**Documentação: CRUD de Estoque**

**Descrição**

Este sistema permite a gestão de um estoque de produtos, permitindo adicionar, editar e excluir produtos. Ele está implementado em PHP, HTML e CSS, com um banco de dados MySQL para armazenar as informações dos produtos.

**Arquitetura**

A aplicação é dividida em duas partes principais:

1. **Formulário de Cadastro/Atualização de Produtos**: Permite ao usuário adicionar ou editar os produtos.
2. **Lista de Produtos**: Exibe os produtos cadastrados no banco de dados com as opções de editar ou excluir.

**Banco de Dados**

A aplicação interage com uma tabela chamada produtos\_tbl no banco de dados estoque. A estrutura da tabela é a seguinte:

* **id** (INT, AUTO\_INCREMENT): Identificador único do produto.
* **nome** (VARCHAR): Nome do produto.
* **quantidade** (INT): Quantidade disponível do produto no estoque.
* **preco** (DECIMAL): Preço do produto.

**Funcionalidades**

**1. Conexão com o Banco de Dados**

O código se conecta ao banco de dados MySQL usando PDO. Caso a conexão falhe, uma mensagem de erro é exibida.

php

Copiar

$pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $user, $pass);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

**2. Adicionar Produto**

Quando um usuário preenche o formulário com os dados do produto (nome, quantidade e preço) e envia o formulário, um novo registro é inserido na tabela produtos\_tbl.

php

Copiar

$sql = "INSERT INTO produtos\_tbl (nome, quantidade, preco) VALUES (:nome, :quantidade, :preco)";

$stmt = $pdo->prepare($sql);

$stmt->execute(['nome' => $nome, 'quantidade' => $quantidade, 'preco' => $preco]);

**3. Editar Produto**

Quando um usuário clica no botão "Editar", o sistema preenche o formulário com os dados do produto selecionado para que ele possa ser modificado. Após a alteração, os dados são atualizados no banco de dados.

php

Copiar

$sql = "UPDATE produtos\_tbl SET nome = :nome, quantidade = :quantidade, preco = :preco WHERE id = :id";

$stmt = $pdo->prepare($sql);

$stmt->execute(['nome' => $nome, 'quantidade' => $quantidade, 'preco' => $preco, 'id' => $id]);

**4. Excluir Produto**

Quando o usuário clica em "Excluir", um formulário POST é enviado com a ação delete, que exclui o produto da tabela.

php

Copiar

$sql = "DELETE FROM produtos\_tbl WHERE id = :id";

$stmt = $pdo->prepare($sql);

$stmt->execute(['id' => $id]);

**5. Exibição da Lista de Produtos**

Os produtos cadastrados são exibidos em uma tabela HTML. Para cada produto, são exibidos o ID, nome, quantidade, preço e as opções de editar e excluir.

php

Copiar

$stmt = $pdo->query('SELECT \* FROM produtos\_tbl');

while ($row = $stmt->fetch(PDO::FETCH\_ASSOC)):

**6. Validação e Cancelamento**

O formulário de cadastro e atualização possui uma função JavaScript para validar o preenchimento dos campos. Além disso, se o usuário estiver editando um produto, pode cancelar a edição e retornar à lista de produtos.

**HTML e CSS**

A interface foi desenvolvida com HTML e CSS para fornecer um design simples e funcional para a aplicação. O CSS é utilizado para estruturar e estilizar os elementos da página, como formulários, tabelas e botões.

**Fluxo de Funcionamento**

1. **Adicionar Produto:**
   * O usuário preenche os campos de nome, quantidade e preço.
   * Ao clicar no botão "Salvar", um novo produto é adicionado ao banco de dados.
2. **Editar Produto:**
   * O usuário clica no botão "Editar" ao lado de um produto na lista.
   * O sistema preenche o formulário com os dados do produto selecionado.
   * O usuário pode modificar os campos e clicar em "Atualizar" para salvar as alterações.
3. **Excluir Produto:**
   * O usuário clica em "Excluir" ao lado do produto.
   * O produto é removido do banco de dados após a confirmação.

**Possíveis Melhorias e Considerações**

* **Validação de Dados**: A validação no lado do cliente poderia ser expandida para garantir que os dados inseridos no formulário sejam válidos, como checar se o preço e a quantidade são números positivos.
* **Segurança**: É importante aplicar medidas de segurança, como o uso de prepared statements para evitar SQL injection, o que já está sendo feito corretamente.
* **Feedback do Usuário**: A aplicação poderia fornecer feedback visual mais claro, como alertas em cores diferentes para diferentes tipos de ações (sucesso, erro).
* **Responsividade**: O layout poderia ser aprimorado para ser responsivo e funcionar bem em dispositivos móveis.