Universidade Estácio de Sá

Polo Bangu - Rio de Janeiro

- **Curso:** Desenvolvimento Full-Stack
- **Disciplina:** Vamos integrar sistemas
- **Semestre:** 3
- **Turma:** 2024.2
- **Aluna:** Clara Martins Azevedo

---

1. Missão Prática - Nível 4

2. Objetivos da Missão 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.

3. Códigos do Projeto:

- Movimento.java:

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

/**
 *
 * @author okidata
 */
@Entity
@Table(name = "Movimento")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Movimento.findAll", query = "SELECT m FROM Movimento m"),
    @NamedQuery(name = "Movimento.findByIdMovimento", query = "SELECT m FROM Movimento m WHERE
m.idMovimento = :idMovimento"),
    @NamedQuery(name = "Movimento.findByQuantidade", query = "SELECT m FROM Movimento m WHERE
m.quantidade = :quantidade"),
    @NamedQuery(name = "Movimento.findByTipo", query = "SELECT m FROM Movimento m WHERE m.tipo =
:tipo"),
    @NamedQuery(name = "Movimento.findByValorUnitario", query = "SELECT m FROM Movimento m WHERE
m.valorUnitario = :valorUnitario")})
public class Movimento implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "idMovimento")
    private Integer idMovimento;
    @Column(name = "quantidade")
    private Integer quantidade;
    @Size(max = 15)
    @Column(name = "tipo")
    private String tipo;
    @Column(name = "valorUnitario")
    private Long valorUnitario;
    @JoinColumn(name = "Pessoa_idPessoa", referencedColumnName = "idPessoa")
    @ManyToOne(optional = false)
```

```
private Pessoa pessoaidPessoa;
@JoinColumn(name = "Produto_idProduto", referencedColumnName = "idProduto")
@ManyToOne(optional = false)
private Produto produtoidProduto;
@JoinColumn(name = "Usuario_idUsuario", referencedColumnName = "idUsuario")
@ManyToOne(optional = false)
private Usuario usuarioidUsuario;

public Movimento() {
}

public Movimento(Integer idMovimento) {
    this.idMovimento = idMovimento;
}

public Integer getIdMovimento() {
    return idMovimento;
}

public void setIdMovimento(Integer idMovimento) {
    this.idMovimento = idMovimento;
}

public Integer getQuantidade() {
    return quantidade;
}

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

public String getTipo() {
    return tipo;
}

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

public Long getValorUnitario() {
    return valorUnitario;
}

public void setValorUnitario(Long valorUnitario) {
    this.valorUnitario = valorUnitario;
}

public Pessoa getPessoaidPessoa() {
    return pessoaidPessoa;
}

public void setPessoaidPessoa(Pessoa pessoaidPessoa) {
    this.pessoaidPessoa = pessoaidPessoa;
}

public Produto getProdutoidProduto() {
    return produtoidProduto;
}

public void setProdutoidProduto(Produto produtoidProduto) {
    this.produtoidProduto = produtoidProduto;
}

public Usuario getUsuarioidUsuario() {
    return usuarioidUsuario;
}

public void setUsuarioidUsuario(Usuario usuarioidUsuario) {
```

```
        this.usuarioidUsuario = usuarioidUsuario;
    }

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

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

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

}
```

• Pessoa.java:

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;
import java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author okidata
 */
@Entity
@Table(name = "Pessoa")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Pessoa.findAll", query = "SELECT p FROM Pessoa p"),
    @NamedQuery(name = "Pessoa.findByIdPessoa", query = "SELECT p FROM Pessoa p WHERE p.idPessoa =
:idPessoa"),
```

```
    @NamedQuery(name = "Pessoa.findByNome", query = "SELECT p FROM Pessoa p WHERE p.nome = :nome"),
    @NamedQuery(name = "Pessoa.findByLocadouro", query = "SELECT p FROM Pessoa p WHERE p.locadouro =
:locadouro"),
    @NamedQuery(name = "Pessoa.findByCidade", query = "SELECT p FROM Pessoa p WHERE p.cidade =
:cidade"),
    @NamedQuery(name = "Pessoa.findByEstado", query = "SELECT p FROM Pessoa p WHERE p.estado =
:estado"),
    @NamedQuery(name = "Pessoa.findByTelefone", query = "SELECT p FROM Pessoa p WHERE p.telefone =
:telefone"),
    @NamedQuery(name = "Pessoa.findByEmail", query = "SELECT p FROM Pessoa p WHERE p.email = :email")})
public class Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Size(max = 255)
    @Column(name = "nome")
    private String nome;
    @Size(max = 255)
    @Column(name = "locadouro")
    private String locadouro;
    @Size(max = 255)
    @Column(name = "cidade")
    private String cidade;
    @Size(max = 2)
    @Column(name = "estado")
    private String estado;
    @Size(max = 11)
    @Column(name = "telefone")
    private String telefone;
    //
@Pattern(regexp="[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0-
9](?:[a-z0-9-]*[a-z0-9])?", message="Invalid email")//if the field contains email address consider using this annotation to enforce
field validation
    @Size(max = 255)
    @Column(name = "email")
    private String email;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaJuridica pessoaJuridica;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaFisica pessoaFisica;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "pessoaidPessoa")
    private Collection<Movimento> movimentoCollection;

    public Pessoa() {
    }

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

    public Integer getIdPessoa() {
        return idPessoa;
    }

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

    public String getNome() {
        return nome;
    }

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

```
}

public String getLocadouro() {
    return locadouro;
}

public void setLocadouro(String locadouro) {
    this.locadouro = locadouro;
}

public String getCidade() {
    return cidade;
}

public void setCidade(String cidade) {
    this.cidade = cidade;
}

public String getEstado() {
    return estado;
}

public void setEstado(String estado) {
    this.estado = estado;
}

public String getTelefone() {
    return telefone;
}

public void setTelefone(String telefone) {
    this.telefone = telefone;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public PessoaJuridica getPessoaJuridica() {
    return pessoaJuridica;
}

public void setPessoaJuridica(PessoaJuridica pessoaJuridica) {
    this.pessoaJuridica = pessoaJuridica;
}

public PessoaFisica getPessoaFisica() {
    return pessoaFisica;
}

public void setPessoaFisica(PessoaFisica pessoaFisica) {
    this.pessoaFisica = pessoaFisica;
}

@XmlTransient
public Collection<Movimento> getMovimentoCollection() {
    return movimentoCollection;
}

public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
    this.movimentoCollection = movimentoCollection;
}

@Override
```

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

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

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

}
```

• PessoaFisica.java:

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

/**
 *
 * @author okidata
 */
@Entity
@Table(name = "PessoaFisica")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "PessoaFisica.findAll", query = "SELECT p FROM PessoaFisica p"),
    @NamedQuery(name = "PessoaFisica.findByIdPessoaFisica", query = "SELECT p FROM PessoaFisica p
WHERE p.idPessoaFisica = :idPessoaFisica"),
    @NamedQuery(name = "PessoaFisica.findByCpf", query = "SELECT p FROM PessoaFisica p WHERE p.cpf =
:cpf")})
public class PessoaFisica implements Serializable {

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

```
    @NotNull
    @Column(name = "idPessoaFisica")
    private Integer idPessoaFisica;
    @Size(max = 255)
    @Column(name = "cpf")
    private String cpf;
    @JoinColumn(name = "idPessoaFisica", referencedColumnName = "idPessoa", insertable = false, updatable =
false)
    @OneToOne(optional = false)
    private Pessoa pessoa;

    public PessoaFisica() {
    }

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

    public Integer getIdPessoaFisica() {
        return idPessoaFisica;
    }

    public void setIdPessoaFisica(Integer idPessoaFisica) {
        this.idPessoaFisica = idPessoaFisica;
    }

    public String getCpf() {
        return cpf;
    }

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

    public Pessoa getPessoa() {
        return pessoa;
    }

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

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

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

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

}

• PessoaJuridica.java:

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

/**
 *
 * @author okidata
 */
@Entity
@Table(name = "PessoaJuridica")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "PessoaJuridica.findAll", query = "SELECT p FROM PessoaJuridica p"),
    @NamedQuery(name = "PessoaJuridica.findByIdPessoaJuridica", query = "SELECT p FROM PessoaJuridica p
WHERE p.idPessoaJuridica = :idPessoaJuridica"),
    @NamedQuery(name = "PessoaJuridica.findByCnpj", query = "SELECT p FROM PessoaJuridica p WHERE
p.cnpj = :cnpj")})
public class PessoaJuridica implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "idPessoaJuridica")
    private Integer idPessoaJuridica;
    @Size(max = 255)
    @Column(name = "cnpj")
    private String cnpj;
    @JoinColumn(name = "idPessoaJuridica", referencedColumnName = "idPessoa", insertable = false, updatable =
false)
    @OneToOne(optional = false)
    private Pessoa pessoa;

    public PessoaJuridica() {
    }

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

    public Integer getIdPessoaJuridica() {
        return idPessoaJuridica;
    }

    public void setIdPessoaJuridica(Integer idPessoaJuridica) {
        this.idPessoaJuridica = idPessoaJuridica;
    }
```

```
    public String getCnpj() {
        return cnpj;
    }

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

    public Pessoa getPessoa() {
        return pessoa;
    }

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

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

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

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

}
```

• Produto.java:

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
```

```
import jakarta.xml.bind.annotation.XmlTransient;
import java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author okidata
 */
@Entity
@Table(name = "Produto")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Produto.findAll", query = "SELECT p FROM Produto p"),
    @NamedQuery(name = "Produto.findByIdProduto", query = "SELECT p FROM Produto p WHERE p.idProduto =
:idProduto"),
    @NamedQuery(name = "Produto.findByNome", query = "SELECT p FROM Produto p WHERE p.nome = :nome"),
    @NamedQuery(name = "Produto.findByQuantidade", query = "SELECT p FROM Produto p WHERE p.quantidade
= :quantidade"),
    @NamedQuery(name = "Produto.findByPrecoVenda", query = "SELECT p FROM Produto p WHERE
p.precoVenda = :precoVenda")})
public class Produto implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "idProduto")
    private Integer idProduto;
    @Size(max = 255)
    @Column(name = "nome")
    private String nome;
    @Column(name = "quantidade")
    private Integer quantidade;
    @Column(name = "precoVenda")
    private Float precoVenda;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "produtoidProduto")
    private Collection<Movimento> movimentoCollection;

    public Produto() {
    }

    public Produto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public Integer getIdProduto() {
        return idProduto;
    }

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

    public String getNome() {
        return nome;
    }

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

    public Integer getQuantidade() {
        return quantidade;
    }

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

```
    public Float getPrecoVenda() {
        return precoVenda;
    }

    public void setPrecoVenda(Float precoVenda) {
        this.precoVenda = precoVenda;
    }

    @XmlTransient
    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }

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

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

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

}
```

- Usuario.java:

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;
```

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

/**
 *
 * @author okidata
 */
@Entity
@Table(name = "Usuario")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM Usuario u"),
    @NamedQuery(name = "Usuario.findByIdUsuario", query = "SELECT u FROM Usuario u WHERE u.idUsuario =
:idUsuario"),
    @NamedQuery(name = "Usuario.findByLoginUsuario", query = "SELECT u FROM Usuario u WHERE
u.loginUsuario = :loginUsuario"),
    @NamedQuery(name = "Usuario.findBySenha", query = "SELECT u FROM Usuario u WHERE u.senha =
:senha")})
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "idUsuario")
    private Integer idUsuario;
    @Size(max = 20)
    @Column(name = "loginUsuario")
    private String loginUsuario;
    @Size(max = 20)
    @Column(name = "senha")
    private String senha;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "usuarioidUsuario")
    private Collection<Movimento> movimentoCollection;

    public Usuario() {
    }

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

    public Integer getIdUsuario() {
        return idUsuario;
    }

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

    public String getLoginUsuario() {
        return loginUsuario;
    }

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

    public String getSenha() {
        return senha;
    }

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

    @XmlTransient
    public Collection<Movimento> getMovimentoCollection() {
```

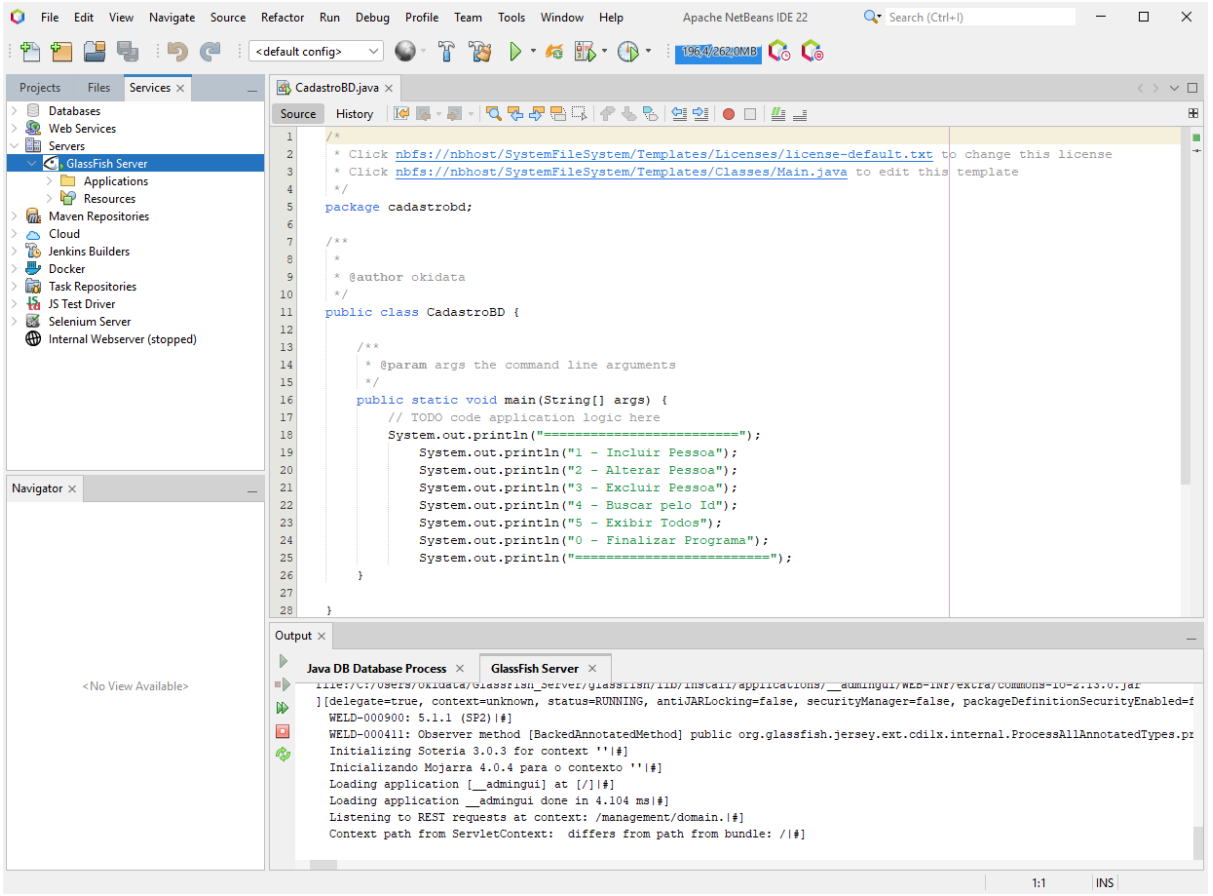
```
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }

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

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

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

}
```

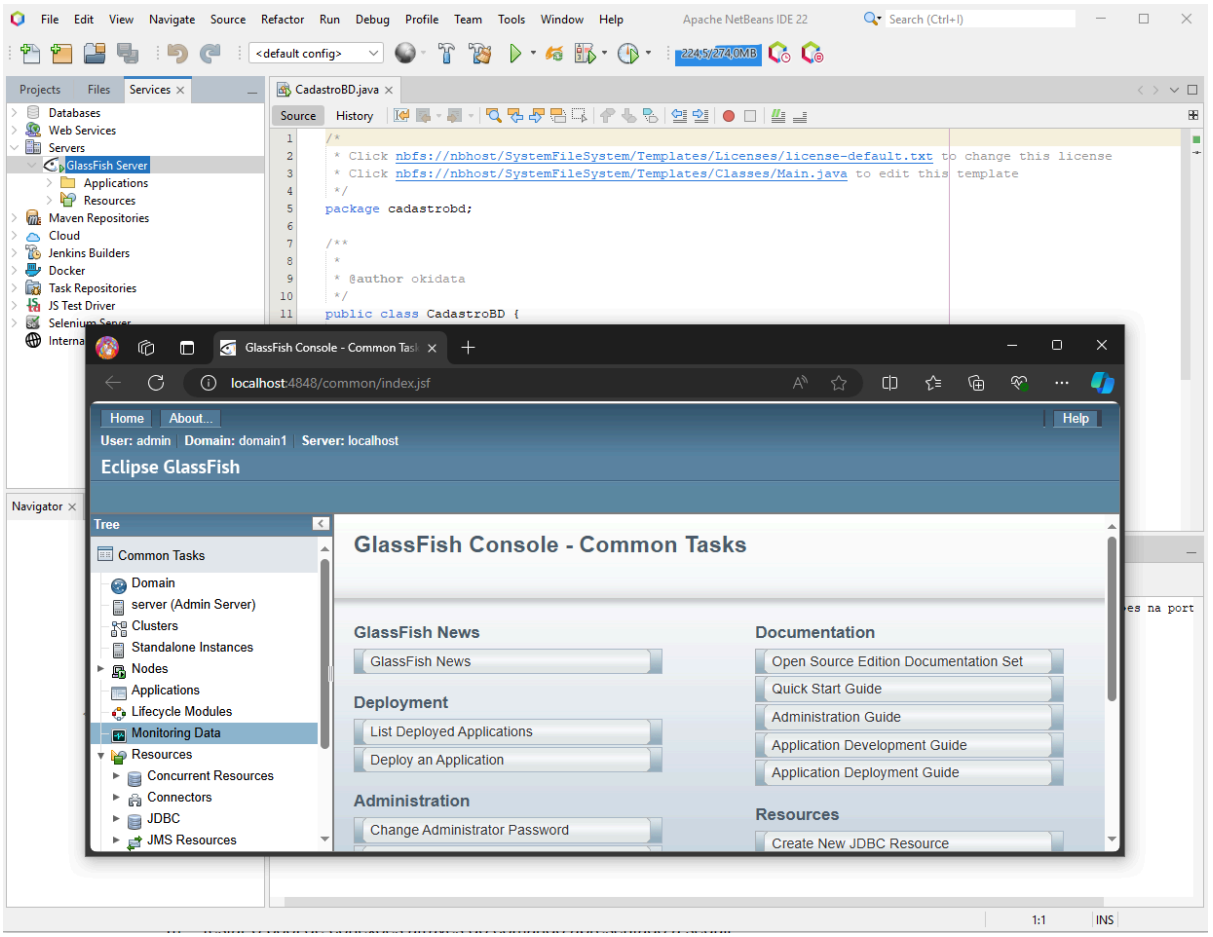
• ServletProduto.java:

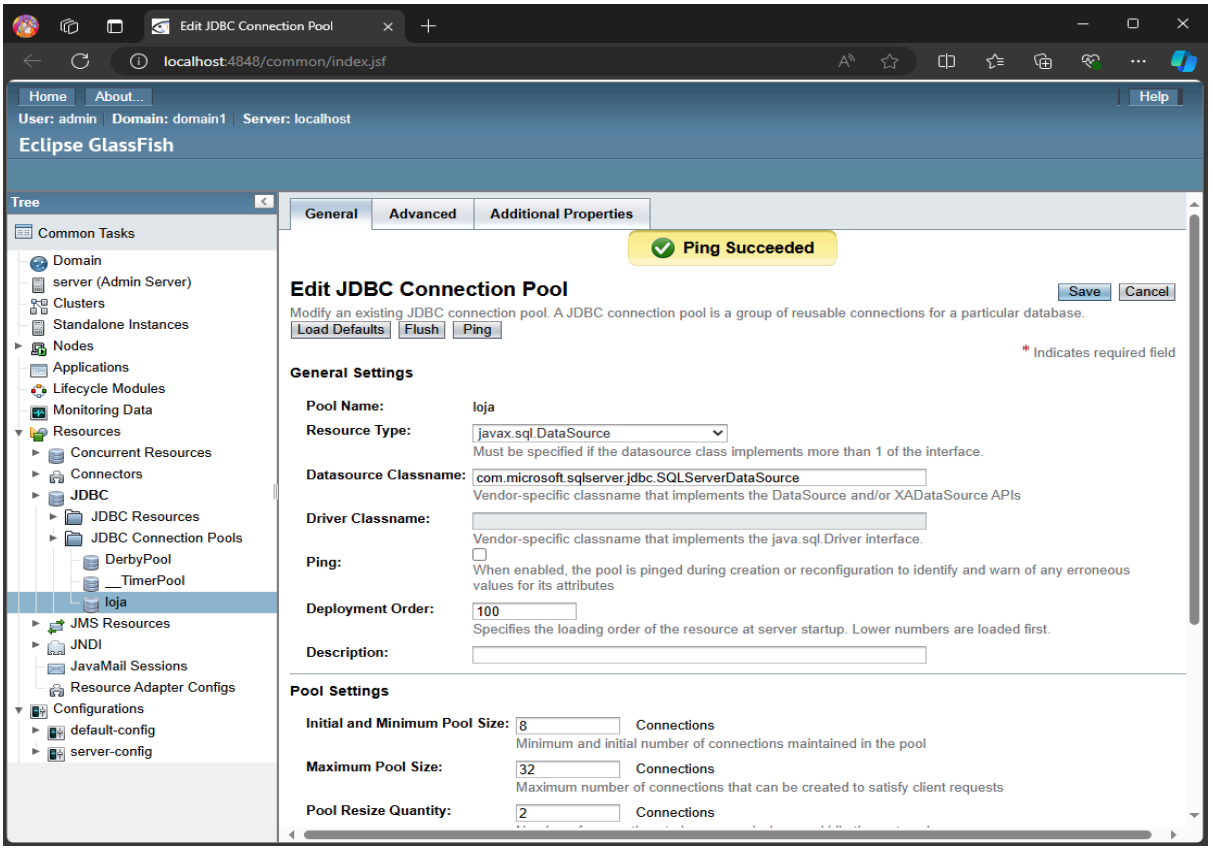
```
package cadastroee.servlets;

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

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

    @EJB
    private ProdutoFacadeLocal facade;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
```

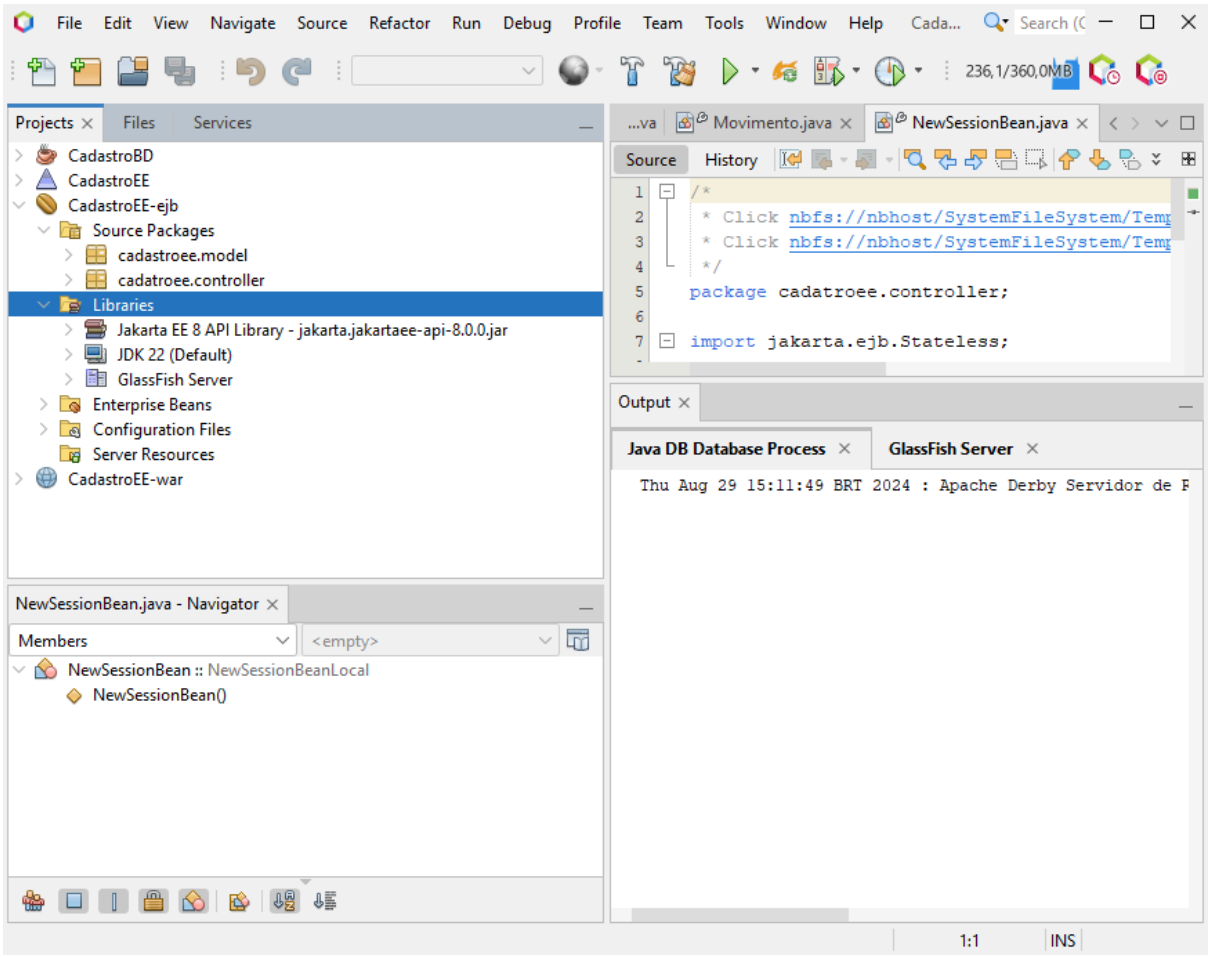
```
        List<Produto> produtos = facade.findAll();

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Lista de Produtos</h1>");
        out.println("<ul>");
        for (Produto produto : produtos) {
            out.println("<li>" + produto.getNome() + "</li>");
        }
        out.println("</ul>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

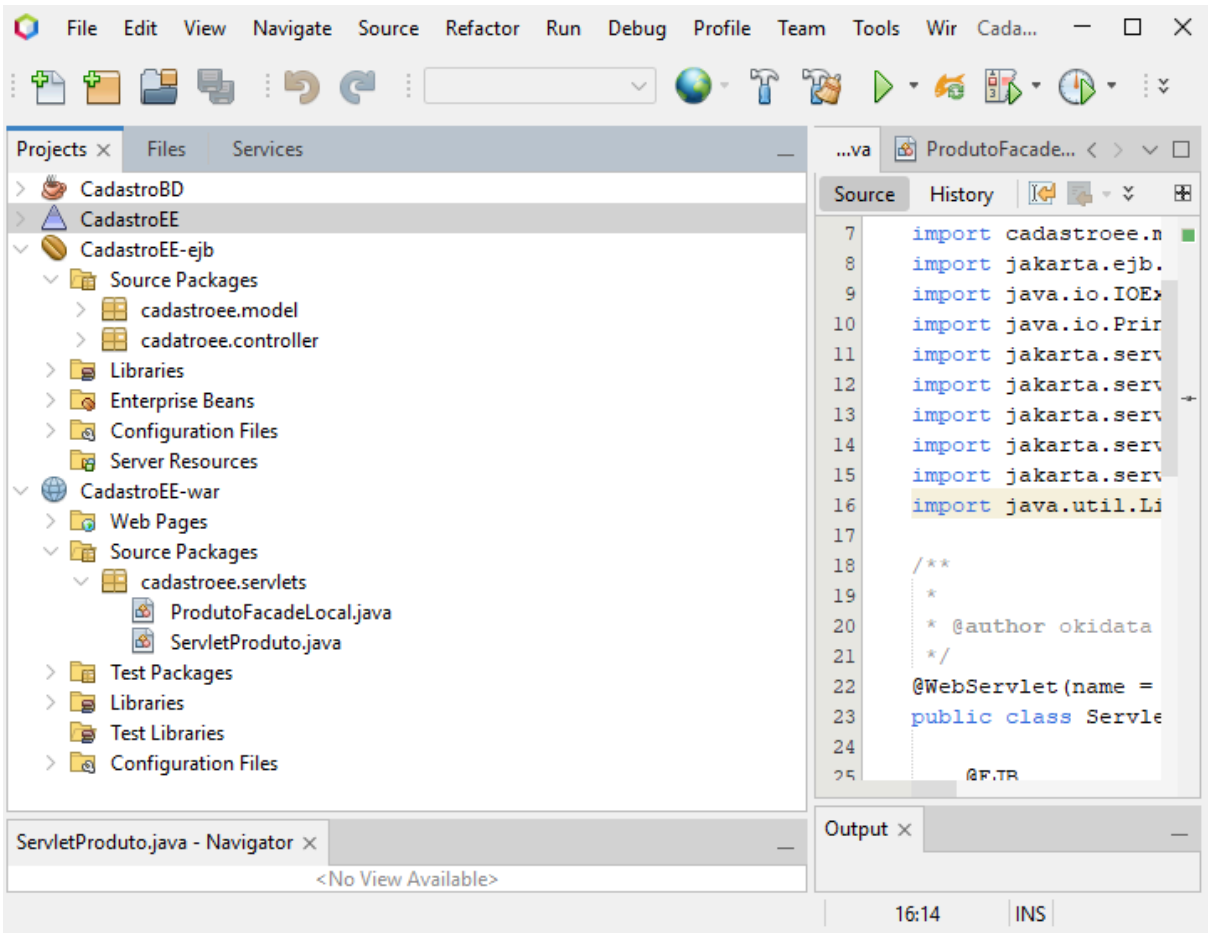
4. Resultado da execução do código:




File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Apache NetBeans IDE 22
Search (Ctrl+I)
Projects Files Services
Databases
Web Services
Servers
GlassFish Server
Applications
Resources
Maven Repositories
Cloud
Jenkins Builders
Docker
Task Repositories
JS Test Driver
Selenium Server
CadastroBD.java
Source History
/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
*/
package cadastrobd;
/**
*
* @author okidata
*/
public class CadastroBD {
GlassFish Console - Common Tasks
localhost:4848/common/index.jsf
Home About...
Help
User: admin | Domain: domain1 | Server: localhost
Eclipse GlassFish
Tree
Common Tasks
Domain
server (Admin Server)
Clusters
Standalone Instances
Nodes
Applications
Lifecycle Modules
Monitoring Data
Resources
Concurrent Resources
Connectors
JDBC
JMS Resources
GlassFish Console - Common Tasks
GlassFish News
GlassFish News
Deployment
List Deployed Applications
Deploy an Application
Administration
Change Administrator Password
Documentation
Open Source Edition Documentation Set
Quick Start Guide
Administration Guide
Application Development Guide
Application Deployment Guide
Resources
Create New JDBC Resource
Navigator
1:1
INS


Edit JDBC Connection Pool
localhost:4848/common/index.jsf
Home About...
Help
User: admin | Domain: domain1 | Server: localhost
Eclipse GlassFish
Tree
Common Tasks
Domain
server (Admin Server)
Clusters
Standalone Instances
Nodes
Applications
Lifecycle Modules
Monitoring Data
Resources
Concurrent Resources
Connectors
JDBC
JDBC Resources
JDBC Connection Pools
DerbyPool
__TimerPool
loja
JMS Resources
JNDI
JavaMail Sessions
Resource Adapter Configs
Configurations
default-config
server-config
General
Advanced
Additional Properties
Ping Succeeded
Edit JDBC Connection Pool
Save
Cancel
Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.
Load Defaults
Flush
Ping
* Indicates required field
General Settings
Pool Name:
loja
Resource Type:
javax.sql.DataSource
Must be specified if the datasource class implements more than 1 of the interface.
Datasource Classname:
com.microsoft.sqlserver.jdbc.SQLServerDataSource
Vendor-specific classname that implements the DataSource and/or XADataSource APIs
Driver Classname:
Vendor-specific classname that implements the java.sql.Driver interface.
Ping:
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes
Deployment Order:
100
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.
Description:
Pool Settings
Initial and Minimum Pool Size:
8
Connections
Minimum and initial number of connections maintained in the pool
Maximum Pool Size:
32
Connections
Maximum number of connections that can be created to satisfy client requests
Pool Resize Quantity:
2
Connections


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Files
Services
CadastroBD
CadastroEE
CadastroEE-ejb
Source Packages
cadastroee.model
cadatroee.controller
Libraries
Jakarta EE 8 API Library - jakarta.jakartaee-api-8.0.0.jar
JDK 22 (Default)
GlassFish Server
Enterprise Beans
Configuration Files
Server Resources
CadastroEE-war
NewSessionBean.java - Navigator
Members
NewSessionBean :: NewSessionBeanLocal
NewSessionBean()
Movimento.java
NewSessionBean.java
Source
History
/*
* Click nbfs://nbhost/SystemFileSystem/Temp
* Click nbfs://nbhost/SystemFileSystem/Temp
*/
package cadatroee.controller;
import jakarta.ejb.Stateless;
Output
Java DB Database Process
GlassFish Server
Thu Aug 29 15:11:49 BRT 2024 : Apache Derby Servidor de R
1:1
INS


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Wir
Projects
Files
Services
CadastroBD
CadastroEE
CadastroEE-ejb
Source Packages
cadastroee.model
cadatroee.controller
Libraries
Enterprise Beans
Configuration Files
Server Resources
CadastroEE-war
Web Pages
Source Packages
cadastroee.servlets
ProdutoFacadeLocal.java
ServletProduto.java
Test Packages
Libraries
Test Libraries
Configuration Files
ServletProduto.java - Navigator
<No View Available>
ProdutoFacade...
Source
History
import cadastroee.m
import jakarta.ejb.
import java.io.IOEx
import java.io.Prir
import jakarta.serv
import jakarta.serv
import jakarta.serv
import jakarta.serv
import jakarta.serv
import java.util.Li
/**
*
* @author okidata
*/
@WebServlet(name =
public class Servle
Output
16:14
INS

5. Análise e Conclusão:

a. *Como é organizado um projeto corporativo no NetBeans?*

   Através de uma estrutura modular hierárquica. Com o encapsulamento o projeto é dividido em funcionalidades e o código é protegido protegido.

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

   JPA (Jakart Persistence API): Usado para persistência dos dados.

   EBJ (Enterprise JavaBeans): Usado para o encapsulamento e proteção do projeto.

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

   Através do suporte integrado que oferece templates e ferramentas que facilitam o uso das tecnologias e poupam um tempo considerável já que reduzem a necessidade de linhas de código escritas manualmente.

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

   Servlets são componentes em forma de classe que respondem a requisições. O netbeans suporta sim, e inclusive oferece suporte, para Servlets através da assistentes para o construção e desenvolvimento que automatizam o processo de criação.

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

É realizada através do Java Naming and Directory Interface, ou JNDI.

1. Missão Prática - Nível 4

2. Objetivos da Missão 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.

3. Códigos do Projeto:

4. Resultado da execução do código:

5. Análise e Conclusão:

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

É um padrão arquitetural que funciona como um controlador tratando todas as solicitações para um site Web, implementado como Servlet.

b. *Quais as diferenças e semelhanças entre Servlets e JSPs?*

Diferenças: Servlets são mais focados em controle e JSPs na parte visual.

Semelhanças: Ambos trabalham juntos para dinamizar páginas web.

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?*

1. Missão Prática - Nível 4

2. Objetivos da Missão 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.

3. Códigos do Projeto:

4. Resultado da execução do código:

5. Análise e Conclusão:

a. *Como o framework Bootstrap é utilizado?*

   Bootstrap é um framework que se popularizou principalmente por sua facilidade e agilidade, poupando um tempo considerável no desenvolvimento de interfaces web responsivas. Ele oferece algumas opções de personalização em CSS que apesar de serem um pouco mais limitadas, são sólidas e bem coesas.

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

   Pois ele usa de classes CSS, apesar de não ser feito separadamente como uma estrutura CSS padrão, por utilizar de classes de estilo não altera em nada na estrutura do HTML.

c. *Qual a relação entre o Boostrap e a responsividade da página?*

   Através principalmente de um sistema de grid flexível que adapta os componentes para o tamanho da página.