RELATÓRIO TÉCNICO - aplicação.php

- Deleção em cascata: função que faz com que, ao excluir um registro “pai”, todos os registros filhos também sejam excluídos:

```php
$user_id = $_SESSION['user_id'];

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Exclui o usuário do banco de dados
    $stmt = $conexao->prepare("DELETE FROM usuarios WHERE id = ?");
    $stmt->bind_param("i", $user_id);

    if ($stmt->execute()) {
        session_destroy(); // Destroi a sessão após exclusão
        echo "<script>alert('Sua conta foi excluída com sucesso.'); window.location.href='index.html';</script>";
    } else {
        echo "<script>alert('Erro ao excluir conta.'); window.history.back();</script>";
    }
}
?>
```

- HASH da senha: armazena a senha de forma segura.

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email = $_POST['email'];
    $senha = password_hash($_POST['senha'], PASSWORD_BCRYPT);
```

- Sessions: mantém as requisições da aplicação em funcionamento, armazenando as informações necessárias.

```php
session_start();
include 'conexao.php';
include 'mail.php'; // Arquivo PHPMailer configurado
```

```php
<?php
session_start();
session_unset(); // Remove todas as variáveis de sessão
session_destroy(); // Destroi a sessão
header("Location: index.html"); // Redireciona para a página index
exit();
?>
```

- Cookies: são arquivos pequenos de armazenamento individual do usuário.

```php
// Verifica o cookie de contador de visitas
if (isset($_COOKIE['login_count'])) {
    $loginCount = $_COOKIE['login_count'] + 1; // Incrementa o contador
} else {
    $loginCount = 1; // Primeira visita
}

// Atualiza o cookie com o novo valor e define validade de 30 dias
setcookie('login_count', $loginCount, time() + (30 * 24 * 60 **60), "/");
```

- Functions:

```php
function enviarCodigoEmail($destinatario, $codigo) {
    $mail = new PHPMailer(true);

    try {
        $mail->isSMTP();
        $mail->Host = 'smtp.gmail.com';
        $mail->SMTPAuth = true;

    $mail->Username = ''; //Preencher com e-mail para envio
        $mail->Password = ''zv; //Preencher a senha do app

        $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
        $mail->Port = 465;

    // Define o idioma e o charset
        $mail->setLanguage('pt_br');
        $mail->CharSet = 'UTF-8';

        $mail->setFrom('', 'Sistema de Login'); //Preencher com e-mail para envio
        $mail->addAddress($destinatario);

        $mail->isHTML(true);
        $mail->Subject = 'Código de Verificação';
        $mail->Body = "Seu código de verificação é: <b>$codigo</b>";

        $mail->send();
        echo 'Código enviado com sucesso!';
    } catch (Exception $e) {
        echo "Erro ao enviar código: {$mail->ErrorInfo}";
    }
}
?>
```

```php
function validarFormulario() {
    const novaSenha = document.getElementById("nova_senha").value;
    const confirmarSenha = document.getElementById("confirmar_senha").value;

    if (novaSenha !== confirmarSenha) {
        alert("As senhas não coincidem.");
        return false;
    }

    return true;
}
```

- Include e Require: o include, ao não encontrar o que procura gera um aviso, o require exige encontrar o que procura pra continuar funcionando.

```php
session_start();
include 'conexao.php';


require 'src/PHPMailer.php';
require 'src/SMTP.php';
require 'src/Exception.php';
```

- Conexão mysqli: é a extensão do PHP para interagir com bancos de dados.

```php
$conexao = new mysqli($host, $usuario, $senha, $banco, $porta);
if ($conexao->connect_error) {
    die("Erro na conexão: " . $conexao->connect_error);
}
?>
```

- Try...catch: trata os erros sem interromper o funcionamento do código.

```php
try {
        $mail->isSMTP();
        $mail->Host = 'smtp.gmail.com';
        $mail->SMTPAuth = true;

    $mail->Username = ''; //Preencher com e-mail para envio
        $mail->Password = ''zv; //Preencher a senha do app

        $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
        $mail->Port = 465;

    // Define o idioma e o charset
        $mail->setLanguage('pt_br');
        $mail->CharSet = 'UTF-8';

        $mail->setFrom('', 'Sistema de Login'); //Preencher com e-mail para envio
        $mail->addAddress($destinatario);

        $mail->isHTML(true);
        $mail->Subject = 'Código de Verificação';
        $mail->Body = "Seu código de verificação é: <b>$codigo</b>";

        $mail->send();
        echo 'Código enviado com sucesso!';
    } catch (Exception $e) {
        echo "Erro ao enviar código: {$mail->ErrorInfo}";
    }
}
```

- Estruturas if, elseif e else:

```php
    if (password_verify($senha, $senha_hash)) {
            if ($verificado) {
                $_SESSION['user_id'] = $id; // Armazena o ID do usuário na sessão
                header("Location: pagina_principal.php"); // Redireciona para a página principal
                exit();
            } else {
                echo "<script>alert('Conta ainda não verificada. Verifique seu e-mail.'); window.location.href='verificar.html';</script>";
            }
        } else {
            echo "<script>alert('Senha incorreta.'); window.history.back();</script>";
        }
    } else {
        echo "<script>alert('Usuário não encontrado.'); window.history.back();</script>";
    }
}
?>
```

1. IF: A estrutura if é usada para testar uma condição. Se a condição for verdadeira, o bloco de código dentro do if será executado.
2. ELSEIF: O elseif (ou else if, dependendo da linguagem) é utilizado quando queremos testar uma segunda condição caso a primeira não seja verdadeira. Ou seja, se a condição do if não for verdadeira, a condição do elseif será avaliada.
3. ELSE: O else é utilizado para definir um bloco de código que será executado caso todas as condições anteriores sejam falsas. Ele é opcional e é sempre colocado após o if e/ou elseif.

RESUMO:

- if: Testa uma condição inicial. Se for verdadeira, executa o bloco de código.
- elif: Permite testar uma nova condição se a anterior for falsa.
- else: Executa um bloco de código caso todas as condições anteriores sejam falsas.

```php
PHP
        if ($stmt->num_rows > 0) {
            $stmt->bind_result($usuario_id);
            $stmt->fetch();
```

- **Referências com ->**
  1. Operador de Comparação: Em programação, > é usado para verificar se um valor é maior que outro. Exemplo: a > b.
  2. HTML (Links de Referência): Em HTML, o > pode aparecer em seletores CSS ou em listas hierárquicas de navegação.
  3. Ponteiros e Referências (C++): Em C++ e outras linguagens, > pode aparecer em templates ou em outras construções com ponteiros, mas não diretamente relacionado a referências.
  4. Hierarquia ou Estrutura de Caminho: > pode ser usado para mostrar uma hierarquia ou caminho (ex: Sistema > Aplicações > Banco de Dados).

5. Redirecionamento em Shell (Bash): Em terminais, > serve para redirecionar a saída de um comando para um arquivo, por exemplo: echo "Texto" > arquivo.txt.

- Estruturas while:

```php
$questao = null;
if (isset($_POST['load_editar'])) {
    $id = $_POST['id'];
    $questao = $conexao->query("SELECT * FROM questoes WHERE id=$id")->fetch_assoc();
}
?>
```

  - while executa um bloco de código enquanto a condição for verdadeira.
  - Pode ser usado para loops que precisam de uma condição de parada.
  - Use break para sair do loop e continue para pular a iteração atual.

- Métodos POST:

```php
/ Obter assuntos únicos para o filtro
$assuntos_resultado = $conexao->query("SELECT DISTINCT assunto FROM questoes");

// Filtro de assunto
$filtro_assunto = '';
if (isset($_POST['filtro_assunto'])) {
    $filtro_assunto = $_POST['filtro_assunto'];
}

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

// Carregar dados da questão para edição
$questao = null;
if (isset($_POST['load_editar'])) {
    $id = $_POST['id'];
    $questao = $conexao->query("SELECT * FROM questoes WHERE id=$id")->fetch_assoc();
}
?>
```

```php
<?php while ($assunto = $assuntos_resultado->fetch_assoc()) : ?>
                <option value="<?php echo $assunto['assunto']; ?>" <?php echo ($assunto['assunto'] == $filtro_assunto) ? 'selected' : ''; ?>>
                    <?php echo $assunto['assunto']; ?>
                </option>
            <?php endwhile; ?>
}
```

- POST é usado para enviar dados ao servidor, geralmente para criar ou atualizar recursos.

- Os dados são enviados no corpo da requisição (não na URL).
- O POST é mais seguro que o GET para enviar dados sensíveis.
- Ele é utilizado em formulários de sites, APIs RESTful, login, cadastro, etc.

- **Finalidade da Aplicação:**

A aplicação descrita tem como objetivo principal gerenciar um banco de questões, onde é possível adicionar, editar, deletar e filtrar questões de diferentes assuntos. A interface é simples e permite ao usuário gerenciar o conteúdo de forma intuitiva.

- **Autenticação em 2 etapas:**

A autenticação em 2 etapas (2FA) é um método de segurança que exige dois fatores para verificar a identidade do usuário ao acessar uma conta. Isso adiciona uma camada extra de proteção, dificultando o acesso não autorizado, mesmo que a senha seja comprometida. Como Funciona:

1. Primeiro fator: O usuário insere sua senha.
2. Segundo fator: O usuário fornece uma segunda forma de autenticação, como:

- Código gerado por um aplicativo (ex: Google Authenticator).
- Código enviado por SMS ou e-mail.
- Chave de segurança física (ex: Yubikey).
- Biometria (ex: impressão digital ou reconhecimento facial).

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email = $_POST['email'];
    $codigo = $_POST['codigo'];
    $nova_senha = password_hash($_POST['nova_senha'], PASSWORD_BCRYPT);
```

Fluxo de 2FA:

1. O usuário faz login com sua senha.
2. O sistema solicita o segundo fator (como o código do Google Authenticator ou o SMS enviado).
3. Se o segundo fator for validado, o acesso é concedido. Se falhar, o acesso é negado.

- **Identificação das rotas:**

Identificação de rotas em uma aplicação web refere-se ao processo de mapear URLs a ações específicas que o servidor deve executar quando uma requisição é feita. Cada rota é identificada por uma URL e um método HTTP (como GET, POST, PUT, DELETE), que determina qual operação o servidor realizará.