Documentação do Sistema “Doces da Gio”

Geovanna Moura SP3137465

Miqueias Nascimento SP3131785

09 de fevereiro de 2025

Introdução

O sistema "Doces da Gio" foi desenvolvido com o objetivo de fornecer uma plataforma simples e eficaz para o gerenciamento de usuários de uma doceria.

Com uma interface intuitiva e funcionalidades básicas de CRUD (Criar, Ler, Atualizar e Deletar), ele permite que administradores realizem o cadastro, login e manutenção de usuários de forma rápida e segura. O sistema foi estruturado utilizando tecnologias amplamente utilizadas no desenvolvimento web: HTML e CSS para o front-end, PHP para o back-end e MySQL para o gerenciamento do banco de dados.

A solução foi projetada para ser fácil de usar, com uma navegação fluida entre as páginas, permitindo que os usuários se cadastrem e acessem suas informações de maneira eficiente. Com foco na experiência do usuário e em boas práticas de segurança, o sistema oferece funcionalidades essenciais de forma clara e objetiva.

Arquitetura do Sistema

1. Front-End:
    - HTML: Estruturação das páginas da aplicação.
    - CSS: Estilo visual das páginas da aplicação.
2. Back-End:
    - PHP: Responsável pelo processamento das informações, incluindo login, cadastro e manipulação de dados de usuários.
3. Banco de Dados:
    - SQL: Utilizado para o armazenamento e gerenciamento dos dados dos usuários.

Front End e Back End

Tela Inicial

Descrição:
A tela inicial apresenta uma visão geral do sistema, com informações básicas sobre a loja "Doces da Gio", como nome da loja e o botão de login para os usuários autenticados.

Tecnologias:

- **HTML**: Estruturação do conteúdo da página.
- **CSS**: Estilo visual e layout.

index.html | styles.css

Tela de Login

Descrição:
Permite que o usuário faça login utilizando seu nome de usuário e senha.

Tecnologias:

- **HTML**: Estrutura do formulário de login.
- **PHP**: Processamento da autenticação.
- **SQL**: Consulta ao banco de dados para verificar a existência do usuário.

login.php | login.css

Tela de Cadastro

Descrição:
Permite que o usuário se cadastre com informações como nome de usuário e senha.

Tecnologias:

- **HTML**: Estruturação do formulário de cadastro.
- **PHP**: Processamento e inserção de dados no banco de dados.
- **SQL**: Inserção dos dados no banco de dados.

cadastro.php | cadastro.css

Tela de Usuário Logado

Descrição:
Esta página exibe as opções para que o usuário logado possa inserir, consultar, editar ou excluir informações de outros usuários.

Tecnologias:

- **PHP**: Processamento das operações de CRUD (Create, Read, Update, Delete).
- **SQL**: Consultas e manipulação dos dados no banco de dados.

testelogin.php | delete.php | edit.php | sair.php | saveedit.php |

config.php | sistema.php

Banco de dados

Tabelas

1. Tabela `users`:

- **Descrição**: Armazena informações dos usuários do sistema.
- **Campos**:
  - `id`: Identificador único para cada usuário.
  - `nome`: Nome completo do usuário.
  - `email`: Endereço de e-mail do usuário.
  - `senha`: Senha do usuário.
  - `cpf`: CPF do usuário (único).
  - `telefone`: Número de telefone do usuário.
  - `endereco`: Endereço completo do usuário.
  - `tipo`: Tipo de usuário (admin ou comum), com valor padrão `comum`.

2. Tabela `tb_contato`:

- **Descrição**: Armazena informações de contato adicionais para os usuários.
- **Campos**:
  - `telefone`: Número de telefone do usuário.
  - `email`: Endereço de e-mail do usuário.
  - `fk_id_user`: Chave estrangeira que referencia o `id` da tabela `users`.

3. Tabela `tb_endereco`:

- **Descrição**: Armazena endereços adicionais para os usuários.
- **Campos**:
  - `endereco`: Endereço do usuário.
  - `fk_id_user`: Chave estrangeira que referencia o `id` da tabela `users`.

4. Tabela `tb_doceria`:

- **Descrição**: Armazena informações sobre a doceria (loja de doces).
- **Campos**:
  - `pk_cnpj`: CNPJ da doceria (identificador único).
  - `nome`: Nome da doceria.

5. Tabela `tb_endereco_doceria`:

- **Descrição**: Armazena os endereços das docerias.
- **Campos**:
  - `pk_cep`: Código postal (CEP) da doceria.
  - `rua`, `bairro`, `numero`, `complemento`, `cidade`: Informações detalhadas do endereço da doceria.
  - `fk_cnpj`: Chave estrangeira que referencia o `pk_cnpj` da tabela `tb_doceria`.

6. Tabela `tb_funcionarios`:

- **Descrição**: Armazena informações dos funcionários da doceria.
- **Campos**:
  - `pk_id_funcionario`: Identificador único do funcionário.
  - `nome`: Nome do funcionário.
  - `funcao`: Função que o funcionário exerce.
  - `fk_cnpj`: Chave estrangeira que referencia o `pk_cnpj` da tabela `tb_doceria`.

7. Tabela `tb_itens_pedidos`:

- **Descrição**: Armazena os itens de um pedido feito pelos usuários.
- **Campos**:
  - `pk_itens_pedidos`: Identificador único do item de pedido.
  - `fk_id_user`: Chave estrangeira que referencia o `id` da tabela `users`.
  - `fk_id_produto`: Chave estrangeira que referencia o `pk_id_produto` da tabela `tb_produto`.
  - `fk_id_pedido`: Chave estrangeira que referencia o `pk_id_pedido` da tabela `tb_pedido`.

    - `subtotal_itens_pedidos`: Subtotal do item no pedido.
    - `data_hora`: Data e hora em que o item foi pedido.

8. Tabela `tb_nota_fiscal`:

- **Descrição**: Armazena as notas fiscais geradas para os itens de pedidos.
- **Campos**:
    - `pk_nota_fiscal`: Identificador único da nota fiscal.
    - `fk_itens_pedidos`: Chave estrangeira que referencia o `pk_itens_pedidos` da tabela `tb_itens_pedidos`.
    - `fk_cnpj`: Chave estrangeira que referencia o `pk_cnpj` da tabela `tb_doceria`.
    - `fk_id_user`: Chave estrangeira que referencia o `id` da tabela `users`.

9. Tabela `tb_pedido`:

- **Descrição**: Armazena os pedidos realizados pelos usuários.
- **Campos**:
    - `pk_id_pedido`: Identificador único do pedido.
    - `fk_id_user`: Chave estrangeira que referencia o `id` da tabela `users`.
    - `fk_cnpj`: Chave estrangeira que referencia o `pk_cnpj` da tabela `tb_doceria`.

10. Tabela `tb_produto`:

- **Descrição**: Armazena informações sobre os produtos disponíveis para venda na doceria.
- **Campos**:
    - `pk_id_produto`: Identificador único do produto.
    - `nome_produto`: Nome do produto.
    - `valor_produto`: Preço do produto.
    - `qntd_estoque_produto`: Quantidade do produto em estoque.
    - `descricao_produto`: Descrição do produto.

Procedures

1. `AlterarParaAdmin`

```
CREATE PROCEDURE `AlterarParaAdmin` (IN `userId` INT)
BEGIN
    UPDATE users
```

```
    SET tipo = 'admin'
    WHERE id = userId;
END
```

- **Objetivo**: Essa procedure tem como objetivo alterar o tipo de usuário de um usuário específico para "admin". Ela recebe como parâmetro o `userId` (identificador do usuário), e quando chamada, atualiza o campo `tipo` da tabela `users`, definindo-o como `'admin'` para o usuário com o id correspondente.

2. `ObterContato`

```
CREATE PROCEDURE `ObterContato` (IN `id_user` INT)
BEGIN
    SELECT * FROM tb_contato WHERE fk_id_user = id_user;
END
```

- **Objetivo**: A **procedure** `ObterContato` recebe o `id_user` (id do usuário) e retorna todos os dados da tabela `tb_contato` onde o campo `fk_id_user` corresponde ao `id_user` fornecido. Ou seja, ela retorna as informações de contato de um usuário específico, como telefone e e-mail.

3. `ObterEndereco`

```
CREATE PROCEDURE `ObterEndereco` (IN `id_user` INT)
BEGIN
    SELECT * FROM tb_endereco WHERE fk_id_user = id_user;
END
```

- **Objetivo**: A **procedure** `ObterEndereco` funciona de maneira similar à `ObterContato`, mas no caso do endereço. Ela recebe o `id_user` (id do usuário) e retorna os dados da tabela `tb_endereco` onde o campo `fk_id_user` é igual ao `id_user` fornecido, ou seja, retorna o endereço de um usuário específico.

4. `PreencherContato`

```
CREATE PROCEDURE `PreencherContato` ()
BEGIN
    INSERT INTO tb_contato (telefone, email, fk_id_user)
    SELECT telefone, email, id
    FROM users
```

```
    WHERE id NOT IN (SELECT fk_id_user FROM tb_contato);
END
```

- **Objetivo**: Esta **procedure** preenche a tabela `tb_contato` com os dados de contato (telefone e e-mail) dos usuários que ainda não possuem um registro na tabela de contatos. Ela faz uma inserção dos dados da tabela `users` (telefone, e-mail e o id do usuário) na tabela `tb_contato`, mas somente para os usuários cujos `id` ainda não estão na tabela `tb_contato` (verificado pela subconsulta).

5. `PreencherEndereco`

```
CREATE PROCEDURE `PreencherEndereco` ()
BEGIN
    INSERT INTO tb_endereco (endereco, fk_id_user)
    SELECT endereco, id
    FROM users
    WHERE id NOT IN (SELECT fk_id_user FROM tb_endereco);
END
```

- **Objetivo**: Assim como a **procedure** anterior, esta preenche a tabela `tb_endereco` com os dados de endereço dos usuários que ainda não têm um registro na tabela de endereços. Ela insere o `endereco` e o `id` do usuário da tabela `users` na tabela `tb_endereco`, mas apenas para os usuários que ainda não estão na tabela `tb_endereco`.

Functions

1. `VerificarSeAdmin`

```
CREATE FUNCTION `VerificarSeAdmin` (IN `userId` INT)
RETURNS BOOLEAN
BEGIN
    DECLARE isAdmin BOOLEAN;
    SELECT tipo INTO isAdmin
    FROM users
    WHERE id = userId;
    RETURN isAdmin = 'admin';
END
```

- **Objetivo**: A **function** `VerificarSeAdmin` recebe como parâmetro o `userId` (id do usuário) e retorna um valor booleano indicando se o usuário com esse id é um administrador ou não.
    - O processo consiste em verificar se o campo `tipo` da tabela `users` (que indica o tipo de usuário) é igual a `'admin'`.
    - Se for um administrador, a função retorna `TRUE`, caso contrário, retorna `FALSE`.

2. `ObterTelefone`

```
CREATE FUNCTION `ObterTelefone` (IN `id_user` INT)
RETURNS VARCHAR(15)
BEGIN
    DECLARE telefone VARCHAR(15);
    SELECT telefone INTO telefone
    FROM tb_contato
    WHERE fk_id_user = id_user;
    RETURN telefone;
END
```

- **Objetivo**: A **function** `ObterTelefone` recebe o `id_user` (id do usuário) e retorna o número de telefone associado a esse usuário.
    - Ela seleciona o telefone da tabela `tb_contato` onde o campo `fk_id_user` (referência ao id do usuário) corresponde ao `id_user` passado como argumento.
    - O valor do telefone é então retornado pela função.

3. `ObterEmail`

```
CREATE FUNCTION `ObterEmail` (IN `id_user` INT)
RETURNS VARCHAR(100)
BEGIN
    DECLARE email VARCHAR(100);
    SELECT email INTO email
    FROM tb_contato
    WHERE fk_id_user = id_user;
    RETURN email;
END
```

- **Objetivo**: A **function** `ObterEmail` é muito semelhante à `ObterTelefone`, mas no caso, ela retorna o endereço de e-mail de um usuário específico.

    - Ela recebe o `id_user` como parâmetro e retorna o valor de `email` da tabela `tb_contato` onde o campo `fk_id_user` corresponde ao `id_user` fornecido.

4. `ObterEnderecoUsuario`

```
CREATE FUNCTION `ObterEnderecoUsuario` (IN `id_user` INT)
RETURNS VARCHAR(255)
BEGIN
    DECLARE endereco VARCHAR(255);
    SELECT endereco INTO endereco
    FROM tb_endereco
    WHERE fk_id_user = id_user;
    RETURN endereco;
END
```

- **Objetivo**: A **function** `ObterEnderecoUsuario` funciona de maneira similar às anteriores, mas em vez de retornar telefone ou e-mail, ela retorna o endereço do usuário.
    - Ela recebe o `id_user` como parâmetro e retorna o valor de `endereco` da tabela `tb_endereco`, onde o campo `fk_id_user` é igual ao `id_user` fornecido.

Conclusão

O sistema "Doces da Gio" é uma ferramenta eficiente para gerenciar usuários, oferecendo funcionalidades essenciais de forma acessível. A combinação de tecnologias como HTML, CSS, PHP e MySQL proporciona uma base sólida e flexível para futuras melhorias ou adições de recursos. A implementação do CRUD permite um controle completo sobre os dados dos usuários, garantindo uma administração simples e segura.

Embora o sistema seja básico, ele serve como um excelente ponto de partida para o desenvolvimento de soluções mais complexas e personalizadas para o gerenciamento de usuários. Com uma estrutura bem definida e fácil manutenção, ele pode ser facilmente expandido ou integrado a outras plataformas conforme a necessidade do negócio.

Link para apresentação:
https://drive.google.com/drive/folders/1wXjvl5c8ojcl1ac0uc4oH91m5tSzb52u?usp=sharing

Link para baixar o xampp: https://www.apachefriends.org/pt_br/index.html

Link Github: https://github.com/Miqueiasnasc/banco-de-dados/tree/main