Relatório de Atividade Prática Nível 3

Idelmar Fernando de Souza - 202302114431

Campus Santa Rosa - RS

Vamos integrar sistemas – 3º Semestre

Objetivo da Prática

A atividade tem como objetivo desenvolver um sistema de cadastro de produtos usando o Jakarta EE, JPA e EJB. Com integração com banco de dados (CRUD);

1º Procedimento | Camadas de Persistência e Controle

Nessa primeira parte os componentes foram gerados automoticamente conforme seguidos os passos da atividade, para não se extender segue os principais códigos:

Classe Produto:

```
package cadastroee.model;
import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.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;

@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.findByProduto", query = "SELECT p FROM Produto p WHERE p.produto = :produto"),
    @NamedQuery(name = "Produto.findByEstoque", query = "SELECT p FROM Produto p WHERE p.estoque = :estoque"),
    @NamedQuery(name = "Produto.findByPreco", query = "SELECT p FROM Produto p WHERE p.preco = :preco")
})
public class Produto implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
@Basic(optional = false)
@Column(name = "idproduto")
private Integer idproduto;
@Basic(optional = false)
@NotNull
@Size(min = 1, max = 20)
@Column(name = "produto")
private String produto;
@Basic(optional = false)
@NotNull
@Column(name = "estoque")
private int estoque;
@Basic(optional = false)
@NotNull
@Column(name = "preco")
private Float preco;
@OneToMany(mappedBy = "produto")
private Collection<Negociacao> negociacaoCollection;

public Produto() {
}
public Produto(Integer idproduto) {
    this.idproduto = idproduto;
}
public Produto(Integer idproduto, String produto, int estoque, Float preco) {
    this.idproduto = idproduto;
    this.produto = produto;
    this.estoque = estoque;
    this.preco = preco;
}
public Integer getIdproduto() {
    return idproduto;
}
public void setIdproduto(Integer idproduto) {
    this.idproduto = idproduto;
}
public String getProduto() {
    return produto;
}

public void setProduto(String produto) {
    this.produto = produto;
}
public int getEstoque() {
```

```
        return estoque;
    }
    public void setEstoque(int estoque) {
        this.estoque = estoque;
    }
    public Float getPreco() {
        return preco;
    }
    public void setPreco(Float preco) {
        this.preco = preco;
    }
    @XmlTransient
    public Collection<Negociacao> getNegociacaoCollection() {
        return negociacaoCollection;
    }
    public void setNegociacaoCollection(Collection<Negociacao> negociacaoCollection) {
        this.negociacaoCollection = negociacaoCollection;
    }
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idproduto != null ? idproduto.hashCode() : 0);
        return hash;
    }
    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Produto)) {
            return false;
        }
        Produto other = (Produto) object;
                    return !((this.idproduto == null && other.idproduto != null) || (this.idproduto != null && !
this.idproduto.equals(other.idproduto)));
    }
    @Override
    public String toString() {
        return "cadastroee.model.Produto[ idproduto=" + idproduto + " ]";
    }
}
```

A classe Produto é componente que espelha a tabela do banco de dados, possuindo alguns métodos para manipular o banco;

ProdutoFacade:

```
package cadastroee.controller;
import cadastroee.model.Produto;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
@Stateless
public class ProdutoFacade extends AbstractFacade<Produto> implements ProdutoFacadeLocal {
    @PersistenceContext(unitName = "CadastroEE-ejbPU")
```

```
    private EntityManager em;
    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public ProdutoFacade() {
        super(Produto.class);
    }
}
```

A classe atua como intermédio entre a camada de negócios e a persistencia para a classe Produto. Usa o EntityMananger para realizar operações de banco de dados.

ProdutoFacadeLocal

```
package cadastroee.controller;
import cadastroee.model.Produto;
import jakarta.ejb.Local;
import java.util.List;
@Local
public interface ProdutoFacadeLocal {
    void create(Produto produto);
    void edit(Produto produto);
    void remove(Produto produto);
    Produto find(Object id);
    List<Produto> findAll();
    List<Produto> findRange(int[] range);
    int count();
}
```

A classe adiciona um conjunto de métodos para serem utilizados pela classe Produto.

ServletProduto

```
package cadastroee.controller;
import cadastroee.model.Produto;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
@Stateless
public class ProdutoFacade extends AbstractFacade<Produto> implements ProdutoFacadeLocal {
    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;
```

```
    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public ProdutoFacade() {
        super(Produto.class);
    }
}
```

Classe para testar o sistema em desenvolimento, interage com o banco e traz os produtos cadastrados.

Parte 1
Respostas:

Como é organizado um projeto corporativo no NetBeans?

A organização é feita por pacotes, e os utilitários como css e imagens ficam no source packages ou src

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

A JPA faz a interação com o banco relacional e faz a gestão dos dados; E o EJB serve para constuir os componentes do projeto, traz em sua composição componentes prontos o que facilita e agiliza o desenvolvimento.

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

O NetBeans auxilia os desenvolvedores com ferramentas como a depuração e o assistente de codigo e interface amigável e conhecida dos usuários.

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

É o componente responsável pela interação com a web, fazem o processamento das requisitções HTTP.

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

O EJBs faz injeção da session bean marcado com o comentáio @EJB, basicamente o serlvets separa a lógica de negócios o que é uma boa prática;

2º Procedimento | Interface Cadastral com Servlet e JSPs

ServletProdutoFC

```
package cadastroee.servlets;

import cadastroee.controller.ProdutoFacadeLocal;

import cadastroee.model.Produto;

import java.io.IOException;

import java.util.List;

import jakarta.ejb.EJB;

import jakarta.servlet.RequestDispatcher;
```

```
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet(name = "ServletProdutoFC", urlPatterns = {"/ServletProdutoFC"})
public class ServletProdutoFC extends HttpServlet {
   @EJB
   private ProdutoFacadeLocal facade;
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      String acao = request.getParameter("acao");
      String destino;
      System.out.println("Ação recebida: " + acao);
      try {
         switch (acao) {
            case "listar":
               List<Produto> produtos = facade.findAll();
               System.out.println("Número de produtos listados: " + (produtos != null ? produtos.size() : "null"));
               request.setAttribute("produtos", produtos);
               destino = "ProdutoLista.jsp";
               break;
            case "formIncluir":
               destino = "ProdutoDados.jsp";
               break;
            case "formAlterar":
               String idStr = request.getParameter("idproduto");
               System.out.println("ID para alterar: " + idStr);
               if (idStr != null && !idStr.isEmpty()) {
                  Produto produtoAlterar = facade.find(Integer.valueOf(idStr));
                  System.out.println("Produto a ser alterado: " + (produtoAlterar != null ? produtoAlterar.toString() : "null"));
                  request.setAttribute("produto", produtoAlterar);
               }
               destino = "ProdutoDados.jsp";
               break;
            case "incluir":
```

```
                Produto novoProduto = new Produto();
                novoProduto.setProduto(request.getParameter("produto"));
                novoProduto.setEstoque(Integer.parseInt(request.getParameter("quantidade")));
                novoProduto.setPreco(Float.valueOf(request.getParameter("preco")));
                System.out.println("Novo produto a ser incluído: " + novoProduto);
                facade.create(novoProduto);
                request.setAttribute("produtos", facade.findAll());
                destino = "ProdutoLista.jsp";
                break;
            case "alterar":
                String idAlterar = request.getParameter("idproduto");
                System.out.println("ID para alterar: " + idAlterar);
                if (idAlterar != null && !idAlterar.isEmpty()) {
                    Produto produtoExistente = facade.find(Integer.valueOf(idAlterar));
                        System.out.println("Produto existente para alterar: " + (produtoExistente != null ? produtoExistente.toString() :
"null"));
                    if (produtoExistente != null) {
                        produtoExistente.setProduto(request.getParameter("produto"));
                        produtoExistente.setEstoque(Integer.parseInt(request.getParameter("quantidade")));
                        produtoExistente.setPreco(Float.valueOf(request.getParameter("preco")));
                        facade.edit(produtoExistente);
                    }
                }
                request.setAttribute("produtos", facade.findAll());
                destino = "ProdutoLista.jsp";
                break;
            case "excluir":
                String idExcluir = request.getParameter("idproduto");
                System.out.println("ID para excluir: " + idExcluir);
                if (idExcluir != null && !idExcluir.isEmpty()) {
                    Produto produtoExcluir = facade.find(Integer.valueOf(idExcluir));
                    System.out.println("Produto a ser excluído: " + (produtoExcluir != null ? produtoExcluir.toString() : "null"));
                    if (produtoExcluir != null) {
                        facade.remove(produtoExcluir);
                    }
                }
                request.setAttribute("produtos", facade.findAll());
                destino = "ProdutoLista.jsp";
                break;
            default:
                destino = "ProdutoLista.jsp";
                break;
        }
```

```
        } catch (Exception e) {
            System.err.println("Erro ao processar a ação: " + e.getMessage());
            e.printStackTrace();
            request.setAttribute("erro", "Erro ao processar a ação: " + e.getMessage());
            destino = "erro.jsp"; // Uma página de erro genérica que você pode criar
        }
        RequestDispatcher dispatcher = request.getRequestDispatcher(destino);
        dispatcher.forward(request, response);
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @override
    public String getServletInfo() {
        return "Servlet para gerenciar produtos";
    }
}
```

O servlet gerencia as operações CRUD e encaminha para as respostas para as páginas (JPS) correspondentes.

ProdutoDados

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
                <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>
    <title>Dados do Produto</title>
</head>
<body>
    <h1>${produto != null ? 'Alterar Produto' : 'Novo Produto'}</h1>
```

```
  <form action="ServletProdutoFC" method="post">
    <c:set var="acao" value="${produto != null ? 'alterar' : 'incluir'}"/>
        <input type="hidden" name="acao" value="${acao}"/>
        <c:if test="${acao == 'alterar'}">
      <input type="hidden" name="idproduto" value="${produto.idproduto}"/>
    </c:if>
    Produto: <input type="text" name="produto" value="${produto != null ? produto.produto : "}"/><br/>
    Quantidade: <input type="number" name="quantidade" value="${produto != null ? produto.estoque : "}"/><br/>
    Preço de Venda: <input type="text" name="preco" value="${produto != null ? produto.preco : "}"/><br/>
    <input type="submit" value="${acao == 'alterar' ? 'Alterar Produto' : 'Incluir Produto'}"/>
  </form>
</body>
```

O ProdutoDados é a pagina de interação do usuário, permitindo inserir, alterar os dados do produto.

ProdutoLista

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
                <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>
  <title>Lista de Produtos</title>
</head>
<body class="container">
  <h1>Lista de Produtos</h1>
  <a href="ServletProdutoFC?acao=formIncluir" class="btn btn-primary m-2">Novo Produto</a>
  <table class="table table-striped">
    <thead class="table-dark">
      <tr>
        <th>ID</th>
        <th>Produto</th>
        <th>Quantidade</th>
        <th>Preço</th>
        <th>Ações</th>
      </tr>
    </thead>
    <tbody>
      <c:forEach var="produto" items="${produtos}">
        <tr>
          <td>${produto.idproduto}</td>
          <td>${produto.produto}</td>
```

```
            <td>${produto.estoque}</td>
            <td>${produto.preco}</td>
            <td>
                <a href="ServletProdutoFC?acao=formAlterar&idproduto=${produto.idproduto}" class="btn btn-primary btn-sm">Alterar</a>
                <a href="ServletProdutoFC?acao=excluir&idproduto=${produto.idproduto}" class="btn btn-danger btn-sm">Excluir</a>
            </td>
          </tr>
        </c:forEach>
      </tbody>
    </table>
</body>
</html>
```

A pagina de produto lista permite consultar os dados dos produtos cadastrados.

Parte 2
Respostas:

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

O Front Controller centraliza as solicitações do usuário e distribui para o componente específico.

Quais as diferenças e semelhanças entre Servlets e JSPs?

O servlets são classes escritas em java e recebem a parte lógica e encaminham para o JSP; e os JSPs são as views do projeto.

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 response.sendRedirect envia-se uma resposta http e o browser fara a requisição para URL, e no requestDispatcher a requisição é encaminhada para ser atendida por um servlet e este devolverá outra página ao cliente, ou seja, é feita pelo lado o servidor. Os parâmetros servem para armazenar os dados recebidos do cliente para enviar ao servidor, os atributos são dados definidos pelo servidor e são usados para compartilhamento de dados entre os componentes no servidor.

Parte 3
Respostas:

Como o framework Bootstrap é utilizado?
Em java, utiliza-se o CDN que é basicamente uma forma de usar o framework adicionando o link para os arquivos do Bootstrap.

Por que o Bootstrap garante a independência estrutural do HTML?

O Bootstrap permite modificar os estilos da pagina HTML sem alterar a estrutura da original da pagina.

Qual a relação entre o Boostrap e a responsividade da página?

O Bootstrap possui ferramentas e componentes parar facilitar a criação de páginas responsivas, ou seja, se usado corretamente se adaptam automaticamente a diferentes tipos de telas.

Conclusão

O projeto de cadastro de produtos utilizando Java Ant Enterprise Edition, apresenta com legibilidade e uma estutrura de fácil manutenção. Utilizando tecnologias como EJB (Enterprise JavaBeans), JPA (Java Persistence API) e JSP (JavaServer Pages), o projeto permite lidar com a persistência de dados, lógica de negócios e interface do usuário.