

**UNIPÊ- CENTRO UNIVERSITÁRIO DE JOÃO PESSOA**

**CURSO: CIÊNCIA DA COMPUTAÇÃO**

**TURMA: P2-B**

**Relatório de Projeto da Disciplina programação Web**

**Agenda Telefônica**

**Alunos:**

Diego da Silva Feitosa (RGM 37625446)

José Kleyton Francelino da silva Filho (RGM 38577305)

**João Pessoa- PB 2024**

## Introdução ao Projeto: Agenda Telefônica

O projeto **Agenda Telefônica** é uma aplicação web desenvolvida em **PHP**, com integração a um banco de dados **MySQL**, projetada para gerenciar contatos de forma prática e eficiente. A ferramenta permite que usuários cadastrem, editem, excluam e consultem informações de contatos, como nome, telefone, e-mail, endereço e data de nascimento.

O código PHP foi desenvolvido para configurar automaticamente um banco de dados MySQL denominado **agenda\_telefonica**, criando sua estrutura básica para armazenar informações de contatos e usuários.

---

### Etapas Realizadas Arquivo db\_agenda.php

#### 1. Conexão com o Banco de Dados

```
1  <?php
2
3  $host = 'localhost';
4  $user = 'root';
5  $password = '';
6  $dbname = 'agenda_telefonica';
7
8  // Conecta ao servidor de banco de dados
9  $conexao = new mysqli($host, $user, $password);
10
11 // Verifica se a conexão foi bem-sucedida
12 if ($conexao->connect_error) {
13     die("Falha na conexão: " . $conexao->connect_error);
14 }
15
```

- Foi estabelecida uma conexão com o servidor MySQL utilizando o host **localhost**, o usuário **root** e senha vazia.
- A conexão foi validada para evitar erros antes de prosseguir.

#### 2. Criação do Banco de Dados

- O banco de dados **agenda\_telefonica** foi criado caso não existisse.

```
16 // Cria o banco de dados se não existir
17 $sql = "CREATE DATABASE IF NOT EXISTS $dbname";
18 if ($conexao->query($sql) === TRUE) {
19     // Seleciona o banco de dados
20     $conexao->select_db($dbname);
21
```

### 3. Criação das Tabelas

- **Tabela `tb_contato`**: Responsável por armazenar dados dos contatos, incluindo:
  - Identificador único (`idContato`).
  - Nome, e-mail, telefone, endereço e data de nascimento.
- **Tabela `tb_usuarios`**: Responsável por gerenciar os usuários do sistema, incluindo:
  - Nome de login (`loginUser`), senha (`senhaUser`) e nome do usuário (`nomeUser`).

```
// Cria a tabela tb_contato se não existir
$sql = "CREATE TABLE IF NOT EXISTS tb_contato (
    idContato INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    nomeContato VARCHAR(200) NOT NULL,
    emailContato VARCHAR(100) NOT NULL,
    telefoneContato VARCHAR(50) NOT NULL,
    enderecoContato VARCHAR(200) NOT NULL,
    data_nasc_Contato DATE NOT NULL
);
```

```
// Cria a tabela tb_usuarios se não existir
$sql = "CREATE TABLE IF NOT EXISTS tb_usuarios (
    loginUser VARCHAR(45) NOT NULL,
    senhaUser VARCHAR(64) NOT NULL,
    nomeUser VARCHAR(45) NOT NULL,
    PRIMARY KEY (loginUser)
);
```

---

### Validações

- Todas as operações de criação foram verificadas, e erros potenciais (conexão, criação de tabelas ou banco) foram tratados com mensagens específicas.

```
if ($conexao->query($sql) === FALSE) {
    die("Erro na criação da tabela tb_contato: " . $conexao->error);
}
```

## Resultados

- Banco de dados e tabelas criados ou validados com sucesso, permitindo que o sistema esteja preparado para armazenar dados.
  - Estrutura flexível para novos contatos e usuários.
- 

## Funcionalidades arquivo Index.php

### 1. Autenticação do Usuário

- Verifica se o usuário está logado por meio de variáveis de sessão (`loginUser`, `senhaUser` e `nomeUser`).
- Confirma a existência do usuário no banco de dados (`tb_usuarios`) antes de conceder acesso ao sistema.
- Redireciona para a página de login caso a autenticação falhe.

```
1 <?php
2 // Inclui o arquivo que contém a conexão com o banco de dados
3 include("db/db_agenda.php");
4
5 // Inicia a sessão para acessar as variáveis de sessão
6 session_start();
7
8 // Verifica se as variáveis de login e senha estão configuradas na sessão
9 if (isset($_SESSION["loginUser"]) && isset($_SESSION["senhaUser"])) {
10     // Atribui os valores da sessão para variáveis locais
11     $loginUser = $_SESSION["loginUser"];
12     $senhaUser = $_SESSION["senhaUser"];
13     $nomeUser = $_SESSION["nomeUser"];
14
15     // Executa a consulta no banco de dados para verificar o usuário
16     $sql = "SELECT * FROM tb_usuarios WHERE loginUser = '{$loginUser}' AND senhaUser = '{$senhaUser}'";
17     $rs = mysqli_query($conexao, $sql);
18     $dados = mysqli_fetch_assoc($rs); // Recupera os dados do usuário
19     $linha = mysqli_num_rows($rs); // Conta o número de registros encontrados
```

```
    // Se não encontrar o usuário no banco de dados, desconecta a sessão e redireciona para o login
    if ($linha == 0) {
        session_unset(); // Limpa as variáveis de sessão
        session_destroy(); // Destroi a sessão
        header('Location: login.php'); // Redireciona para a página de login
        exit(); // Interrompe a execução do código
    }
    else {
        // Caso o usuário não esteja logado, redireciona para a página de login
        header('Location: login.php');
        exit(); // Interrompe a execução do código
    }
}
```

## 2. Navegação Dinâmica

- A funcionalidade de menu dinâmico carrega diferentes páginas com base no parâmetro `menuop` passado pela URL:
  - `home`: Página inicial.
  - `contatos`: Lista de contatos.
  - `cad-contatos`: Cadastro de contatos.
  - `inserir-contato`: Inserção de contatos.
  - `editar-contato`: Edição de contatos.
  - `excluir-contato`: Exclusão de contatos.
  - `atualizar-contato`: Atualização de contatos.
  - Padrão: Página inicial.

```
<main>
  <div class="container">
    <?php
      // Verifica qual página do menu foi selecionada e inclui o conteúdo correspondente
      $menuop = (isset($_GET["menuop"])) ? $_GET["menuop"] : "home";
      switch ($menuop) {
        case 'home':
          include("paginas/home/home.php"); // Inclui a página home
          break;
        case 'contatos':
          include("paginas/contatos/contatos.php"); // Inclui a página de contatos
          break;
        case 'cad-contatos':
          include("paginas/contatos/cad-contato.php"); // Inclui a página para cadastro de contatos
          break;
        case 'inserir-contato':
          include("paginas/contatos/inserir-contato.php"); // Inclui a página para inserir novo contato
          break;
        case 'editar-contato':
          include("paginas/contatos/editar-contato.php"); // Inclui a página para editar um contato
          break;
        case 'excluir-contato':
          include("paginas/contatos/excluir-contato.php"); // Inclui a página para excluir um contato
          break;
        case 'atualizar-contato':
          include("paginas/contatos/atualizar-contato.php"); // Inclui a página para atualizar um contato
          break;
        default:
          include("paginas/home/home.php"); // Inclui a página home como padrão
          break;
      }
    >
  </div>
</main>
```

## 3. Interface do Usuário

- Interface estilizada com **Bootstrap**:
    - Barra de navegação responsiva.
    - Ícones para facilitar a usabilidade.
    - Links para logout e exibição do nome do usuário logado.
  - Rodapé fixo com a versão do sistema.
  - Inclusão de estilos personalizados em `css/estilo-padrao.css`.
-

## Funcionalidades Implementadas

### Página para os arquivos referente a contatos

#### 1. Pesquisa de Contatos:

- Permite que o usuário pesquise contatos por ID ou nome. O termo de pesquisa é enviado via formulário **POST**.
- Caso nenhum termo seja informado, todos os contatos são exibidos.

```
1 <?php
2     include ("db/db_agenda.php");
3
4     // Variável da pesquisa - Se não houver um valor passado, define como uma string vazia
5     $txt_pesquisa = isset($_POST["txt_pesquisa"]) ? $_POST["txt_pesquisa"] : "";
6
7     ?>
```

#### 2. Paginação:

- Exibe os contatos em páginas com um limite de 5 registros por página.
- Implementa links para navegar entre páginas, incluindo:
  - Primeira página
  - Página anterior
  - Próximas páginas (com limite dinâmico de 5 páginas ao redor da atual)
  - Última página

```
<!-- Exibição da paginação -->
<ul class="pagination justify-content-center">
<?php
// Conta o total de registros de contatos
$sqlTotal = "SELECT idContato FROM tb_contato";
$qrTotal = mysqli_query($conexao, $sqlTotal) or die(mysqli_error($conexao));
$numTotal = mysqli_num_rows($qrTotal);

// Calcula o número total de páginas
$totalPagina = ceil($numTotal / $quantidade);

// Exibe o total de registros
echo '<li class="page-item"><span class="page-link">Total de registros: ' . $numTotal . '</span></li>';

// Link para a primeira página
echo '<li class="page-item"><a class="page-link" href="?menuop=contatos&pagina=1">Primeira Página</a></li>';

// Link para a página anterior, se houver
if ($pagina > 1) {
    ?>
    <li class="page-item"><a class="page-link" href="?menuop=contatos&pagina=?php echo $pagina - 1; ?>"><< </a></li>
    <?php
}
```

```
// Exibe os números das páginas de forma dinâmica, limitando o alcance para 5 páginas ao redor da página atual
for ($i = 1; $i <= $totalPagina; $i++) {
    if($i >= ($pagina-5) && $i <= ($pagina+5)){
        if ($i == $pagina) {
            echo "<li class='page-item active'><span class='page-link'>$i</span></li>";
        } else {
            echo "<li class='page-item'><a class='page-link' href='\"?menuop=contatos&pagina=$i\">$i</a></li>";
        }
    }
}
```



### 3. Exibição dos Contatos:

- Os dados dos contatos são apresentados em uma tabela estilizada com Bootstrap.
- As colunas incluem:
  - ID do contato
  - Nome
  - Telefone
  - E-mail
  - Endereço
  - Data de nascimento (formatada para `dd/mm/yy`)
- Para cada contato, são fornecidos botões:
  - Editar: Redireciona para a página de edição do contato.
  - Excluir: Redireciona para a funcionalidade de exclusão do contato.

### 4. Contagem de Registros:

- Exibe o total de registros encontrados no banco de dados.
- 

## Formulário de Cadastro de Contato:

### 1. Estrutura do Formulário:

- Inclui campos para:
  - Nome
  - Telefone
  - E-mail
  - Endereço
  - Data de nascimento
- Cada campo está acompanhado de mensagens de feedback para orientar o usuário.

### 2. Validação de Entrada:

- Usa classes do Bootstrap para fornecer validação visual de campos:
  - `valid-feedback` para entradas corretas.
  - `invalid-feedback` para mensagens de erro específicas, como campos obrigatórios ou formatos inválidos.

### 3. Envio dos Dados:

- Os dados do formulário são enviados ao servidor usando o método `POST` para o endpoint `index.php?menuop=inserir-contato`.

### 4. Design Responsivo:

- Adota uma estrutura responsiva com classes do Bootstrap (`col-*` e `d-flex`) para uma boa experiência em diferentes tamanhos de tela.
- Centraliza o formulário na tela com as classes `justify-content-center`, `align-items-center` e `min-vh-100`.

### 5. Estilização com Ícones:

- Usa ícones do Bootstrap (`bi-*`) para melhorar a experiência visual e identificar os campos de forma intuitiva.
-

## Inserção de Contato no Banco de Dados:

### Cabeçalho HTML:

- Inclui um título `<h3>` para identificar a página como a de inserção de contatos.

### Recebimento e Sanitização dos Dados:

- Os dados são recebidos por meio do método **POST** e sanitizados utilizando `mysqli_real_escape_string`. Essa abordagem previne ataques de SQL Injection, escapando caracteres especiais.

### Campos Sanitizados:

- `nomeContato`
- `emailContato`
- `telefoneContato`
- `enderecoContato`
- `data_nasc_Contato`

```
<?php
// Recebe os dados do formulário via POST e utiliza a função 'mysqli_real_escape_string' para evitar ataques de SQL Injection
$nomeContato = mysqli_real_escape_string($conexao, $_POST["nomeContato"]); // Escapa o nome do contato
$emailContato = mysqli_real_escape_string($conexao, $_POST["emailContato"]); // Escapa o e-mail do contato
$telefoneContato = mysqli_real_escape_string($conexao, $_POST["telefoneContato"]); // Escapa o telefone do contato
$enderecoContato = mysqli_real_escape_string($conexao, $_POST["enderecoContato"]); // Escapa o endereço do contato
$data_nasc_Contato = mysqli_real_escape_string($conexao, $_POST["data_nasc_Contato"]); // Escapa a data de nascimento do contato
```

### Criação do Comando SQL:

- Monta um comando **INSERT INTO** para adicionar os dados sanitizados à tabela `tb_contato`.
- A instrução SQL define as colunas e os valores correspondentes para o novo registro.
- `{telefoneCo`

```
$sql = "INSERT INTO tb_contato (
    nomeContato,
    emailContato,
    telefoneContato,
    enderecoContato,
    data_nasc_Contato)
VALUES(
    '{$nomeContato}',
    '{$emailContato}',
    '{$telefoneContato}',
    '{$enderecoContato}',
    '{$data_nasc_Contato}'
);"
```



### Execução do Comando SQL:

- Usa a função `mysqli_query` para executar a consulta no banco de dados.
- Caso ocorra algum erro durante a execução, o script é interrompido e uma mensagem de erro é exibida utilizando `die()`.

### Confirmação de Sucesso:

- Exibe uma mensagem estática, indicando que o registro foi inserido com sucesso, caso não haja erros durante o processo.

```
mysqli_query($conexao, $sql) or die("Erro ao executar a consulta." . mysqli_error($conexao));

// Exibe uma mensagem informando que o registro foi inserido com sucesso
echo "O registro foi inserido com sucesso.";
?>
```

---

### Edição do contato cadastrado:

#### Recebimento do ID do Contato:

- O ID do contato é obtido via `GET` na URL.
- A função `mysqli_real_escape_string` deveria ser usada para proteger contra SQL Injection, mas está ausente nesse trecho, o que requer atenção.

```
<?php
// Recebe o ID do contato a ser editado via parâmetro GET, usando 'mysqli_real_escape_string' para prevenir SQL Injection
$idContato = $_GET["idContato"];

// Consulta SQL para selecionar os dados do contato que será editado com base no ID
$sql = "SELECT * FROM agenda_telefonica.tb_contato tbcontato WHERE idContato= {$idContato}";
```

#### Consulta SQL:

- Um comando `SELECT` busca os dados do contato com base no `idContato`.
- A função `mysqli_query` executa a consulta, e os resultados são armazenados em um array associativo usando `mysqli_fetch_assoc`.

```
// Consulta SQL para selecionar os dados do contato que será editado com base no ID
$sql = "SELECT * FROM agenda_telefonica.tb_contato tbcontato WHERE idContato= {$idContato}";

// Executa a consulta SQL e verifica se houve erro na execução
$rs = mysqli_query($conexao, $sql) or die("Erro ao executar a consulta." . mysqli_error($conexao));

// Recupera os dados do contato retornado pela consulta SQL em formato de array associativo
$dados = mysqli_fetch_assoc($rs);
?>
```

### Estrutura HTML:

- Um formulário HTML exibe os dados do contato, permitindo sua edição. Os valores são preenchidos dinamicamente com os dados retornados pela consulta SQL.

### Campos do Formulário:

- ID: Exibido apenas para leitura.
- Nome, Telefone, E-mail, Endereço, Data de Nascimento: Permitem a edição dos valores, com validações de formulário fornecidas pelo Bootstrap.
- Mensagens de feedback (positivas e negativas) são fornecidas para melhorar a experiência do usuário.

### Envio do Formulário:

- O formulário é submetido via método **POST** para a rota **index.php?menuop=atualizar-contato**, que processará as atualizações no banco.

### Validação do Formulário:

- Inclui validações HTML5 e mensagens customizadas, garantindo que os campos obrigatórios sejam preenchidos corretamente antes de envio.

### Inclusão de Scripts:

- Um script externo **js/validation.js** para validações adicionais.
- O JavaScript do Bootstrap é carregado para funcionalidades interativas.

```
// Exemplo de Javascript para desabilitar o envio de formulários se houver campos inválidos
(() => {
  'use strict' // Ativa o modo estrito do JavaScript, prevenindo erros comuns

  // Seleciona todos os formulários que possuem a classe 'needs-validation' (necessitam de validação personalizada)
  const forms = document.querySelectorAll('.needs-validation')

  // Itera sobre todos os formulários selecionados
  Array.from(forms).forEach(form => {
    // Adiciona um ouvinte de evento para o evento de 'submit' (envio) do formulário
    form.addEventListener('submit', event => {
      // Verifica se o formulário é válido (se todos os campos obrigatórios foram preenchidos corretamente)
      if (!form.checkValidity()) {
        // Se o formulário não for válido, previne o envio do formulário
        event.preventDefault()
        // Impede que o evento continue propagando
        event.stopPropagation()
      }

      // Adiciona a classe 'was-validated' ao formulário, ativando os estilos de validação do Bootstrap
      form.classList.add('was-validated')
    }, false) // Define a captura do evento como falsa (não captura em fases anteriores do evento)
  })
})()
```

Implementação da funcionalidade de atualização de dados de um contato no banco de dados. Ele recebe os dados atualizados de um formulário e realiza a atualização do registro correspondente na tabela **tb\_contato**.

#### Recepção e Sanitização de Dados:

- Os dados são recebidos via **POST** e processados por **mysqli\_real\_escape\_string** para prevenir SQL Injection.
- Campos processados:
  - ID do contato (**idContato**).
  - Informações do contato: Nome, E-mail, Telefone, Endereço e Data de Nascimento.

#### Comando SQL:

- Um comando **UPDATE** é usado para modificar o registro na tabela **tb\_contato** correspondente ao **idContato**.
- 

```
// Comando SQL para atualizar os dados na tabela tb_contato
$sql = "UPDATE agenda_telefonica.tb_contato SET
    nomeContato = '{$nomeContato}',
    emailContato = '{$emailContato}',
    telefoneContato = '{$telefoneContato}',
    enderecoContato = '{$enderecoContato}',
    data_nasc_Contato = '{$data_nasc_Contato}'
    WHERE idContato = '{$idContato}'";
```

#### Execução da Consulta:

- A consulta é executada utilizando a função **mysqli\_query**.
- Caso haja um erro, a função **die** exibe a mensagem de erro retornada pelo banco.

#### Mensagem ao Usuário:

- Se a operação for bem-sucedida, uma mensagem é exibida: "O registro foi atualizado com sucesso."

```
// Executando a consulta no banco de dados
mysqli_query($conexao, $sql) or die("Erro ao executar a consulta." . mysqli_error($conexao));

// Mensagem de sucesso
echo "O registro foi atualizado com sucesso.";
>
```

---

Exclusão de um contato do banco de dados, com base no ID fornecido. O registro correspondente na tabela **tb\_contato** é removido permanentemente.

Recepção do ID do Contato:

- O ID do contato a ser excluído é recebido via parâmetro **GET** na URL.
- Utiliza **mysqli\_real\_escape\_string** para sanitizar o dado recebido e evitar SQL Injection.

```
<?php
// Recebe o ID do contato a ser excluído via parâmetro GET, e usa 'mysqli_real_escape_string' para prevenir SQL Injection
$idContato = mysqli_real_escape_string($conexao, $_GET["idContato"]);
```

Comando SQL:

- Um comando **DELETE** é utilizado para remover o registro correspondente ao ID fornecido.
- Exemplo de comando gerado:

```
// Comando SQL para excluir o registro da tabela 'tb_contato' com o ID correspondente
$sql = "DELETE FROM tb_contato WHERE idContato= '{$idContato}'";

// Executa a consulta SQL para excluir o registro do banco de dados
// Caso haja um erro, a função 'mysqli_query' interrompe o script e exibe uma mensagem de erro
mysqli_query($conexao, $sql) or die("Erro ao excluir o registro. ". mysqli_error($conexao));

// Exibe uma mensagem informando que o registro foi excluído com sucesso
echo "Registro excluído com sucesso!";
?>
```

---

Tela de Login:

#### 1. Validação do Login

- Verifica se os campos **loginUser** e **senhaUser** foram enviados via método **POST**.
- Sanitiza as entradas utilizando **mysqli\_escape\_string** para prevenir SQL Injection.
- Aplica um hash SHA256 na senha fornecida, comparando-a com o hash armazenado no banco.

## 2. Consulta ao Banco de Dados

- O código utiliza a consulta SQL:  
sql  
Copiar código
- Retorna o usuário correspondente (se existir).
- Conta o número de registros encontrados usando `mysqli_num_rows`.

```
// SQL para verificar a existência do usuário e da senha no banco de dados
$sql = "SELECT * FROM tb_usuarios WHERE loginUser = '{$loginUser}' AND senhaUser = '{$senhaUser}'";
$rs = mysqli_query($conexao, $sql); // Executa a consulta
$dados = mysqli_fetch_assoc($rs); // Armazena os dados do usuário
$linha = mysqli_num_rows($rs); // Conta quantos resultados foram encontrados
```

## 3. Autenticação do Usuário

- Sucesso:
  - Inicia uma sessão com `session_start()`.
  - Armazena informações do usuário (`loginUser`, `senhaUser`, e `nomeUser`) na variável global `$_SESSION`.
  - Redireciona para a página principal (`index.php`) com `header('Location: index.php')`.

```
// Se o usuário for encontrado, inicia a sessão e redireciona para a página principal
if ($linha > 0) {
    session_start(); // Inicia a sessão
    $_SESSION["loginUser"] = $loginUser; // Armazena o login do usuário na sessão
    $_SESSION["senhaUser"] = $senhaUser; // Armazena a senha criptografada na sessão
    $_SESSION["nomeUser"] = $dados["nomeUser"]; // Armazena o nome do usuário na sessão

    header('Location: index.php'); // Redireciona para a página principal
} else {
```

- Falha:
  - Exibe uma mensagem de erro indicando que o login falhou.

## 4. Feedback Visual

- Mensagem de erro exibida dinamicamente no formulário, usando a variável `$msg_error`.
- Validação de campos no formulário com Bootstrap (mensagens de feedback de sucesso ou erro para entradas inválidas).

## 5. Interface do Usuário

- Uso de Bootstrap para estilização:
  - Estrutura responsiva com a classe `container` e colunas ajustáveis.

- Ícones do Bootstrap Icons para campos de entrada.
- Formulário limpo e amigável, exibindo campos de login e senha.

## 6. Scripts Adicionais

- Bootstrap JS e Popper.js para funcionalidade interativa.
- Inclusão de um script de validação personalizada em **validation.js**.

```
<!-- Inclusão dos scripts JS do Bootstrap -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaA"
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jiz"
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689a"

<!-- Script de validação personalizada do formulário -->
<script src="./js/validation.js"></script>
body>
html>
```

gerador de senha criptografada usando o algoritmo SHA-256, com o objetivo de proteger a senha original antes de armazená-la ou utilizá-la em um contexto como autenticação.

## 1. Entrada

- A senha original é armazenada na variável **\$senha** com o valor "abcde"

```
<?php
//gerador de senha para o formulário de login.php
// Armazena a senha original
$senha = "abcde";
```

## 2. Criação do Hash

- O código utiliza a função **hash('sha256', \$senha)** para gerar um hash da senha.
  - SHA-256 é um algoritmo de hash criptográfico que gera uma saída fixa de 256 bits (64 caracteres em hexadecimal) independente do tamanho da entrada.
  - Essa função é unidirecional, ou seja, não é possível reverter o hash para recuperar a senha original.

```
// Cria um hash SHA-256 da senha
$senhaCriptografada = hash('sha256', $senha);
```



### 3. Saída

- O código exibe:
  - A senha original, como referência.
  - A versão criptografada da senha gerada pelo algoritmo SHA-256.

```
// Exibe a senha original e a senha criptografada
echo "Senha original: " . $senha . "<br>";
echo "Senha criptografada: " . $senhaCriptografada;
?>
```

---

### Logout da agenda telefônica:

#### 1. Encerrando a Sessão

##### 1. Remoção de Dados da Sessão:

- O comando `session_unset()` remove todas as variáveis armazenadas na sessão atual.
- Ele limpa o conteúdo, mas a sessão em si ainda existe até ser destruída.

#### Destruição da Sessão:

- O comando `session_destroy()` encerra a sessão, removendo completamente os dados e tornando o identificador de sessão inválido.

#### 2. Redirecionamento

- Após encerrar a sessão, o código redireciona o usuário para a página de login

`header()` envia um cabeçalho HTTP para redirecionar o navegador para outra página.

É uma prática comum após logout para garantir que o usuário tenha que se autenticar novamente.

```
<?php

session_unset();

session_destroy();

header('Location: login.php');
```

