Documentação do Projeto - Back-end

1. Introdução

Este projeto foi desenvolvido utilizando a arquitetura MVC (Model-View-Controller) e segue a estrutura de três camadas, separando a apresentação, a lógica de negócios e os dados.

O sistema consiste em um sistema de gerenciamento de usuários, onde é possível realizar login, visualizar, editar e excluir informações de usuários. A autenticação e as permissões são verificadas para garantir que os usuários possam modificar apenas seus próprios dados ou os dados de outros usuários caso sejam administradores.

2. Arquitetura do Sistema

Camada de Apresentação (View)

A camada de apresentação é composta pelas páginas HTML, CSS, e JavaScript responsáveis pela interface do usuário. As páginas são:

login.php: Página que permite ao usuário realizar login informando o e-mail e a senha.

sistema.php: Página principal onde os usuários logados podem visualizar os dados dos usuários cadastrados. Também permite a edição e exclusão de usuários, conforme as permissões.

edit.php: Página de edição onde os dados do usuário selecionado podem ser alterados.

Camada de Lógica de Negócio (Controller)

A camada de lógica de negócio é responsável por coordenar a interação entre a camada de apresentação e a camada de dados. Em nosso projeto, a lógica de controle está distribuída nas páginas PHP que gerenciam a autenticação e a manipulação dos dados dos usuários.

As principais funcionalidades de controle são:

login.php: Verifica se o usuário e a senha estão corretos e, em seguida, inicia uma sessão.

sistema.php: Exibe a lista de usuários e permite a pesquisa, edição e exclusão.

saveedit.php: Atualiza os dados de um usuário no banco de dados.

sair.php: Encerra a sessão do usuário.

Camada de Dados (Model)

A camada de dados é responsável pela comunicação com o banco de dados, utilizando consultas SQL para recuperar e atualizar informações dos usuários. A interação com o banco de dados é feita diretamente nas páginas PHP que implementam a lógica de negócio. Não foi utilizado um arquivo separado de modelo (como uma classe) no seu código, mas a interação com o banco de dados ocorre diretamente nas páginas.

Exemplo de interação com o banco de dados (consulta):

php

CopyEdit

```
$sql = "SELECT id, tipo FROM users WHERE email = '$email' AND senha = '$senha'";
$result = $conexao->query($sql);
```

3. Descrição das Funcionalidades

Login e Autenticação (testelogin.php)

Objetivo: Validar as credenciais do usuário e iniciar uma sessão.

Funcionamento: O sistema verifica se os campos email e senha estão preenchidos. Caso a consulta no banco de dados retorne um usuário válido, a sessão é iniciada com as variáveis email, senha, id e tipo. Caso contrário, o usuário é redirecionado para a página de login.

Exemplo de código para verificação de login (testelogin.php):

php

CopyEdit

```
$email = $_POST['email'];
$senha = $_POST['senha'];
$sql = "SELECT id, tipo FROM users WHERE email = '$email' AND senha = '$senha'";
$result = $conexao->query($sql);

if ($result->num_rows > 0) {
   // Sessão iniciada
   session_start();
   $_SESSION['email'] = $email;
   $_SESSION['senha'] = $senha;
   $_SESSION['id'] = $id;
   $_SESSION['tipo'] = $tipo;
   header("Location: sistema.php");
```

```
} else {
   echo "Usuário ou senha inválidos!";
}
```

Sistema de Gerenciamento de Usuários (sistema.php)

Objetivo: Exibir a lista de usuários registrados, com opções de pesquisa e de ações (editar e excluir).

Funcionamento: O sistema verifica se o usuário está autenticado. Caso contrário, é redirecionado para a página de login. A lista de usuários é carregada com uma consulta SQL, permitindo pesquisa por id, nome ou email. O administrador pode editar ou excluir qualquer usuário, enquanto os usuários comuns podem editar apenas seus próprios dados.

Exemplo de código para exibição de usuários (sistema.php):

php

CopyEdit

```
session_start();
if (!isset($_SESSION['email'])) {
   header("Location: login.php");
   exit();
}

$sql = "SELECT id, nome, email, tipo FROM users";
$result = $conexao->query($sql);

while ($row = $result->fetch_assoc()) {
   echo "<tr>
         <td>" . $row['id'] . "</td>
         <td>" . $row['nome'] . "</td>
         <td>" . $row['email'] . "</td>
         <td>" . $row['tipo'] . "</td>
         <td>
            <a href='edit.php?id=" . $row['id'] . "'>Editar</a> |
            <a href='delete.php?id=" . $row['id'] . "'>Excluir</a>
         </td>
      </tr>";
}
```

Pesquisa de Usuários

Objetivo: Permitir a busca de usuários pelo nome, e-mail ou ID.

Funcionamento: A pesquisa é realizada através da URL, passando o termo de pesquisa como parâmetro GET. Quando o usuário digita um termo e pressiona "Enter", o sistema realiza a pesquisa no banco de dados e exibe os resultados.

Exemplo de código para pesquisa de usuários (sistema.php):

php

CopyEdit

```
$pesquisa = $_GET['pesquisa'];
$sql = "SELECT id, nome, email, tipo FROM users WHERE nome LIKE '%$pesquisa%' OR email LIKE '%$pesquisa%'";
$result = $conexao->query($sql);
```

Edição de Dados de Usuário (saveedit.php)

Objetivo: Permitir que um usuário ou administrador edite os dados de um usuário.

Funcionamento: Ao editar os dados de um usuário, a página saveedit.php recebe os dados via POST e executa uma consulta SQL para atualizar o usuário no banco de dados. A alteração é feita apenas se o usuário tiver permissão para editar aquele usuário (um administrador pode editar qualquer usuário, enquanto um usuário comum pode editar apenas seu próprio cadastro).

Exemplo de código para edição de usuário (saveedit.php):

php

CopyEdit

```
$id = $_POST['id'];
$nome = $_POST['nome'];
$email = $_POST['email'];
$telefone = $_POST['telefone'];
$endereco = $_POST['endereco'];

$sql = "UPDATE users SET nome='$nome', email='$email', telefone='$telefone', endereco='$endereco' WHERE id='$id'";
if ($conexao->query($sql) === TRUE) {
   echo "Usuário atualizado com sucesso!";
} else {
   echo "Erro ao atualizar usuário: " . $conexao->error;
```

}

Logout (sair.php)

Objetivo: Encerra a sessão do usuário e o redireciona para a página de login.

Funcionamento: Quando o usuário clica no botão "Sair", as variáveis de sessão relacionadas ao email e senha são removidas, encerrando a sessão. O usuário é então redirecionado para a página de login.

Exemplo de código para logout (sair.php):

php

CopyEdit

```
session_start();
session_unset();
session_destroy();
header("Location: login.php");
```

4. Fluxo de Execução

O fluxo do sistema pode ser descrito da seguinte forma:

Login (testelogin.php):

O usuário insere seu email e senha no formulário de login.

O sistema consulta o banco de dados para verificar as credenciais.

Se o login for bem-sucedido, o usuário é redirecionado para a página sistema.php.

Página Principal (sistema.php):

O sistema exibe a lista de usuários registrados.

O administrador pode editar ou excluir qualquer usuário, enquanto os usuários comuns podem editar apenas seus próprios dados.

Edição de Dados (saveedit.php):

O usuário ou administrador altera os dados de um usuário.

A alteração é salva no banco de dados.

Logout (sair.php):

O usuário clica no botão "Sair" para encerrar a sessão e é redirecionado para login.php.

5. Banco de Dados

O banco de dados utilizado para armazenar as informações é o MySQL. A tabela users contém os seguintes campos:

id: Identificador único do usuário (auto incremento).

nome: Nome completo do usuário.

email: E-mail do usuário (único).

senha: Senha do usuário (armazenada de forma simples, sem criptografia no código fornecido).

telefone: Número de telefone do usuário.

cpf: CPF do usuário.

endereco: Endereço do usuário.

tipo: Tipo de usuário (admin ou comum).

6. Considerações de Segurança

Autenticação Simples: O código atual valida as credenciais diretamente no banco de

4o mini

Documentação do Projeto - Back-end

1. Introdução

Este projeto foi desenvolvido utilizando a arquitetura MVC (Model-View-Controller) e segue a estrutura de três camadas, separando a apresentação, a lógica de negócios e os dados.

O sistema consiste em um sistema de gerenciamento de usuários, onde é possível realizar login, visualizar, editar e excluir informações de usuários. A autenticação e as permissões são verificadas para garantir que os usuários possam modificar apenas seus próprios dados ou os dados de outros usuários caso sejam administradores.

2. Arquitetura do Sistema

Camada de Apresentação (View)

A camada de apresentação é composta pelas páginas **HTML**, **CSS**, e **JavaScript** responsáveis pela interface do usuário. As páginas são:

- **login.php**: Página que permite ao usuário realizar login informando o e-mail e a senha.
- **sistema.php**: Página principal onde os usuários logados podem visualizar os dados dos usuários cadastrados. Também permite a edição e exclusão de usuários, conforme as permissões.
- **edit.php**: Página de edição onde os dados do usuário selecionado podem ser alterados.

Camada de Lógica de Negócio (Controller)

A camada de lógica de negócio é responsável por coordenar a interação entre a camada de apresentação e a camada de dados. Em nosso projeto, a lógica de controle está distribuída nas páginas PHP que gerenciam a autenticação e a manipulação dos dados dos usuários.

As principais funcionalidades de controle são:

- **login.php**: Verifica se o usuário e a senha estão corretos e, em seguida, inicia uma sessão.
- **sistema.php**: Exibe a lista de usuários e permite a pesquisa, edição e exclusão.
- **saveedit.php**: Atualiza os dados de um usuário no banco de dados.
- **sair.php**: Encerra a sessão do usuário.

Camada de Dados (Model)

A camada de dados é responsável pela comunicação com o banco de dados, utilizando consultas SQL para recuperar e atualizar informações dos usuários. A interação com o banco de dados é feita diretamente nas páginas PHP que implementam a lógica de negócio. Não foi utilizado um arquivo separado de modelo (como uma classe) no seu código, mas a interação com o banco de dados ocorre diretamente nas páginas.

Exemplo de interação com o banco de dados (consulta):

```
php
CopyEdit
$sql = "SELECT id, tipo FROM users WHERE email = '$email' AND senha
= '$senha'";
$result = $conexao->query($sql);
```

3. Descrição das Funcionalidades

Login e Autenticação (testelogin.php)

- **Objetivo:** Validar as credenciais do usuário e iniciar uma sessão.
- **Funcionamento:** O sistema verifica se os campos **email** e **senha** estão preenchidos. Caso a consulta no banco de dados retorne um usuário válido, a sessão é iniciada com as variáveis **email**, **senha**, **id** e **tipo**. Caso contrário, o usuário é redirecionado para a página de login.

Exemplo de código para verificação de login (testelogin.php):

```
php
CopyEdit
$email = $_POST['email'];
$senha = $_POST['senha'];
$sql = "SELECT id, tipo FROM users WHERE email = '$email' AND senha
= '$senha'";
$result = $conexao->query($sql);

if ($result->num_rows > 0) {
    // Sessão iniciada
    session_start();
    $_SESSION['email'] = $email;
    $_SESSION['senha'] = $senha;
    $_SESSION['id'] = $id;
    $_SESSION['tipo'] = $tipo;
    header("Location: sistema.php");
} else {
    echo "Usuário ou senha inválidos!";
}
```

Sistema de Gerenciamento de Usuários (sistema.php)

- **Objetivo:** Exibir a lista de usuários registrados, com opções de pesquisa e de ações (editar e excluir).
- **Funcionamento:** O sistema verifica se o usuário está autenticado. Caso contrário, é redirecionado para a página de login. A lista de usuários é carregada com uma consulta SQL, permitindo pesquisa por **id**, **nome** ou **email**. O administrador pode editar ou excluir qualquer usuário, enquanto os usuários comuns podem editar apenas seus próprios dados.

Exemplo de código para exibição de usuários (sistema.php):

```
php
CopyEdit
session_start();
if (!isset($_SESSION['email'])) {
    header("Location: login.php");
    exit();
}

$sql = "SELECT id, nome, email, tipo FROM users";
$result = $conexao->query($sql);

while ($row = $result->fetch_assoc()) {
    echo "<tr>
            <td>" . $row['id'] . "</td>
            <td>" . $row['nome'] . "</td>
            <td>" . $row['email'] . "</td>
            <td>" . $row['tipo'] . "</td>
            <td>
                <a href='edit.php?id=" . $row['id'] . "'>Editar</a>
|
                <a href='delete.php?id=" . $row['id'] .
"'>Excluir</a>
            </td>
        </tr>";
}
```

Pesquisa de Usuários

- **Objetivo:** Permitir a busca de usuários pelo nome, e-mail ou ID.
- **Funcionamento:** A pesquisa é realizada através da URL, passando o termo de pesquisa como parâmetro GET. Quando o usuário digita um termo e pressiona "Enter", o sistema realiza a pesquisa no banco de dados e exibe os resultados.

Exemplo de código para pesquisa de usuários (sistema.php):

```
php
CopyEdit
$pesquisa = $_GET['pesquisa'];
$sql = "SELECT id, nome, email, tipo FROM users WHERE nome LIKE
'%$pesquisa%' OR email LIKE '%$pesquisa%'";
```

```
$result = $conexao->query($sql);
```

Edição de Dados de Usuário (saveedit.php)

- **Objetivo:** Permitir que um usuário ou administrador edite os dados de um usuário.
- **Funcionamento:** Ao editar os dados de um usuário, a página **saveedit.php** recebe os dados via **POST** e executa uma consulta SQL para atualizar o usuário no banco de dados. A alteração é feita apenas se o usuário tiver permissão para editar aquele usuário (um administrador pode editar qualquer usuário, enquanto um usuário comum pode editar apenas seu próprio cadastro).

Exemplo de código para edição de usuário (saveedit.php):

```
php
CopyEdit
$id = $_POST['id'];
$nome = $_POST['nome'];
$email = $_POST['email'];
$telefone = $_POST['telefone'];
$endereco = $_POST['endereco'];

$sql = "UPDATE users SET nome='$nome', email='$email',
telefone='$telefone', endereco='$endereco' WHERE id='$id'";
if ($conexao->query($sql) === TRUE) {
    echo "Usuário atualizado com sucesso!";
} else {
    echo "Erro ao atualizar usuário: " . $conexao->error;
}
```

Logout (sair.php)

- **Objetivo:** Encerra a sessão do usuário e o redireciona para a página de login.
- **Funcionamento:** Quando o usuário clica no botão "Sair", as variáveis de sessão relacionadas ao **email** e **senha** são removidas, encerrando a sessão. O usuário é então redirecionado para a página de login.

Exemplo de código para logout (sair.php):

```
php
CopyEdit
```

```
session_start();
session_unset();
session_destroy();
header("Location: login.php");
```

4. Fluxo de Execução

O fluxo do sistema pode ser descrito da seguinte forma:

1. **Login (testelogin.php)**:
    a. O usuário insere seu **email** e **senha** no formulário de login.
    b. O sistema consulta o banco de dados para verificar as credenciais.
    c. Se o login for bem-sucedido, o usuário é redirecionado para a página **sistema.php**.
2. **Página Principal (sistema.php)**:
    a. O sistema exibe a lista de usuários registrados.
    b. O administrador pode editar ou excluir qualquer usuário, enquanto os usuários comuns podem editar apenas seus próprios dados.
3. **Edição de Dados (saveedit.php)**:
    a. O usuário ou administrador altera os dados de um usuário.
    b. A alteração é salva no banco de dados.
4. **Logout (sair.php)**:
    a. O usuário clica no botão "Sair" para encerrar a sessão e é redirecionado para **login.php**.

5. Banco de Dados

O banco de dados utilizado para armazenar as informações é o **SQL**. A tabela **users** contém os seguintes campos:

- **id**: Identificador único do usuário (auto incremento).
- **nome**: Nome completo do usuário.
- **email**: E-mail do usuário (único).
- **senha**: Senha do usuário (armazenada de forma simples, sem criptografia no código fornecido).
- **telefone**: Número de telefone do usuário.
- **cpf**: CPF do usuário.
- **endereco**: Endereço do usuário.
- **tipo**: Tipo de usuário (`admin` ou `comum`).

6. Considerações de Segurança

- **Autenticação Simples:** O código atual valida as credenciais diretamente no banco de