



Projeto Servlet/Front-end de Exemplo – PARTE 1

Professor Vicente Paulo de Camargo

Projeto Servlet/Front-end de Exemplo

Esse projeto é uma continuação do projeto **SistemaEstoque**

O ideal é criar outro projeto web, deixando o anterior intacto

Dessa forma, crie um projeto web

Provavelmente, você já construiu o banco de dados do projeto anterior

Caso não tenha construído, segue a estrutura da tabela do banco de dados

#	Nome	Tipo de dados	Tamanho/It...	Unsign...	Permitir...	Zerofill	Padrão
1	id	INT	11	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	descricao	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	precounit	DOUBLE		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	quantidade	DOUBLE		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Esse banco de dados foi construído com o SGBD MySQL

Utilize qualquer SGBD para construir o seu banco de dados

Projeto Servlet/Front-end de Exemplo

CODIFICAÇÃO DO BACK-END

Classes e Servlets

Projeto Servlet/Front-end de Exemplo

Crie o pacote **br.edu.pucgoias.sistemaestoque.modelo**

Em seguida, crie a classe **Estoque.java**, com a seguinte estrutura:

```
1 package br.edu.pucgoias.sistemaestoque.modelo;
2
3 public class Estoque{
4
5     private int id;
6     private String descricao;
7     private double quantidade;
8     private double precounit;
9
10    private String mensagem;
11    private int controle; //1=inclusão, 2=alteração, 3=exclusão
12
13    public int getId() {
14        return id;
15    }
16    public void setId(int id) {
17        this.id = id;
18    }
19    public String getDescricao() {
20        return descricao;
21    }
22    public void setDescricao(String descricao) {
23        this.descricao = descricao;
24    }
25    public double getQuantidade() {
26        return quantidade;
27    }
28    public void setQuantidade(double quantidade) {
29        this.quantidade = quantidade;
30    }
31    public double getPrecounit() {
32        return precounit;
33    }
34    public void setPrecounit(double precounit) {
35        this.precounit = precounit;
36    }
37
38    public String getMensagem() {
39        return mensagem;
40    }
41
42    public void setMensagem(String mensagem) {
43        this.mensagem = mensagem;
44    }
45    public int getControle() {
46        return controle;
47    }
48    public void setControle(int controle) {
49        this.controle = controle;
50    }
51    @Override
52    public String toString() {
53        return "Estoque [id=" + id + ", descricao=" + descricao + ", quantidade=" + quantidade + ", precounit="
54            + precounit + "]\n";
55    }
56
57 }
```

Projeto Servlet/Front-end de Exemplo

Crie o pacote **br.edu.pucgoias.sistemaestoque.dao**

Em seguida, crie a classe **BaseDao.java** para conexão com o banco de dados, com a seguinte estrutura:

```
1 package br.edu.pucgoias.sistemaestoque.dao;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class BaseDao
8 {
9
10     public BaseDao()
11     {
12         try {
13             Class.forName("com.mysql.jdbc.Driver");
14         }
15         catch(ClassNotFoundException e) {
16
17         }
18     }
19
20     public Connection getConnection() throws SQLException{
21         String url="jdbc:mysql://localhost/sistema_estoque_web";
22         Connection conn = (Connection) DriverManager.getConnection(url, "vicente", "vicente");
23         return conn;
24     }
25
26     public static void main(String[] args) throws SQLException {
27         BaseDao bd = new BaseDao();
28         Connection conn = bd.getConnection();
29         if (conn != null)
30             System.out.println("Conectou !");
31         else
32             System.out.println("Não conectou");
33     }
34 }
35
```

Ajuste apenas as configurações para acessar o seu banco de dados



Projeto Servlet/Front-end de Exemplo

Em seguida, crie a classe **EstoqueDao.java** no pacote **br.edu.pucgoias.sistemaestoque.dao**

Essa classe possui dois métodos para salvar dados e dois métodos para excluir item do estoque

No entanto, apenas os métodos correspondentes que retornam objetos de Estoque serão explicados nesse conteúdo, pois utilizam o encapsulamento de mensagem dentro do próprio objeto Estoque, facilitando o envio e visualização de mensagens para o usuário

Os outros dois métodos foram desenvolvidos e explicados no projeto anterior

Eles se assemelham com os explicados aqui, apenas retornam um boolean para indicar se a operação foi ou não bem sucedida e, neste caso, não deixa a aplicação muito amigável para o usuário, pois retorna apenas um true ou false e o correto é retornar uma mensagem indicando se a operação foi ou não bem sucedida e também indicar qual operação foi realizada (inclusão, alteração ou exclusão)

Projeto Servlet/Front-end de Exemplo

Início do código da classe `EstoqueDao.java`

```
1 package br.edu.pucgoias.sistemaestoque.servlets;
2
3 import java.io.IOException;
4
5 @WebServlet("/servletconsultaPorId")
6 public class ServletConsultaPorId extends HttpServlet {
7     private static final long serialVersionUID = 1L;
8     private Gson gson;
9     public ServletConsultaPorId() {
10         super();
11     }
12
13     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
14
15         gson = new GsonBuilder().setPrettyPrinting().create();
16         Estoque estoque = new Estoque();
17
18         int id=0;
19         String strId= request.getParameter("id");
20
21         id = Integer.parseInt(strId);
22
23         String retorno = "ERRO";
24         boolean acao=false;
25         if ((strId == null || strId.length()==0) || strId.isEmpty() && id == 0)
26             estoque.setMensagem("Informação inválida");
27         else
28         {
29             EstoqueControle ec = new EstoqueControle();
30             estoque = ec.getEstoquePorId(id);
31
32         }
33         String retornoJsonString = this.gson.toJson(estoque);
34         PrintWriter out = response.getWriter();
35         response.setContentType("application/json");
36         response.setCharacterEncoding("UTF-8");
37         out.print(retornoJsonString);
38         out.flush();
39     }
40 }
```

Esse método é responsável por receber o código de um item do estoque (id), consultar esse item e retornar o seu objeto para a classe de controle. Mesmo não encontrando um objeto, retorna-o com alguma mensagem. O Front-end detecta se o id é diferente de zero. Se isso ocorrer, então encontrou um item do estoque

Projeto Servlet/Front-end de Exemplo

EstoqueDao.java (Cont.)

```
38
39 public Estoque criaEstoque(ResultSet rs) throws SQLException {
40     Estoque estoque = new Estoque();
41     estoque.setDescricao(rs.getString("descricao"));
42     estoque.setId(rs.getInt("id"));
43     estoque.setQuantidade(rs.getDouble("quantidade"));
44     estoque.setPrecounit(rs.getDouble("precounit"));
45
46     return estoque;
47 }
48
49 public List<Estoque> getEstoqueViaNome(String nome){
50
51     List<Estoque> lista = new ArrayList<>();
52     Estoque estoque = new Estoque();
53     PreparedStatement pstm = null;
54     Connection conn = null;
55     try {
56         conn = this.getConnection();
57         String sql="select * from estoque where lower(descricao) like ? order by descricao";
58         pstm = conn.prepareStatement(sql);
59         pstm.setString(1,"%"+nome.toLowerCase()+"%");
60         ResultSet rs = pstm.executeQuery();
61         while (rs.next())
62         {
63             estoque = criaEstoque(rs);
64             lista.add(estoque);
65         }
66     }
67     catch(SQLException e)
68     {
69         lista=null;
70     }
71     return lista;
72 }
73 }
```

O método `criaEstoque` permite criar uma instância de um objeto, popular esse objeto com os dados de um registro e retornar esse objeto para o local que chamou o método.

O método `getEstoqueViaNome` recebe um nome ou parte de um nome, faz a consulta e retorna um `ArrayList` dos objetos dos itens encontrados. Retorna um `null` se não encontrou itens. Esse método não é utilizado na página `index.html`

Projeto Servlet/Front-end de Exemplo

EstoqueDao.java (Cont.)

O método `salvarEstoqueMsg` recebe um objeto `Estoque`. Caso o `id` desse objeto for `= 0`, então é uma inclusão. Caso contrário será uma alteração. As linhas 155 e 156 obtêm o `id` do novo item conforme designado na linha 129. Para inclusão ou edição, o objeto `Estoque` é retornado com uma mensagem que será utilizada pelo controle que a enviará para o Servlet retornar via Json para a página apresentar o conteúdo da mensagem. O método `getGeneratedId` obtém o `id` gerado pelo `PreparedStatement` e o retorna para o local que chamou esse método.

```
118 public Estoque salvarEstoqueMsg(Estoque estoque) {
119     String sql="";
120     PreparedStatement pstmt = null;
121     Connection conn = null;
122     try {
123         conn = this.getConnection();
124         if (estoque.getId() == 0)
125         {
126             //6 interessante fazer uma consulta para ver se já existe item com essa descrição
127             sql="insert into estoque (descricao, precounit, quantidade) ";
128             sql+=" values (?, ?, ?)";
129             pstmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
130             estoque.setControl(1);
131         }
132         else
133         {
134             sql="update estoque set descricao=?, precounit=?, quantidade=? where id = ?";
135             pstmt = conn.prepareStatement(sql);
136             estoque.setControl(2);
137         }
138         pstmt.setString(1, estoque.getDescricao());
139         pstmt.setDouble(2, estoque.getPrecounit());
140         pstmt.setDouble(3, estoque.getQuantidade());
141         if (estoque.getId() != 0)
142         {
143             pstmt.setInt(4, estoque.getId());
144         }
145         int idAux = pstmt.executeUpdate();
146         if (idAux==0)
147         {
148             if (estoque.getControl()==1) estoque.setMensagem("Inclusão não foi realizada");
149             if (estoque.getControl()==2) estoque.setMensagem("Alteração não foi realizada");
150         }
151         else
152         {
153             if (estoque.getId()==0)
154             {
155                 int idInserir = getGeneratedId(pstmt);
156                 estoque.setId(idInserir);
157             }
158             if (estoque.getControl()==1) estoque.setMensagem("Inclusão realizada com sucesso!!");
159             if (estoque.getControl()==2) estoque.setMensagem("Alteração realizada com sucesso!!");
160         }
161     }
162     catch (SQLException e)
163     {
164         estoque.setMensagem("Erro de atualização");
165     }
166     return estoque;
167 }
```

```
169 public int getGeneratedId(PreparedStatement stm) throws SQLException {
170     ResultSet rs = stm.getGeneratedKeys();
171     if (rs.next())
172     {
173         int id = rs.getInt(1);
174         return id;
175     }
176     return 0;
177 }
```

Projeto Servlet/Front-end de Exemplo

EstoqueDao.java (Cont.)

```
198 public Estoque excluirMsg(int id) {
199     Estoque estoque=new Estoque();
200
201     PreparedStatement pstmt= null;
202     Connection conn = null;
203     try {
204         conn = this.getConnection();
205         String sql="delete from estoque where id = ?";
206         pstmt = conn.prepareStatement(sql);
207         pstmt.setInt(1,id);
208         int conta =pstmt.executeUpdate();
209         if (conta>0)
210         {
211             estoque.setControle(3);
212             estoque.setMensagem("Exclusão efetuada com sucesso");
213         }
214     }
215     catch(SQLException e)
216     {
217         estoque.setMensagem("Erro de exclusão");
218     }
219     return estoque;
220 }
```

O método `excluirMsg` recebe o código de um item do estoque (`id`) e efetua a sua exclusão do banco de dados, retornando um objeto `Estoque` com a mensagem que será apresentada pelo front-end

Projeto Servlet/Front-end de Exemplo

EstoqueDao.java (Cont.)

```
241 public List<Estoque> getTodos(){
242
243     List<Estoque> lista = new ArrayList<>();
244     Estoque estoque = new Estoque();
245     PreparedStatement pstm = null;
246     ResultSet rs;
247     Connection conn = null;
248     try {
249         conn = this.getConnection();
250         String sql="select * from estoque order by descricao";
251         pstm = conn.prepareStatement(sql);
252         rs = pstm.executeQuery();
253         while (rs.next())
254         {
255             estoque = criaEstoque(rs);
256             lista.add(estoque);
257         }
258     }
259     catch(SQLException e)
260     {
261         return lista;
262     }
263     return lista;
264 }
265
266 }
```

O método `getTodos` retorna uma lista com todos os itens cadastrados no banco de dados. Note que o `select` faz com que a busca se apresente por ordem alfabética da descrição do item.

Se a lista retornada estiver vazia, o front-end terá que tratar essa questão, apresentando alguma mensagem para o usuário

Projeto Servlet/Front-end de Exemplo

Crie o pacote **br.edu.pucgoias.sistemaestoque.controle**

Em seguida, crie a classe **EstoqueControle.java**, com a seguinte estrutura:

```
1 package br.edu.pucgoias.sistemaestoque.controle;
2
3 import java.util.List;
4
5 import br.edu.pucgoias.sistemaestoque.dao.EstoqueDao;
6 import br.edu.pucgoias.sistemaestoque.modelo.Estoque;
7
8 public class EstoqueControle {
9
10     private EstoqueDao ed = new EstoqueDao();
11
12     public List<Estoque> getEstoque(){
13         List<Estoque> estoques = ed.getTodos();
14         return estoques;
15     }
16
17     public Estoque getEstoquePorId(int id) {
18         return ed.getEstoqueViaId(id);
19     }
20
21     public boolean excluir(int id) {
22         return ed.excluir(id);
23     }
24
25     public Estoque excluirMsg(int id) {
26         return ed.excluirMsg(id);
27     }
28
29     public boolean salvar(Estoque estoque) {
30         return ed.salvarEstoque(estoque);
31     }
32
33     public Estoque salvarComMsg(Estoque estoque) {
34         return ed.salvarEstoqueMsg(estoque);
35     }
36
37     public List<Estoque> buscaEstoquePorNome(String nome){
38         return ed.getEstoqueVialNome(nome);
39     }
40 }
```

Os métodos excluir e salvar não estão sendo utilizados pela aplicação, pois não utilizam a mensagem dentro do objeto Estoque. Os métodos dessa classe são auto explicativos. Ressalta-se que essa classe é instanciada nos respectivos Servlets que recebem os retornos. Estes são enviados para as chamadas via Ajax na página index.html.



Projeto Servlet/Front-end de Exemplo

Crie o pacote **br.edu.pucgoias.sistemaestoque.servlets**

Esse pacote armazenará todos os Servlets da aplicação.

Em seguida, pesquise na Internet o arquivo **gson-2.3.1.jar**, que é uma biblioteca da Google que permite manipular as informações para que possam ser retornadas no formato JSON

Copie esse arquivo para a pasta **WebContent/WEB-INF/lib** do seu projeto

Projeto Servlet/Front-end de Exemplo

Em seguida, crie a classe (Servlet) **ServletAll.java**, com a seguinte estrutura:

```
1 package br.edu.pucgoias.sistemaestoque.servlets;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 import javax.servlet.ServletException;
9 import javax.servlet.annotation.WebServlet;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 import com.google.gson.Gson;
15 import com.google.gson.GsonBuilder;
16
17 import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
18 import br.edu.pucgoias.sistemaestoque.modelo.Estoque;
19
20 @WebServlet("/servletall")
21 public class ServletAll extends HttpServlet {
22     private static final long serialVersionUID = 1L;
23     private Gson gson;
24     public ServletAll() {
25         super();
26     }
27
28     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
29
30         gson = new GsonBuilder().setPrettyPrinting().create();
31         Estoque estoque;
32         List<Estoque> lista = new ArrayList<>();
33
34         String retorno = "ERRO";
35         boolean acao = false;
36         EstoqueControle ec = new EstoqueControle();
37         lista = ec.getEstoque();
38
39         String retornoJsonString = this.gson.toJson(lista);
40         PrintWriter out = response.getWriter();
41         response.setContentType("application/json");
42         response.setCharacterEncoding("UTF-8");
43         out.print(retornoJsonString);
44         out.flush();
45     }
46
47     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
48         doGet(request, response);
49     }
50
51 }
```

O objeto **gson** da classe **Gson** é declarado na linha 23. Em seguida, esse objeto é criado na linha 30. Essa forma de criação permite apresentar os objetos Json mais estruturados. Note a chamada ao método **getEstoque** da classe **EstoqueControle.java**, que retorna uma lista de objetos do estoque, a qual é convertida em um array de objetos na linha 39.

Projeto Servlet/Front-end de Exemplo

Crie a classe (Servlet) **ServletConsultaPorID.java**, com a seguinte estrutura:

```
1 package br.edu.pucgoias.sistemaestoque.servlets;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 import com.google.gson.Gson;
13 import com.google.gson.GsonBuilder;
14
15 import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
16 import br.edu.pucgoias.sistemaestoque.modelo.Estoque;
17
18 @WebServlet("/servletconsultaporid")
19 public class ServletConsultaPorID extends HttpServlet {
20     private static final long serialVersionUID = 1L;
21     private Gson gson;
22     public ServletConsultaPorID() {
23         super();
24     }
25
26     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
27
28         gson = new GsonBuilder().setPrettyPrinting().create();
29         Estoque estoque = new Estoque();
30
31         int id=0;
32         String strId= request.getParameter("id");
33
34         id = Integer.parseInt(strId);
35
36         String retorno ="ERRO";
37         boolean acao=false;
38         if ((strId == null || strId.length()==0) || strId.isEmpty() && id == 0)
39             estoque.setMensagem("Informação inválida");
40         else
41         {
42             EstoqueControle ec = new EstoqueControle();
43             estoque = ec.getEstoquePorId(id);
44
45         }
46         String retornoJsonString = this.gson.toJson(estoque);
47         PrintWriter out = response.getWriter();
48         response.setContentType("application/json");
49         response.setCharacterEncoding("UTF-8");
50         out.print(retornoJsonString);
51         out.flush();
52     }
53
54     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
55         doGet(request, response);
56     }
57
58 }
```

Esse Servlet retorna um objeto instanciado para armazenar o item pesquisado para que seus dados possam ser manipulados no front-end ou retorna uma mensagem caso o item não seja encontrado no banco de dados.

Projeto Servlet/Front-end de Exemplo

Crie a classe (Servlet) **ServletEditarPorID.java**, com a seguinte estrutura:

```
1 package br.edu.pucgoias.sistemaestoque.servlets;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 import com.google.gson.Gson;
13 import com.google.gson.GsonBuilder;
14
15 import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
16 import br.edu.pucgoias.sistemaestoque.modelo.Estoque;
17
18 @WebServlet("/servleteditarporid")
19 public class ServletEditarPorID extends HttpServlet {
20     private static final long serialVersionUID = 1L;
21     private Gson gson;
22
23     public ServletEditarPorID() {
24         super();
25     }
26
27     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
28
29         gson = new GsonBuilder().setPrettyPrinting().create();
30         Estoque estoque = null;
31         String descricao = request.getParameter("descricao");
32         double precounit = 0;
33         double quantidade = 0;
34         int id = 0;
35         String strPu = request.getParameter("precounit");
36         String strQt = request.getParameter("quantidade");
37         String strId = request.getParameter("id");
38
39         precounit = Double.parseDouble(strPu);
40         quantidade = Double.parseDouble(strQt);
41         id = Integer.parseInt(strId);
42     }
```

Esse Servlet atualiza os campos de um item de estoque que foram editados no front end. Em seguida, retorna o objeto com os dados atualizados, no formato Json com mensagem específica de erro ou não.

```
43     if ((descricao == null || descricao.length() == 0) && id == 0)
44     {
45         estoque = new Estoque();
46         estoque.setId(0);
47         estoque.setMensagem("Descrição inválida");
48     }
49     else
50     {
51         estoque = new Estoque();
52         estoque.setDescricao(descricao);
53         estoque.setPrecounit(precounit);
54         estoque.setQuantidade(quantidade);
55         estoque.setId(id);
56         EstoqueControle ec = new EstoqueControle();
57         estoque = ec.salvarComMsg(estoque);
58     }
59
60     String retornoJsonString = this.gson.toJson(estoque);
61     PrintWriter out = response.getWriter();
62     response.setContentType("application/json");
63     response.setCharacterEncoding("UTF-8");
64     out.print(retornoJsonString);
65     out.flush();
66 }
67
68 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
69     doGet(request, response);
70 }
71 }
```

Projeto Servlet/Front-end de Exemplo

Crie a classe (Servlet) **ServletExcluirPorID.java**, com a seguinte estrutura:

```
1 package br.edu.pucgoias.sistemaestoque.servlets;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 import com.google.gson.Gson;
13 import com.google.gson.GsonBuilder;
14
15 import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
16 import br.edu.pucgoias.sistemaestoque.modelo.Estoque;
17
18 @WebServlet("/servletExcluirPorID")
19 public class ServletExcluirPorID extends HttpServlet {
20     private static final long serialVersionUID = 1L;
21     private Gson gson;
22     public ServletExcluirPorID() {
23         super();
24     }
25
26     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
27
28         gson = new GsonBuilder().setPrettyPrinting().create();
29         Estoque estoque = new Estoque();
30
31         int id=0;
32         String strId= request.getParameter("id");
33
34         id = Integer.parseInt(strId);
35         boolean acao=false;
36         String retorno = "ERRO";
37         if ((strId == null || strId.length()==0) || strId.isEmpty() && id == 0)
38         {
39             estoque.setMensagem("Informação Inválida");
40         }
41         else
42         {
43             EstoqueControle ec = new EstoqueControle();
44             estoque = ec.excluirMsg(id);
45         }
46         String retornoJsonString = this.gson.toJson(estoque);
47         PrintWriter out = response.getWriter();
48         response.setContentType("application/json");
49         response.setCharacterEncoding("UTF-8");
50         out.print(retornoJsonString);
51         out.flush();
52     }
53
54     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
55         doGet(request, response);
56     }
57 }
58 }
```

Esse Servlet recebe o código de um item do estoque e efetua sua exclusão, retornando um objeto, no formato Json, com uma mensagem para o front-end.

Projeto Servlet/Front-end de Exemplo

Crie a classe (Servlet) **ServletSalvarDados.java**, com a seguinte estrutura:

```
1 package br.edu.pucgoias.sistemaestoque.servlets;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 import javax.servlet.ServletException;
9 import javax.servlet.annotation.WebServlet;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 import com.google.gson.Gson;
15 import com.google.gson.GsonBuilder;
16
17 import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
18 import br.edu.pucgoias.sistemaestoque.modelo.Estoque;
19
20 @WebServlet("/servletsalvados")
21 public class ServletSalvarDados extends HttpServlet {
22     private static final long serialVersionUID = 1L;
23     private Gson gson;
24
25     public ServletSalvarDados() {
26         super();
27     }
28
29     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
30
31         gson = new GsonBuilder().setPrettyPrinting().create();
32         String descricao=request.getParameter("descricao");
33         double precounit=0;
34         double quantidade=0;
35         int id=0;
36         String strPu = request.getParameter("precounit");
37         String strQt = request.getParameter("quantidade");
38         String strId= request.getParameter("id");
39
40         if (strPu ==null || strPu.length() ==0 || strPu.isEmpty())
41             precounit=0;
42         else
43             precounit = Double.parseDouble(strPu);
44
45         if (strQt == null || strQt.length() ==0 || strQt.isEmpty())
46             quantidade = 0;
47         else
48             quantidade = Double.parseDouble(strQt);
49
50         if (strId ==null || strId.length() ==0 || strId.isEmpty())
51             id=0;
52         else
53             id = Integer.parseInt(strId);
54
55         Estoque estoque=new Estoque();
56         List<Estoque> lista = new ArrayList<>();
57         booleanacao=false;
58         if (descricao == null || descricao.length()==0 || descricao.isEmpty())
59         {
60             estoque.setMensagem("Descrição inválida");
61         }
62         else
63         {
64             estoque.setDescricao(descricao);
65             estoque.setPrecounit(precounit);
66             estoque.setQuantidade(quantidade);
67             estoque.setId(id);
68             EstoqueControle ec = new EstoqueControle();
69             estoque = ec.salvarComLog(estoque);
70         }
71         String retornoJsonString ="";
72         retornoJsonString=this.gson.toJson(estoque);
73         PrintWriter out = response.getWriter();
74         response.setContentType("application/json");
75         response.setCharacterEncoding("UTF-8");
76         out.print(retornoJsonString);
77         out.flush();
78     }
79
80     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
81         doGet(request, response);
82     }
83 }
84 }
```

Esse Servlet recebe os dados para serem atualizados no banco de dados, podendo ser inclusão ou edição. Retorna um objeto do item do estoque com uma mensagem para ser visualizada no front-end.

Projeto Servlet/Front-end de Exemplo

CODIFICAÇÃO DO FRONT-END



Projeto Servlet/Front-end de Exemplo

O front-end foi desenvolvido com HTML, CSS, Bootstrap, JavaScript e jQuery

Apresenta apenas uma página (index.html) com código JavaScript dentro da mesma

Ao ser carregada, essa página apresenta uma tabela HTML com todos os itens do estoque, em ordem alfabética de descrição

O usuário poderá efetuar: um novo cadastro (botão Novo), editar (no botão Edite da tabela HTML) ou excluir (botão Exclui da tabela HTML)

Ao tentar excluir, será apresentada uma modal perguntando se o usuário deseja realmente excluir o item selecionado.

Após uma atualização (inclusão, exclusão ou alteração) a tabela HTML é atualizada

Projeto Servlet/Front-end de Exemplo

A parte HTML da página de index.html é a seguinte:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5
6 <title>Cadastro</title>
7 <link href="css/bootstrap.min.css" rel="stylesheet">
8 <link href="css/index.css" rel="stylesheet">
9 <style>
10 body {
11     background: white;
12 }
13
14 .alinhar {
15     text-align: center;
16 }
17 </style>
18 </head>
19 <body>
20 <!-- div para apresentar mensagens na parte superior da página -->
21 <div id="mensagem" class="alinhar"></div>
22 <div id="divTabela" class="alinhar"></div>
23
24 <div id="divForm" class="alinhar"></div>
25
26
27 <div class="modal fade" id="modalMensagem">
28 <div class="modal-dialog">
29 <div class="modal-content">
30 <div class="modal-header">
31 <h4 class="modal-title">Aviso</h4>
32 <button type="button" class="close" data-dismiss="modal">
33 <span>x</span>
34 </button>
35 </div>
36 <div class="modal-body">
37 <div id="msgModal"></div>
38 </div>
39 <div class="modal-footer">
40 <button type="button" class="btn btn-danger" data-dismiss="modal">Fechar</button>
41 </div>
42 </div>
43 </div>
44 </div>
45
46 <div class="modal fade" id="modalConfirma">
47 <div class="modal-dialog">
48 <div class="modal-content">
49 <div class="modal-header">
50 <h4 class="modal-title">Confirmação</h4>
51 <button type="button" class="close" data-dismiss="modal">
52 <span>x</span>
53 </button>
54 </div>
55 <div class="modal-body">Confirma exclusão</div>
56 <div class="modal-footer">
57 <button type="button" class="btn btn-danger" id="btn-fechar"
58 data-dismiss="modal">Fechar</button>
59 <button type="button" class="btn btn-success" id="btn-confirmar"
60 data-dismiss="modal">Confirmar</button>
61 </div>
62 </div>
63 </div>
64 </div>
65
66
67
68
69 <script src="jquery-3.5.1.min.js"></script>
70 <script src="js/bootstrap.min.js"></script>
```

A div mensagem é responsável de apresentar mensagem quando o Ajax é acionado (linha 21). A divTabela é alimentada com a table HTML, gerada dinamicamente via JavaScript(linha 22).A divForm é populada com o formulário para edição de dados(linha 24). A primeira modal (linhas 22 a 44) é responsável para apresentar mensagens e a segunda modal (linha 46 a 64) é utilizada para se confirmar ou não a exclusão de um item do estoque

Projeto Servlet/Front-end de Exemplo

Index.html parte JavaScript

```
72@
73  <script>
74    function excluir(msg)
75    {
76      $.excluir(msg);
77    }
78
79    function editar(msg)
80    {
81      $.mostraForm();
82      $.populaForm(msg);
83    }
84
85    function novo()
86    {
87      $.mostraForm();
88    }
89
90    function salvar()
91    {
92      $.salvar();
93    }
94
95    function preparaForm()
96    {
97      return(
98        '<br><br>'+
99        '<b>CADASTRO DE ESTOQUE</b><br>'+
100        '<div class="form-group">'+
101          '<strong>Código</strong><input class="form-control" style="text-align:center" id="idForm" readonly>'+
102          '</div>'+
103          '<div class="form-group">'+
104            '<strong>Descrição</strong><input class="form-control" style="text-align:center" id="descricaoForm">'+
105            '</div>'+
106            '<div class="form-group">'+
107              '<strong>Quantidade</strong><input class="form-control" style="text-align:center" id="quantForm">'+
108              '</div>'+
109              '<div class="form-group">'+
110                '<strong>P.Unit.</strong><input class="form-control" style="text-align:center" id="pUnitForm">'+
111                '</div>'+
112                '<button class="btn btn-success" onclick="salvar()">SALVAR</button>'+
113                '<button class="btn btn-danger" onclick="$.limpar()">LIMPAR</button>'+
114              );
115    }
116  </script>
```

A função excluir chama \$.excluir(msg), onde msg é o código do item a excluir.

A função editar(msg), visualiza o formulário (via função \$.mostraForm(msg) e preenche o formulário com os dados via função \$.mostraForm(), onde msg é o objeto pesquisado.

A função preparaForm retorna uma string contendo a estrutura do formulário a ser visualizado

Projeto Servlet/Front-end de Exemplo

Index.html parte JavaScript(continuação)

```
116 function mostraCabTabela ()
117 {
118     return(
119         '<table class="table table-striped-primary table-hover" >' +
120         '<thead>' +
121         '<tr>' +
122         '<th style="text-align: center" scope="col">CÓDIGO</th>' +
123         '<th style="text-align: center" scope="col">DESCRIÇÃO</th>' +
124         '<th style="text-align: center" scope="col">P.UNIT.</th>' +
125         '<th style="text-align: center" scope="col">QUANTIDADE</th>' +
126         '<th style="text-align: center" scope="col">EDITAR</th>' +
127         '<th style="text-align: center" scope="col">EXCLUIR</th>' +
128         '</tr>' +
129         '</thead>'
130     );
131 }
132
133 function mostraLinhaTabela(data)
134 {
135     return(
136         '<tr><td style="text-align: center">' + data.id + '</td>' +
137         '<td style="text-align: center">' + data.descricao + '</td>' +
138         '<td style="text-align: center">' + data.precounit + '</td>' +
139         '<td style="text-align: center">' + data.quantidade + '</td>' +
140         '<td style="text-align: center"><button class="btn btn-success" id="btnEditar" onclick="editar(' + data.id + ')">Edita</button></td>' +
141         '<td style="text-align: center"><button class="btn btn-danger" id="btnExcluir" onclick="$.$excluir(' + data.id + ')">Exclui</button></td></tr>'
142     );
143 }
```

A função `mostraCabTela` retorna uma string com o cabeçalho da tabela HTML. A função `mostraLinhaTabela(data)` recebe os dados de um item, constrói a string da linha com esse item e retorna a string com a estrutura da linha para ser visualizada na página

Projeto Servlet/Front-end de Exemplo

Index.html parte JavaScript(continuação)

```
145 function carregaTabela ()
146 {
147     $.ajax({
148         url: "http://localhost:8282/SistemaEstoque/servletall",
149         type: 'get',
150         data: {},
151         beforeSend: function() { $("#mensagem").html("Analisando dados..."); },
152         success: function(msg)
153         {
154
155             let table = mostraCabTabela() + '<tbody>';
156
157             for(let i=0;i < msg.length; i++) { table += mostraLinhaTabela(msg[i]); }
158             table += '</tbody></table>';
159             table += '</br><button class="btn btn-primary" onclick="novo()">NOVO</button>';
160             $('#divTabela').html(table);
161             $('#mensagem').html('');
162             $('#divForm').html('');
163             if (msg.length == 0)
164                 $.mostraMensagem("Sem itens no estoque");
165         },
166         error: function(msg)
167         {
168             $("#mensagem").html(" ");
169             $.mostraMensagem("erro");
170         }
171     });
172 }
173 //fim carregaTabela
```

A função `carregaTabela` executa o ajax para acessar o servlet que a lista de itens do estoque (linha 148). Concatena o cabeçalho da tabela na variável **table** pela função `mostraCabTabela` (linha 155) e concatena cada uma das linhas (via função `mostraLinhaTabela`)(linha 157). Complementa a concatenação com as linhas 158 e 159. Mostra uma mensagem de erro caso a lista não possuir itens (linha 164).

Projeto Servlet/Front-end de Exemplo

Index.html parte JavaScript(continuação)

```
174 $(document).ready(function()
175 {
176     carregaTabela();
177
178     $.excluir = function (codigo)
179     {
180         $("#modalConfirma").modal();
181         $("#btn-confirmar").click(function()
182         {
183             //$("#modalConfirma").modal('none');
184             $.ajax(
185                 {
186                     url : "http://localhost:8282/SistemaEstoque/servletexcluirporid",
187                     type : 'get',
188                     data :{ id : codigo },
189                     beforeSend : function()
190                     {
191                         $("#mensagem").text("Analisando a solicitação...");
192                     },
193                     success: function(msg)
194                     {
195                         $("#divForm").html("");
196                         $("#mensagem").html(" ");
197                         $.mostraMensagem("Exclusão efetuada com sucesso");
198                         carregaTabela();
199                     },
200                     error: function(msg)
201                     {
202                         $("#mensagem").html(" ");
203                         $.mostraMensagem(msg.mensagem);
204                     }
205                 }
206             );
207         });
208     });
209 };//fim $.excluir
```

A linha 174 inicia o bloco de código do jQuery. A função \$.excluir(codigo) permite excluir o item com o código informador. As linhas 180 e 181 mostra uma modal que pergunta ao usuário se deseja excluir o item. Se o botão btn-confirmar for selecionado (linha 181) o ajax (linhas 184 a 205) é executado. Se ocorrer sucesso, a tabela é carregada novamente sem o item excluído (linha 198), apresentando uma mensagem de sucesso da operação (linha 197).

Projeto Servlet/Front-end de Exemplo

Index.html parte JavaScript(continuação)

```
210
211 $.editar = function (codigo)
212 {
213     $.ajax(
214     {
215         url : "http://localhost:8282/SistemaEstoque/servleteditarporid",
216         type : 'get',
217         data : { id : codigo },
218         beforeSend : function()
219         {
220             $("#mensagem").html("Analisando a solicitação...");
221         },
222         success: function(msg)
223         {
224             $("#mensagem").html(" ");
225             $("#msgModal").text(msg);
226             $('#modalMensagem').modal('show');
227             carregaTabela();
228         },
229         error: function(msg)
230         {
231             $("#mensagem").html(" ");
232             $("#msgModal").text(msg);
233             $('#modalMensagem').modal('show');
234         }
235     });
236 }; //fim $.editar
237
```

A função \$.editar recebe o código do item selecionado (parâmetro codigo) e envia-o para o servlet servleteditarporid (linha 215), efetuando a atualização dos dados do item. Em seguida, chama a função carregaTabela para atualizar a tabela HTML na página.

Projeto Servlet/Front-end de Exemplo

Index.html parte JavaScript(continuação)

```
238     $.mostraForm = function ()
239     {
240         $("#divForm").html(preparaForm());
241         $("#mensagem").html("");
242     };
243
244     $.salvar = function()
245     {
246         let codigo = $("#idForm").val();
247         let descricao = $("#descricaoForm").val();
248         descricao=descricao.toUpperCase();
249         let quantidade = $("#quantForm").val();
250         let preco = $("#pUnitForm").val();
251
252         if (codigo === "")
253         {
254             codigo=0;
255         }
256         if (descricao.length ==0 || descricao == "")
257         {
258
259             $.mostraMensagem("descrição deve ser informada");
260         }
261         else
262         {
263             $.enviarDadosCadastro(codigo,descricao,quantidade,preco);
264         }
265     }
266 } //fim $salvar
```

A função \$.mostraForm chama o método preparaForm para visualizar o formulário para edição ou inclusão de dados. A função \$.salvar permite obter os dados que foram editados no formulário e enviá-los para a função \$.enviarDadosCadastro que se encarregará de enviar os dados para serem atualizados no servidor.

Projeto Servlet/Front-end de Exemplo

Index.html parte JavaScript(continuação)

```
267     $.enviarDadosCadastro = function(codigo,descricao,quantidade,preco)
268     {
269         $.ajax(
270         {
271             url : "http://localhost:8282/SistemaEstoque/servletsalvados",
272             type : 'get',
273             data : {
274                 id: codigo,
275                 descricao: descricao,
276                 quantidade: quantidade,
277                 precounit : preco
278             },
279             beforeSend : function()
280             {
281                 $("#mensagem").html("Enviando / salvando dados...");
282             },
283             success: function(msg)
284             {
285
286                 $("#mensagem").html(" ");
287                 $.mostraMensagem(msg.mensagem);
288                 carregaTabela();
289             },
290             error: function(msg)
291             {
292                 $("#mensagem").html(" ");
293                 $.mostraMensagem(msg.mensagem);
294             }
295         }); //fim ajax
296     } //fim enviarDadosCadastro
297
```

A função `$.enviarDadosCadastro` recebe os dados para atualização, enviando-os para `servletsalvados`. Em seguida, em caso de sucesso, carrega a tabela novamente (linha 288) e mostra uma modal com mensagem de sucesso (linha 287).

Projeto Servlet/Front-end de Exemplo

Index.html parte JavaScript(continuação)

```
298     $.populaForm = function (codigo)
299     {
300         $.ajax(
301         {
302             url : "http://localhost:8282/SistemaEstoque/servletconsultaporid",
303             type : 'get',
304             data : {
305                 id: codigo,
306             },
307             beforeSend : function()
308             {
309                 $("#mensagem").html("Consultando dados...");
310             },
311             success: function(msg)
312             {
313                 $("#mensagem").html(" ");
314                 if (msg.id !=0)
315                 {
316                     $("#idForm").val(msg.id);
317                     $("#descricaoForm").val(msg.descricao);
318                     $("#quantForm").val(msg.quantidade);
319                     $("#pUnitForm").val(msg.precounit);
320                 }
321             },
322             error: function(msg)
323             {
324                 $("#mensagem").html(" ");
325                 $.mostraMensagem(msg[0].mensagem);
326             }
327         })
328     }; //fim $.populaForm
329
```

A função \$.populaForm recebe o código do item e envia-o para servletconsultaporid (linha 302). Recebe os dados e alimenta-os no formulário (linhas 316 a 319) para que o usuário possa efetuar as devidas alterações.

Projeto Servlet/Front-end de Exemplo

Index.html parte JavaScript(continuação)

```
330     $.limpar = function()
331     {
332         $("#divForm").html("");
333     }
334
335     $.mostraMensagem = function(msg)
336     {
337         $("#msgModal").text(msg);
338         $("#modalMensagem").modal();
339     }
340
341 }); //fim jquery
342
```

A função \$.limpar remove o conteúdo da div divForm. A função \$.mostraMensagem recebe o parâmetro msg(uma string de mensagem) e o visualiza na modal modalMensagem, dentro do componente msgModal (linha 337)

Projeto Servlet/Front-end de Exemplo

FIM