Fernanda Gabrielly de Padua Martins
Turma: 1A-SESI-TI
Data: 16/10/2024

Sprint 2 - CRUD

Introdução

Este projeto implementa um sistema CRUD (Create, Read, Update, Delete) para gerenciar dados de clientes usando uma aplicação web. O sistema é dividido em duas partes principais: a interface do usuário (frontend) e a API (backend). A comunicação entre eles é realizada via requisições HTTP.

Tecnologias Utilizadas

- **HTML: Para estruturar a interface do usuário.**
- **CSS: Para estilização da interface.**
- **JavaScript: Para manipulação do DOM e requisições assíncronas (AJAX).**
- **PHP: Para o desenvolvimento da API que lida com as operações no banco de dados.**
- **MySQL: Para o armazenamento de dados.**

Estrutura do Código

1. Interface HTML

A interface HTML apresenta campos de entrada e botões que permitem ao usuário executar operações CRUD. Os campos disponíveis incluem:

- **Id: Um campo somente leitura que exibe o ID do cliente selecionado.**
- **Nome: Campo para o nome do cliente.**
- **Categoria: Campo para a categoria do cliente.**
- **Preço de Venda: Campo para o preço de venda do produto.**
- **Preço de Custo: Campo para o preço de custo do produto.**
- **Descrição: Campo para uma breve descrição do produto.**
- **Lucro: Campo para o valor do lucro.**
- **Estoque: Campo para a quantidade em estoque.**

```html
<h1>C.R.U.D. padrão API</h1>
**Id <input type="text" id="id" name="id" value="0" size="3" readonly />

**
**Nome <input type="text" id="nome" name="nome" value="" size="35" />

**
**Categoria <input type="text" id="categoria" name="categoria" value="" size="20" />

**
**Preco de Venda <input type="text" id="preco" name="preco" value="" size="20" />

**
**Preco de Custo <input type="text" id="custo" name="custo" value="" size="20" />

**

Descrição <input type="text" id="descricao" name="descricao" value="" size="40" />

Lucro <input type="text" id="lucro" name="lucro" value="" size="20" />

Estoque <input type="text" id="estoque" name="estoque" value="" size="20" />


```

2. Funções CRUD em JavaScript

**GET (Consultar/Listar)**

A função `get()` realiza uma requisição GET para buscar todos os clientes cadastrados. Em caso de sucesso, os dados são exibidos em uma tabela. Cada linha da tabela é clicável e preenche os campos de entrada para facilitar a edição.

```javascript
async function get() {
    try {
        const response = await fetch("http://localhost/api2/testeApi.php/cliente/list", {
            method: "GET",
            headers: {
                "Content-Type": "application/json",
            },
        });
        const data = await response.json();

        let tableHtml =
"<table><tr><th>ID</th><th>Nome</th><th>Categoria</th><th>Preço</th><th>Custo</th><th>Descrição</th><th>Lucro</th><th>Estoque</th></tr>";
        data.forEach((item) => {
            tableHtml += `<tr onclick='selectRow(${item.id}, "${item.nome}", "${item.categoria}", "${item.preco}", "${item.custo}", "${item.descricao}", "${item.lucro}", "${item.estoque}")'>`;
            tableHtml += `<td>${item.id}</td>`;
            tableHtml += `<td>${item.nome}</td>`;
            tableHtml += `<td>${item.categoria}</td>`;
            tableHtml += `<td>${item.preco}</td>`;
            tableHtml += `<td>${item.custo}</td>`;
            tableHtml += `<td>${item.descricao}</td>`;
            tableHtml += `<td>${item.lucro}</td>`;
            tableHtml += `<td>${item.estoque}</td>`;
            tableHtml += `</tr>`;
        });
        tableHtml += "</table>";

        document.getElementById("table-container").innerHTML = tableHtml;
    } catch (error) {
        console.error("Erro ao executar solicitação GET:", error);
```
```

```
    }
}
```

Possíveis Erros:
- Falha na conexão com a API.
- Resposta não no formato JSON esperado.

POST (Cadastrar)
A função `post()` é acionada quando o usuário deseja adicionar um novo cliente. Os dados dos campos de entrada são coletados e enviados em formato JSON para a API.

```javascript
async function post() {
    try {
        const response = await fetch("http://localhost/api2/testeApi.php/cliente", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify({
                nome: document.getElementById("nome").value,
                categoria: document.getElementById("categoria").value,
                preco: document.getElementById("preco").value,
                custo: document.getElementById("custo").value,
                descricao: document.getElementById("descricao").value,
                lucro: document.getElementById("lucro").value,
                estoque: document.getElementById("estoque").value,
            }),
        });
        const data = await response.json();
        alert(JSON.stringify(data.message));
        get(); // Atualiza a tabela após adicionar
    } catch (error) {
        console.error("Erro ao executar solicitação POST:", error);
    }
}
```

Possíveis Erros:
- Falta de preenchimento nos campos obrigatórios.
- Erro na conexão ou na consulta SQL no backend.

PUT (Atualizar)

A função `put()` é utilizada para modificar os dados de um cliente existente. Os dados são enviados através de uma requisição PUT.

```javascript
async function put() {
    try {
        const response = await fetch("http://localhost/api2/testeApi.php/cliente/" +
document.getElementById("id").value, {
            method: "PUT",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify({
                nome: document.getElementById("nome").value,
                categoria: document.getElementById("categoria").value,
                preco: document.getElementById("preco").value,
                custo: document.getElementById("custo").value,
                descricao: document.getElementById("descricao").value,
                lucro: document.getElementById("lucro").value,
                estoque: document.getElementById("estoque").value,
            }),
        });
        const data = await response.json();
        alert(JSON.stringify(data.message));
        get(); // Atualiza a tabela após editar
    } catch (error) {
        console.error("Erro ao executar solicitação PUT:", error);
    }
}
```

Possíveis Erros:
- ID inválido.
- Erros de validação de dados.

DELETE (Excluir)

A função `del()` permite a exclusão de um cliente. Uma requisição DELETE é enviada para a API com o ID do cliente a ser removido.

```javascript
async function del() {
    try {
        const response = await fetch("http://localhost/api2/testeApi.php/cliente/" +
document.getElementById("id").value, {
            method: "DELETE",
        });
        const data = await response.json();
        alert(JSON.stringify(data.message));
        get(); // Atualiza a tabela após excluir
    } catch (error) {
```

```
        console.error("Erro ao executar solicitação DELETE:", error);
    }
}
```

Possíveis Erros:
- ID não encontrado no banco de dados.
- Erro de conexão com a API.

3. Seleção de Linha

A função `selectRow()` preenche os campos do formulário com os dados do cliente selecionado, facilitando a edição.

```javascript
function selectRow(id, nome, categoria, preco, custo, descricao, lucro, estoque) {
    document.getElementById("id").value = id;
    document.getElementById("nome").value = nome;
    document.getElementById("categoria").value = categoria;
    document.getElementById("preco").value = preco;
    document.getElementById("custo").value = custo;
    document.getElementById("descricao").value = descricao;
    document.getElementById("lucro").value = lucro;
    document.getElementById("estoque").value = estoque;
}
```

4. API PHP

A API PHP é responsável por receber as requisições do frontend e interagir com o banco de dados. As principais seções incluem:

Conexão com o Banco de Dados

O PHP se conecta ao banco de dados MySQL usando as credenciais definidas.

```php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "api";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Erro na conexão com o banco de dados: " . $conn->connect

_error);
```

}
```

Manipulação de Dados

Dependendo do método da requisição, a API executa operações no banco de dados, como inserções, atualizações, deleções e consultas.

```php
switch ($_SERVER['REQUEST_METHOD']) {
case 'POST':
// Código para inserir cliente...
break;
case 'PUT':
// Código para atualizar cliente...
break;
case 'DELETE':
// Código para deletar cliente...
break;
case 'GET':
// Código para obter lista de clientes...
break;
default:
$response['message'] = "Método não suportado.";
}
```

5. Criação do Banco de Dados

Um script PHP é fornecido para criar o banco de dados e a tabela `cliente`, se ainda não existirem. Isso é útil para a inicialização do sistema.

```php
$sql = "CREATE DATABASE api";
if ($conn->query($sql) === TRUE) {
**echo "Banco de dados 'api' criado com sucesso.
";**
}
```

Mensagens de Feedback

O sistema fornece mensagens de feedback ao usuário após cada operação, indicando o sucesso ou falha de uma operação.

Exemplos de Mensagens

- "Cliente adicionado com sucesso!"
- "Erro ao adicionar cliente: [detalhes do erro]"

- "Cliente atualizado com sucesso!"
- "Cliente excluído com sucesso!"

Conclusão

Este sistema CRUD é um exemplo prático de como desenvolver uma aplicação web interativa com um backend em PHP e um banco de dados MySQL. A separação entre frontend e backend permite uma arquitetura mais escalável e modular.

Se precisar de mais detalhes ou de uma parte específica, é só avisar!