Documentação

▼ index.php

```
<?php

session_start();
```

- **Objetivo:** Inicia uma sessão PHP para permitir o armazenamento de variáveis de sessão.
- **Explicação:**
  - `session_start();`: Inicia ou resume uma sessão PHP existente, permitindo o armazenamento de dados de sessão.

```
$_SESSION["id"] = $_GET["id"];
$_SESSION["ap"] = $_GET["ap"];

?>
```

- **Objetivo:** Obtém e armazena os endereços MAC do ponto de acesso (AP) e do usuário na sessão.
- **Explicação:**
  - `$_SESSION["id"] = $_GET["id"];`: Obtém o endereço MAC do usuário a partir dos parâmetros da URL (`id`) e o armazena na variável de sessão `$_SESSION["id"]`.
  - `$_SESSION["ap"] = $_GET["ap"];`: Obtém o endereço MAC do ponto de acesso (AP) a partir dos parâmetros da URL (`ap`) e o armazena na variável de sessão `$_SESSION["ap"]`.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title>Portal Wi-Fi</title>
        <meta name="viewport" content="width=device-width, minimum-scale=1.0, maximum-scal
        <link rel="stylesheet" type="text/css" href="css/style.css">
    </head>
```

  - **Configurações Gerais (`<head>`):**
    - **Meta Tags e Título:**
      - `meta charset="utf-8"`: Define a codificação de caracteres UTF-8.
      - `meta name="viewport" ...`: Configura a viewport para melhor suporte a dispositivos móveis.
      - `title`: Define o título da página como "Portal Wi-Fi".
    - **Inclusão de Estilos Externos:**
      - `link rel="stylesheet"`: Inclui o arquivo externo de estilo `style.css`.

```
<body>

        <div class="logo-container">
            <img src="./cest.png" alt="Logo da Empresa" class="logo">
        </div>


        <form method="post" action="connecting.php">

            <div class="radio-group">
                <input type="radio" id="aluno" name="tipoUsuario" value="aluno" required>
```

```
                <label for="aluno">Aluno</label>

                <input type="radio" id="visitante" name="tipoUsuario" value="visitante" re
                <label for="visitante">Visitante</label>
            </div>

            <div id="inputContainer">
                <!-- Input inicial (será substituído dinamicamente) -->
                <input type="text" name="cpfMatricula" placeholder="Inserir CPF ou Matrícu
            </div>
            Senha
            <input type="password" name="senha" placeholder="Inserir Senha" required>
            <br>
            <input type="submit" value="Entrar">

            <!-- Adicionando link para a página de alteração de senha -->
            <a href="cadastro/esqueceu_senha.php">Redefinir senha</a>


        </form>
        <br><br>


        <!-- Botão de redirecionamento para a página de cadastro.php -->
        <button class="signup-button" onclick="window.location.href='cadastro/cadastro.php

        <script>
    // Adiciona um ouvinte de evento aos radio buttons para alterar o tipo de entrada
            var radioAluno = document.getElementById('aluno');
            var radioVisitante = document.getElementById('visitante');
            var inputContainer = document.getElementById('inputContainer');
            var radioGroup = document.querySelector('.radio-group');

            radioAluno.addEventListener('change', function () {
                if (radioAluno.checked) {
                    inputContainer.innerHTML = '<input type="text" name="cpfMatricula" pla
                }
            });

            radioVisitante.addEventListener('change', function () {
                if (radioVisitante.checked) {
                    inputContainer.innerHTML = '<input type="text" name="cpfMatricula" pla
                }
            });
        </script>
    </body>
    </html>
```

- **Corpo da Página (`<body>`):**
  - **Logo e Formulário:**
    - `<div class="logo-container">`: Contêiner para a logo da empresa.
    - `<form method="post" action="connecting.php">`: Formulário de autenticação, com método POST e ação para o arquivo `connecting.php`.
    - **Radio Group:**
      - `<div class="radio-group">`: Contêiner para os botões de rádio.

- `<input type="radio" ...>`: Botão de rádio para selecionar o tipo de usuário (Aluno ou Visitante).
- `<label for="aluno">...</label>`: Rótulo associado ao botão de rádio "Aluno".
- Similarmente para o botão "Visitante".

- **Input Container Dinâmico:**
  - `<div id="inputContainer">`: Contêiner para a entrada de CPF ou Matrícula, substituído dinamicamente com JavaScript.
  - `<input type="text" ...>`: Campo de entrada inicial para CPF ou Matrícula.
- **Campo de Senha:**
  - `<input type="password" ...>`: Campo de entrada para a senha do usuário.
- **Botão de Envio:**
  - `<input type="submit" ...>`: Botão de envio do formulário.
- **Link para Redefinição de Senha:**
  - `<a href="cadastro/esqueceu_senha.php">Redefinir senha</a>`: Link para a página de redefinição de senha.
- **Botão de Cadastro:**
  - `<button class="signup-button" ...>`: Botão de redirecionamento para a página de cadastro.
- **Script JavaScript:**
  - `<script> ... </script>`: Script JavaScript para dinamicamente alterar o campo de entrada com base na seleção do tipo de usuário.

▼ connecting.php

```
<?php

// Inicia a sessão
session_start();
```

- Inicia uma sessão PHP. As sessões são usadas para armazenar informações do lado do servidor que podem ser acessadas entre diferentes páginas.

```
// Obtém os endereços MAC do ponto de acesso (AP) e do usuário da sessão
$mac = $_SESSION["id"];
$ap = $_SESSION["ap"];
```

- **Obtenção de Endereços MAC (`$mac = $_SESSION["id"]; $ap = $_SESSION["ap"];`):**
  - **Objetivo:** Obtém os endereços MAC do ponto de acesso (AP) e do usuário armazenados na sessão.
  - **Explicação:** Recupera os valores armazenados nas variáveis de sessão `$_SESSION["id"]` e `$_SESSION["ap"]` para utilizar posteriormente.

```
// Carrega a biblioteca UniFi
require __DIR__ . '/vendor/autoload.php';
```

- **Carregamento da Biblioteca UniFi (`require __DIR__ . '/vendor/autoload.php';`):**
  - **Objetivo:** Carrega a biblioteca UniFi API.
  - **Explicação:** Inclui a biblioteca necessária para interagir com o UniFi Controller.

```
// Configurações para autorização no UniFi Controller
$duracao_autorizacao = 20; // Tempo em minutos que usuario ficara logado
$id_site = 'default'; // ID do site encontrado na URL exemplo servidor:8443/manage/site/defau
```

- Define algumas configurações específicas para a autorização no UniFi Controller, como a duração da autorização em minutos e o ID do site.

```
$usuario_controller = 'usuarioUnifi'; // Nome de usuário do UniFi Controller Cadastrado no si
$senha_controller = 'SenhaUnifi'; // Senha para acesso ao UniFi Controller
$url_controller = 'https://servidor:8443'; // URL do UniFi Controller
$versao_controller = '7.1.68'; // Versão do software do Controller
$depuracao = false;
```

- Configura as informações de autenticação para o UniFi Controller, como nome de usuário, senha, URL e versão do software.

```
// Cria uma instância da classe UniFi API Client
$unifi_conexao = new UniFi_API\Client($usuario_controller, $senha_controller, $url_controller
```

- Cria uma instância da classe UniFi API Client usando as informações fornecidas anteriormente.

```
// Define o modo de depuração
$definir_modo_debug = $unifi_conexao->set_debug($depuracao);
```

- Define o modo de depuração da conexão UniFi API Client com base na configuração anterior.

```
// Faz o login no UniFi Controller
$resultados_login = $unifi_conexao->login();
```

- **Objetivo:** Realiza o login no UniFi Controller.
- **Explicação:** Chama o método `login()` da instância da classe UniFi API Client, armazenando os resultados em `$resultados_login`.

Autenticação do Usuário

```
// Obtém os dados do formulário se é aluno ou visitante usuario vai selecionar se é aluno ou
$tipoUsuario = $_POST['tipoUsuario'];
$cpfMatricula = $_POST['cpfMatricula'];
$senha = $_POST['senha'];
```

- **Objetivo:** Obtém os dados do formulário HTML via método POST.
- **Explicação:** Utiliza `$_POST` para coletar informações do tipo de usuário, CPF/Matrícula e senha fornecidos pelo formulário.

1. **Conectar ao Banco de Dados**

```
include 'conexao.php';
```

- Inclui um arquivo chamado `conexao.php`, que contém informações ou funções relacionadas à conexão com o banco de dados.

```
$sql = "SELECT * FROM usuarios WHERE ";

if ($tipoUsuario == 'aluno') {
    $sql .= "matricula = '$cpfMatricula'";
} elseif ($tipoUsuario == 'visitante') {
    $sql .= "cpf = '$cpfMatricula'";
} else {
    // Tipo de usuário não reconhecido
    echo "Tipo de usuário não reconhecido.";
    exit;
```

```
}

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

- **Objetivo:** Executa uma consulta SQL para verificar o usuário no banco de dados.
- **Explicação:** Constrói dinamicamente uma consulta SQL com base no tipo de usuário. O resultado é armazenado em `$result`.

```
// Verifica se o usuário foi encontrado no banco de dados
if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();

    // Verifica a senha hasheada
    if (password_verify($senha, $row['senha'])) {
        // Usuário autenticado com sucesso

        // Autoriza o acesso do usuário na rede Wi-Fi
        $resultado_autorizacao = $unifi_conexao->authorize_guest($mac, $duracao_autorizacao,

        // Registra os dados na tabela "usuarios_log"
        $nome = $row['nome'];
        $cpf = $row['cpf'];
        $telefone = $row['telefone'];
        $ip = $_SERVER['REMOTE_ADDR']; // Obtém o IP do cliente
        $hora_login = date('Y-m-d H:i:s');

        // Insere os dados na tabela "usuarios_log"
        $sql_insert_log = "INSERT INTO usuarios_log (nome, cpf, telefone, ip, mac, hora_logir
        $conn->query($sql_insert_log);

        // Redireciona para a página desejada
        header('Location: https://www.google.com.br');
        exit(); // Certifica-se de que o script não continua a ser executado após o redirecio
    } else {
        // Senha incorreta
        $mensagem = "Senha incorreta. Por favor, tente novamente.";
        echo "<script>
            alert('$mensagem');
            window.location.href = 'index.php'; // redireciona para a página desejada
            </script>";
        }
} else {
    // Usuário não encontrado
    $mensagem = "Usuário não encontrado. Por favor, verifique o CPF ou a Matrícula e a senha
```

- **Objetivo:** Autentica o usuário, autoriza o acesso Wi-Fi, e registra dados no banco de dados.
- **Explicação:** Verifica se o usuário foi encontrado no banco de dados e realiza as ações correspondentes, como autorização no UniFi Controller e registro de dados na tabela `usuarios_log`.

```
echo "<script>
        alert('$mensagem');
        window.location.href = 'index.php'; // redireciona para a página desejada
        </script>";
}
```

- **Objetivo:** Lida com possíveis erros, como tipo de usuário não reconhecido ou senhas incorretas.

- **Explicação:** Exibe mensagens de erro e, em alguns casos, redireciona o usuário para a página inicial.

```
// Fecha a conexão com o banco de dados
$conn->close();
?>
```

- **Objetivo:** Fecha a conexão com o banco de dados após a execução das operações.
- **Explicação:** Utiliza `$conn->close()` para liberar os recursos e encerrar a conexão com o banco de dados.

▼ atualizarsenha.php

Conexão com o banco de dados

```
<?php
// Conectar ao banco de dados
include '../conexao.php';
```

- **Objetivo:** Estabelecer a conexão com o banco de dados.
- **Explicação:** Inclui o arquivo "conexao.php", que contém as configurações de conexão com o banco de dados.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $cpf = $_POST['cpf'];
    $novaSenha = password_hash($_POST['novaSenha'], PASSWORD_DEFAULT);
```

- **Objetivo:** Identificar se o formulário foi submetido.
- **Explicação:** Verifica se a requisição foi feita usando o método POST para determinar se o formulário foi enviado.

```
$cpf = $_POST['cpf'];
    $novaSenha = password_hash($_POST['novaSenha'], PASSWORD_DEFAULT);

    // Atualizar a senha no banco de dados
    $sql = "UPDATE usuarios SET senha = '$novaSenha' WHERE cpf = '$cpf'";
    if ($conn->query($sql) === TRUE) {
        // Senha atualizada com sucesso, redirecionar para a página de login
        header("Location: ../index.php");
        exit();
    } else {
        // Erro ao atualizar senha, exibir mensagem de erro
        $error_message = "Erro ao atualizar a senha: " . $conn->error;
    }
}
```

- **Objetivo:** Processar os dados do formulário e atualizar a senha no banco de dados.
- **Explicação:** Obtém o CPF e a nova senha do formulário, realiza o hash da nova senha e executa uma instrução SQL para atualizar a senha no banco de dados. Em caso de sucesso, redireciona para a página de login. Em caso de erro, armazena uma mensagem de erro.

```
// Obter CPF da URL
$cpf = $_GET['cpf'];
```

- **Objetivo:** Obter o CPF da URL para preencher automaticamente o campo no formulário.
- **Explicação:** Obtém o CPF da URL usando o método GET para pré-preencher o campo correspondente no formulário.

```
// Fechar a conexão
$conn->close();
```

```
?>
```

- **Objetivo:** Encerrar a conexão com o banco de dados.
- **Explicação:** Utiliza o método `close()` para liberar os recursos associados à conexão com o banco de dados.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Atualizar Senha</title>
    <link rel="stylesheet" href="css/style.css">
    <link rel="stylesheet" type="text/css" href="../css/style.css">
</head>
<body>
<div class="logo-container">
    <img src="../cest.png" alt="Logo do cest" class="logo">
</div>
<div class="form-container">

    <?php if (isset($error_message)) echo "<p class='error'>$error_message</p>"; ?>
    <form method="post" action="">
        <input type="hidden" name="cpf" value="<?php echo $cpf; ?>">
        <label for="novaSenha">Nova Senha:</label>
        <input type="password" name="novaSenha" required>
        <label for="confirmarSenha">Confirmar Senha:</label>
        <input type="password" name="confirmarSenha" required>
        <input type="submit" value="Atualizar">
    </form>

     <a href="../esqueceu_senha.php"><button>Voltar</button></a>



</div>

</body>
</html>
```

- **Objetivo:** Estruturação da página HTML para exibir o formulário de atualização de senha.
- **Explicação:** Criação da estrutura HTML com formulário para a atualização da senha. Exibe mensagens de erro, se houver, e pré-preenche o campo do CPF com o valor obtido da URL.

▼ esqueceu_senha.php

```
<?php
// Conectar ao banco de dados
include '../conexao.php';
```

- **Objetivo:** Estabelecer a conexão com o banco de dados.
- **Explicação:** Inclui o arquivo "conexao.php", que contém as configurações de conexão com o banco de dados.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $cpf = $_POST['cpf'];
    $telefone = $_POST['telefone'];
```

- **Objetivo:** Identificar se o formulário foi submetido.

- **Explicação:** Verifica se a requisição foi feita usando o método POST para determinar se o formulário foi enviado.

```
$cpf = $_POST['cpf'];
    $telefone = $_POST['telefone'];

    // Verificar se o CPF e o telefone existem no banco de dados
    $sql = "SELECT * FROM usuarios WHERE cpf = '$cpf' AND telefone = '$telefone'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        // CPF e telefone correspondem, redirecionar para a página de atualização de senha
        header("Location: atualizarsenha.php?cpf=$cpf");
        exit();
    } else {
        // CPF ou telefone incorretos, exibir mensagem de erro
        $error_message = "CPF e/ou telefone incorretos. Por favor, tente novamente.";
    }
}
```

- **Objetivo:** Validar se o CPF e o telefone fornecidos existem no banco de dados.
- **Explicação:** Realiza uma consulta SQL para verificar se há uma correspondência entre o CPF e o telefone na tabela "usuarios". Se houver, redireciona para a página "atualizarsenha.php" com o CPF como parâmetro. Caso contrário, exibe uma mensagem de erro.

```
// Fechar a conexão
$conn->close();
?>
```

- **Objetivo:** Encerrar a conexão com o banco de dados.
- **Explicação:** Utiliza o método `close()` para liberar os recursos associados à conexão com o banco de dados.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Esqueceu a Senha</title>
    <link rel="stylesheet" href="css/style.css">
    <link rel="stylesheet" type="text/css" href="../css/style.css">
</head>
<body>

<div class="form-container">

    <div class="logo-container">
    <img src="../cest.png" alt="Logo do cest" class="logo">
</div>

    <?php if (isset($error_message)) echo "<p class='error'>$error_message</p>"; ?>
    <form method="post" action="">
        <label for="cpf">CPF:</label>
        <input type="text" name="cpf" required>
        <label for="telefone">Telefone:</label>
        <input type="text" name="telefone" required>
        <input type="submit" value="Próximo">
        <!-- Adicionando botão de voltar -->
    </form>
    <a href="../index.php"><button>Voltar</button></a>
```

```
</div>

</body>
</html>
```

- **Objetivo:** Estruturação da página HTML para exibir o formulário de recuperação de senha.
- **Explicação:** Criação da estrutura HTML com formulário para a recuperação de senha. Exibe mensagens de erro, se houver, e inclui um botão para voltar à página inicial.

Conclusão:

Este script PHP e HTML permite a recuperação de senha dos usuários. Ele interage com o banco de dados, processa formulários e fornece uma interface de usuário para iniciar o processo de recuperação de senha.

▼ conexao.php

```
<?php

$servername = "localhost";
$username = "usuariobanco";
$password = "senhabanco";
$dbname = "nomebanco";
```

- **Objetivo:** Define as credenciais necessárias para a conexão com o banco de dados.
- **Explicação:** Atribui os valores para `$servername` (endereço do servidor de banco de dados), `$username` (nome de usuário do banco), `$password` (senha do banco) e `$dbname` (nome do banco de dados).

```
// Cria a conexão
$conn = new mysqli($servername, $username, $password, $dbname);
```

- **Objetivo:** Estabelece a conexão com o banco de dados usando a classe `mysqli`.
- **Explicação:** Utiliza o construtor da classe `mysqli` para criar uma nova instância de conexão com o banco de dados, armazenada na variável `$conn`.

```
// Verifica a conexão
if ($conn->connect_error) {
    die("Conexão falhou: " . $conn->connect_error);
}
```

- **Objetivo:** Verifica se a conexão com o banco de dados foi estabelecida com sucesso.
- **Explicação:** Utiliza a propriedade `connect_error` da instância `$conn` para checar se houve algum erro na conexão. Caso positivo, encerra o script PHP e exibe uma mensagem de erro.

Função para Hashear Senha:

```
// Função para hashear a senha (use a que corresponde ao seu método de hash no banco de dados
function hashSenha($senha) {
    return password_hash($senha, PASSWORD_BCRYPT);
}
```

- **Objetivo:** Gera um hash seguro para a senha utilizando o algoritmo bcrypt.
- **Explicação:** A função `password_hash()` é utilizada para gerar um hash seguro da senha fornecida como argumento. O método de hash utilizado é `PASSWORD_BCRYPT`.

```
// Função para verificar a senha
function verificarSenha($senha, $hash) {
    return password_verify($senha, $hash);
```

```
}

?>
```

- **Objetivo:** Verifica se a senha corresponde ao hash fornecido.
- **Explicação:** Utiliza a função `password_verify()` para comparar a senha com o hash armazenado. Retorna `true` se a senha for válida, `false` caso contrário.

▼ cadastro.php

Página de Cadastro.

A página é projetada para coletar informações do usuário, como nome, CPF, telefone, e-mail e senha, e possui validações em tempo real, como a validação do CPF.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="../css/cad.style.css">
    <title>Cadastro de Visitante</title>
</head>
```

- **Objetivo:** Declaração e configuração do documento HTML.
- **Explicação:** Define a versão do HTML, o idioma da página, a codificação de caracteres, e inclui uma folha de estilo CSS externa.

Cabeçalho e Logo:

```
<div class="logo-container">
    <img src="../cest.png" alt="Logo do cest" class="logo">
</div>
```

- Cria um contêiner para a logo, exibindo uma imagem com o caminho "../cest.png" e um texto alternativo ("Logo do cest").

```
<form action="processa_cadastro.php" method="post" onsubmit="return validarCPF()">
    <label for="nome">NOME COMPLETO</label>
    <input type="text" name="nome" maxlength="50" required>

    <label for="cpf">CPF:</label>
    <input type="text" name="cpf" id="cpf" maxlength="11" oninput="validarNumero(this)" requ

    <label for="telefone">TEL/WHATS APP</label>
    <input type="text" name="telefone" maxlength="11" required>

    <label for="email">EMAIL</label>
    <input type="email" name="email" required>

    <label for="senha">SENHA</label>
    <input type="password" name="senha" required>

    <input type="submit" value="CADASTRAR">

    <button onclick="window.location.href='../index.php'">VOLTAR</button>
</form>
```

- Contém campos para inserção de nome, CPF, telefone, e-mail e senha, além de botões para enviar o formulário e voltar à página inicial. Formulário que envia os dados para o script "processa_cadastro.php" usando o método POST. A função

`validarCPF()` é chamada no evento de envio do formulário.

```
<script>
    function validarCPF() {
        // Recupera o valor do CPF
        var cpf = document.getElementById('cpf').value;

        // Remove caracteres não numéricos
        cpf = cpf.replace(/\D/g, '');

        // Verifica se o CPF tem 11 dígitos
        if (cpf.length !== 11) {
            alert('CPF inválido. Certifique-se de inserir 11 dígitos.');
            return false;
        }

        // Validação do CPF
        var soma = 0;
        for (var i = 0; i < 9; i++) {
            soma += parseInt(cpf.charAt(i)) * (10 - i);
        }
        var resto = 11 - (soma % 11);
        var digitoVerificador1 = (resto >= 10) ? 0 : resto;

        soma = 0;
        for (var i = 0; i < 10; i++) {
            soma += parseInt(cpf.charAt(i)) * (11 - i);
        }
        resto = 11 - (soma % 11);
        var digitoVerificador2 = (resto >= 10) ? 0 : resto;

        if (digitoVerificador1 !== parseInt(cpf.charAt(9)) || digitoVerificador2 !== parseInt
            alert('CPF inválido. Verifique os dígitos verificadores.');
            document.getElementById('cpf').value = '';  // Limpa o campo CPF
            return false;
        }

        // Se passou nas validações, retorna true para permitir o envio do formulário
        return true;
    }

        function validarNumero(input) {
        // Remove caracteres não numéricos do campo
        input.value = input.value.replace(/\D/g, '');
    }

</script>

</body>
</html>
```

- Inclusão de scripts JavaScript que fornecem funcionalidades de validação de CPF e remoção de caracteres não numéricos do campo CPF.
- A função `validarCPF()` realiza a validação do CPF, exibindo alertas em caso de erro e retornando true ou false, dependendo do resultado.

- A função `validarNumero(input)` remove caracteres não numéricos do campo associado, garantindo que apenas números sejam aceitos.

Conclusão:

Esta página HTML fornece um formulário de cadastro com validação de CPF utilizando JavaScript, e a estilização é feita através de uma folha de estilo externa.

▼ processa_cadastro.php

```
<?php
// Conectar ao banco de dados
include '../conexao.php';
```

- **Objetivo:** Estabelecer a conexão com o banco de dados.
- **Explicação:** Inclui o arquivo "conexao.php", que contém as configurações de conexão com o banco de dados.

```
// Receber os dados do formulário
$nome = $_POST['nome'];
$cpf = $_POST['cpf'];
$telefone = $_POST['telefone'];
$email = $_POST['email'];
$senha = password_hash($_POST['senha'], PASSWORD_DEFAULT);
```

- **Objetivo:** Obter os dados submetidos pelo formulário de cadastro.
- **Explicação:** Utiliza o método POST para obter os valores dos campos do formulário e hash da senha para armazenamento seguro.

```
// Obter data e hora atuais
$dataHoraCadastro = date("Y-m-d H:i:s");
```

- **Objetivo:** Registrar a data e hora exatas do momento do cadastro.
- **Explicação:** Utiliza a função `date()` para obter a data e hora atuais no formato apropriado para armazenamento no banco de dados.

```
// Obter endereço IP do cliente
$ip = $_SERVER['REMOTE_ADDR'];
```

- **Objetivo:** Registrar o endereço IP do usuário que está realizando o cadastro.
- **Explicação:** Utiliza a variável `$_SERVER['REMOTE_ADDR']` para obter o endereço IP do cliente.

```
// Função para obter o endereço MAC
function getMACAddress(){
    exec("ipconfig /all", $output);
    foreach($output as $line){
        if (preg_match('/Physical.*: (.*)/', $line, $mac)){
            return $mac[1];
        }
    }
    return null;
}

// Obter endereço MAC
$mac = getMACAddress();
```

- **Objetivo:** Recuperar o endereço MAC da máquina do cliente.
- **Explicação:** Utiliza a função `getMACAddress()` para executar comandos do sistema e recuperar o endereço MAC.

```
// Verificar se o CPF já foi cadastrado
$sql_cpf = "SELECT * FROM usuarios WHERE cpf = '$cpf'";
$result_cpf = $conn->query($sql_cpf);

if ($result_cpf->num_rows > 0) {
    // CPF já cadastrado, exibir mensagem de alerta
    echo '<script>alert("CPF já cadastrado. Por favor, escolha outro.");</script>';
    echo '<script>window.location.href="cadastro.php";</script>';
    $conn->close();
    exit();
}

// Verificar se o e-mail já foi cadastrado
$sql_email = "SELECT * FROM usuarios WHERE email = '$email'";
$result_email = $conn->query($sql_email);

if ($result_email->num_rows > 0) {
    // E-mail já cadastrado, exibir mensagem de alerta
    echo '<script>alert("E-mail já cadastrado. Por favor, escolha outro.");</script>';
    echo '<script>window.location.href="cadastro.php";</script>';
    $conn->close();
    exit();
}
```

- **Objetivo:** Garantir a unicidade de CPF e e-mail no banco de dados.
- **Explicação:** Realiza consultas SQL para verificar se o CPF ou e-mail já existem na tabela de usuários. Exibe mensagens de alerta e redireciona em caso de duplicidade.

```
// Inserir os dados no banco de dados
$sql = "INSERT INTO usuarios (nome, cpf, telefone, email, senha, data_hora_cadastro, ip, mac)
        VALUES ('$nome', '$cpf', '$telefone', '$email', '$senha', '$dataHoraCadastro', '$ip',
```

- **Objetivo:** Armazenar os dados do novo usuário no banco de dados.
- **Explicação:** Executa uma instrução SQL para inserir os dados do usuário na tabela "usuarios".

```
if ($conn->query($sql) === TRUE) {
    // Cadastro bem-sucedido, exibir mensagem de alerta e redirecionar
    echo '<script>alert("Cadastro bem-sucedido!");</script>';
    echo '<script>window.location.href="index.php";</script>';
    $conn->close();
    exit(); // Certifique-se de parar a execução do script após o redirecionamento
} else {
    // Erro ao cadastrar, exibir mensagem de alerta com detalhes do erro
    echo '<script>alert("Erro ao cadastrar: ' . $conn->error . '");</script>';
}
```

- **Objetivo:** Tratar o resultado da inserção no banco de dados.
- **Explicação:** Se a inserção for bem-sucedida, exibe uma mensagem de sucesso, redireciona para a página inicial e encerra a execução. Em caso de erro, exibe uma mensagem de alerta com detalhes do erro.

```
// Fechar a conexão
$conn->close();
?>
```

- **Objetivo:** Encerrar a conexão com o banco de dados.

- **Explicação:** Utiliza o método `close()` para liberar os recursos associados à conexão com o banco de dados.

Conclusão

Este script PHP trata do processamento do formulário de cadastro, realizando a validação de dados, verificação de duplicatas (CPF e e-mail) no banco de dados e inserção segura dos dados do novo usuário. As mensagens de alerta são utilizadas para fornecer feedback ao usuário.

▼ cad.style.css

Estilo Geral

corpo **(`body`):**

```
body {
   background: linear-gradient(to bottom, #3498db, #2980b9);
```

- Define um fundo gradiente azul escuro para azul claro na página.

```
color: #333;
```

- Define a cor do texto principal como um tom de cinza escuro.

```
font-family: 'Arial', sans-serif;
```

- Especifica a fonte do texto como Arial ou, se indisponível, qualquer fonte sans-serif padrão.

```
margin: 0;
padding: 0;
```

- Remove as margens e o preenchimento padrão para evitar espaçamentos indesejados.

Estilo do Logotipo

```
.logo-container {
    text-align: center;
```

- Centraliza o texto (no caso, o logotipo) no centro.

```
margin: 20px 0; /* Margem superior e inferior de 20 pixels */
}
```

- Define margens superior e inferior de 20 pixels para o contêiner do logotipo.

```
.logo {
    max-width: 100%; /* Garante que a logo nÃ£o ultrapasse o contÃªiner */
    height: auto; /* Altura automÃ¡tica para manter a proporÃ§Ã£o original */
}
```

- Garante que a logo não ultrapasse o contêiner, ajustando automaticamente a altura para manter a proporção original.

Estilo do Formulário:

```
form {
    background-color: rgba(255, 255, 255, 0.9);
    padding: 20px; /* Preenchimento interno do formulÃ¡rio */
    border-radius: 8px; /* Borda arredondada com raio de 8 pixels */
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Sombra ao redor do formulÃ¡rio */
    width: 300px; /* Largura do formulÃ¡rio */
```

```
    margin: 0 auto; /* Centraliza o formulÃ¡rio na pÃ¡gina */
}
```

- Define estilos para o formulário, como cor de fundo com opacidade, preenchimento interno, borda arredondada, sombra e centralização na página.

```
form label {
    display: block; /* xibe labels em uma nova linha */
    margin-bottom: 8px; /* Margem inferior entre as labels */
}
```

• Configura as labels para serem exibidas em uma nova linha e adiciona uma margem inferior.

```
form input {
    width: 100%; /* Largura total do campo de entrada */
    padding: 10px; /* Preenchimento interno do campo de entrada */
    margin-bottom: 15px; /* Margem inferior entre os campos de entrada */
    box-sizing: border-box; /* O tamanho total do elemento inclui a borda e o preenchimento,
}
```

- Define estilos para os campos de entrada, como largura total, preenchimento interno, margem inferior e box-sizing para incluir borda e preenchimento no tamanho total do elemento.

```
form input[type="submit"] {
background-color: #3498db; /* Cor de fundo para o botÃ£o de envio */
color: #fff; /* Cor do texto do botÃ£o de envio */
cursor: pointer; /* Mostra o cursor como um ponteiro ao passar por cima */
}
```

• Estiliza o botão de envio do formulário, definindo cor de fundo, cor do texto e cursor.

```
form input[type="submit"]:hover {
    background-color: #2980b9; /* Cor de fundo mais escura ao passar o mouse por cima do botÃ
}
```

• Modifica a cor de fundo do botão de envio ao passar o mouse por cima.

```
button {
    background-color: #ddd; /* Cor de fundo para o botÃ£o VOLTAR */
    padding: 10px; /* Preenchimento interno do botÃ£o VOLTAR */
    border: none; /* Remove a borda padrÃ£o do botÃ£o */
    cursor: pointer; /* Mostra o cursor como um ponteiro ao passar por cima */
}
```

- Define estilos para o botão "VOLTAR", como cor de fundo, preenchimento interno, remoção da borda padrão e cursor.

```
button:hover {
    background-color: #ccc; /* Cor de fundo mais escura ao passar o mouse por cima do botÃ£o
}
```

- Muda a cor de fundo do botão "VOLTAR" para um tom mais escuro ao passar o mouse por cima.

▼ style.css

```
body {
            background: linear-gradient(to bottom, #3498db, #2980b9);
            color: #333;
            font-family: 'Arial', sans-serif;
```

```
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }
```

• Define o estilo geral do corpo da página, incluindo um gradiente de fundo, cor de texto, fonte, configuração de layout flexível (coluna), centralização vertical e horizontal (usando `align-items` e `justify-content`), e altura de 100% da viewport (`100vh`).

Estilo do logotipo

```
.logo-container {
            margin-bottom: 20px;
        }
```

• Adiciona uma margem inferior ao contêiner da logo para dar espaço entre a logo e o formulário.

```
.logo {
            max-width: 100%;
            height: auto;
        }
```

- Garante que a logo não ultrapasse o contêiner, ajustando automaticamente a altura para manter a proporção original.

```
form {
            background-color: rgba(255, 255, 255, 0.9);
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            width: 300px;
            text-align: center;
        }
```

- Define estilos para o formulário, como cor de fundo com opacidade, preenchimento interno, borda arredondada, sombra e largura fixa e centralização do texto.

```
form p {
            margin-bottom: 20px;
        }
```

- Adiciona uma margem inferior para dar espaçamento entre os parágrafos dentro do formulário.

```
form input {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            box-sizing: border-box;
        }
```

- Define estilos para os campos de entrada, como largura total, preenchimento interno, margem inferior e configuração de box-sizing para incluir borda e preenchimento no tamanho total do elemento

```
form input[type="submit"] {
            background-color: #3498db;
```

```
            color: #fff;
            cursor: pointer;
        }
```

- Define o estilo do botão de envio com uma cor de fundo azul, texto branco e cursor de ponteiro.

```
form input[type="submit"]:hover {
            background-color: #2980b9;
        }
```

- Muda a cor de fundo do botão para um tom mais escuro ao passar o mouse por cima.

Estilo na mensagem de erro

```
.error-message {
            color: #e74c3c;
            margin-top: 10px;
        }
```

• Define estilos para a mensagem de erro, como cor vermelha e uma margem superior para separação.

```
.radio-group {
            display: flex;
        }

        .radio-group input {
            margin-right: 0.5px; /* Adicione margem entre os botões de rádio, se necessário *


        .radio-group label {
            margin-right: 1px; /* Adicione margem entre os rótulos, se necessário */
        }
```

- Esse trecho adiciona estilos para um grupo de botões de rádio, utilizando o layout flexível (`display: flex`). Ele também inclui margens entre os botões de rádio e rótulos para melhor espaçamento visual

▼ rel.style.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    display: flex;
    align-items: center;
    justify-content: center;
    height: 40vh;
}
```

- **Objetivo:** Definir estilos gerais para o corpo da página.
- **Explicação:**
  - `font-family`: Define a família de fontes utilizada na página como Arial ou fontes sans-serif.
  - `background-color`: Estabelece a cor de fundo como um tom de cinza claro (#f4f4f4).
  - `margin`, `padding`: Remove margens e preenchimentos padrão do corpo.
  - `display`: Utiliza o modelo flex para centralizar conteúdo na vertical e horizontal.

- `height`: Define a altura do corpo como 40% da altura da visualização da janela (viewport height).

```
.container {
    text-align: center;
}
```

- **Objetivo:** Ajustar o estilo do container principal.
- **Explicação:**
  - `text-align`: Centraliza o texto no container.

```
h1 {
    color: #333;
}
```

- **Objetivo:** Estilizar o título principal.
- **Explicação:**
  - `color`: Define a cor do texto do título como um tom escuro de cinza (#333).

```
table {
    width: 100%; /* Ajuste a largura conforme necessário */
    border-collapse: collapse;
    margin-top: 20px;
}
```

- **Objetivo:** Estabelecer estilos para a tabela.
- **Explicação:**
  - `width`: Define a largura da tabela como 100% da largura disponível.
  - `border-collapse`: Remove espaçamento entre as bordas das células.
  - `margin-top`: Adiciona um espaçamento superior de 20 pixels.

```
table, th, td {
    border: 1px solid #ddd;
}
```

- **Objetivo:** Adicionar bordas às células da tabela.
- **Explicação:**
  - `border`: Adiciona uma borda de 1 pixel com uma cor cinza clara (#ddd).

```
th {
    background-color: #4CAF50;
    color: white;
}
```

- **Objetivo:** Estilizar as células do cabeçalho da tabela.
- **Explicação:**
  - `background-color`: Define a cor de fundo como verde (#4CAF50).
  - `color`: Define a cor do texto como branco.

```
tr:hover {
    background-color: #f5f5f5;
}
```

- **Objetivo:** Adicionar um destaque quando o mouse passa sobre uma linha da tabela.
- **Explicação:**
  - `background-color`: Muda a cor de fundo para um tom mais claro (#f5f5f5) ao passar o mouse sobre uma linha.

▼ relatorio.php

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Página de Relatórios</title>
    <link rel="stylesheet" type="text/css" href="rel.style.css">
</head>
```

- **Objetivo:** Definir a estrutura inicial do documento HTML.
- **Explicação:**
  - `<!DOCTYPE html>`: Declaração do tipo de documento.
  - `<html lang="pt-br">`: Elemento raiz do HTML com atributo de idioma.
  - `<head>`: Contém informações sobre o documento, como o título e os estilos.
  - `<meta charset="UTF-8">`: Define o conjunto de caracteres como UTF-8 para suportar caracteres especiais.
  - `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Configura a visualização responsiva em dispositivos móveis.
  - `<title>`: Título da página exibido na barra de título do navegador.
  - `<link rel="stylesheet" type="text/css" href="rel.style.css">`: Vínculo ao arquivo de estilo CSS.

```
<body>



    <div class="container">

        <h1>RELATÓRIO DE ACESSOS</h1>
```

- **Objetivo:** Estruturar o conteúdo principal da página.
- **Explicação:**
  - `<body>`: Contém todo o conteúdo visível da página.
  - `<div class="container">`: Container principal que agrupa o conteúdo.
  - `<h1>`: Título principal da página.

```
<form method="post" action="" class="search-form">
    <div class="form-group">
        <label for="data">Pesquisar por Data:</label>
        <input type="date" id="data" name="data">
    </div>

    <div class="form-group">
        <label for="cpf">Pesquisar por CPF:</label>
        <input type="text" id="cpf" name="cpf" placeholder="Digite o CPF">
    </div>
```

```
    <div class="form-group">
        <label for="mac">Pesquisar por MAC:</label>
        <input type="text" id="mac" name="mac" placeholder="Digite o MAC">
    </div>

    <div class="form-group">
        <label for="nome">Pesquisar por Nome:</label>
        <input type="text" id="nome" name="nome" placeholder="Digite o Nome">
    </div>

    <div class="form-group">
        <input type="submit" value="Pesquisar">
    </div>
</form>
```

- **Objetivo:** Incluir um formulário para permitir a filtragem dos resultados.
- **Explicação:**
  - `<form method="post" action="" class="search-form">`: Formulário usando o método POST e sem ação definida (será a própria página).
  - Campos de filtro para data, CPF, MAC e Nome.
  - `<input type="submit" value="Pesquisar">`: Botão de envio para acionar a pesquisa.

```
<?php
        include '../conexao.php';

        // Processar os filtros
        $filtro_data = isset($_POST['data']) ? $_POST['data'] : '';
        $filtro_cpf = isset($_POST['cpf']) ? $_POST['cpf'] : '';
        $filtro_mac = isset($_POST['mac']) ? $_POST['mac'] : '';
        $filtro_nome = isset($_POST['nome']) ? $_POST['nome'] : '';

        // Montar a consulta SQL com base nos filtros
        $sql = "SELECT nome, cpf, ip, mac, hora_login, hora_logout FROM usuarios_log WHERE 1'
```

- **Objetivo:** Conectar ao banco de dados e processar os filtros para construir a consulta SQL.
- **Explicação:**
  - `include '../conexao.php';`: Inclui o arquivo de conexão ao banco de dados.
  - Processa os filtros (data, CPF, MAC, Nome) vindos do formulário.
  - Constrói a consulta SQL considerando os filtros.

```
// Montar a consulta SQL com base nos filtros
        $sql = "SELECT nome, cpf, ip, mac, hora_login, hora_logout FROM usuarios_log WHERE 1'

        if (!empty($filtro_data)) {
            $sql .= " AND DATE(hora_login) = '$filtro_data'";
        }

        if (!empty($filtro_cpf)) {
            $sql .= " AND cpf LIKE '%$filtro_cpf%'";
        }


        if (!empty($filtro_mac)) {
            $sql .= " AND mac LIKE '%$filtro_mac%'";
```

```
        }

        if (!empty($filtro_nome)) {
            $sql .= " AND nome LIKE '%$filtro_nome%'";
        }

        $result = $conn->query($sql);

        if (!$result) {
            die("Erro na consulta: " . $conn->error);
        }

        if ($result->num_rows > 0) {
            // Exibir os resultados
            echo '<table>';
            echo '<tr>
                    <th>Nome</th>
                    <th>CPF</th>
                    <th>IP</th>
                    <th>MAC</th>
                    <th>Hora de Login</th>
                    <th>Hora de Logout</th>
                </tr>';

            while ($row = $result->fetch_assoc()) {
                echo '<tr>';
                echo '<td>' . $row['nome'] . '</td>';
                echo '<td>' . $row['cpf'] . '</td>';
                echo '<td>' . $row['ip'] . '</td>';
                echo '<td>' . $row['mac'] . '</td>';
                echo '<td>' . $row['hora_login'] . '</td>';
                echo '<td>' . $row['hora_logout'] . '</td>';
                echo '</tr>';
            }

            echo '</table>';
        } else {
            echo 'Nenhum registro encontrado com os filtros aplicados.';
        }
        ?>
```

- **Objetivo:** Exibir os resultados da consulta em uma tabela.
- **Explicação:**
  - Verifica se há resultados na consulta.
  - Se houver resultados, exibe-os em uma tabela HTML.
  - Caso contrário, informa que nenhum registro foi encontrado com os filtros aplicados.

```
</div>

</body>
</html>
```

- **Objetivo:** Finalizar o corpo do documento HTML.
- **Explicação:**
  - `<div class="container">`: Fecha o container principal.

- `</body>`: Fecha o corpo do documento HTML.
- `</html>`: Fecha o elemento raiz do HTML.