.println("<ul { list-style: none; padding-left: 0; }>");
74             out.println("</style>");
75             out.println("</head>");
76             out.println("<body>");
77             out.println("<h1>Servlet ServletProduto at " + request.getContextPath() + "</h1>");
78             out.println("<ul>");
79             for (Produtos produto : produtos) {
80                 out.println("<li> " + produto.getNome() + "</li>");
81             }
82             out.println("</ul>");
83             out.println("</body>");
84             out.println("</html>");
85         }
86         processRequest(request, response);
87     }
88
89     /**
90      * Handles the HTTP POST method.
91      *
92      * @param request servlet request
93      * @param response servlet response
94      * @throws ServletException if a servlet-specific error occurs
95      * @throws IOException if an I/O error occurs
96      */
97 }
```

```

96  @Override
97  protected void doPost(HttpServletRequest request, HttpServletResponse response)
98      throws ServletException, IOException {
99      processRequest(request, response);
100  }
101  /**
102   * Returns a short description of the servlet.
103   *
104   * @return a String containing servlet description
105   */
106  @Override
107  public String getServletInfo() {
108      return "Short description";
109  } // </editor-fold>
110
111  }

```

Resultado da execução:



a) Como é organizado um projeto corporativo no NetBeans?

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

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

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

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

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

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

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

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

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

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

2º Procedimento – Interface Cadastral com Servlet e JSPs

ServletProdutoFC.java:

```
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 /**
20  * @author leosc
21  */
22 @WebServlet(name = "ServletProdutoFC", urlPatterns = {"/ServletProdutoFC"})
23 public class ServletProdutoFC extends HttpServlet {
24     @EJB
25     ProdutosFacadeLocal facade;
26     /**
27      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
28      * methods.
29      *
30      * @param request servlet request
31      * @param response servlet response
32      * @throws ServletException if a servlet-specific error occurs
33      * @throws IOException if an I/O error occurs
34      */
35 }
```



```

35 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
36     throws ServletException, IOException {
37 }
38 // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit t
39 /**
40  * Handles the HTTP <code>GET</code> method.
41  *
42  * @param request servlet request
43  * @param response servlet response
44  * @throws ServletException if a servlet-specific error occurs
45  * @throws IOException if an I/O error occurs
46  */
47 @Override
48 protected void doGet(HttpServletRequest request, HttpServletResponse response)
49     throws ServletException, IOException {
50     String acao = request.getParameter(string: "acao");
51     String destino = null;
52
53     switch (acao) {
54         case "listar":
55             List<Produtos> produtos = facade.findAll();
56             request.setAttribute(string: "produtos", o: produtos);
57             destino = "ProdutoLista.jsp";
58             break;
59         case "formIncluir":
60             destino = "ProdutoDados.jsp";
61             break;
62         case "formAlterar":
63             int idAlterar = Integer.parseInt(s: request.getParameter(string: "id"));
64             Produtos produtoAlterar = facade.find(id: idAlterar);
65             request.setAttribute(string: "produto", o: produtoAlterar);
66             destino = "ProdutoDados.jsp";
67             break;
68         case "excluir":
69             int idExcluir = Integer.parseInt(s: request.getParameter(string: "id"));
70             Produtos produtoExcluir = facade.find(id: idExcluir);
71             facade.remove(produtos: produtoExcluir);
72             List<Produtos> produtosExcluidos = facade.findAll();
73             request.setAttribute(string: "produtos", o: produtosExcluidos);
74             destino = "ProdutoLista.jsp";
75             break;
76         default:
77             break;
78     }
79     RequestDispatcher rd = request.getRequestDispatcher(string: destino);
80     rd.forward(s: request, s: response);
81 }
82 /**
83  * Handles the HTTP <code>POST</code> method.
84  *
85  * @param request servlet request
86  * @param response servlet response
87  * @throws ServletException if a servlet-specific error occurs
88  * @throws IOException if an I/O error occurs
89  */
90 @Override
91 protected void doPost(HttpServletRequest request, HttpServletResponse response)
92     throws ServletException, IOException {
93     String acao = request.getParameter(string: "acao");
94     List<Produtos> produtos = null;
95
96     switch (acao) {
97         case "incluir":
98             String nome = request.getParameter(string: "nome");
99             int quantidade = Integer.parseInt(s: request.getParameter(string: "quantidade"));
100             float preco = Float.parseFloat(s: request.getParameter(string: "preco"));
101             produtos = facade.findAll();
102             int ultimoId = 0;
103             Produtos ultimoProduto = produtos.get(produtos.size() - 1);
104             ultimoId = ultimoProduto.getIdProduto();
105             int id = ultimoId + 1;
106
107             Produtos novoProduto = new Produtos();
108             novoProduto.setNome(nome);
109             novoProduto.setQuantidade(quantidade);
110             novoProduto.setPrecoVenda(precoVenda: preco);
111             novoProduto.setIdProduto(idProduto: id);
112
113             facade.create(produtos: novoProduto);
114
115             produtos = facade.findAll();
116             request.setAttribute(string: "produtos", o: produtos);
117
118             break;
119
120         case "alterar":
121             int idAlterar = Integer.parseInt(s: request.getParameter(string: "id"));
122             Produtos produtoExistente = facade.find(id: idAlterar);
123
124             if (produtoExistente != null) {
125                 String novoNome = request.getParameter(string: "nome");
126                 String novaQuantidadeStr = request.getParameter(string: "quantidade");
127                 String novoPrecoStr = request.getParameter(string: "preco");
128
129                 int novaQuantidade =
130                     (novaQuantidadeStr != null && !novaQuantidadeStr.isEmpty())
131                     ? Integer.parseInt(s: novaQuantidadeStr) : produtoExistente.getQuantidade();
132                 float novoPreco =
133                     (novoPrecoStr != null && !novoPrecoStr.isEmpty())
134                     ? Float.parseFloat(s: novoPrecoStr) : produtoExistente.getPrecoVenda();
135
136                 produtoExistente.setNome(nome: novoNome);
137                 produtoExistente.setQuantidade(quantidade: novaQuantidade);
138                 produtoExistente.setPrecoVenda(precoVenda: novoPreco);

```

```

139         facade.edit(produtos: produtoExistente);
140         produtos = facade.findAll();
141         request.setAttribute(string: "produtos", o: produtos);
142     } else {
143         produtos = facade.findAll();
144         request.setAttribute(string: "produtos", o: produtos);
145     }
146     break;
147 }
148
149 RequestDispatcher rd = request.getRequestDispatcher(string: "ProdutoLista.jsp");
150 rd.forward(sp: request, sp1: response);
151 }
152
153 /**
154  * Returns a short description of the servlet.
155  *
156  * @return a String containing servlet description
157  */
158 @Override
159 public String getServletInfo() {
160     return "Short description";
161 }

```

ProdutoList.jsp:

```

6 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
7 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
8 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9 "http://www.w3.org/TR/html4/loose.dtd">
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
13 <title>Lista de Produtos</title>
14 </head>
15 <body class="">
16 <h1>Listagem de Produtos</h1>
17 <a class="" href="ServletProdutoFC?acao=formIncluir">Novo Produto</a>
18 <table class="">
19 <tr class="">
20 <th></th>
21 <th>Nome</th>
22 <th>Quantidade</th>
23 <th>Preço</th>
24 <th>Opções</th>
25 </tr>
26 <!-- Aqui virá a lista de produtos -->
27 <!-- Loop para exibir a lista de produtos -->
28 <c:forEach items="${produtos}" var="produto">
29 <tr>
30 <td>${produto.idProduto}</td>
31 <td>${produto.nome}</td>
32 <td>${produto.quantidade}</td>
33 <td>${produto.precoVenda}</td>
34 <td>
35 <a class="" href="ServletProdutoFC?acao=formAlterar&id=${produto.idProduto}">Alterar</a>
36 <a class="" href="ServletProdutoFC?acao=excluir&id=${produto.idProduto}">Excluir</a>
37 </td>
38 </tr>
39 </c:forEach>
40 </table>
41 </body>
42 </html>

```

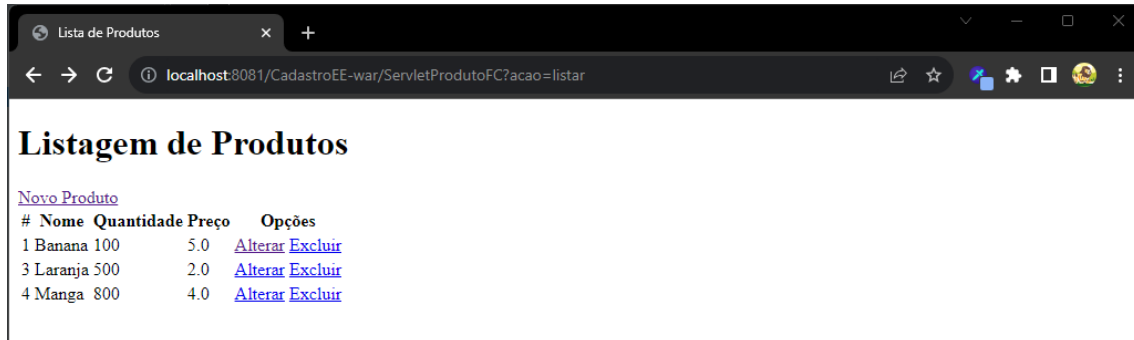
ProdutoDados.jsp:

```

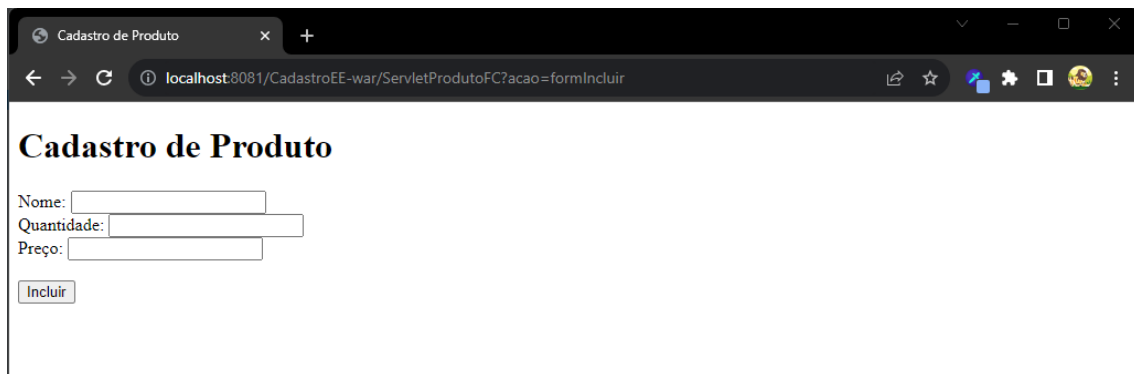
6 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
7 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta charset="UTF-8">
12 <title>Cadastro de Produto</title>
13 </head>
14 <body class="">
15 <h1>${empty produto ? 'Cadastro de Produto' : 'Dados do produto'}</h1>
16 <form class="" action="ServletProdutoFC" method="post">
17 <input type="hidden" name="acao" value="${not empty produto ? 'alterar' : 'incluir'}">
18 <c:if test="${not empty produto}">
19 <input type="hidden" name="id" value="${produto.idProduto}">
20 </c:if>
21 <div class="">
22 <label class="" for="nome">Nome:</label>
23 <input class="" type="text" id="nome" name="nome" value="${not empty produto ? produto.nome : ''}">
24 </div>
25 <div class="">
26 <label class="" for="quantidade">Quantidade:</label>
27 <input class="" type="text" id="quantidade" name="quantidade" value="${not empty produto ? produto.quantidade : ''}">
28 </div>
29 <div class="">
30 <label class="" for="preco">Preço:</label>
31 <input class="" type="text" id="preco" name="preco" value="${not empty produto ? produto.precoVenda : ''}">
32 </div>
33 <br>
34 <input class="" type="submit" value="${not empty produto ? 'Alterar' : 'Incluir'}">
35 </form>
36 </body>
37 </html>

```

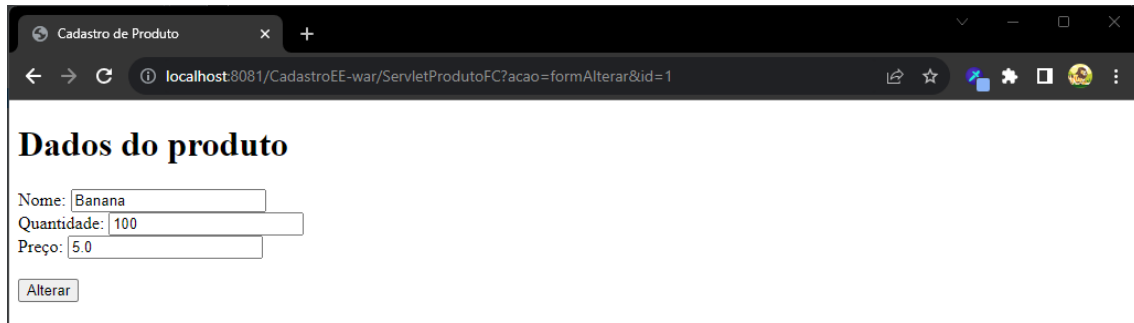
Resultados da execução:



Novo produto:



Alterar:



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

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

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

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

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

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

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

3º Procedimento – Melhorando o Design da Interface

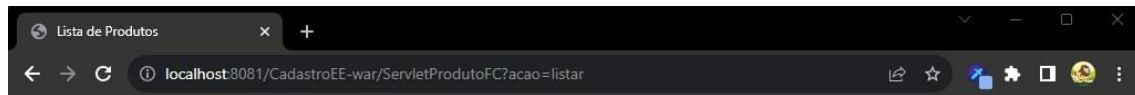
ProdutoList.jsp:

```
6 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
7 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
8 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9 "http://www.w3.org/TR/html4/loose.dtd">
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
13 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+aeP/YC94hEpV"
14 <title>Lista de Produtos</title>
15 </head>
16 <body class="container">
17 <h1>Listagem de Produtos</h1>
18 <a class="btn btn-primary m-2" href="ServletProdutoFC?acao=formIncluir">Novo Produto</a>
19 <table class="table table-striped" border="1">
20 <tr class="table-dark">
21 <th></th>
22 <th>Nome</th>
23 <th>Quantidade</th>
24 <th>Preço</th>
25 <th>Opções</th>
26 </tr>
27 <!-- Aqui virá a lista de produtos -->
28 <!-- Loop para exibir a lista de produtos -->
29 <c:forEach items="${produtos}" var="produto">
30 <tr>
31 <td>${produto.idProduto}</td>
32 <td>${produto.nome}</td>
33 <td>${produto.quantidade}</td>
34 <td>${produto.precoVenda}</td>
35 <td>
36 <a class="btn btn-primary btn-sm" href="ServletProdutoFC?acao=formAlterar&id=${produto.idProduto}">Alterar</a>
37 <a class="btn btn-danger btn-sm" href="ServletProdutoFC?acao=excluir&id=${produto.idProduto}">Excluir</a>
38 </td>
39 </tr>
40 </c:forEach>
41 </table>
42 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-Hwwvtg8No3l"
43 </body>
44 </html>
```

ProdutoDados.jsp:

```
6 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
7 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta charset="UTF-8">
12 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+aeP/YC94hEpV"
13 <title>Cadastro de Produto</title>
14 </head>
15 <body class="container">
16 <h1>${empty produto ? 'Cadastro de Produto' : 'Dados do produto'}</h1>
17 <form class="form" action="ServletProdutoFC" method="post">
18 <input type="hidden" name="acao" value="${not empty produto ? 'alterar' : 'incluir'}">
19
20 <c:if test="${not empty produto}">
21 <input type="hidden" name="id" value="${produto.idProduto}">
22 </c:if>
23 <div class="mb-3">
24 <label class="form-label" form="nome">Nome:</label>
25 <input class="form-control" type="text" id="nome" name="nome" value="${not empty produto ? produto.nome : ''}">
26 </div>
27 <div class="mb-3">
28 <label class="form-label" form="quantidade">Quantidade:</label>
29 <input class="form-control" type="text" id="quantidade" name="quantidade" value="${not empty produto ? produto.quantidade : ''}">
30 </div>
31 <div class="mb-3">
32 <label class="form-label" form="preco">Preço:</label>
33 <input class="form-control" type="text" id="preco" name="preco" value="${not empty produto ? produto.precoVenda : ''}">
34 </div>
35 <input class="btn btn-primary" type="submit" value="${not empty produto ? 'Alterar' : 'Incluir'}">
36 </form>
37 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-Hwwvtg8No3l"
38 </body>
39 </html>
```

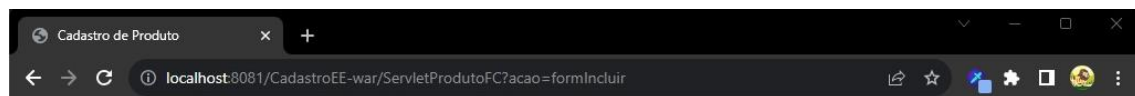
Resultados da execução:



Listagem de Produtos

[Novo Produto](#)

#	Nome	Quantidade	Preço	Opções
1	Banana	100	5.0	Alterar Excluir
3	Laranja	500	2.0	Alterar Excluir
4	Manga	800	4.0	Alterar Excluir



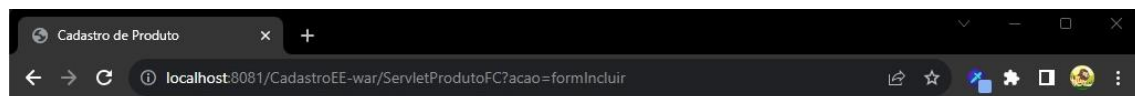
Cadastro de Produto

Nome:

Quantidade:

Preço:

[Incluir](#)



Cadastro de Produto

Nome:

Quantidade:

Preço:

[Incluir](#)

a) Como o framework Bootstrap é utilizado?

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

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

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

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

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

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

Conclusão

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