Faculdade Brasília - FBr

Análise e Desenvolvimento de Sistemas - ADS

Pedro Henrique Da Silva

Ana Beatriz Frazão

Davi Rodrigues

Felipe Andrade

Rafael Alves

Welisson Henrique

PROJETO DE ANÁLISE E MODELAGEM DE SISTEMAS

Santa Maria - DF

2025

Pedro Henrique Da Silva
Ana Beatriz Frazão
Davi Rodrigues
Felipe Andrade
Rafael Alves
Welisson Henrique

PROJETO DE ANÁLISE E MODELAGEM DE SISTEMAS

Projeto De Análise e Desenvolvimento de Sistemas Destinado a disciplina de Análise e Modelagem de Sistemas como requisito para avaliação bimestral do 3ºSemestre de ADS.

Orientador: Prof. Willians Luiz Gomes

Santa Maria - DF
2025

SUMÁRIO

1. OBJETIVO

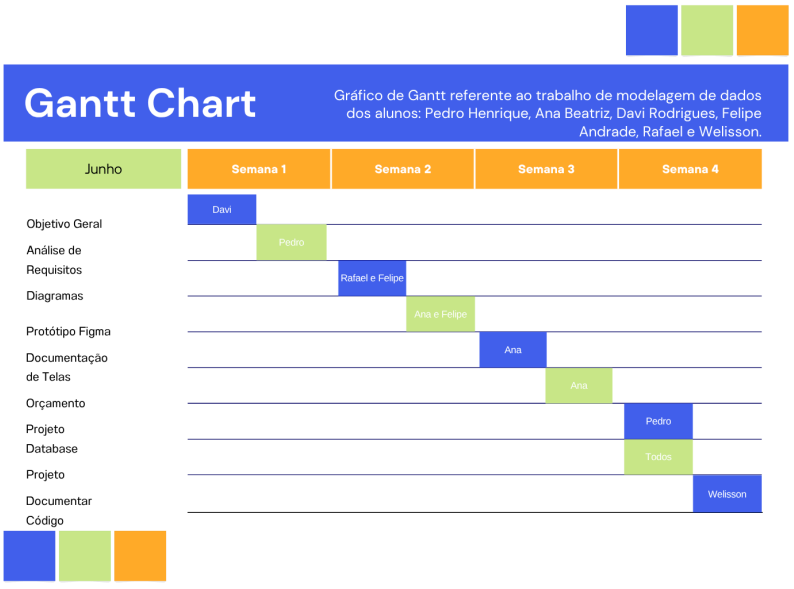
Objetivo Geral

O desenvolvimento deste projeto teve como principal objetivo aprimorar as habilidades da equipe na criação de uma tela de cadastro, algo fundamental dentro do universo da programação e desenvolvimento de sistemas. Mais do que simplesmente construir a interface, o processo ajudou o grupo a entender a importância de planejar, organizar e pensar estrategicamente cada etapa, desenvolvendo uma visão mais ampla sobre como conduzir um projeto do início ao fim.

Além das questões técnicas, um dos maiores aprendizados da equipe foi sobre como se organizar para que tudo saísse conforme o planejado. Dividir as tarefas, definir prioridades, resolver problemas que surgiram no caminho e administrar bem o tempo foram ações fundamentais durante todo o processo.

De forma geral, este projeto não só ajudou a equipe a desenvolver habilidades técnicas importantes, como também fortaleceu a capacidade do grupo de se organizar, planejar e executar ideias de maneira mais eficiente. Foi uma experiência que, sem dúvidas, agregou muito tanto para o desenvolvimento técnico quanto para a visão sobre como tornar qualquer projeto mais claro, organizado e possível de ser realizado.

2. CRONOGRAMA



3. LEVANTAMENTO DE REQUISITOS

3.1. Objetivo do Sistema

Permitir que usuários e administradores possam se cadastrar e interagir com um ambiente de gerenciamento de eventos, com funcionalidades distintas para cada tipo de acesso.

3.2. Requisitos Funcionais

3.2.1. Cadastro e Autenticação

- RF01: O sistema deve permitir o cadastro de usuários comuns.
- RF02: O sistema deve permitir o cadastro de usuários administradores.
- RF03: O sistema deve permitir o login com e-mail e senha para ambos os tipos.
- RF04: O sistema deve validar e-mails para aceitar apenas domínios .com no momento do cadastro.

3.2.2. Funcionalidades do Usuário

- RF05: O usuário autenticado deve ter acesso a um painel de eventos.
- RF06: O usuário pode criar novos eventos com dados como título, descrição e data.
- RF07: O usuário pode concluir eventos já criados.
- RF08: O usuário pode excluir eventos.

3.2.3. Funcionalidades do Administrador

- RF09: O administrador autenticado deve ter acesso a um painel administrativo.
- RF10: O painel do administrador deve exibir uma lista de todos os usuários cadastrados.
- RF11: O sistema deve exibir informações completas dos usuários na tela do administrador (como nome, e-mail, status de admin).
- RF12: O administrador deve ter a funcionalidade de excluir contas de usuários cadastrados.

3.3. Requisitos Não Funcionais

- RNF01: A interface deve estar em português.
- RNF02: O sistema deve seguir boas práticas de segurança no armazenamento de senhas (ex: hash).
- RNF03: O sistema deve ser acessível por navegadores modernos.
- RNF04: Deve haver validação dos dados tanto no frontend quanto no backend (PHP).

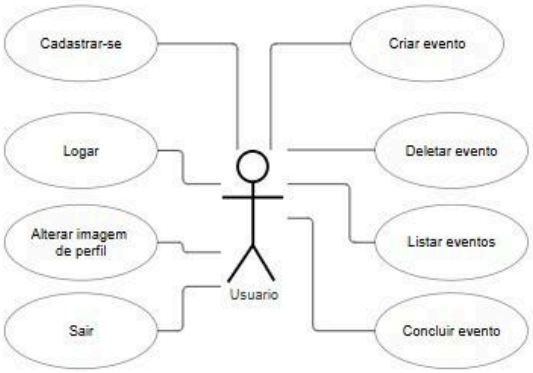
3.4. Regras de Negócio

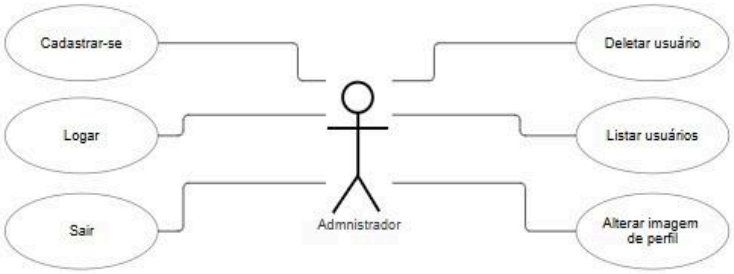
- RN01: Apenas administradores podem visualizar a lista de usuários.
- RN02: Cada evento pertence exclusivamente ao usuário que o criou.

- RN03: Um evento marcado como “concluído” não deve poder ser editado.
- RN04: Um usuário comum não pode acessar o painel de administração.

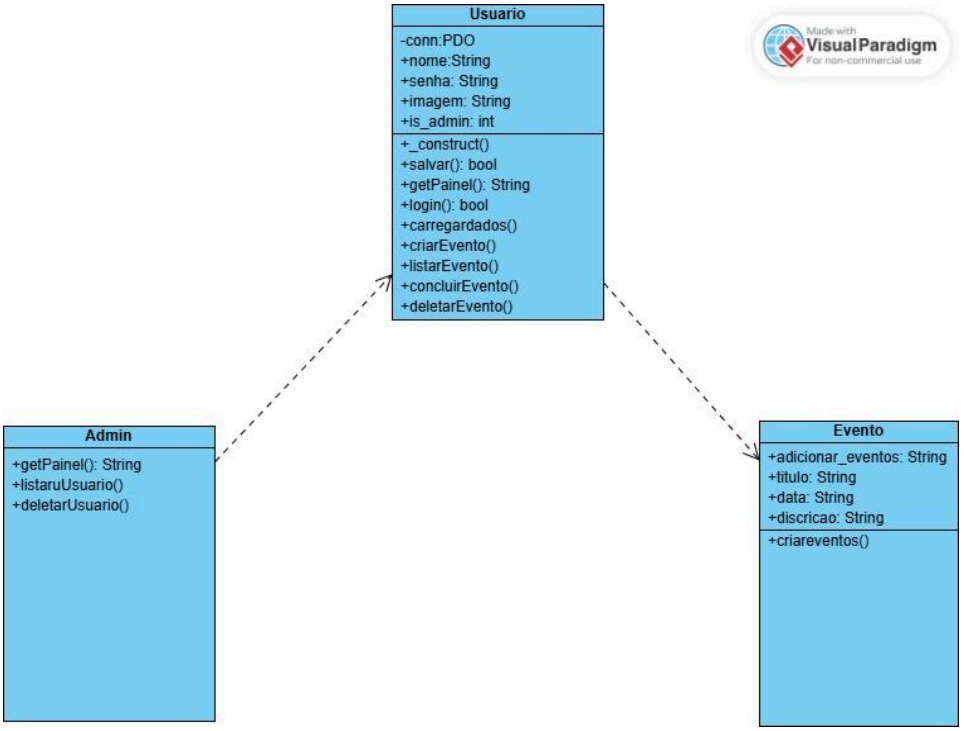
4. DIAGRAMAS

4.1 - DIAGRAMA CASO USO

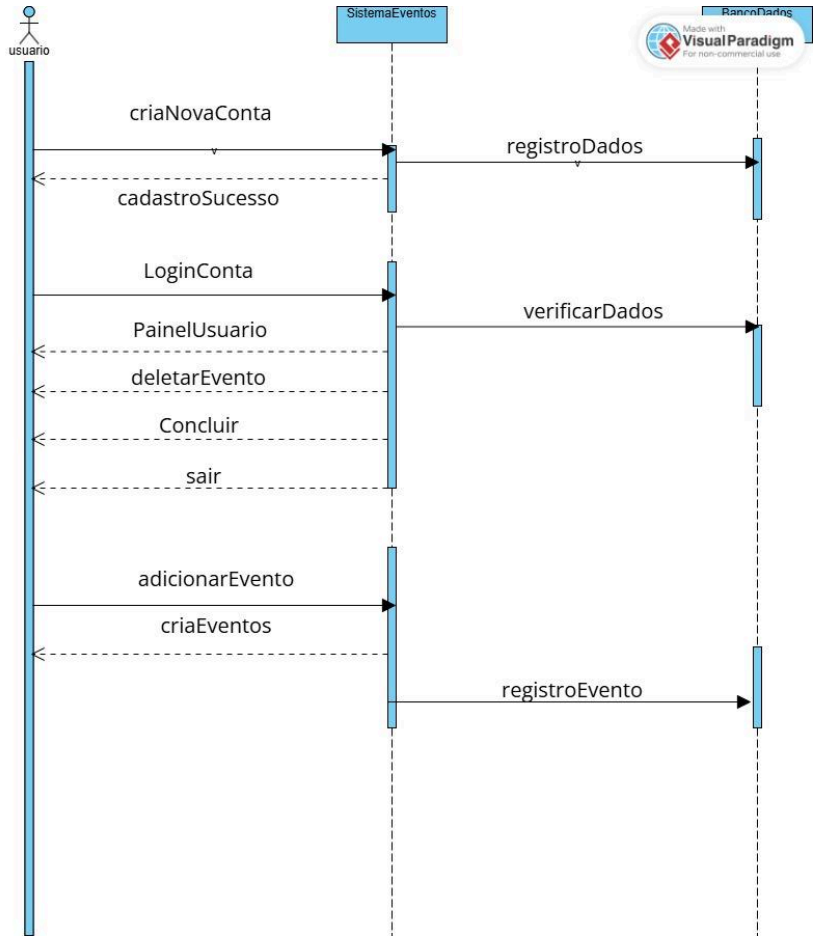




4.2 - DIAGRAMA CASO CLASSE



4.3 - DIAGRAMA DE SEQUÊNCIA



5. PROTÓTIPO FIGMA

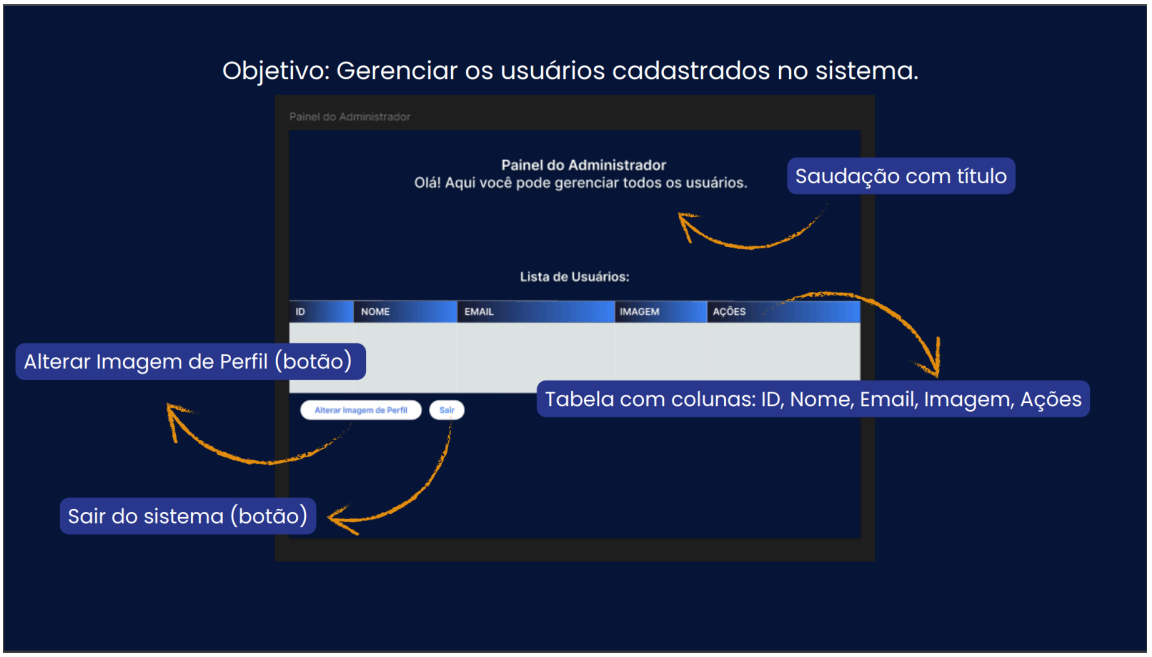
LINK FIGMA - PROTÓTIPOS

https://www.figma.com/design/Gp6elLgmmEiUWQVwLjBNsE/Prot%C3%B3tipo-Cadastro-Login-ADM?node-id=0-1&t=p584yhbH4uNu93aK-1

6. DOCUMENTAÇÃO TELAS

6.1.1 Página de Login do Sistema

6.1.2 Página do Painel do Usuário Administrador



6.1.3 Página do Painel do Usuário Comum

6.1.4 Página Alterar Imagem de Perfil

6.1.5 Página Criar Eventos Do Perfil Usuário Comum

6.1.6 Página de Cadastro de Usuário Comum ou Administrador

6.2 DOCUMENTAÇÃO DE TELA ESCRITA

6.2.1. Tela de Cadastro de Usuário

Objetivo: Permitir que novos usuários se cadastrem no sistema.

Campos:

- Nome (input de texto)
- Email (input de texto)
- Senha (input de senha)

- Imagem de Perfil (upload de arquivo)
- Administrador (checkbox)

Ação principal: Botão 'Cadastrar'

Navegação extra: Link para 'Fazer Login'

6.2.2 Tela de Cadastro de Evento

Objetivo: Permitir a criação de eventos pelos usuários.

Campos:

- Título (input de texto)
- Data (campo de data no formato DD/MM/AAAA)
- Descrição (textarea)

Ação principal: Botão 'Criar Evento'

6.2.3. Tela de Login

Objetivo: Acessar o sistema como usuário ou administrador.

Campos:

- Email
- Senha

Ação principal: Botão 'Login'

Navegação extra: Link para 'Cadastrar-se'

6.2.4. Painel do Administrador

Documentação das Telas do Sistema

Objetivo: Gerenciar os usuários cadastrados no sistema.

Componentes principais:

- Saudação com título
- Tabela com colunas: ID, Nome, Email, Imagem, Ações

Ações disponíveis:

- Alterar Imagem de Perfil (botão)
- Sair do sistema (botão)

6.2.5. Painel do Usuário

Objetivo: Visualizar e gerenciar os próprios eventos.

Componentes principais: - Saudação personalizada

- Botões:
- Adicionar Evento
- Alterar Imagem de Perfil
- Sair

Lista de eventos criados:

- Exibição do título, data e descrição
- Ações: Concluir | Excluir

6.2.6. Tela de Envio de Imagem de Perfil

Objetivo: Permitir o upload de nova imagem de perfil.

Componentes:

- Campo de upload de arquivo
- Botão 'Enviar Imagem'

7. ORÇAMENTOS

Hospedagem de site no plano HostGator

Descrição	Quantidade	Valor Mensal (R$)	Total
Hospedagem HostGator(mensal	1 Mês	28,99	28,99

Observações:

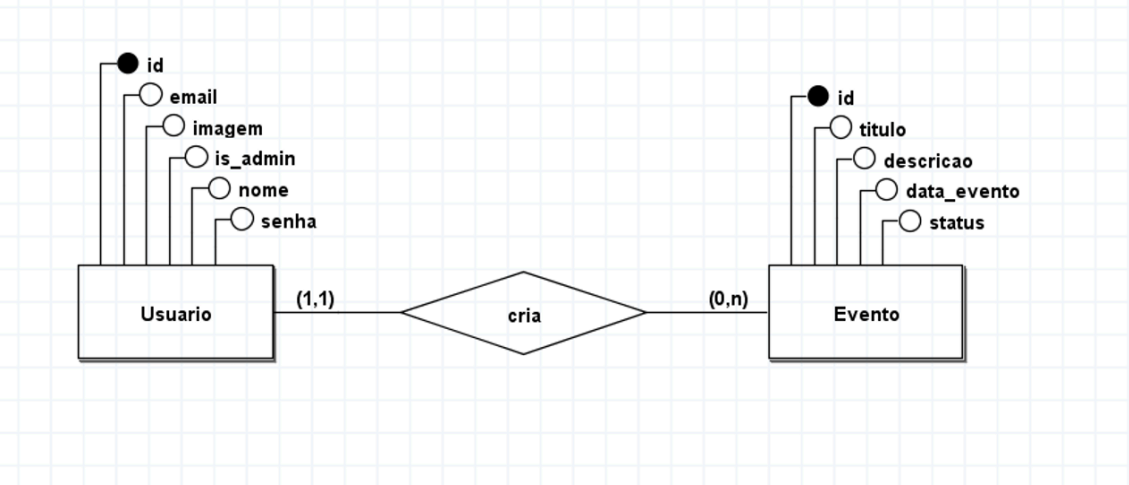
- O valor refere-se ao plano mensal de hospedagem fornecido pela empresa HostGator.

- Caso opte por planos trimestrais, semestrais ou anuais, o valor poderá ser ajustado com desconto proporcional.

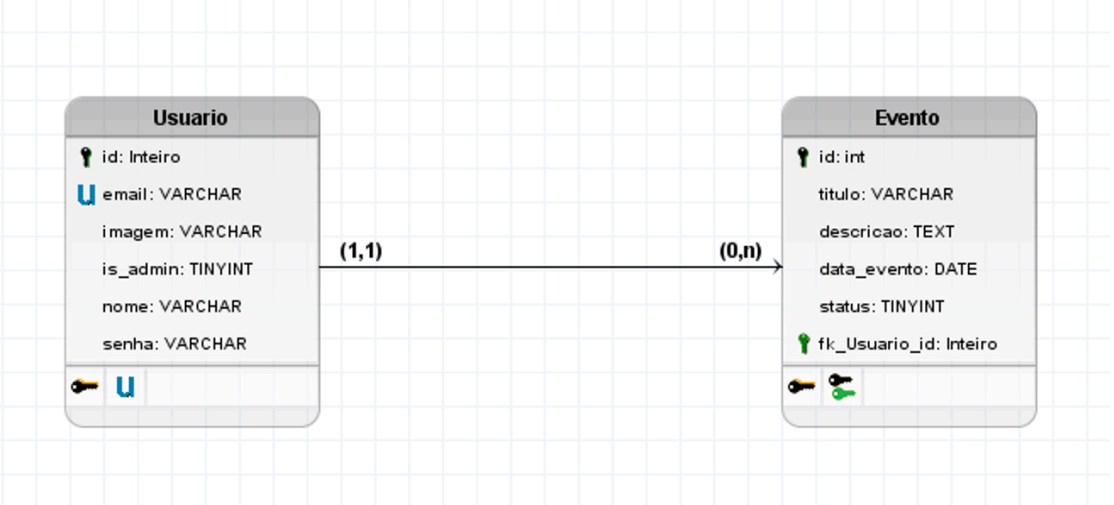
8. PROJETO BANCO DADOS

8.1 - CONCEITUAL

Modelos criado através do brModelo.

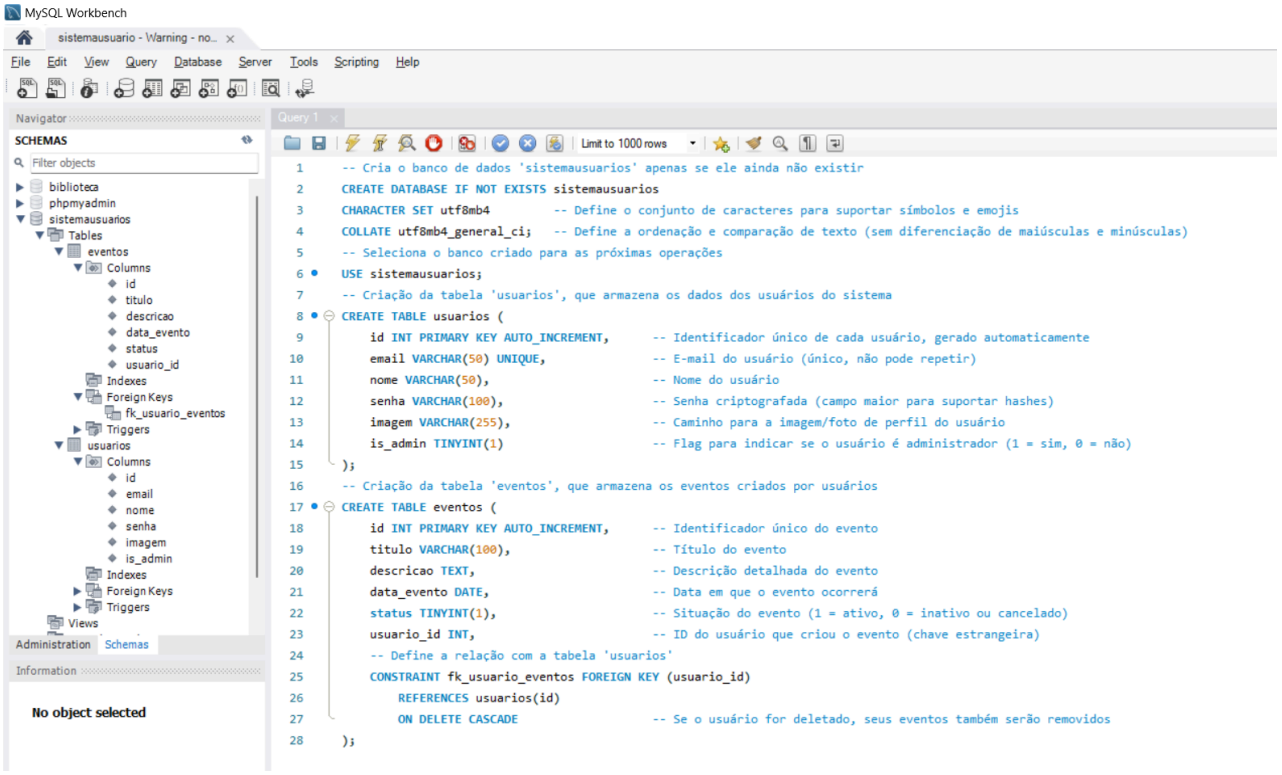


8.2 - LÓGICO



8.3 - FÍSICO

Criado através do MySQL Workbeanch.



9 PROJETO FUNCIONAL

9.1 - Código documentado na Linguagem De Programação utilizada

9.1.1 - Código Database.php Documentado

```php
<?php

// Define a classe Database para configurar a conexão com o banco de dados MySQL usando PDO.

class Database {

    private $host = "localhost"; // Endereço do servidor do banco de dados.

    private $db_name = "sistemausuarios"; // Nome do banco de dados.

    private $username = "root"; // Usuário do banco de dados.

    private $password = ""; // Senha do banco de dados.

    public $conn; // Declaração da variável que armazenará a conexão.

    public function getConnection() {

        // Função para criar e retornar a conexão com o banco de dados.

        $this->conn = null; // Inicializa a variável de conexão como nula.

        try {
```

```
            // Cria a conexão usando a classe PDO, passando o host, nome do banco, usuário e senha.
            $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db_name, $this->username, $this->password);
            $this->conn->exec("set names utf8"); // Define o conjunto de caracteres como UTF-8.
        } catch (PDOException $exception) {
            // Exibe uma mensagem de erro caso a conexão falhe.
            echo "Erro de conexão: " . $exception->getMessage();
        }
        return $this->conn; // Retorna a conexão criada.
    }
}
?>
```

9.1.2 - Código Usuario.php Documentado

```
<?php
// Inclui a conexão com o banco de dados
include_once 'Database.php';
class Usuario {
    // Conexão com o banco protegida (usada internamente pela classe)
    protected $conn;
    // Propriedades públicas do usuário
    public $nome;
    public $email;
    public $senha;
    public $imagem;
    public $is_admin = 0;
```

```php
    // Construtor: estabelece conexão com o banco de dados ao instanciar a classe
    public function __construct() {
        $database = new Database();
        $this->conn = $database->getConnection();
    }
    // Salva os dados do usuário no banco
    public function salvar() {
        $query = "INSERT INTO usuarios SET nome=:nome, email=:email, senha=:senha, imagem=:imagem, is_admin=:is_admin";
        $stmt = $this->conn->prepare($query);
        // Vincula os valores recebidos às variáveis da query
        $stmt->bindParam(":nome", $this->nome);
        $stmt->bindParam(":email", $this->email);
        $senhaHash = password_hash($this->senha, PASSWORD_DEFAULT); // Gera hash seguro da senha
        $stmt->bindParam(":senha", $senhaHash);
        $stmt->bindParam(":imagem", $this->imagem);
        $stmt->bindParam(":is_admin", $this->is_admin);
        return $stmt->execute(); // Executa o insert
    }
    // Realiza login a partir do e-mail e senha informados
    public function login($email, $senha) {
        $query = "SELECT * FROM usuarios WHERE email=:email";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(":email", $email);
        $stmt->execute();
```

```
        // Verifica se encontrou um usuário com o e-mail fornecido
        if ($stmt->rowCount() > 0) {
            $row = $stmt->fetch(PDO::FETCH_ASSOC);
            // Verifica se a senha confere com a hash armazenada
            if (password_verify($senha, $row['senha'])) {
                session_start();
                $_SESSION['usuario'] = $row['id'];
                $_SESSION['is_admin'] = isset($row['is_admin']) ?
(int)$row['is_admin'] : 0;
                return true;
            }
        }
        return false;
    }
    // Retorna HTML básico do painel do usuário
    public function getPainel() {
        return "<h3>Painel do Usuário</h3><p>Bem-vindo, {$this->nome}!</p>";
    }
    // Carrega os dados do usuário com base no ID
    public function carregarDados($id) {
        $query = "SELECT * FROM usuarios WHERE id = :id";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(":id", $id);
        $stmt->execute();
        // Se houver resultado, preenche os dados do objeto
```

```
        if ($stmt->rowCount() > 0) {
            $dados = $stmt->fetch(PDO::FETCH_ASSOC);
            $this->nome = $dados['nome'];
            $this->email = $dados['email'];
            $this->senha = $dados['senha']; // Atenção: senha já vem criptografada
            $this->imagem = $dados['imagem'];
            $this->is_admin = $dados['is_admin'];
        }
    }
    // Atualiza a imagem de perfil do usuário no banco de dados
public function atualizarImagem() {
    $query = "UPDATE usuarios SET imagem = :imagem WHERE email = :email";
    $stmt = $this->conn->prepare($query);
    $stmt->bindParam(":imagem", $this->imagem);
    $stmt->bindParam(":email", $this->email);
    return $stmt->execute();
}
    // Cria um novo evento associado ao usuário logado
    public function criarEvento($titulo, $data_evento, $descricao) {
        $usuario_id = $_SESSION['usuario'];
        $query = "INSERT INTO eventos (usuario_id, titulo, data_evento,
descricao)
                VALUES (:usuario_id, :titulo, :data_evento, :descricao)";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(":usuario_id", $usuario_id);
        $stmt->bindParam(":titulo", $titulo);
```

```
        $stmt->bindParam(":data_evento", $data_evento);
        $stmt->bindParam(":descricao", $descricao);
        return $stmt->execute();
    }

    // Retorna todos os eventos do usuário logado
    public function listarEventos() {
        $usuario_id = $_SESSION['usuario'];
        $query = "SELECT * FROM eventos WHERE usuario_id = :usuario_id
ORDER BY id DESC";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(":usuario_id", $usuario_id);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }
    // Marca o evento como concluído
    public function concluirEvento($evento_id) {
        $usuario_id = $_SESSION['usuario'];
        $query = "UPDATE eventos SET status = TRUE WHERE id = :evento_id
AND usuario_id = :usuario_id";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(":evento_id", $evento_id);
        $stmt->bindParam(":usuario_id", $usuario_id);
        return $stmt->execute();
    }
    // Deleta um evento do usuário
    public function deletarEvento($evento_id) {
```

```
        $usuario_id = $_SESSION['usuario'];
        $query = "DELETE FROM eventos WHERE id = :evento_id AND usuario_id = :usuario_id";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(":evento_id", $evento_id);
        $stmt->bindParam(":usuario_id", $usuario_id);
        return $stmt->execute();
    }
}
?>
```

9.1.3 - Código Admin.php Documentado

```
<?php
// Inclui a classe base 'Usuario', da qual Admin irá herdar
include_once 'Usuario.php';
// A classe Admin estende os comportamentos da classe Usuario
class Admin extends Usuario {
    // Sobrescreve o método getPainel para exibir uma mensagem personalizada para administradores
    public function getPainel() {
        return "<h3>Painel do Administrador</h3><p>Olá! Aqui você pode gerenciar todos os usuários.</p>";
    }
    // Retorna uma lista de todos os usuários do sistema
    public function listarUsuarios() {
        // Consulta o banco de dados para obter id, nome, email e imagem dos usuários
        $query = "SELECT id, nome, email, imagem FROM usuarios"; // A coluna 'genero' foi removida aqui propositalmente
```

```
        $stmt = $this->conn->prepare($query);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC); // Retorna os dados como um array associativo
    }
    // Deleta um usuário com base no ID fornecido
    public function deletarUsuario($id) {
        $query = "DELETE FROM usuarios WHERE id=:id"; // Define o comando SQL
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(":id", $id); // Substitui o parâmetro :id pelo valor recebido
        $resultado = $stmt->execute(); // Executa o comando

        // Verifica se o usuário deletado é o mesmo da sessão atual
        if ($resultado && isset($_SESSION['usuario']) && $_SESSION['usuario'] == $id) {
            // Encerra a sessão e redireciona para a página de login
            session_unset();
            session_destroy();
            header("Location: index.php");
            exit;
        }
        return $resultado; // Retorna true se a exclusão foi bem-sucedida, false caso contrário
    }
}
?>
```

9.1.4 - Código Index.php Documentado

```php
<?php

session_start(); // Inicia a sessão para verificar o login.

if (isset($_SESSION['usuario'])) { // Verifica se o usuário já está logado.

    header("Location: painel.php"); // Redireciona para o painel do usuário logado.

    exit; // Encerra o script.

}


include_once 'includes/Usuario.php'; // Inclui a classe Usuario para realizar operações de login.


if ($_SERVER['REQUEST_METHOD'] === 'POST') { // Verifica se o formulário foi enviado via método POST.

    $usuario = new Usuario(); // Cria uma nova instância da classe Usuario.

    $loginSuccess = $usuario->login($_POST['email'], $_POST['senha']); // Tenta autenticar o usuário usando os dados fornecidos.


    if ($loginSuccess) { // Se o login for bem-sucedido:

        header("Location: painel.php"); // Redireciona para o painel do usuário.

        exit; // Encerra o script.

    } else {

        echo "Credenciais inválidas!"; // Exibe uma mensagem de erro caso as credenciais estejam incorretas.

    }

}

?>

<!DOCTYPE html> <!-- Declaração do tipo de documento HTML5 -->
```

```
<html lang="pt-br"> <!-- Define o idioma da página como português do Brasil -->
<head>
   <meta charset="UTF-8"> <!-- Define o conjunto de caracteres como UTF-8 -->
   <title>Login</title> <!-- Título da página que aparece na aba do navegador -->
   <link rel="stylesheet" href="css/style.css">
</head>
<body>
   <h2>Login</h2> <!-- Cabeçalho da página -->
   <form method="POST" action="">
      <!-- Formulário para login, enviando os dados para o próprio arquivo index.php via POST -->
      <input type="email" name="email" placeholder="Email" required> <!-- Campo para inserir o email -->
      <input type="password" name="senha" placeholder="Senha" required> <!-- Campo para inserir a senha -->
      <button type="submit">Login</button> <!-- Botão para enviar o formulário -->
   </form>
   <p>Não tem uma conta? <a href="cadastro.html">Cadastre-se</a></p>
</body>
</html>
```

9.1.5 - Código Painel.php Documentado

```
<?php
session_start(); // Inicia a sessão para verificar se o usuário está logado
// Redireciona para o login se não houver usuário autenticado
if (!isset($_SESSION['usuario'])) {
```

```php
    header("Location: index.php");
    exit;
}
// Inclui as classes necessárias para manipulação de usuários e admins
include_once 'includes/Usuario.php';
include_once 'includes/Admin.php';
// Verifica se o usuário logado é um administrador
if ($_SESSION['is_admin'] == 1) {
    $usuario = new Admin(); // Se for admin, instancia a classe Admin
} else {
    $usuario = new Usuario(); // Se não for, instancia a classe Usuario
}
// Carrega os dados do usuário logado a partir do ID da sessão
$usuario->carregarDados($_SESSION['usuario']);
// Exibe as informações do painel (mensagem de boas-vindas, nome, etc.)
echo $usuario->getPainel();
// Se for administrador e foi enviado um ID para deletar, executa a remoção
if ($_SESSION['is_admin'] == 1 && isset($_GET['delete_id'])) {
    $usuario->deletarUsuario($_GET['delete_id']); // Deleta o usuário com o ID especificado
    header("Location: painel.php"); // Recarrega a página após a exclusão
    exit;
}
// Exibição específica para administradores: lista completa de usuários
if ($usuario instanceof Admin) {
    $usuarios = $usuario->listarUsuarios(); // Recupera todos os usuários
```

```
    echo '<h3>Lista de Usuários:</h3>';
    echo '<table border="1">';
    echo '<tr><th>ID</th><th>Nome</th><th>Email</th><th>Imagem</th><th>Ações</th></tr>';

    // Loop para exibir cada usuário na tabela
    foreach ($usuarios as $u) {
        echo '<tr>';
        echo '<td>' . $u['id'] . '</td>';
        echo '<td>' . $u['nome'] . '</td>';
        echo '<td>' . $u['email'] . '</td>';
        echo '<td>';
        // Exibe imagem se houver, ou mensagem "Sem imagem"
        echo $u['imagem'] ? '<img src="uploads/' . $u['imagem'] . '" alt="Imagem de Perfil" width="50">' : 'Sem imagem';
        echo '</td>';
        echo '<td>';
        // Botão para deletar o usuário com confirmação
        echo '<a href="painel.php?delete_id=' . $u['id'] . '" onclick="return confirm(\'Tem certeza que deseja deletar este usuário?\');">Deletar</a>';
        echo '</td>';
        echo '</tr>';
    }
    echo '</table>';
} else {
    // Para usuários comuns, exibe eventos
```

```
    echo '<h3>Seus Eventos</h3>';
    echo '<a href="eventos.php?action=criar">Adicionar Evento</a>'; // Link para criar novo evento
    // Verifica se o método listarEventos existe (pode depender da classe)
    if (method_exists($usuario, 'listarEventos')) {
        $eventos = $usuario->listarEventos(); // Obtém os eventos do usuário
    } else {
        echo "Erro: Método listarEventos não existe na classe " . get_class($usuario);
    }
    // Loop para exibir cada evento do usuário com cor de fundo dependendo do status
    foreach ($eventos as $evento) {
        $corFundo = $evento['status'] ? '#e6f9e6' : '#ffe6e6'; // Verde para concluído, vermelho para pendente
        echo "<div style='background-color: $corFundo; padding: 10px; margin-bottom: 10px; border-radius: 5px;'>";
        echo "<strong>{$evento['titulo']}</strong> - {$evento['data_evento']}<br>{$evento['descricao']}";
        echo "<br><a href='eventos.php?action=concluir&id={$evento['id']}'>Concluir</a>";
        echo " | <a href='eventos.php?action=deletar&id={$evento['id']}'>Excluir</a>";
        echo "</div>";
    }
}
?>
<!DOCTYPE html>
<html lang="pt-BR">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Painel do Usuário</title>
    <link rel="stylesheet" href="css/style.css"> <!-- Link para o CSS externo -->
</head>
<body>
    <!-- Navegação do painel -->
    <a href="upload.php">Alterar Imagem de Perfil</a>
    <a href="logout.php">Sair</a>
</body>
</html>
```

9.1.6 - Código Eventos.php Documentado

```
<?php
session_start(); // Inicia a sessão para controlar e acessar dados do usuário logado
include_once 'includes/Usuario.php'; // Inclui a classe que manipula os dados e métodos do usuário
// Verifica se o usuário está logado; se não, redireciona para a página de login
if (!isset($_SESSION['usuario'])) {
    header("Location: index.php"); // Redireciona para login se não estiver autenticado
    exit; // Encerra a execução do script
}
// Cria um novo objeto do tipo Usuario
$usuario = new Usuario();
// Carrega os dados do usuário logado (possivelmente nome, email, ID, etc.)
```

```php
$usuario->carregarDados($_SESSION['usuario']);

// Verifica se a ação foi enviada via GET, como ?action=criar

if (!isset($_GET['action'])) {

    echo "Erro: Nenhuma ação foi especificada."; // Exibe erro se nenhuma ação for informada

    exit;

}

$action = $_GET['action']; // Captura a ação da URL

$dataHoje = date('Y-m-d'); // Armazena a data atual (usada no input de data e validação)

?>

<!DOCTYPE html>

<html lang="pt-BR">

<head>

    <meta charset="UTF-8"> <!-- Suporte a caracteres UTF-8 -->

    <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- Responsividade -->

    <title>Eventos</title>

    <link rel="stylesheet" href="css/style.css"> <!-- Aplica estilo visual externo -->

</head>

<body>

    <div class="container">

        <h2>Eventos</h2>

        <?php

        // Controla o fluxo com base no valor de $action

        switch ($action) {
```

```
        case 'criar': // Se o usuário estiver criando um evento
            if ($_SERVER['REQUEST_METHOD'] === 'POST') { // Verifica se o formulário foi enviado
                $titulo = $_POST['titulo'];
                $data_evento = $_POST['data_evento'];
                $descricao = $_POST['descricao'];
                // Validação da data: não pode ser anterior à atual
                if ($data_evento < date('Y-m-d')) {
                    echo "<div class='message error'>A data do evento não pode ser anterior à data atual.</div>";
                    exit;
                }
                // Tenta criar o evento e redireciona em caso de sucesso
                if ($usuario->criarEvento($titulo, $data_evento, $descricao)) {
                    echo "<div class='message success'>Evento criado com sucesso!</div>";
                    header("Location: painel.php");
                    exit;
                } else {
                    echo "<div class='message error'>Erro ao criar evento.</div>"; // Erro na criação
                }
            } else {
                // Exibe o formulário se ainda não enviado
                echo '<form action="eventos.php?action=criar" method="POST">
                        <label for="titulo">Título:</label>
                        <input type="text" name="titulo" required><br>
```

```
                    <label for="data_evento">Data:</label>
                    <input type="date" name="data_evento" min="' . $dataHoje . '" required><br>

                    <label for="descricao">Descrição:</label>
                    <textarea name="descricao" required></textarea><br>

                    <button type="submit">Criar Evento</button>
                </form>';
            }
            break;
                case 'listar': // Exibe todos os eventos cadastrados pelo usuário
            $eventos = $usuario->listarEventos(); // Busca todos os eventos do banco de dados
            echo "<h3>Lista de Eventos</h3>";
            foreach ($eventos as $evento) {
                // Mostra o título, data e descrição de cada evento com estilo conforme o status
                echo "<div class='evento " . ($evento['status'] ? "success" : "error") . "'>";
                echo "<strong>{$evento['titulo']}</strong> - {$evento['data_evento']}<br>{$evento['descricao']}";
                // Adiciona links para concluir ou excluir o evento
                echo "<br><a href='eventos.php?action=concluir&id={$evento['id']}'>Concluir</a>";
                echo " | <a href='eventos.php?action=deletar&id={$evento['id']}'>Excluir</a>";
```

```
                echo "</div>";
            }
            break;
        case 'concluir': // Marca um evento como concluído
            if (isset($_GET['id'])) { // Verifica se o ID foi passado pela URL
                if ($usuario->concluirEvento($_GET['id'])) {
                    echo "<div class='message success'>Evento concluído!</div>";
                    header("Location: painel.php"); // Redireciona para o painel após sucesso
                    exit;
                } else {
                    echo "<div class='message error'>Erro ao concluir evento.</div>";
                }
            }
            break;
        case 'deletar': // Exclui um evento
            if (isset($_GET['id'])) {
                if ($usuario->deletarEvento($_GET['id'])) {
                    echo "<div class='message success'>Evento excluído!</div>";
                    header("Location: painel.php"); // Redireciona após exclusão
                    exit;
                } else {
                    echo "<div class='message error'>Erro ao excluir evento.</div>";
                }
            }
            break;
```

```
        default: // Caso a ação passada não seja reconhecida
            echo "<div class='message error'>Erro: Ação inválida.</div>";
    }
    ?>
  </div>
</body>
</html>
```

9.1.7 - Código Cadastro.php Documentado

```
<?php
// Inclui o arquivo responsável pela conexão com o banco de dados
include_once 'includes/Database.php';
// Inclui a classe que representa um usuário do sistema
include_once 'includes/Usuario.php';
// Verifica se o formulário foi enviado via método POST
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  // Cria um novo objeto do tipo Usuario
  $usuario = new Usuario();
  // Define as propriedades do usuário com os dados recebidos do formulário
  $usuario->nome = $_POST['nome'];
  $usuario->email = $_POST['email'];
  $usuario->senha = $_POST['senha'];
  $usuario->is_admin = isset($_POST['is_admin']) ? 1 : 0; // Admin = 1, comum
= 0
  // Verifica se uma imagem foi enviada no formulário
  if (!empty($_FILES['imagem']['name'])) {
    $targetDir = "uploads/"; // Pasta onde as imagens são armazenadas
```

```php
        $fileName = basename($_FILES['imagem']['name']); // Nome do arquivo de imagem
        $targetFilePath = $targetDir . $fileName; // Caminho completo do destino do arquivo
        // Move o arquivo enviado para a pasta de uploads
        if (move_uploaded_file($_FILES['imagem']['tmp_name'], $targetFilePath)) {
            $usuario->imagem = $fileName; // Atribui o nome da imagem ao usuário
        } else {
            // Se não conseguir mover a imagem, exibe erro e encerra
            echo "Erro ao fazer upload da imagem.";
            exit;
        }
    }
    // Tenta salvar o usuário no banco de dados
    if ($usuario->salvar()) {
        // Redireciona para a página de sucesso após cadastro bem-sucedido
        header("Location: sucesso.php");
        exit;
    } else {
        // Em caso de erro no cadastro, exibe mensagem com estilo
        ?>
        <!DOCTYPE html>
        <html lang="pt-BR">
        <head>
            <meta charset="UTF-8">
            <title>Erro no Cadastro</title>
            <link rel="stylesheet" href="css/style.css">
```

```
    </head>
    <body>
        <div class="container">
            <h2 class="message error">Erro ao cadastrar usuário. Verifique os dados e tente novamente.</h2>
            <p><a href="cadastro.html">Voltar para o cadastro</a></p>
        </div>
    </body>
    </html>
    <?php
    exit;
  }
}
?>
```

9.1.8 - Código Cadastro.html Documentado

```
<!DOCTYPE html> <!-- Declaração do tipo de documento HTML5 -->
<html lang="pt-br"> <!-- Define o idioma da página como português do Brasil -->
<head>
  <meta charset="UTF-8"> <!-- Define o conjunto de caracteres como UTF-8 -->
  <title>Cadastro de Usuário</title> <!-- Título da página que aparece na aba do navegador -->
  <link rel="stylesheet" href="css/style.css"> <!-- Link para o arquivo de estilos -->
  <script>
    // Validação extra de e-mail no lado do cliente (opcional, reforço)
    document.addEventListener("DOMContentLoaded", () => {
      const form = document.querySelector("form");
```

```
        form.addEventListener("submit", (e) => {
            const email = document.getElementById("email").value;
            const regexEmail = /^[^\s@]+@[^\s@]+\.(com)$/i; // Padrão básico para e-mail válido
            if (!regexEmail.test(email)) {
                alert("Por favor, insira um e-mail válido.");
                e.preventDefault(); // Impede o envio do formulário
            }
        });
    });
  </script>
</head>
<body>
  <h2>Cadastro de Usuário</h2> <!-- Cabeçalho principal da página -->
  <form action="cadastro.php" method="POST" enctype="multipart/form-data">
    <!-- Formulário para enviar dados de cadastro para cadastro.php com suporte para upload de arquivos -->
    <label for="nome">Nome:</label> <!-- Rótulo para o campo do nome -->
    <input type="text" name="nome" id="nome" required><br> <!-- Campo de texto para o nome do usuário -->
    <label for="email">Email:</label> <!-- Rótulo para o campo do email -->
    <input type="email" name="email" id="email" required><br> <!-- Campo de entrada para o e-mail (já tem validação HTML) -->
    <label for="senha">Senha:</label> <!-- Rótulo para o campo da senha -->
    <input type="password" name="senha" id="senha" required><br> <!-- Campo de entrada da senha -->
```

```
        <label for="imagem">Imagem de Perfil:</label> <!-- Rótulo para o campo de upload de imagem -->
        <input type="file" name="imagem" id="imagem"><br> <!-- Campo de seleção de imagem de perfil -->
        <label for="is_admin">Administrador:</label> <!-- Rótulo para o checkbox de status admin -->
        <input type="checkbox" name="is_admin" id="is_admin"><br> <!-- Checkbox: se marcado, define o usuário como admin -->
        <button type="submit">Cadastrar</button> <!-- Botão de envio do formulário -->
    </form>
    <p>Já tem uma conta? <a href="index.php">Fazer Login</a></p> <!-- Link para a tela de login -->
</body>
</html>
```

9.1.9 - Código Upload.php Documentado

```
<?php
session_start(); // Inicia a sessão para garantir que o usuário está logado
// Se o usuário não estiver logado, redireciona para a página de login
if (!isset($_SESSION['usuario'])) {
    header("Location: index.php");
    exit;
}
include_once 'includes/Usuario.php'; // Inclui a classe Usuario
$usuario = new Usuario(); // Instancia um novo objeto Usuario
$usuario->carregarDados($_SESSION['usuario']); // Carrega os dados do usuário logado
$mensagem = ""; // Variável para mensagens de sucesso ou erro
```

```php
// Verifica se o formulário foi enviado e se o campo de imagem está presente
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['imagem'])) {
    $targetDir = "uploads/"; // Diretório de destino
    $fileName = basename($_FILES['imagem']['name']);
    $targetFilePath = $targetDir . $fileName;
    // Move o arquivo e atualiza o banco
    if (move_uploaded_file($_FILES['imagem']['tmp_name'], $targetFilePath)) {
        $usuario->imagem = $fileName;
        // Verifica se existe o método atualizarImagem
        if (method_exists($usuario, 'atualizarImagem') &&
$usuario->atualizarImagem()) {
            $mensagem = '<div class="message success">Imagem enviada e
atualizada com sucesso.</div>';
        } else {
            $mensagem = '<div class="message error">Erro ao atualizar a imagem
no banco de dados.</div>';
        }
    } else {
        $mensagem = '<div class="message error">Erro ao fazer upload do
arquivo.</div>';
    }
}
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
```

```
    <title>Upload de Imagem</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="container">
        <h2>Envio de Imagem de Perfil</h2>
        <?= $mensagem ?>
        <?php if (!empty($usuario->imagem)): ?>
            <p>Imagem atual:</p>
            <img src="uploads/<?= htmlspecialchars($usuario->imagem) ?>"
alt="Imagem atual" width="120"><br><br>
        <?php endif; ?>
        <form method="POST" enctype="multipart/form-data">
            <label for="imagem">Selecione uma nova imagem:</label>
            <input type="file" name="imagem" id="imagem" required>
            <button type="submit">Enviar Imagem</button>
        </form>
        <p><a href="painel.php">← Voltar para o Painel</a></p>
    </div>
</body>
</html>
```

9.1.10 - Código Sucesso.php Documentado

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>Cadastro Bem-Sucedido</title>
```

```
    <!-- Link para a folha de estilos principal do sistema -->
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="container">
        <!-- Mensagem de confirmação com classe de sucesso estilizada -->
        <h2 class="message success">Usuário cadastrado com sucesso!</h2>
        <!-- Link de navegação para voltar à tela de login -->
        <p><a href="index.php">Clique aqui para fazer login</a></p>
    </div>
</body>
</html>
```

9.1.11 - Código Logout.php Documentado

```
<?php
session_start(); // Inicia a sessão para poder destruí-la.
session_destroy(); // Destroi todos os dados da sessão.
header("Location: index.php"); // Redireciona o usuário para a página de login.
exit; // Encerra o script.
?>
```

9.1.12 - Código Styles.css Documentado

```
/* Reset e configurações básicas */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

```css
body {
    font-family: 'Inter', 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background:#071738;
    min-height: 100vh;
    color: #333;
    line-height: 1.6;
    overflow-x: hidden;
}

/* Container principal */
.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
}

/* Títulos */
h2, h3 {
    color: #fff;
    text-align: center;
    margin-bottom: 30px;
    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.3);
}

h2 {
    font-size: 28px;
```

```
    margin-top: 40px;
}

h3 {
    font-size: 22px;
    margin-top: 30px;
    margin-bottom: 20px;
}

/* Formulários */
form {
    background-color: #ffffff;
    padding: 40px;
    border-radius: 15px;
    box-shadow: 0 15px 35px rgba(0, 0, 0, 0.1);
    margin: 20px auto;
    max-width: 500px;
}

/* Labels */
label {
    display: block;
    margin-bottom: 8px;
    font-weight: bold;
    color: #555;
    font-size: 14px;
```

```css
}

/* Inputs */
input[type="text"],
input[type="email"],
input[type="password"],
input[type="date"],
input[type="file"],
textarea {
    width: 100%;
    padding: 12px 15px;
    border: 2px solid #e1e5e9;
    border-radius: 8px;
    font-size: 16px;
    margin-bottom: 20px;
    transition: all 0.3s ease;
    background-color: #f8f9fa;
}

input[type="text"]:focus,
input[type="email"]:focus,
input[type="password"]:focus,
input[type="date"]:focus,
input[type="file"]:focus,
textarea:focus {
    border-color: #667eea;
```

```
    outline: none;
    background-color: #fff;
    box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}
/* Textarea específico */
textarea {
    min-height: 100px;
    resize: vertical;
}
/* Checkbox */
input[type="checkbox"] {
    margin-right: 10px;
    transform: scale(1.2);
}
/* Botões */
button,
input[type="submit"] {
    width: 100%;
    padding: 18px 24px;
    background: linear-gradient(135deg, #1a1a2e 0%, #3b82f6 100%);
    color: white;
    border: none;
    border-radius: 12px;
    font-size: 16px;
    font-weight: 600;
    cursor: pointer;
```

```
    transition: all 0.3s ease;
    margin-top: 10px;
    position: relative;
    overflow: hidden;
}

button:hover,
input[type="submit"]:hover {
    transform: translateY(-3px);
    box-shadow: 0 15px 35px rgba(59, 130, 246, 0.4);
}

/* Links */
a {
    color: #667eea;
    text-decoration: none;
    font-weight: 500;
    transition: color 0.3s ease;
}
a:hover {
    color: #764ba2;
    text-decoration: underline;
}
/* Parágrafos com links */
p {
    text-align: center;
```

```
    margin-top: 20px;
    color: #fff;
    text-shadow: 0 1px 2px rgba(0, 0, 0, 0.3);
}

p a {
    color: #fff;
    background-color: rgba(255, 255, 255, 0.2);
    padding: 8px 16px;
    border-radius: 20px;
    transition: all 0.3s ease;
}

p a:hover {
    background-color: rgba(255, 255, 255, 0.3);
    text-decoration: none;
}

/* Mensagens de feedback */
.message {
    padding: 15px 20px;
    border-radius: 8px;
    margin: 20px 0;
    font-weight: 500;
    text-align: center;
}
```

```css
.message.success {
    background-color: #d4edda;
    color: #155724;
    border: 1px solid #c3e6cb;
}
.message.error {
    background-color: #f8d7da;
    color: #721c24;
    border: 1px solid #f5c6cb;
}
/* Eventos */
.evento {
    background-color: #ffffff;
    padding: 20px;
    margin-bottom: 15px;
    border-radius: 10px;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
    border-left: 4px solid #667eea;
    transition: transform 0.3s ease;
}
.evento:hover {
    transform: translateY(-2px);
}
.evento.success {
    border-left-color: #28a745;
    background-color: #f8fff9;
```

```
}
.evento.error {
    border-left-color: #dc3545;
    background-color: #fff8f8;
}
.evento strong {
    color: #333;
    font-size: 18px;
}
.evento a {
    display: inline-block;
    margin-right: 15px;
    padding: 5px 12px;
    background-color: #667eea;
    color: white;
    border-radius: 5px;
    font-size: 14px;
    transition: background-color 0.3s ease;
}
.evento a:hover {
    background-color: #764ba2;
    text-decoration: none;
}
/* Tabelas */
table {
    width: 100%;
```

```
    background-color: #ffffff;
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.1);
    margin: 20px 0;
}
th, td {
    padding: 15px;
    text-align: left;
    border-bottom: 1px solid #e9ecef;
}

th {
    background: linear-gradient(135deg, #1a1a2e 0%, #3b82f6 100%);
    color: white;
    font-weight: 600;
    text-transform: uppercase;
    font-size: 14px;
    letter-spacing: 0.5px;
}
tr:hover {
    background-color: #f8f9fa;
}
td img {
    border-radius: 50%;
    border: 2px solid #e9ecef;
```

```css
}
/* Links de navegação no painel */
body > a {
    display: inline-block;
    margin: 10px 20px;
    padding: 12px 24px;
    background-color: #ffffff;
    color: #667eea;
    border-radius: 25px;
    font-weight: bold;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
    transition: all 0.3s ease;
}
body > a:hover {
    transform: translateY(-2px);
    box-shadow: 0 8px 25px rgba(102, 126, 234, 0.2);
    text-decoration: none;
}
/* Responsividade */
@media (max-width: 768px) {
    .container {
        padding: 10px;
    }
    form {
        padding: 30px 20px;
        margin: 10px;
```

```
    }

    h2 {
        font-size: 24px;
        margin-top: 20px;
    }

    table {
        font-size: 14px;
    }

    th, td {
        padding: 10px 8px;
    }
    .evento {
        padding: 15px;
    }
    body > a {
        display: block;
        margin: 10px;
        text-align: center;
    }
}
@media (max-width: 480px) {
    form {
        padding: 20px 15px;
```

```
    }
    input[type="text"],
    input[type="email"],
    input[type="password"],
    input[type="date"],
    input[type="file"],
    textarea {
        font-size: 16px; /* Evita zoom no iOS */
    }
    table {
        font-size: 12px;
    }
    .evento a {
        display: block;
        margin: 5px 0;
        text-align: center;
    }
}
/* Animações */
@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(20px);
    }
    to {
        opacity: 1;
```

```
        transform: translateY(0);
    }
}
form, .evento, table, .message {
    animation: fadeIn 0.6s ease-out;
}
/* Melhorias visuais adicionais */
.container > h3 + a {
    display: inline-block;
    margin-bottom: 20px;
    padding: 10px 20px;
    background: linear-gradient(135deg, #28a745 0%, #20c997 100%);
    color: white;
    border-radius: 25px;
    font-weight: bold;
    transition: all 0.3s ease;
}
.container > h3 + a:hover {
    transform: translateY(-2px);
    box-shadow: 0 6px 20px rgba(40, 167, 69, 0.3);
    text-decoration: none;
}
```

9.2 - CÓDIGO BANCO SQL – DOCUMENTADO.

Todos os trechos de documentação do código são apresentados com comentários destacados em negrito, a fim de melhorar a compreensão das instruções SQL. Essa formatação foi adotada para dar clareza visual e não compromete os critérios formais das normas da ABNT.

```
– Cria o banco de dados 'sistemausuarios' apenas se ele ainda não existir
CREATE DATABASE IF NOT EXISTS sistemausuarios
CHARACTER SET utf8mb4          -- Define o conjunto de caracteres para suportar símbolos e emojis
COLLATE utf8mb4_general_ci;   -- Define a ordenação e comparação de texto (sem diferenciação de maiúsculas e minúsculas)
– 4 Seleciona o banco criado para as próximas operações
USE sistemausuarios;
-- Criação da tabela 'usuarios', que armazena os dados dos usuários do sistema
CREATE TABLE usuarios (
    id INT PRIMARY KEY AUTO_INCREMENT,      – Identificador único de cada usuário, gerado automaticamente
    email VARCHAR(50) UNIQUE,   -- E-mail do usuário (único, não pode repetir)
    nome VARCHAR(50),   -- Nome do usuário
    senha VARCHAR(100),   -- Senha criptografada (campo maior para suportar hashes)
    imagem VARCHAR(255),   -- Caminho para a imagem/foto de perfil do usuário
    is_admin TINYINT(1)   -- Flag para indicar se o usuário é administrador (1 = sim, 0 = não)
);
-- Criação da tabela 'eventos', que armazena os eventos criados por usuários
CREATE TABLE eventos (
    id INT PRIMARY KEY AUTO_INCREMENT,   -- Identificador único do evento
    titulo VARCHAR(100),   -- Título do evento
    descricao TEXT,   -- Descrição detalhada do evento
    data_evento DATE,   -- Data em que o evento ocorrerá
    status TINYINT(1),   -- Situação do evento (1 = ativo, 0 = inativo ou cancelado)
    usuario_id INT,   -- ID do usuário que criou o evento (chave estrangeira)
    -- Define a relação com a tabela 'usuarios'
    CONSTRAINT fk_usuario_eventos FOREIGN KEY (usuario_id)
        REFERENCES usuarios(id)
```

ON DELETE CASCADE **-- Se o usuário for deletado, seus eventos também serão removidos**

);

10. LINK GITHUB

https://github.com/PedroAraujoo7/ProjetoModel