



# Estácio

## Estácio - Mundo 3 - Missão Nível 4

Faculdade Estácio - Polo Centro – Belo Horizonte – MG

Curso: Desenvolvimento Full Stack.

Disciplina: Nível 4: Vamos integrar sistemas!

RPG0017.

Semestre Letivo: 3

Integrante: André Luiz Ferreira da Silva

Repositório: <https://github.com/Andre-Luiz22/m3-n4>

IDE: NetBeans

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

## **Título da Prática**

Vamos integrar sistemas !

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

## **Objetivos da Prática**

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

1º Procedimento | Camadas de Persistência e Controle

## **Arquivos do Projeto**

CadastroEE-ejb

Cadastroee.model

Movimento

```

1 package cadastroee.model;
2
3 import jakarta.persistence.Basic;
4 import jakarta.persistence.Column;
5 import jakarta.persistence.Entity;
6 import jakarta.persistence.GeneratedValue;
7 import jakarta.persistence.GenerationType;
8 import jakarta.persistence.Id;
9 import jakarta.persistence.JoinColumn;
10 import jakarta.persistence.ManyToOne;
11 import jakarta.persistence.NamedQueries;
12 import jakarta.persistence.NamedQuery;
13 import jakarta.persistence.Table;
14 import jakarta.validation.constraints.Size;
15 import jakarta.xml.bind.annotation.XmlRootElement;
16 import java.io.Serializable;
17 import java.math.BigDecimal;
18
19 /**
20  *
21  * @author paulomulliton
22  */
23 @Entity
24 @Table(name = "movimento")
25 @XmlRootElement
26 @NamedQueries({
27     @NamedQuery(name = "Movimento.findAll", query = "SELECT m FROM Movimento m"),
28     @NamedQuery(name = "Movimento.findByIdMovimento", query = "SELECT m FROM Movimento m WHERE m.idMovimento = :idMovimento"),
29     @NamedQuery(name = "Movimento.findByIdPorQuantidade", query = "SELECT m FROM Movimento m WHERE m.quantidade = :quantidade"),
30     @NamedQuery(name = "Movimento.findByIdPorTipo", query = "SELECT m FROM Movimento m WHERE m.tipo = :tipo"),
31     @NamedQuery(name = "Movimento.findByIdPorValorUnitario", query = "SELECT m FROM Movimento m WHERE m.valorUnitario = :valorUnitario"))
32 public class Movimento implements Serializable {
33
34     private static final long serialVersionUID = 1L;
35     @Id
36     @GeneratedValue(strategy = GenerationType.IDENTITY)
37     @Basic(optional = false)
38     @Column(name = "idMovimento")
39     private Integer idMovimento;
40     @Column(name = "quantidade")
41     private Integer quantidade;
42     @Size(max = 1)
43     @Column(name = "tipo")
44     private String tipo;
45     // @Max(value=?) @Min(value=?)//If you know range of your decimal fields consider using these annotations to enforce field validation
46     @Column(name = "valorUnitario")
47     private BigDecimal valorUnitario;
48     @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa")
49     @ManyToOne(optional = false)
50     private Pessoa idPessoa;
51     @JoinColumn(name = "idProduto", referencedColumnName = "idProduto")
52     @ManyToOne(optional = false)
53     private Produto idProduto;
54     @JoinColumn(name = "idUsuario", referencedColumnName = "idUsuario")
55     @ManyToOne(optional = false)
56     private Usuario idUsuario;
57
58     public Movimento() {
59     }
60
61     public Movimento(Integer idMovimento) {
62         this.idMovimento = idMovimento;
63     }
64
65     public Integer getIdMovimento() {
66         return idMovimento;
67     }
68
69     public void setIdMovimento(Integer idMovimento) {
70         this.idMovimento = idMovimento;
71     }
72
73     public Integer getQuantidade() {
74         return quantidade;
75     }
76
77     public void setQuantidade(Integer quantidade) {
78         this.quantidade = quantidade;
79     }
80
81     public String getTipo() {
82         return tipo;
83     }
84
85     public void setTipo(String tipo) {
86         this.tipo = tipo;
87     }
88
89     public BigDecimal getValorUnitario() {
90         return valorUnitario;
91     }
92
93     public void setValorUnitario(BigDecimal valorUnitario) {
94         this.valorUnitario = valorUnitario;
95     }
96
97     public Pessoa getIdPessoa() {
98         return idPessoa;
99     }
100
101     public void setIdPessoa(Pessoa idPessoa) {
102         this.idPessoa = idPessoa;
103     }
104
105     public Produto getIdProduto() {
106         return idProduto;
107     }
108
109     public void setIdProduto(Produto idProduto) {
110         this.idProduto = idProduto;
111     }
112
113     public Usuario getIdUsuario() {
114         return idUsuario;
115     }
116
117     public void setIdUsuario(Usuario idUsuario) {
118         this.idUsuario = idUsuario;
119     }
120
121     @Override
122     public int hashCode() {
123         int hash = 0;
124         hash = (idMovimento != null ? idMovimento.hashCode() : 0);
125         return hash;
126     }
127
128     @Override
129     public boolean equals(Object object) {
130         // TODO: Warning! This method may fail in the case the id fields are not set
131         if (!(object instanceof Movimento)) {
132             return false;
133         }
134         Movimento other = (Movimento) object;
135         if (((this.idMovimento == null & other.idMovimento != null) || (this.idMovimento != null & !this.idMovimento.equals(other.idMovimento))) {
136             return false;
137         }
138         return true;
139     }
140
141     @Override
142     public String toString() {
143         return "cadastroee.model.Movimento[ idMovimento=" + idMovimento + " ]";
144     }
145
146 }

```

Pessoa

## Pessoa Fisica

```

1 package cadastroee.model;
2
3 import jakarta.persistence.Basic;
4 import jakarta.persistence.Column;
5 import jakarta.persistence.Entity;
6 import jakarta.persistence.Id;
7 import jakarta.persistence.JoinColumn;
8 import jakarta.persistence.NamedQueries;
9 import jakarta.persistence.NamedQuery;
10 import jakarta.persistence.OneToOne;
11 import jakarta.persistence.Table;
12 import jakarta.validation.constraints.NotNull;
13 import jakarta.validation.constraints.Size;
14 import jakarta.xml.bind.annotation.XmlRootElement;
15 import java.io.Serializable;
16
17 /**
18  *
19  * @author paulomueliton
20  */
21 @Entity
22 @Table(name = "pessoaFisica")
23 @XmlRootElement
24 @NamedQueries({
25     @NamedQuery(name = "PessoaFisica.findAll", query = "SELECT p FROM PessoaFisica p"),
26     @NamedQuery(name = "PessoaFisica.findByIdPessoa", query = "SELECT p FROM PessoaFisica p WHERE p.idPessoa = :idPessoa"),
27     @NamedQuery(name = "PessoaFisica.findByCpf", query = "SELECT p FROM PessoaFisica p WHERE p.cpf = :cpf")}
28 )
29 public class PessoaFisica implements Serializable {
30
31     private static final long serialVersionUID = 1L;
32     @Id
33     @Basic(optional = false)
34     @NotNull
35     @Column(name = "idPessoa")
36     private Integer idPessoa;
37     @Basic(optional = false)
38     @NotNull
39     @Size(min = 1, max = 11)
40     @Column(name = "cpf")
41     private String cpf;
42     @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa", insertable = false, updatable = false)
43     @OneToOne(optional = false)
44     private PessoaFisica pessoa;
45
46     public PessoaFisica() {
47     }
48
49     public PessoaFisica(Integer idPessoa) {
50         this.idPessoa = idPessoa;
51     }
52
53     public PessoaFisica(Integer idPessoa, String cpf) {
54         this.idPessoa = idPessoa;
55         this.cpf = cpf;
56     }
57
58     public Integer getIdPessoa() {
59         return idPessoa;
60     }
61
62     public void setIdPessoa(Integer idPessoa) {
63         this.idPessoa = idPessoa;
64     }
65
66     public String getCpf() {
67         return cpf;
68     }
69
70     public void setCpf(String cpf) {
71         this.cpf = cpf;
72     }
73
74     public PessoaFisica getPessoa() {
75         return pessoa;
76     }
77
78     public void setPessoa(PessoaFisica pessoa) {
79         this.pessoa = pessoa;
80     }
81
82     @Override
83     public int hashCode() {
84         int hash = 0;
85         hash += (idPessoa != null ? idPessoa.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 PessoaFisica)) {
93             return false;
94         }
95         PessoaFisica other = (PessoaFisica) object;
96         if (((this.idPessoa == null && other.idPessoa != null) || (this.idPessoa != null && !this.idPessoa.equals(other.idPessoa))) {
97             return false;
98         }
99         return true;
100     }
101
102     @Override
103     public String toString() {
104         return "cadastroee.model.PessoaFisica[ idPessoa=" + idPessoa + " ]";
105     }
106 }
107

```

Pessoa Juridica

```

1 package cadastr0e.model;
2
3 import jakarta.persistence.Basic;
4 import jakarta.persistence.Column;
5 import jakarta.persistence.Entity;
6 import jakarta.persistence.Id;
7 import jakarta.persistence.JoinColumn;
8 import jakarta.persistence.NamedQueries;
9 import jakarta.persistence.NamedQuery;
10 import jakarta.persistence.OneToOne;
11 import jakarta.persistence.Table;
12 import jakarta.validation.constraints.NotNull;
13 import jakarta.validation.constraints.Size;
14 import jakarta.xml.bind.annotation.XmlRootElement;
15 import java.io.Serializable;
16
17 /**
18  *
19  * @author paulomueliton
20  */
21 @Entity
22 @Table(name = "pessoaJuridica")
23 @XmlRootElement
24 @NamedQueries({
25     @NamedQuery(name = "PessoaJuridica.findAll", query = "SELECT p FROM PessoaJuridica p"),
26     @NamedQuery(name = "PessoaJuridica.findByIdPessoa", query = "SELECT p FROM PessoaJuridica p WHERE p.idPessoa = :idPessoa"),
27     @NamedQuery(name = "PessoaJuridica.findByCnpj", query = "SELECT p FROM PessoaJuridica p WHERE p.cnpj = :cnpj")})
28 public class PessoaJuridica implements Serializable {
29
30     private static final long serialVersionUID = 1L;
31
32     @Id
33     @Basic(optional = false)
34     @NotNull
35     @Column(name = "idPessoa")
36     private Integer idPessoa;
37     @Basic(optional = false)
38     @NotNull
39     @Size(min = 1, max = 10)
40     @Column(name = "cnpj")
41     private String cnpj;
42     @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa", insertable = false, updatable = false)
43     @OneToOne(optional = false)
44     private Pessoa pessoa;
45
46     public PessoaJuridica() {
47     }
48
49     public PessoaJuridica(Integer idPessoa) {
50         this.idPessoa = idPessoa;
51     }
52
53     public PessoaJuridica(Integer idPessoa, String cnpj) {
54         this.idPessoa = idPessoa;
55         this.cnpj = cnpj;
56     }
57
58     public Integer getIdPessoa() {
59         return idPessoa;
60     }
61
62     public void setIdPessoa(Integer idPessoa) {
63         this.idPessoa = idPessoa;
64     }
65
66     public String getCnpj() {
67         return cnpj;
68     }
69
70     public void setCnpj(String cnpj) {
71         this.cnpj = cnpj;
72     }
73
74     public Pessoa getPessoa() {
75         return pessoa;
76     }
77
78     public void setPessoa(Pessoa pessoa) {
79         this.pessoa = pessoa;
80     }
81
82     @Override
83     public int hashCode() {
84         int hash = 0;
85         hash += (idPessoa != null ? idPessoa.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 PessoaJuridica)) {
93             return false;
94         }
95         PessoaJuridica other = (PessoaJuridica) object;
96         if (((this.idPessoa == null && other.idPessoa != null) || (this.idPessoa != null && !this.idPessoa.equals(other.idPessoa))) {
97             return false;
98         }
99         return true;
100     }
101
102     @Override
103     public String toString() {
104         return "cadastr0e.model.PessoaJuridica[ idPessoa=" + idPessoa + " ]";
105     }
106 }
107

```

Produto

```

1 package cadastr0ee.model;
2
3 import jakarta.persistence.Basic;
4 import jakarta.persistence.CascadeType;
5 import jakarta.persistence.Column;
6 import jakarta.persistence.Entity;
7 import jakarta.persistence.GeneratedValue;
8 import jakarta.persistence.GenerationType;
9 import jakarta.persistence.Id;
10 import jakarta.persistence.NamedQueries;
11 import jakarta.persistence.NamedQuery;
12 import jakarta.persistence.OneToOne;
13 import jakarta.persistence.Table;
14 import jakarta.validation.constraints.NotNull;
15 import jakarta.validation.constraints.Size;
16 import jakarta.xml.bind.annotation.XmlRootElement;
17 import jakarta.xml.bind.annotation.XmlTransient;
18 import java.io.Serializable;
19 import java.util.Collection;
20
21 /**
22  *
23  * @author paulomeliton
24  */
25 @Entity
26 @Table(name = "produto")
27 @XmlRootElement
28 @NamedQueries({
29     @NamedQuery(name = "Produto.findAll", query = "SELECT p FROM Produto p"),
30     @NamedQuery(name = "Produto.findByIdProduto", query = "SELECT p FROM Produto p WHERE p.idProduto = :idProduto"),
31     @NamedQuery(name = "Produto.findByName", query = "SELECT p FROM Produto p WHERE p.nome = :nome"),
32     @NamedQuery(name = "Produto.findByQuantidade", query = "SELECT p FROM Produto p WHERE p.quantidade = :quantidade"),
33     @NamedQuery(name = "Produto.findByPrecoVenda", query = "SELECT p FROM Produto p WHERE p.precoVenda = :precoVenda")})
34 public class Produto implements Serializable {
35
36     private static final long serialVersionUID = 1L;
37     @Id
38     @GeneratedValue(strategy = GenerationType.IDENTITY)
39     @Basic(optional = false)
40     @Column(name = "idProduto")
41     private Integer idProduto;
42     @Size(max = 255)
43     @Column(name = "nome")
44     private String nome;
45     @Column(name = "quantidade")
46     private Integer quantidade;
47     // @Max(value=?) @Min(value=?)//if you know range of your decimal fields consider using these annotations to enforce field validation
48     @Column(name = "precoVenda")
49     private Float precoVenda;
50     @OneToOne(cascade = CascadeType.ALL, mappedBy = "idProduto")
51     private Collection<Movimento> movimentoCollection;
52
53     public Produto() {
54     }
55
56     public Produto(Integer idProduto) {
57         this.idProduto = idProduto;
58     }
59
60     public Integer getIdProduto() {
61         return idProduto;
62     }
63
64     public void setIdProduto(Integer idProduto) {
65         this.idProduto = idProduto;
66     }
67
68     public String getNome() {
69         return nome;
70     }
71
72     public void setNome(String nome) {
73         this.nome = nome;
74     }
75
76     public Integer getQuantidade() {
77         return quantidade;
78     }
79
80     public void setQuantidade(Integer quantidade) {
81         this.quantidade = quantidade;
82     }
83
84     public Float getPrecoVenda() {
85         return precoVenda;
86     }
87
88     public void setPrecoVenda(Float precoVenda) {
89         this.precoVenda = precoVenda;
90     }
91
92     @XmlTransient
93     public Collection<Movimento> getMovimentoCollection() {
94         return movimentoCollection;
95     }
96
97     public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
98         this.movimentoCollection = movimentoCollection;
99     }
100
101     @Override
102     public int hashCode() {
103         int hash = 0;
104         hash += (idProduto != null ? idProduto.hashCode() : 0);
105         return hash;
106     }
107
108     @Override
109     public boolean equals(Object object) {
110         // TODO: Warning - this method won't work in the case the id fields are not set
111         if (!(Object instanceof Produto)) {
112             return false;
113         }
114         Produto other = (Produto) object;
115         if (((this.idProduto == null && other.idProduto != null) || (this.idProduto != null && !this.idProduto.equals(other.idProduto))) {
116             return false;
117         }
118         return true;
119     }
120
121     @Override
122     public String toString() {
123         return "Cadastr0ee.model.Produto[ idProduto=" + idProduto + " ]";
124     }
125 }
126
127

```

Usuario



```

1 package cadastreoe.model;
2
3 import jakarta.persistence.Basic;
4 import jakarta.persistence.CascadeType;
5 import jakarta.persistence.Column;
6 import jakarta.persistence.Entity;
7 import jakarta.persistence.GeneratedValue;
8 import jakarta.persistence.GenerationType;
9 import jakarta.persistence.Id;
10 import jakarta.persistence.NamedQueries;
11 import jakarta.persistence.NamedQuery;
12 import jakarta.persistence.OneToMany;
13 import jakarta.persistence.Table;
14 import jakarta.validation.constraints.Size;
15 import jakarta.xml.bind.annotation.XmlRootElement;
16 import jakarta.xml.bind.annotation.XmlTransient;
17 import java.io.Serializable;
18 import java.util.Collection;
19
20 /**
21  *
22  * @author paulomueliton
23  */
24 @Entity
25 @Table(name = "usuario")
26 @XmlRootElement
27 @NamedQueries({
28     @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM Usuario u"),
29     @NamedQuery(name = "Usuario.findByIdUsuario", query = "SELECT u FROM Usuario u WHERE u.idUsuario = :idUsuario"),
30     @NamedQuery(name = "Usuario.findByLogin", query = "SELECT u FROM Usuario u WHERE u.login = :login"),
31     @NamedQuery(name = "Usuario.findBySenha", query = "SELECT u FROM Usuario u WHERE u.senha = :senha")})
32 public class Usuario implements Serializable {
33
34     private static final long serialVersionUID = 1L;
35     @Id
36     @GeneratedValue(strategy = GenerationType.IDENTITY)
37     @Basic(optional = false)
38     @Column(name = "idUsuario")
39     private Integer idUsuario;
40     @Size(max = 100)
41     @Column(name = "login")
42     private String login;
43     @Size(max = 30)
44     @Column(name = "senha")
45     private String senha;
46     @OneToMany(cascade = CascadeType.ALL, mappedBy = "idUsuario")
47     private Collection<Movimento> movimentoCollection;
48
49     public Usuario() {
50     }
51
52     public Usuario(Integer idUsuario) {
53         this.idUsuario = idUsuario;
54     }
55
56     public Integer getIdUsuario() {
57         return idUsuario;
58     }
59
60     public void setIdUsuario(Integer idUsuario) {
61         this.idUsuario = idUsuario;
62     }
63
64     public String getLogin() {
65         return login;
66     }
67
68     public void setLogin(String login) {
69         this.login = login;
70     }
71
72     public String getSenha() {
73         return senha;
74     }
75
76     public void setSenha(String senha) {
77         this.senha = senha;
78     }
79
80     @XmlTransient
81     public Collection<Movimento> getMovimentoCollection() {
82         return movimentoCollection;
83     }
84
85     public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
86         this.movimentoCollection = movimentoCollection;
87     }
88
89     @Override
90     public int hashCode() {
91         int hash = 0;
92         hash += (idUsuario != null ? idUsuario.hashCode() : 0);
93         return hash;
94     }
95
96     @Override
97     public boolean equals(Object object) {
98         // TODO: Warning - this method won't work in the case the id fields are not set
99         if (!(object instanceof Usuario)) {
100             return false;
101         }
102         Usuario other = (Usuario) object;
103         if ((this.idUsuario == null && other.idUsuario != null) || (this.idUsuario != null && !this.idUsuario.equals(other.idUsuario))) {
104             return false;
105         }
106         return true;
107     }
108
109     @Override
110     public String toString() {
111         return "cadastreoe.model.Usuario[ idUsuario=" + idUsuario + " ]";
112     }
113 }
114
115

```

Persistence.xml

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

CadastroEE-war

## Cadastroee.servlets

### ServletProduto

```
1 package cadastroee.servlets;
2
3 import cadastroee.controller.ProdutoFacadeLocal;
4 import jakarta.ejb.EJB;
5 import java.io.IOException;
6 import java.io.PrintWriter;
7 import jakarta.servlet.ServletException;
8 import jakarta.servlet.http.HttpServlet;
9 import jakarta.servlet.http.HttpServletRequest;
10 import jakarta.servlet.http.HttpServletResponse;
11
12 /**
13  *
14  * @author paulowueliton
15  */
16 public class ServletProduto extends HttpServlet {
17
18     @EJB
19     ProdutoFacadeLocal facade;
20
21     /**
22      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
23      * methods.
24      *
25      * @param request servlet request
26      * @param response servlet response
27      * @throws ServletException if a servlet-specific error occurs
28      * @throws IOException if an I/O error occurs
29      */
30     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
31         throws ServletException, IOException {
32         response.setContentType("text/html;charset=UTF-8");
33         try (PrintWriter out = response.getWriter()) {
34             /* TODO output your page here. You may use following sample code. */
35             out.println("<DOCTYPE html>");
36             out.println("<html>");
37             out.println("<head>");
38             out.println("<title>Servlet ServletProduto</title>");
39             out.println("</head>");
40             out.println("<body>");
41             out.println("<h1>Servlet ServletProduto at " + request.getContextPath() + "</h1>");
42             out.println("<ul>");
43             this.facade.findAll().forEach(produto -> out.println("<li>" + produto.getNome() + "</li>"));
44             out.println("</ul>");
45             out.println("</body>");
46             out.println("</html>");
47         }
48     }
49
50     // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
51     /**
52      * Handles the HTTP <code>GET</code> method.
53      *
54      * @param request servlet request
55      * @param response servlet response
56      * @throws ServletException if a servlet-specific error occurs
57      * @throws IOException if an I/O error occurs
58      */
59     @Override
60     protected void doGet(HttpServletRequest request, HttpServletResponse response)
61         throws ServletException, IOException {
62         processRequest(request, response);
63     }
64
65     /**
66      * Handles the HTTP <code>POST</code> method.
67      *
68      * @param request servlet request
69      * @param response servlet response
70      * @throws ServletException if a servlet-specific error occurs
71      * @throws IOException if an I/O error occurs
72      */
73     @Override
74     protected void doPost(HttpServletRequest request, HttpServletResponse response)
75         throws ServletException, IOException {
76         processRequest(request, response);
77     }
78
79     /**
80      * Returns a short description of the servlet.
81      *
82      * @return a String containing servlet description
83      */
84     @Override
85     public String getServletInfo() {
86         return "Short description";
87     } // </editor-fold>
88
89 }
```

## Execução



## Analise e Conclusão

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

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

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

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

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

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

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

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

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

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

## 2º Procedimento | Interface cadastral com servlets e JSPs

CadastroEE-war

Cadastroee.servlets

ServletProdutoFC

```

1 package cadastreose.servlets;
2
3 import cadastreose.controller.ProdutoFacadeLocal;
4 import cadastreose.model.Produto;
5 import jakarta.ejb.EJB;
6 import jakarta.servlet.RequestDispatcher;
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
14 /**
15  *
16  * @author paulomeliton
17  */
18 public class ServletProdutoFC extends HttpServlet {
19
20     @EJB
21     ProdutoFacadeLocal facade;
22
23     /**
24      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
25      * methods
26      *
27      * @param request servlet request
28      * @param response servlet response
29      * @throws ServletException if a servlet-specific error occurs
30      * @throws IOException if an I/O error occurs
31      */
32     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
33         throws ServletException, IOException {
34         String acao = request.getParameter("acao");
35         String destino = "ProdutoLista.jsp";
36
37         if (acao == null) {
38             acao = "listar";
39         }
40
41         switch (acao) {
42             case "listar" -> {
43                 request.setAttribute("produtos", facade.findAll());
44                 break;
45             }
46             case "incluir" -> {
47                 Produto produto = this.getProduto(request);
48                 facade.create(produto);
49                 request.setAttribute("produtos", facade.findAll());
50                 break;
51             }
52             case "alterar" -> {
53                 Produto produto = this.getProduto(request);
54                 facade.edit(produto);
55                 request.setAttribute("produtos", facade.findAll());
56                 break;
57             }
58             case "excluir" -> {
59                 Integer id = Integer.valueOf(request.getParameter("id"));
60                 Produto produto = facade.findById(id);
61                 facade.remove(produto);
62                 request.setAttribute("produtos", facade.findAll());
63                 break;
64             }
65             case "reincluir" -> {
66                 destino = "ProdutoBados.jsp";
67                 break;
68             }
69             case "formAlterar" -> {
70                 destino = "ProdutoBados.jsp";
71                 Integer id;
72                 id = Integer.valueOf(request.getParameter("id"));
73                 request.setAttribute("produto", facade.findById(id));
74                 break;
75             }
76         }
77
78         RequestDispatcher dispatcher = request.getRequestDispatcher(destino);
79         dispatcher.forward(request, response);
80     }
81
82     private Produto getProduto(HttpServletRequest request) {
83         String idParametro = request.getParameter("id");
84         Integer id = Integer.valueOf(idParametro);
85         String nome = request.getParameter("nome");
86         Integer quantidade = Integer.valueOf(request.getParameter("quantidade"));
87         Float precoVenda = Float.valueOf(request.getParameter("precoVenda"));
88
89         Produto produto = new Produto();
90         produto.setNome(nome);
91         produto.setPrecoVenda(precoVenda);
92         produto.setQuantidade(quantidade);
93
94         if (idParametro != null) {
95             produto.setIdProduto(id);
96         }
97         return produto;
98     }
99
100     /**
101      * <code>editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
102      */
103     /**
104      * Handles the HTTP <code>GET</code> method.
105      *
106      * @param request servlet request
107      * @param response servlet response
108      * @throws ServletException if a servlet-specific error occurs
109      * @throws IOException if an I/O error occurs
110      */
111     @Override
112     protected void doGet(HttpServletRequest request, HttpServletResponse response)
113         throws ServletException, IOException {
114         processRequest(request, response);
115     }
116
117     /**
118      * Handles the HTTP <code>POST</code> method.
119      *
120      * @param request servlet request
121      * @param response servlet response
122      * @throws ServletException if a servlet-specific error occurs
123      * @throws IOException if an I/O error occurs
124      */
125     @Override
126     protected void doPost(HttpServletRequest request, HttpServletResponse response)
127         throws ServletException, IOException {
128         processRequest(request, response);
129     }
130
131     /**
132      * Returns a short description of the servlet.
133      *
134      * @return a String containing servlet description
135      */
136     @Override
137     public String getServletInfo() {
138         return "Short description";
139     }
140 }

```

ProdutoLista.jsp

```

1  %--
2  Document : ProdutoLista
3  Created on : 11 de mai. de 2024, 12:24:36
4  Author : andreluiz
5  --%
6
7  %page import="java.util.List"%
8  %page import="cadastroe.model.Produto"%
9  %page contentType="text/html" pageEncoding="UTF-8"%
10 <!DOCTYPE html>
11 <html>
12 <head>
13 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14 <title>Listagem de Produtos</title>
15 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWtZyJpPEjISv5WaRU90FeRpok6YctnYvdr5p8lyT2b6jXh0Mh53jYfVGNV" crossorigin="anonymous">
16 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9F0D2LESAAS5SDzOxhy9GkIdks9q9kIdK90" crossorigin="anonymous"></script>
17 </head>
18 <body class="container">
19 <div>
20 <div>
21 <a href="#">Novo Produto</a>
22 <table class="table table-striped">
23 <thead>
24 <tr>
25 <th>Nome</th>
26 <th>Quantidade</th>
27 <th>Preço de Venda</th>
28 <th>Opções</th>
29 </thead>
30 <tbody>
31 <% List<Produto> produtos = (List<Produto>) request.getAttribute("produtos"); %>
32 <% for (Produto produto : produtos) { %>
33 <tr>
34 <td>=<%= produto.getIdProduto() %></td>
35 <td>=<%= produto.getNome() %></td>
36 <td>=<%= produto.getQuantidade() %></td>
37 <td>=<%= produto.getPrecoVenda() %></td>
38 <td>
39 <a href="#">Alterar</a>
40 <a href="#">Excluir</a>
41 </td>
42 </tr>
43 <% } %>
44 </tbody>
45 </table>
46 </div>
47 </div>
48 </html>

```

## ProdutoDados.jsp

```

1  %--
2  Document : ProdutoDados
3  Created on : 13 de mai. de 2024, 12:30:14
4  Author : andreluiz
5  --%
6
7  %page import="cadastroe.model.Produto"%
8  %page contentType="text/html" pageEncoding="UTF-8"%
9  % String acao = "incluir"; %
10 <!DOCTYPE html>
11 <html>
12 <head>
13 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14 <title>=<%= acao == "incluir" ? "Adicionar" : "Editar" %> Produto</title>
15 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWtZyJpPEjISv5WaRU90FeRpok6YctnYvdr5p8lyT2b6jXh0Mh53jYfVGNV" crossorigin="anonymous">
16 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9F0D2LESAAS5SDzOxhy9GkIdks9q9kIdK90" crossorigin="anonymous"></script>
17 </head>
18 <body class="container">
19 <div>
20 <div>
21 <div>
22 <div>
23 <div>
24 <div>
25 <div>
26 <div>
27 <form action="#">
28 <div>
29 <div>
30 <div>
31 <div>
32 <div>
33 <div>
34 <div>
35 <div>
36 <div>
37 <div>
38 <div>
39 <div>
40 <div>
41 <div>
42 <div>
43 <div>
44 <div>
45 <div>
46 <div>
47 <div>
48 <div>
49 <div>
50 <div>
51 <div>
52 <div>
53 <div>
54 <div>
55 <div>
56 <div>
57 <div>
58 <div>

```

## Testes

### Listagem



Novo Produto

#	Nome	Quantidade	Preço de Venda	Opções
2	Laranja	500	2.0	<a href="#">Alterar</a> <a href="#">Excluir</a>
4	Abacate	100	10.0	<a href="#">Alterar</a> <a href="#">Excluir</a>
5	Bergamota	90	3.0	<a href="#">Alterar</a> <a href="#">Excluir</a>
6	Alface	500	9.0	<a href="#">Alterar</a> <a href="#">Excluir</a>
7	Tomate	50	22.0	<a href="#">Alterar</a> <a href="#">Excluir</a>

Incluir

Dados do Produto

Nome:

Ameixa

Quantidade:

100

Preço de Venda:

9.9

Adicionar Produto

Alterar

Dados do Produto

Nome:

Ameixa

Quantidade:

100

Preço de Venda:

7.8

Alterar Produto

Excluir

The screenshot shows a web browser at the URL `localhost:8080/CadastroEE-war/ServletProdutoFC`. The page title is "Listagem de Produtos". There is a blue button labeled "Novo Produto". A dark modal dialog is open, displaying the text "localhost:8080 diz" and "Tem certeza que deseja excluir este produto?", with "Cancelar" and "OK" buttons. Below the dialog is a table of products.

#	Nome	Quantidade		Opções
2	Laranja	500	2.0	<a href="#">Alterar</a> <a href="#">Excluir</a>
4	Abacate	100	10.0	<a href="#">Alterar</a> <a href="#">Excluir</a>
5	Bergamota	90	3.0	<a href="#">Alterar</a> <a href="#">Excluir</a>
6	Alface	500	9.0	<a href="#">Alterar</a> <a href="#">Excluir</a>
7	Tomate	50	22.0	<a href="#">Alterar</a> <a href="#">Excluir</a>
8	Ameixa	100	7.8	<a href="#">Alterar</a> <a href="#">Excluir</a>

## Análise e Conclusão

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

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

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

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

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

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

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

## **Análise e Conclusão**

a) Como o framework Bootstrap é utilizado?

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

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

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

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

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

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