

Trabalho em PHP

1) Modelagem de dados

-- Criação do banco de dados se não existir

```
CREATE DATABASE IF NOT EXISTS autenticacao;  
USE autenticacao;
```

-- Criação da tabela `usuarios`

```
CREATE TABLE IF NOT EXISTS usuarios (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    email VARCHAR(100) NOT NULL UNIQUE,  
    senha_hash VARCHAR(255) NOT NULL,  
    verificado TINYINT(1) DEFAULT 0  
);
```

-- Criação da tabela `codigos_verificacao`

```
CREATE TABLE IF NOT EXISTS codigos_verificacao (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    usuario_id INT NOT NULL,  
    codigo VARCHAR(6) NOT NULL,  
    criado_em TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE  
);
```

-- Criação da tabela `codigos_redefinicao`

```
CREATE TABLE IF NOT EXISTS codigos_redefinicao (  
    usuario_id INT PRIMARY KEY,  
    codigo VARCHAR(6),  
    criado_em TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE  
);
```

-- Criação da tabela `questoes` com a coluna `assunto`

```
CREATE TABLE IF NOT EXISTS questoes (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    enunciado TEXT NOT NULL,  
    alternativa1 VARCHAR(255) NOT NULL,  
    alternativa2 VARCHAR(255) NOT NULL,  
    alternativa3 VARCHAR(255) NOT NULL,  
    alternativa4 VARCHAR(255) NOT NULL,  
    alternativa5 VARCHAR(255) NOT NULL,  
    alternativa_correta INT NOT NULL,  
    assunto VARCHAR(100) NOT NULL
```

);

-- Inserção de dados na tabela `questoes`

```
INSERT INTO `questoes` (`id`, `enunciado`, `alternativa1`, `alternativa2`,  
`alternativa3`, `alternativa4`, `alternativa5`, `alternativa_correta`, `assunto`) VALUES  
(1, 'Qual das operações de CRUD é usada para adicionar um novo registro ao  
banco de dados?', 'Atualizar', 'Excluir', 'Ler', 'Criar', 'Modificar', 4, 'PHP - CRUD'),  
(2, 'Em um sistema CRUD, qual operação permite visualizar dados armazenados  
sem alterá-los?', 'Atualizar', 'Ler', 'Excluir', 'Criar', 'Modificar', 2, 'PHP - CRUD'),  
(3, 'Qual é a função do comando SQL DELETE em operações CRUD?', 'Atualizar  
registros', 'Exibir registros', 'Deletar registros', 'Criar registros', 'Editar registros', 3,  
'PHP - CRUD'),  
(4, 'Qual das operações de CRUD geralmente envolve o uso do comando SQL  
UPDATE?', 'Ler', 'Excluir', 'Atualizar', 'Criar', 'Adicionar', 3, 'PHP - CRUD'),  
(5, 'Qual é a principal função de session_start() em PHP?', 'Criar uma nova variável  
de sessão.', 'Iniciar uma nova sessão ou retomar uma sessão existente.', 'Enviar  
informações de sessão para o navegador.', 'Validar a autenticidade do usuário  
logado.', 'Enviar dados do servidor para o cliente.', 2, 'PHP_SESSION'),  
(6, 'O que acontece se session_start() for chamado após enviar dados para o  
navegador em PHP?', 'A sessão será iniciada corretamente, sem problemas.', 'O  
PHP ignora a função session_start() e a sessão não é criada.', 'A sessão será  
criada, mas o navegador não receberá cookies.', 'Um erro será gerado e o script  
será interrompido.', 'O servidor automaticamente gera um novo cookie de sessão.',  
4, 'PHP_SESSION');
```

```
30 • ○ CREATE TABLE `codigos_redefinicao` (  
31     `usuario_id` int(11) NOT NULL,  
32     `codigo` varchar(6) DEFAULT NULL,  
33     `criado_em` timestamp NOT NULL DEFAULT current_timestamp()  
34 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
35
```

```
42 • ○ CREATE TABLE `codigos_verificacao` (  
43     `id` int(11) NOT NULL,  
44     `usuario_id` int(11) NOT NULL,  
45     `codigo` varchar(6) NOT NULL,  
46     `criado_em` timestamp NOT NULL DEFAULT current_timestamp()  
47 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```

55 • CREATE TABLE `questoes` (
56     `id` int(11) NOT NULL,
57     `enunciado` text NOT NULL,
58     `alternativa1` varchar(255) NOT NULL,
59     `alternativa2` varchar(255) NOT NULL,
60     `alternativa3` varchar(255) NOT NULL,
61     `alternativa4` varchar(255) NOT NULL,
62     `alternativa5` varchar(255) NOT NULL,
63     `alternativa_correta` int(11) NOT NULL,
64     `assunto` varchar(100) NOT NULL
65 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

73 • CREATE TABLE `usuarios` (
74     `id` int(11) NOT NULL,
75     `email` varchar(100) NOT NULL,
76     `senha_hash` varchar(255) NOT NULL,
77     `verificado` tinyint(1) DEFAULT 0
78 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

```

2) Os arquivos

2.1) A pasta *src*

- É necessário ter a pasta *src* com os códigos em php, pois se fosse para fazer os códigos a mão, seriam muito demorados pois são mais de 1.500 linhas de código e também é disponível pelos próprios desenvolvedores do PHP, ou seja, são confiáveis. São necessários para conseguir enviar os emails e receber os códigos. Na pasta *src*, existem 7 arquivos, sendo eles:

- *DSNConfigurator.php* - é usado para configurar a conexão com o banco de dados usando o DSN, que é o *Data Source Name*. Ele contém informações sobre como se conectar com uma base de dados específica, no caso o banco de dados do Sistema de Autenticação de 2 Fatores.
- *Exception.php* - é usado para lidar com exceções, que são erros ou situações inesperadas que ocorrem durante a execução do código.
- *OAuth.php* - usado para implementar autenticação e autorização usando o protocolo *OAuth* (Open Authorization). É um padrão que permite que os usuários se autenticuem no sistema e autorizem o acesso a certos recursos sem precisar compartilhar suas credenciais diretamente com a aplicação. O *OAuth* tem algumas funções, como: configurar credenciais e URL, gerar o URL de autorização, gerenciar o token de acesso e realizar requisições autenticadas.
- *OAuthTokenProvider.php* - geralmente é o responsável por gerenciar tokens de autenticação e autorização que utilizam o protocolo *OAuth*. É usado para lidar com o processo de obtenção, armazenamento, renovação e verificação dos tokens de acesso.
- *PHPMailer.php* - gerencia o envio de emails em aplicações usando a biblioteca *PHPMailer*. É uma biblioteca popular que facilita o envio de emails com suporte de

autenticação SMTP, anexos, HTML, entre outros recursos, simplificando o processo de envio de emails em comparação com a função nativa *mail.php*. O PHPMailer tem como algumas funcionalidades: configuração do servidor SMTP, configuração do remetente e do destinatário, configuração do assunto e corpo do email, anexos e tratamento de erros.

- *POP3.php* - faz parte da biblioteca PHPMailer e é utilizado para gerenciar a conexão com servidores de email utilizando o protocolo POP3 (Post Office Protocol version 3). É um protocolo de recebimento de emails padrão que permite que os clientes de emails acessem suas caixas de entrada e baixem mensagens de um servidor de email.
- *SMTP.php* - responsável por fazer a configuração da conexão SMTP, que contém as funções necessárias para configurar a conexão de um servidor de email via SMTP, fazer autenticação, o envio de emails e gerenciar erros e respostas ao servidor, fazendo parte da biblioteca PHPMailer.

2.2) aplicacao.php - neste código, fazemos o **início da sessão**, que são as variáveis gravadas no servidor e **a conexão com o banco de dados**. Podemos também, **adicionar, editar, remover** ou **filtrar uma questão por determinado tema**, além de **exibir as questões em uma tabela**.

Sessão e Proteção de Acesso:

```
aplicacao.php 9+ X
PHP_BDQ > aplicacao.php > ...
1  <?php
2  session_start();
3  if (!isset($_SESSION['user_id'])) {
4      header(header: "Location: login.html");
5      exit();
6  }
7
8  include 'conexao.php';
```

Conexão com o Banco de Dados:

```
7
8  include 'conexao.php';
9
```

Adicionar uma Nova Questão:

```

10 // Adicionar nova questão
11 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['adicionar'])) {
12     $assunto = $_POST['assunto'];
13     $enunciado = $_POST['enunciado'];
14     $alternativa1 = $_POST['alternativa1'];
15     $alternativa2 = $_POST['alternativa2'];
16     $alternativa3 = $_POST['alternativa3'];
17     $alternativa4 = $_POST['alternativa4'];
18     $alternativa5 = $_POST['alternativa5'];
19     $alternativa_correta = $_POST['alternativa_correta'];
20
21     $sql = "INSERT INTO questoes (assunto, enunciado, alternativa1, alternativa2, alternativa3, alternativa4, alternativa5,
22         alternativa_correta)
23         VALUES ('$assunto', '$enunciado', '$alternativa1', '$alternativa2', '$alternativa3', '$alternativa4', '$alternativa5',
24             '$alternativa_correta')";
25
26     if ($conexao->query(query: $sql) === TRUE) {
27         echo "Questão adicionada com sucesso!";
28     } else {
29         echo "Erro: " . $sql . "<br>" . $conexao->error;
30     }
31 }

```

Editar uma questão:

```

31 // Editar questão existente
32 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['editar'])) {
33     $id = $_POST['id'];
34     $assunto = $_POST['assunto'];
35     $enunciado = $_POST['enunciado'];
36     $alternativa1 = $_POST['alternativa1'];
37     $alternativa2 = $_POST['alternativa2'];
38     $alternativa3 = $_POST['alternativa3'];
39     $alternativa4 = $_POST['alternativa4'];
40     $alternativa5 = $_POST['alternativa5'];
41     $alternativa_correta = $_POST['alternativa_correta'];
42
43     $sql = "UPDATE questoes SET assunto='$assunto', enunciado='$enunciado', alternativa1='$alternativa1',
44         alternativa2='$alternativa2',
45         alternativa3='$alternativa3', alternativa4='$alternativa4', alternativa5='$alternativa5',
46         alternativa_correta='$alternativa_correta' WHERE id=$id";
47
48     if ($conexao->query(query: $sql) === TRUE) {
49         echo "Questão atualizada com sucesso!";
50         header(header: "Location: aplicacao.php");
51         exit();
52     } else {
53         echo "Erro: " . $sql . "<br>" . $conexao->error;
54     }
55 }

```

Deletar uma questão:

```

56 // Deletar questão
57 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['deletar'])) {
58     $id = $_POST['id'];
59     $conexao->query(query: "DELETE FROM questoes WHERE id=$id");
60     header(header: "Location: aplicacao.php");
61     exit();
62 }

```

Filtrar questões:

```

64 // Obter assuntos únicos para o filtro
65 $assuntos_resultado = $conexao->query(query: "SELECT DISTINCT assunto FROM questoes");
66
67 // Filtro de assunto
68 $filtro_assunto = '';
69 if (isset($_POST['filtro_assunto'])) {
70     $filtro_assunto = $_POST['filtro_assunto'];
71 }

```

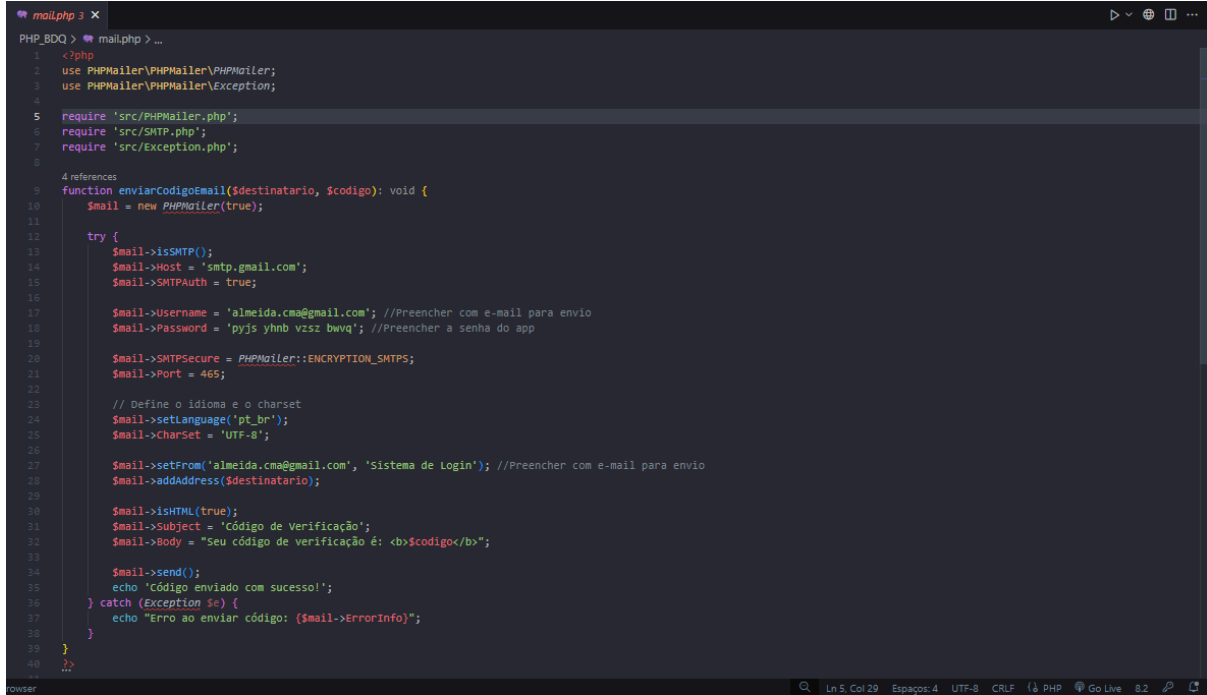
Exibir questões em tabela:

```

73 // Obter todas as questões, aplicando filtro se necessário
74 $sql = "SELECT * FROM questoes";
75 if ($filtro_assunto) {
76     $sql .= " WHERE assunto = '$filtro_assunto'";
77 }
78 $resultado = $conexao->query(query: $sql);

```

2.3) mail.php - é onde faz as configurações para poder enviar e receber seus emails.

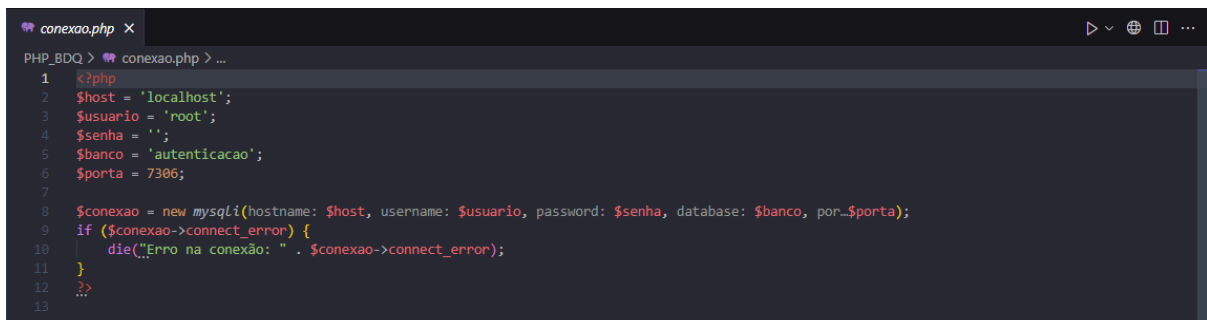


```

mail.php X
PHP_BDQ > mail.php > ...
1 <?php
2 use PHPMailer\PHPMailer\PHPMailer;
3 use PHPMailer\PHPMailer\Exception;
4
5 require 'src/PHPMailer.php';
6 require 'src/SMTP.php';
7 require 'src/Exception.php';
8
9 4 references
10 function enviarCodigoEmail($destinatario, $codigo): void {
11     $mail = new PHPMailer(true);
12
13     try {
14         $mail->isSMTP();
15         $mail->Host = 'smtp.gmail.com';
16         $mail->SMTPAuth = true;
17
18         $mail->Username = 'almeida.cma@gmail.com'; //Preencher com e-mail para envio
19         $mail->Password = 'pyjs yhnb vzs2 bwvq'; //Preencher a senha do app
20
21         $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
22         $mail->Port = 465;
23
24         // Define o idioma e o charset
25         $mail->setLanguage('pt_br');
26         $mail->Charset = 'UTF-8';
27
28         $mail->setFrom('almeida.cma@gmail.com', 'Sistema de Login'); //Preencher com e-mail para envio
29         $mail->addAddress($destinatario);
30
31         $mail->isHTML(true);
32         $mail->Subject = 'Código de Verificação';
33         $mail->Body = 'Seu código de verificação é: <b>$codigo</b>';
34
35         $mail->send();
36         echo 'Código enviado com sucesso!';
37     } catch (Exception $e) {
38         echo "Erro ao enviar código: {$mail->ErrorInfo}";
39     }
40 }
41
42 }
43
44 }

```

2.4) conexao.php - é onde faz a conexão com o MySQL, o banco de dados.



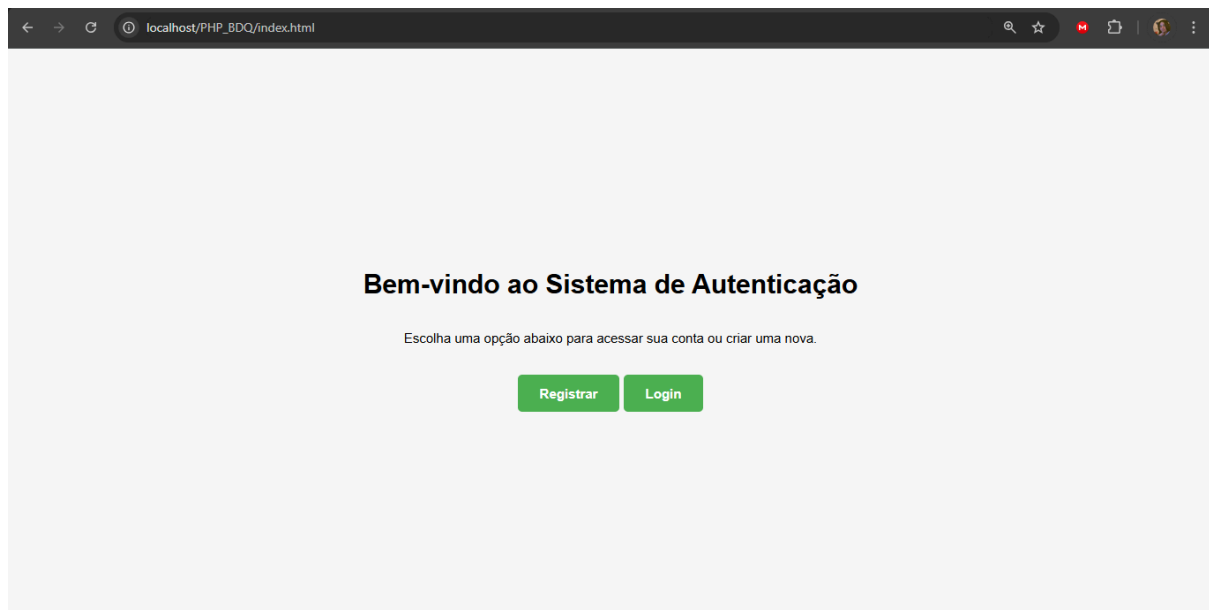
```

conexao.php X
PHP_BDQ > conexao.php > ...
1 <?php
2 $host = 'localhost';
3 $usuario = 'root';
4 $senha = '';
5 $banco = 'autenticacao';
6 $porta = 7306;
7
8 $conexao = new mysqli(hostname: $host, username: $usuario, password: $senha, database: $banco, port:$porta);
9 if ($conexao->connect_error) {
10     die("Erro na conexão: " . $conexao->connect_error);
11 }
12
13 }

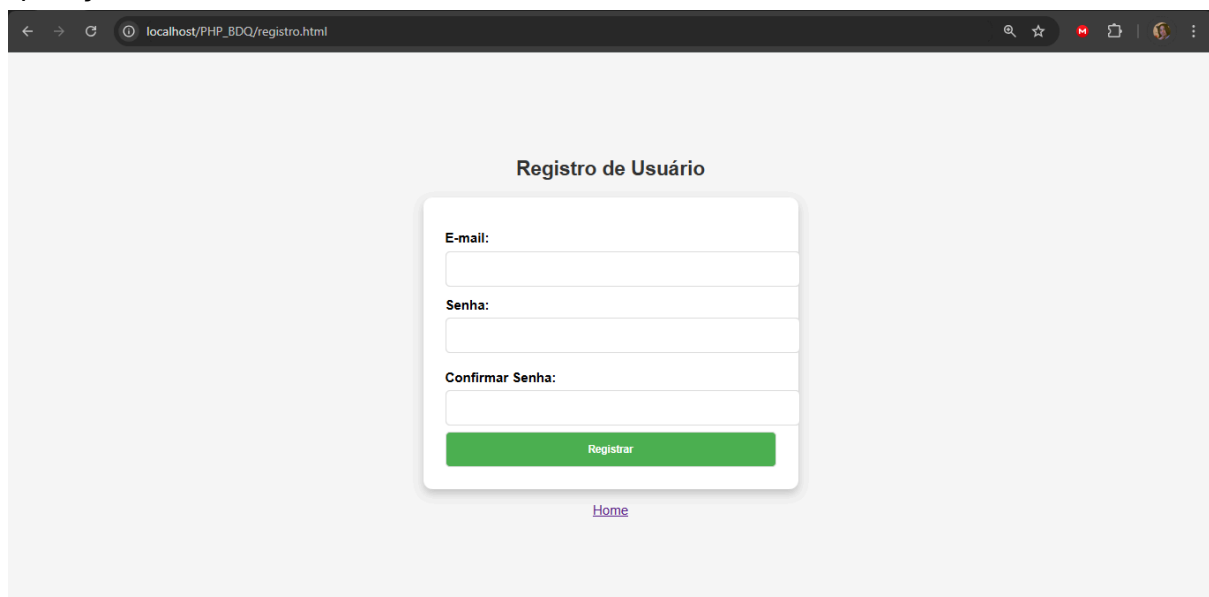
```

3) A página web e sua conexão com PHP

3.1) index.html - é a tela inicial do projeto, o frontend onde o usuário vai ter o seu primeiro contato com a página.

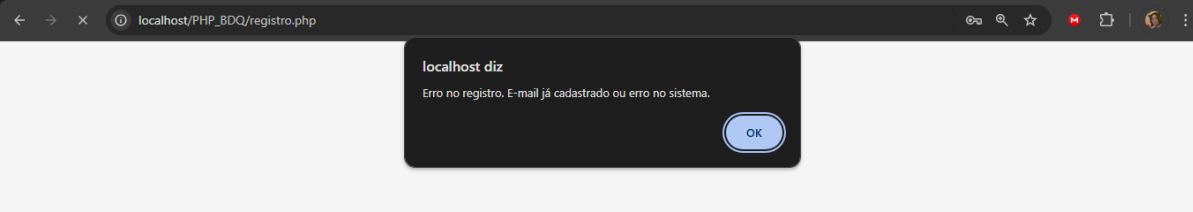


3.2.) registro.html - é a parte da página onde você se registra para conseguir usar a aplicação.



3.2.1) registro.php - é onde você é redirecionado quando esquece sua senha, fazendo conexão pelo *include* com *mail.php* (que é a configuração dos emails) e *conexao.php* (que é a conexão com o banco de dados); faz a verificação do método *\$POST*, e se for, faz a execução do código; coleta e processa dados do formulário, faz a inserção do usuário no banco de dados e a verificação de sucesso na inserção; gera e armazena o código de verificação (que é aleatório, variando entre 6 dígitos); faz o envio do código de verificação por email e redireciona para a página de verificação (**verificar.html**), e por fim, faz o tratamento de erros, caso houver algum erro durante o processo de inserção (como por exemplo, o email já estar cadastrado), o código exibe uma mensagem de erro e redireciona para a página anterior (**registro.html**).

```
PHP_BDQ > registro.php > ...
1  <?php
2  session_start();
3  include 'conexao.php';
4  include 'mail.php'; // Arquivo PHPMailer configurado
5
6  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
7      $email = $_POST['email'];
8      $senha = password_hash(password: $_POST['senha'], algo: PASSWORD_BCRYPT);
9
10     $stmt = $conexao->prepare(query: "INSERT INTO usuarios (email, senha_hash) VALUES (?, ?)");
11     $stmt->bind_param(types: "ss", var: &$email, vars: &$senha);
12
13     if ($stmt->execute()) {
14         $codigo = rand(min: 100000, max: 999999);
15
16         $usuario_id = $conexao->insert_id;
17         $stmtCodigo = $conexao->prepare(query: "INSERT INTO codigos_verificacao (usuario_id, codigo) VALUES (?, ?)");
18         $stmtCodigo->bind_param(types: "is", var: &$usuario_id, vars: &$codigo);
19         $stmtCodigo->execute();
20
21         // Enviar e-mail com código
22         enviarCodigoEmail(destinatario: $email, codigo: $codigo);
23
24         // Redireciona o usuário para a página de verificação
25         echo "<script>alert('Registro realizado! Verifique seu e-mail para confirmar.');" window.location.href='verificar.html';</script>";
26     } else {
27         echo "<script>alert('Erro no registro. E-mail já cadastrado ou erro no sistema.');" window.history.back();</script>";
28     }
29 }
30 >>
```



3.3) login.html - como o próprio nome diz, é usado para fazer login, permitindo que os usuários se autenticuem. Contém um formulário onde o usuário insere seu email e senha para acessar.

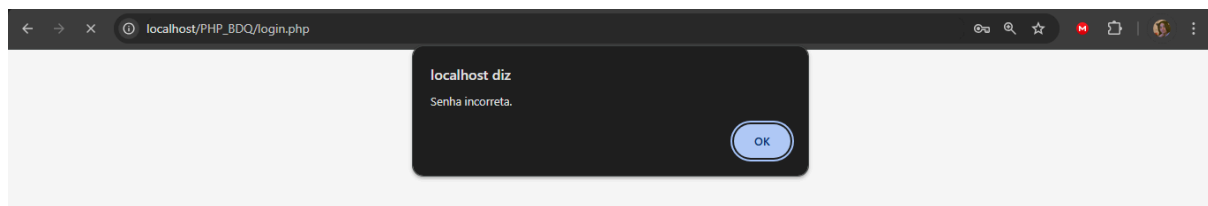
```
PHP_BDQ > login.html > ...
1  <!DOCTYPE html>
2  <html Lang="pt-BR">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Login</title>
7      <link rel="stylesheet" href="estilos.css"> <!-- Opcional: Adicione um arquivo de estilo externo -->
8  </head>
9  <body>
10     <h2>Login</h2>
11     <form action="login.php" method="POST">
12         <label for="email">E-mail:</label>
13         <input type="email" id="email" name="email" required>
14
15         <label for="senha">Senha:</label>
16         <input type="password" id="senha" name="senha" required>
17
18         <button type="submit">Entrar</button>
19     </form>
20     <p><a href="esqueceu_senha.html">Esqueceu sua senha?</a></p>
21     <p><a href="index.html">Home</a></p>
22 </body>
23 </html>
24
```

3.3.1) login.php - faz o início da sessão, permitindo o armazenamento de informações na sessão, como email; se conecta com o banco de dados usando *include*; recebe os dados do formulário de *login.html* usando o método **\$POST**; prepara uma consulta SQL para buscar

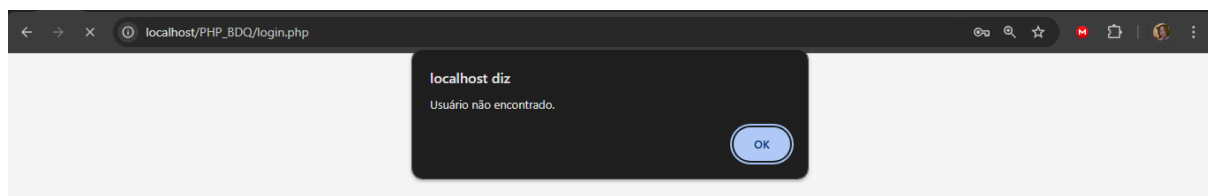
id, *senha_hash* (que é a senha criptografada) e *verificado* (o indicador se a conta foi verificada via email) do banco de dados com base no email fornecido; verifica se o usuário já existe no banco de dados; verifica a senha; verifica se a conta já foi verificada (*\$verificado*), armazena o id do usuário na variável de sessão **\$SESSION['user_id']** e redireciona o usuário para a página principal. Se a conta não estiver verificada, o usuário é alertado com um popup e redirecionado para a página de verificação(*verificar.html*); caso a senha não seja a correta, o código exibe um alerta informando que a senha está incorreta e retorna para a página de login(*login.html*). Caso o email informado não corresponda a nenhum usuário do banco de dados, um alerta é exibido, informando que o usuário não foi encontrado.

```
PHP_BDQ > login.php > ...
1  <?php
2  session_start(); // Inicia a sessão no início do arquivo
3  include 'conexao.php';
4
5  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
6      $email = $_POST['email'];
7      $senha = $_POST['senha'];
8
9      $stmt = $conexao->prepare(query: "SELECT id, senha_hash, verificado FROM usuarios WHERE email = ?");
10     $stmt->bind_param(types: "s", var: &$email);
11     $stmt->execute();
12     $stmt->store_result();
13
14     if ($stmt->num_rows > 0) {
15         $stmt->bind_result(var: &$id, vars: &$senha_hash, $verificado);
16         $stmt->fetch();
17
18         if (password_verify(password: $senha, hash: $senha_hash)) {
19             if ($verificado) {
20                 $_SESSION['user_id'] = $id; // Armazena o ID do usuário na sessão
21                 header(header: "Location: pagina_principal.php"); // Redireciona para a página principal
22                 exit();
23             } else {
24                 echo "<script>alert('Conta ainda não verificada. Verifique seu e-mail.');
```

senha incorreta:

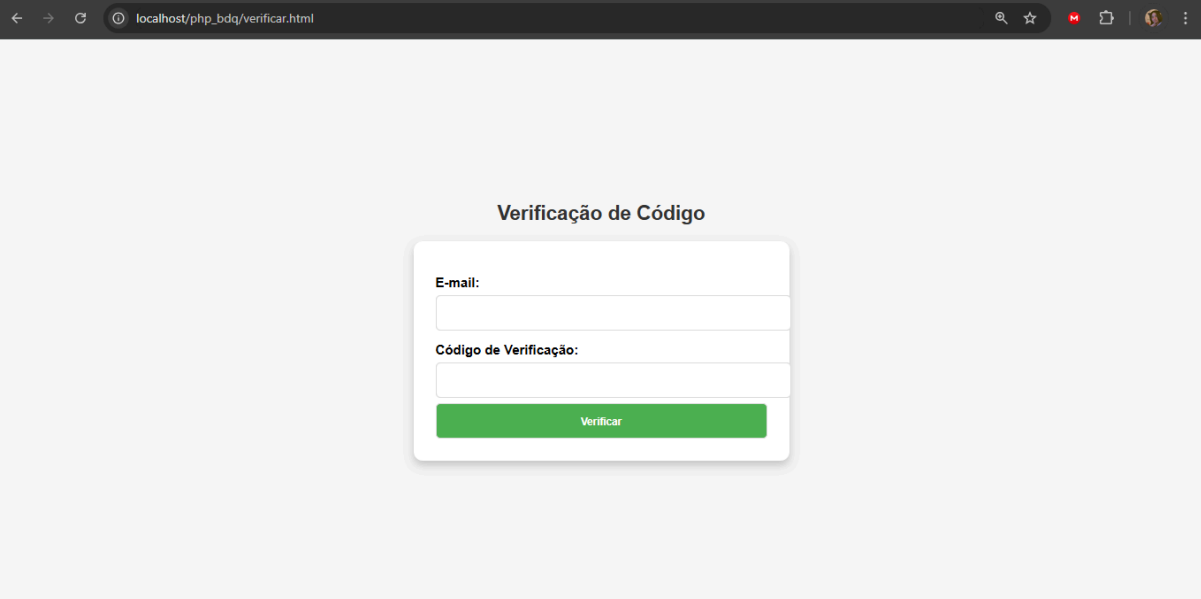


usuário não encontrado:



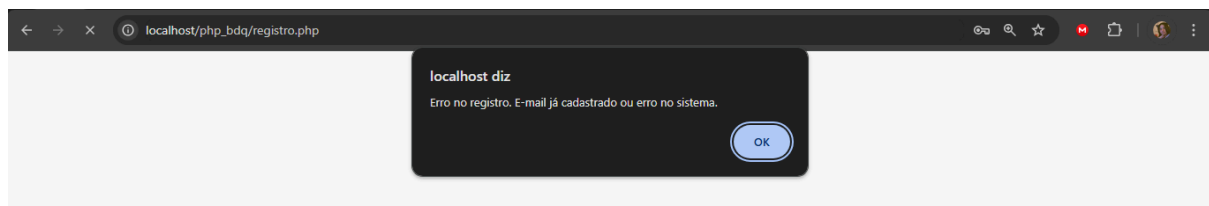
3.4) verificar.html - um formulário para verificar se o usuário recebeu o código de verificação. Assim que o botão “*verificar*” é clicado, os dados são enviados para o servidor, especificamente para o arquivo *verificar.php*, usando o método **\$POST**. Caso haja um

usuário já registrado com o mesmo email, é exibido uma tela informando que já existe o email cadastrado no sistema.



The screenshot shows a web browser window with the address bar displaying 'localhost/php_bdq/verificar.html'. The main content area features a centered form titled 'Verificação de Código'. The form has two input fields: 'E-mail:' and 'Código de Verificação:'. Below these fields is a green button labeled 'Verificar'.

email cadastrado no sistema:



3.4.1) verificar.php - inicia a sessão e faz a conexão com o banco de dados usando *include*; verifica o método usando **\$POST**; faz a captura dos dados do formulário e consulta o banco de dados; atualiza o status de verificação e redireciona após a verificação bem sucedida; caso o código de verificação seja inválido ou tenha expirado, é exibido um alerta informando o usuário e volta para a página anterior.

```

1  <?php
2  session_start();
3  include 'conexao.php';
4
5  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
6      $email = $_POST['email'];
7      $codigo = $_POST['codigo'];
8
9      $stmt = $conexao->prepare(query: "SELECT u.id FROM usuarios u JOIN codigos_verificacao c ON u.id = c.usuario_id WHERE u.email = ?
10     AND c.codigo = ? AND TIMESTAMPDIFF(MINUTE, c.criado_em, NOW()) <= 10");
11     $stmt->bind_param(types: "ss", var: &$email, vars: &$codigo);
12     $stmt->execute();
13     $stmt->store_result();
14
15     if ($stmt->num_rows > 0) {
16         $stmt->bind_result(var: &$usuario_id);
17         $stmt->fetch();
18
19         $stmtUpdate = $conexao->prepare(query: "UPDATE usuarios SET verificado = 1 WHERE id = ?");
20         $stmtUpdate->bind_param(types: "i", var: &$usuario_id);
21         $stmtUpdate->execute();
22
23         // Redireciona o usuário para a página de login após verificação bem-sucedida
24         echo "<script>alert('Conta verificada com sucesso! Faça login.');

```

3.5) pagina_principal.php: faz a verificação de sessão, com a variável `$SESSION['user_id']`. Se o usuário não estiver autenticado (ou seja, a variável de sessão não estiver definida), ele será redirecionado para *login.html*; na tela, é mostrado um contador de visitas com cookie; o código verifica se já existe um cookie chamado *login_count*, e caso exista, ele incrementa o contador de visitas, caso contrário o contador é iniciado em 1; o valor do contador é armazenado novamente em um cookie com validade de 30 dias. É exibido um texto de boas vindas e o número de vezes em que a página foi visitada (o número de logins feitos no mesmo dispositivo). Há também a possibilidade de acessar outra aplicação ou excluir conta, que é destinado a exclusão permanente da conta, conforme a Lei Geral de Proteção de Dados (LGPD).

```

1  <?php
2  session_start();
3
4  // Verifica se o usuário está logado
5  if (!isset($_SESSION['user_id'])) {
6      header(header: "Location: login.html");
7      exit();
8  }
9
10 // Verifica o cookie de contador de visitas
11 if (isset($_COOKIE['login_count'])) {
12     $loginCount = $_COOKIE['login_count'] + 1; // Incrementa o contador
13 } else {
14     $loginCount = 1; // Primeira visita
15 }
16
17 // Atualiza o cookie com o novo valor e define validade de 30 dias
18 setcookie(name: 'login_count', value: $loginCount, expires_or_options: time() + (30 * 24 * 60 * 60), path: "/");
19
20 >>
21
22 <!DOCTYPE html>
23 <html Lang="pt-BR">
24 <head>
25     <meta charset="UTF-8">
26     <meta name="viewport" content="width=device-width, initial-scale=1.0">
27     <title>Página Principal</title>
28     <link rel="stylesheet" href="estilos.css">
29 </head>
30 <body>

```

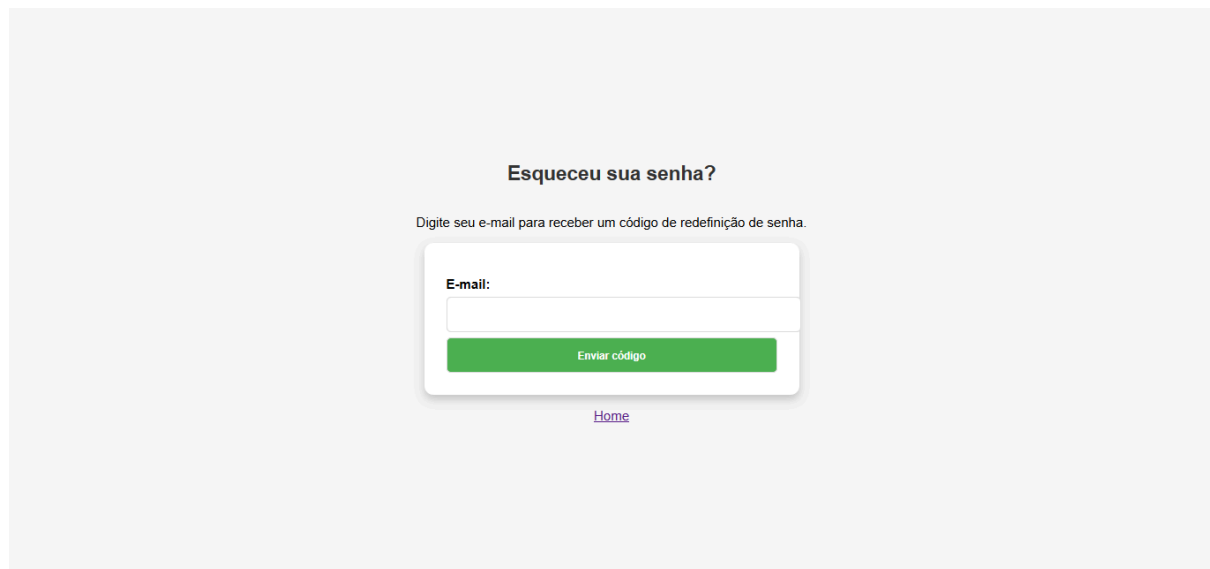
```

30 <body>
31   <h2>Bem-vindo à sua conta!</h2>
32   <p>Você está autenticado com sucesso e agora pode acessar a área principal da aplicação.</p>
33
34   <!-- Exibe o número de logins -->
35   <p>Esta página foi visitada <?= $loginCount ?> vez(es) neste dispositivo.</p>
36
37   <!-- Botão para acessar outra aplicação -->
38   <a href="aplicacao.php" class="btn">Acessar Aplicação</a>
39
40   <!-- Botão para acessar a área de exclusão de conta do usuário logado -->
41   <a href="exclusao_usuario.php" class="btn">Excluir Conta (LGPD)</a>
42
43   <!-- Botão de logout -->
44   <a href="logout.php" class="logout-btn">Sair</a>
45 </body>
46 </html>

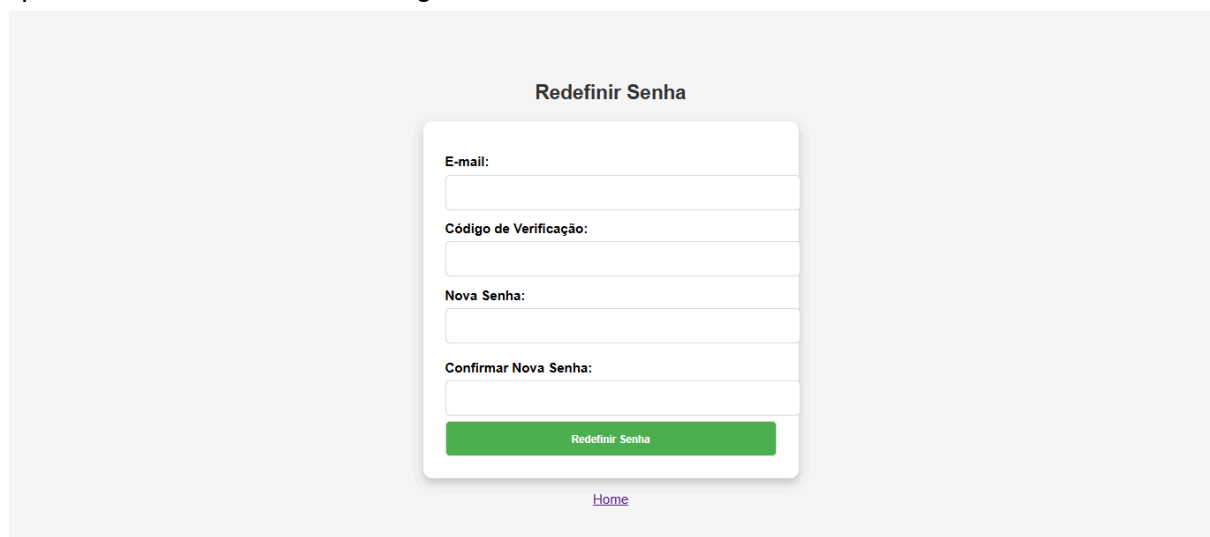
```

3.6) redefinir_senha.html - fornece uma interface para o usuário redefinir sua senha, com a funcionalidade de verificar a força da senha e validar a confirmação da senha.

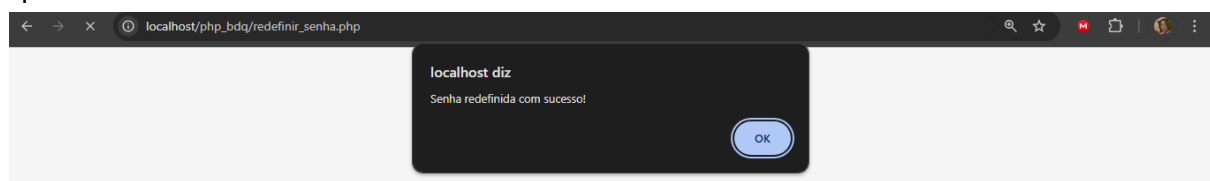
redefinir_senha.php - faz o início da sessão; inclui o arquivo de conexão com o banco de dados usando *include*; verifica o método **\$POST**; recebe e remove os dados do formulário; valida o código de verificação, realizando uma consulta SQL para verificar se o código de verificação corresponde ao código registrado no banco de dados para o email informado, e também verifica se o código não expirou (expira em 10 minutos); se a consulta retornar algum resultado (indicado por **\$stmt->num_rows > 0**), significa que o código ainda é válido. Se o código for válido, a senha do usuário será atualizada no banco de dados, com o hash da nova senha gerada. Se a senha for redefinida com sucesso, o código exibe um alerta em JavaScript e redireciona o usuário para a página de login (*login.html*). Caso contrário, se o código de verificação for inválido ou expirado, será exibido uma mensagem de erro e retorna para a página anterior.



após clicar no botão *enviar código*:



após clicar no botão *redefinir senha*:



3.7) esqueceu_senha.html: permite que o usuário redefina sua senha, fornecendo o email, um código de verificação para a redefinição de senha e a nova senha. Inclui uma validação de senha para garantir que a senha seja forte o suficiente e verifica se a senha foi confirmada corretamente.

Esqueceu sua senha?

Digite seu e-mail para receber um código de redefinição de senha.

E-mail:

Enviar código

[Home](#)

3.7.1) esqueceu_senha.php - serve para realizar a redefinição de senha de usuário após ele fornecer o código de verificação válido que foi enviado a ele. Faz o início da sessão e a conexão com o banco de dados, a verificação do método com **\$POST**, recebe os dados do formulário, faz a verificação do código de redefinição fazendo uma consulta SQL e se o código é válido; atualiza a senha no banco de dados, que será armazenada na coluna *senha_hash* da tabela *usuarios* com o valor *\$nova_senha* que foi criptografada com *password_hash()*. Por fim, mostra um alerta na tela informando ao usuário que a redefinição de senha foi bem sucedida, e o usuário é redirecionado a tela de login onde pode entrar com a nova senha.

```

1  <?php
2  session_start();
3  include 'conexao.php';
4
5  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
6      $email = $_POST['email'];
7      $codigo = $_POST['codigo'];
8      $nova_senha = password_hash(password: $_POST['nova_senha'], algo: PASSWORD_BCRYPT);
9
10     $stmt = $conexao->prepare(query: "SELECT u.id FROM usuarios u JOIN codigos_redefinicao c ON u.id = c.usuario_id WHERE u.email = ?
11     AND c.codigo = ? AND TIMESTAMPDIFF(MINUTE, c.criado_em, NOW()) <= 10");
12     $stmt->bind_param(types: "ss", var: &$email, vars: &$codigo);
13     $stmt->execute();
14     $stmt->store_result();
15
16     if ($stmt->num_rows > 0) {
17         $stmt->bind_result(var: &$usuario_id);
18         $stmt->fetch();
19
20         $stmtUpdate = $conexao->prepare(query: "UPDATE usuarios SET senha_hash = ? WHERE id = ?");
21         $stmtUpdate->bind_param(types: "si", var: &$nova_senha, vars: &$usuario_id);
22         $stmtUpdate->execute();
23
24         echo "<script>alert('Senha redefinida com sucesso!'); window.location.href='login.html';</script>";
25     } else {
26         echo "<script>alert('Código inválido ou expirado.');

```