Upload de Arquivos PHP

Upload de arquivos em PHP

- O upload de arquivos em PHP é uma funcionalidade muito importante para permitir que os usuários de um site possam enviar arquivos para o servidor:
  - como fotos
  - Documentos
  - Vídeos
  - entre outros
- Isso pode ser útil em diversas situações, como em plataformas de armazenamento de arquivos, fóruns, redes sociais, e-commerces, entre outros.

Upload de Arquivos em PHP

- Por isso, é importante que os desenvolvedores tenham um bom conhecimento das melhores práticas e técnicas para lidar com o upload de arquivos em PHP, a fim de garantir a segurança e integridade dos dados dos usuários e evitar problemas técnicos e de segurança.

Tipos de arquivos que podem ser enviados

- Em geral, os tipos de arquivos mais comuns que são permitidos para upload em PHP incluem:
  - Imagens (por exemplo, .jpg, .png, .gif)
  - Documentos (por exemplo, .pdf, .doc, .docx)
  - Arquivos de áudio e vídeo (por exemplo, .mp3, .mp4)
  - Arquivos compactados (por exemplo, .zip, .rar)

Upload de Arquivos em PHP

- O PHP possui funções nativas que facilitam o processo de upload de arquivos, permitindo que os desenvolvedores possam criar formulários que permitam que os usuários enviem arquivos para o servidor de maneira segura e eficiente.
- No entanto, o processo de upload de arquivos pode ser complexo e apresentar diversos desafios:
  - como a validação dos tipos e tamanhos de arquivos permitidos;
  - a prevenção de ataques de injeção de código;
  - a segurança dos arquivos enviados;
  - e a organização dos arquivos no servidor.

Tipos de arquivos que podem ser enviados

- Basicamente, **todos** os tipos de arquivos podem ser enviados, desde que sejam **permitidos** pela configuração do servidor web.
- No entanto, é importante considerar que alguns tipos de arquivos podem apresentar **riscos à segurança** do servidor e dos usuários, como **arquivos executáveis** (por exemplo, .exe) e arquivos com **código malicioso** (por exemplo, .php).
- Por isso, em alguns servidores web, esses tipos de arquivos são **bloqueados por padrão.**

Tipos de arquivos que podem ser enviados

- Ao permitir o upload de arquivos em uma aplicação web, é importante verificar se os arquivos enviados pelos usuários são **válidos** e **não representam um risco à segurança da aplicação ou dos usuários**.
- Isso pode ser feito por meio da validação do tipo de arquivo e da análise do conteúdo do arquivo antes de salvá-lo no servidor.

Preparação do ambiente

- Salve o arquivo HTML em um diretório do servidor web, acessível pela URL, por exemplo, http://localhost/upload/upload.html.

- Com a criação deste formulário, é possível permitir que os usuários enviem arquivos para o seu servidor web através de um formulário web.
- No entanto, ainda é necessário criar um script PHP que receba os arquivos enviados e os armazene no servidor.

Preparação do ambiente

- Para preparar o ambiente para o upload de arquivos em PHP, siga os seguintes passos:
  - Instale um servidor web em sua máquina, como o Apache.
  - Instale o PHP na sua máquina. Lembre-se de verificar a instalação do servidor web e a configuração do PHP.ini.
  - Crie uma página HTML básica com um formulário para envio de arquivos. Nesta página, é necessário que seja criado um formulário HTML que permita que o usuário selecione o arquivo a ser enviado. Por exemplo:

Upload de arquivos simples

Preparação do ambiente

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Envio de arquivo</title>
 </head>
 <body>
  <form action="upload.php" method="post" enctype="multipart/form-data">
   <label for="file">Escolha um arquivo:</label>
   <input type="file" id="file" name="file"><br><br>
   <input type="submit" value="Enviar">
  </form>
 </body>
</html>
```

Passos para upload de arquivos em PHP

- Criação do formulário HTML com um campo do tipo **`file`**, que permitirá ao usuário selecionar o arquivo a ser enviado para o servidor:

```
<form action="upload.php" method="post" enctype="multipart/form-data">
   <input type="file" name="arquivo">
   <input type="submit" value="Enviar">
</form>
```

Passos para upload de arquivo em PHP

- Processamento do envio do arquivo no lado do servidor, no arquivo **`upload.php`**:

```php
<?php
// Verifica se o arquivo foi enviado pelo formulário
if(isset($_FILES['arquivo'])) {

    // Define o diretório onde o arquivo será armazenado
    $diretorio = "uploads/";

    // Define o nome do arquivo a partir do nome original
    $nome_arquivo = basename($_FILES['arquivo']['name']);

    // Define o caminho completo do arquivo
    $caminho_arquivo = $diretorio . $nome_arquivo;

    // Move o arquivo para o diretório definido
    if(move_uploaded_file($_FILES['arquivo']['tmp_name'], $caminho_arquivo)) {
        echo "Arquivo enviado com sucesso!";
    } else {
        echo "Ocorreu um erro ao enviar o arquivo.";
    }

}
?>
```

Passos para upload de arquivos em PHP

- A linha de código **`$nome_arquivo = basename($_FILES['arquivo']['name']);`** tem como objetivo obter o nome do arquivo enviado pelo formulário HTML.
- O array **`$_FILES`** é preenchido com informações do arquivo enviado pelo formulário, como nome, tipo, tamanho, entre outras informações.
- A função **`basename()`** é usada para retornar apenas o nome do arquivo sem o caminho completo do diretório em que o arquivo está localizado.
- Isso é útil para garantir que apenas o nome do arquivo seja salvo em um **banco de dados**, por exemplo, em vez de todo o caminho do arquivo no sistema de arquivos do servidor.
- Então, no código em questão, **`$_FILES['arquivo']['name']`** retorna o nome original do arquivo, que é então passado para a função **`basename()`** para obter apenas o nome do arquivo. O resultado final é armazenado na variável **`$nome_arquivo`**.

Passos para upload de arquivos em PHP

- O que é $_FILES
- **$_FILES** é uma variável super global do PHP que é usada para armazenar informações sobre arquivos enviados pelo usuário através de um formulário de envio de arquivos.
- Ela é um array associativo que contém vários campos com informações sobre o arquivo, como nome, tipo, tamanho, nome temporário, erro e assim por diante.
- Esses campos são preenchidos automaticamente pelo PHP durante o processo de upload do arquivo.
- Com a ajuda da variável **$_FILES**, é possível manipular o arquivo enviado, como movê-lo para uma pasta específica, fazer validações e redimensioná-lo, entre outras coisas.

Passos para upload de arquivo em PHP

- No exemplo, verificamos se o arquivo foi enviado pelo formulário, definimos o diretório e o nome do arquivo a partir do nome original e movemos o arquivo para o diretório definido usando a função **`move_uploaded_file`**.
- Validação do arquivo enviado, verificando se ele atende aos requisitos desejados, como tipo de arquivo, tamanho máximo, entre outros.
- Armazenamento do arquivo em um diretório seguro no servidor, para que ele possa ser acessado posteriormente.
- Exibição de mensagens de erro ou sucesso para o usuário, informando se o upload foi realizado com sucesso ou não.

Passos para upload de arquivos em PHP

- A função **`move_uploaded_file`** em PHP é usada para mover um arquivo enviado para um formulário de upload de arquivos do diretório temporário para o diretório de destino no servidor.
- Ela recebe dois parâmetros: o primeiro é o caminho do arquivo temporário, que pode ser obtido através do array **`$_FILES`** (por exemplo, **`$_FILES['arquivo']['tmp_name']`**) e o segundo é o caminho de destino para onde o arquivo deve ser movido.
- Por exemplo, suponha que você queira mover um arquivo enviado por um formulário de upload com o nome "arquivo.txt" para a pasta "uploads" no servidor. Você pode fazer isso usando a função **`move_uploaded_file`**

Criação de Script PHP

- segue um exemplo de um formulário que permite o envio de um arquivo juntamente com outros campos de entrada de texto:

```
<!DOCTYPE html>
<html>
<head>
  <title>Formulário de Envio de Arquivo</title>
</head>
<body>

  <h1>Formulário de Envio de Arquivo</h1>

  <form action="upload.php" method="POST" enctype="multipart/form-data">
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome"><br><br>

    <label for="email">E-mail:</label>
    <input type="email" id="email" name="email"><br><br>

    <label for="arquivo">Arquivo:</label>
    <input type="file" id="arquivo" name="arquivo"><br><br>

    <input type="submit" value="Enviar">
  </form>

</body>
</html>
```

Criação de Script PHP

- Nesse exemplo, criamos um formulário HTML com três campos de entrada: "Nome", "E-mail" e "Arquivo". O campo "Arquivo" é do tipo **`file`**, o que permite ao usuário selecionar um arquivo do seu computador para ser enviado ao servidor.
- O atributo **`enctype="multipart/form-data"`** é necessário para que o formulário possa enviar arquivos.
- Ao enviar o formulário, ele será processado pelo script PHP "upload.php", que deve ser criado para receber e processar o arquivo enviado e os outros dados do formulário.
- Segue um exemplo básico de como poderia ser implementado o script PHP "upload.php":

Criação de Script PHP

```php
<?php
// Verifica se o formulário foi submetido
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Verifica se o arquivo foi enviado sem erros
    if (isset($_FILES["arquivo"]) && $_FILES["arquivo"]["error"] == 0) {
        $nome_arquivo = basename($_FILES["arquivo"]["name"]);
        $diretorio_destino = "uploads/" . $nome_arquivo;
        // Move o arquivo do diretório temporário para o diretório de destino
        if (move_uploaded_file($_FILES["arquivo"]["tmp_name"], $diretorio_destino)) {
            // Arquivo movido com sucesso
            // Recupera os outros campos do formulário
            $nome = $_POST["nome"];
            $email = $_POST["email"];
            // Faz o que for necessário com o arquivo e os outros campos
            echo "Arquivo enviado com sucesso!";
        } else {
            // Erro ao mover arquivo
            echo "Erro ao enviar arquivo.";
        }
    } else {
        // Erro no envio do arquivo
        echo "Erro no envio do arquivo.";
    }
}
?>
```

Criação de Script PHP

- Note que o script deve ter permissões de escrita no diretório de destino para que possa salvar o arquivo enviado.

Criação de script PHP

- Esse script primeiro verifica se o formulário foi submetido por meio do método **`POST`**.
- Em seguida, verifica se o arquivo foi enviado sem erros.
- Se o arquivo foi enviado com sucesso, ele é movido do diretório temporário para o diretório de destino especificado na variável **`$diretorio_destino`**.
- Depois disso, o script recupera os outros campos do formulário (nome e e-mail, nesse caso) e pode fazer o que for necessário com esses dados e com o arquivo enviado.

Verificação do tipo e tamanho do arquivo

- Para verificar o tipo e tamanho do arquivo enviado, podemos utilizar as variáveis **`$_FILES['arquivo']['type']`** e **`$_FILES['arquivo']['size']`**, respectivamente.
- Por exemplo, para verificar se o arquivo enviado é uma imagem do tipo JPEG e tem tamanho máximo de 2MB, podemos fazer o seguinte:

```
if($_FILES['arquivo']['type'] != 'image/jpeg') {
  echo "Apenas arquivos do tipo JPEG são permitidos";
  exit;
}

if($_FILES['arquivo']['size'] > 2000000) {
  echo "O arquivo deve ter no máximo 2MB";
  exit;
}
```

Exibição de mensagens de Erro

- Para exibir mensagens de erro para o usuário em caso de problemas com o envio do arquivo em PHP, podemos utilizar as variáveis globais **`$_FILES`** e **`$_FILES['arquivo']['error']`**.
- A variável **`$_FILES`** contém informações sobre o arquivo enviado, incluindo o código de erro que pode ser verificado na chave **`'error'`**. Os códigos de erro possíveis são:
  - UPLOAD_ERR_OK: Valor 0, indicando que não houve erros no upload do arquivo.
  - UPLOAD_ERR_INI_SIZE: Valor 1, indicando que o arquivo enviado excede o tamanho máximo definido na diretiva **`upload_max_filesize`** do php.ini.
  - UPLOAD_ERR_FORM_SIZE: Valor 2, indicando que o arquivo enviado excede o tamanho máximo definido no formulário HTML.
  - UPLOAD_ERR_PARTIAL: Valor 3, indicando que o arquivo foi enviado parcialmente.
  - UPLOAD_ERR_NO_FILE: Valor 4, indicando que nenhum arquivo foi enviado.
  - UPLOAD_ERR_NO_TMP_DIR: Valor 6, indicando que não foi possível salvar o arquivo no diretório temporário.
  - UPLOAD_ERR_CANT_WRITE: Valor 7, indicando que não foi possível salvar o arquivo no disco.
  - UPLOAD_ERR_EXTENSION: Valor 8, indicando que uma extensão do PHP interrompeu o upload do arquivo.

Upload de arquivos com tratamento de erros

Exibição de mensagens de erro

Para exibir mensagens de erro para o usuário, podemos usar uma estrutura condicional para verificar se há algum erro na variável **`$_FILES['arquivo']['error']`**. Em seguida, podemos exibir uma mensagem de erro correspondente ao código de erro encontrado. Por exemplo:

```
if ($_FILES['arquivo']['error'] !== UPLOAD_ERR_OK) {
  switch ($_FILES['arquivo']['error']) {
    case UPLOAD_ERR_INI_SIZE:
    case UPLOAD_ERR_FORM_SIZE:
      echo 'O arquivo enviado é muito grande.';
      break;
    case UPLOAD_ERR_PARTIAL:
      echo 'O arquivo foi enviado parcialmente.';
      break;
    case UPLOAD_ERR_NO_FILE:
      echo 'Nenhum arquivo foi enviado.';
      break;
    case UPLOAD_ERR_NO_TMP_DIR:
    case UPLOAD_ERR_CANT_WRITE:
      echo 'Não foi possível salvar o arquivo.';
      break;
    case UPLOAD_ERR_EXTENSION:
      echo 'O upload do arquivo foi interrompido por uma extensão do PHP.';
      break;
    default:
      echo 'Ocorreu um erro ao enviar o arquivo.';
  }
}
```

Tratamento de erros no upload de arquivos, incluindo verificações de segurança para prevenir ataques do tipo "file inclusion"

- Para tratar erros no upload de arquivos e prevenir ataques do tipo "file inclusion", é importante seguir algumas boas práticas de segurança. Algumas dicas importantes incluem:
  - Validar o tipo de arquivo: antes de aceitar o arquivo, é importante verificar se o seu tipo é permitido. Isso pode ser feito verificando a extensão do arquivo ou usando a função **`finfo_file`** para verificar o tipo real do arquivo.
  - Limitar o tamanho do arquivo: é importante definir um limite para o tamanho do arquivo que pode ser enviado para o servidor. Isso pode ser feito definindo a diretiva **`upload_max_filesize`** no arquivo php.ini ou usando a função **`ini_set`**.

Tratamento de erros

- Usar bibliotecas de upload seguras: existem várias bibliotecas de upload disponíveis para PHP que oferecem recursos avançados de segurança, como verificação de tipo de arquivo e remoção de caracteres inválidos no nome do arquivo. Algumas das bibliotecas mais populares incluem o Dropzone.js, o Fine Uploader e o Blueimp jQuery File Upload.

Tratamento de erros

- Renomear o arquivo: ao salvar o arquivo, é importante renomeá-lo para evitar que ele seja sobrescrito ou que o nome do arquivo contenha caracteres maliciosos. O nome do arquivo pode ser gerado aleatoriamente usando a função **`uniqid`** ou baseado no nome original do arquivo, mas removendo caracteres inválidos.
- Armazenar o arquivo fora do diretório web: ao salvar o arquivo, é importante armazená-lo fora do diretório web, para evitar que ele seja acessado diretamente pelos usuários. Isso pode ser feito definindo o diretório de destino como uma pasta fora do diretório raiz do servidor.
- Verificar a integridade do arquivo: ao receber o arquivo, é importante verificar sua integridade para garantir que ele não tenha sido corrompido durante o upload. Isso pode ser feito usando a função **`md5_file`** para calcular o hash MD5 do arquivo original e comparando com o hash do arquivo recebido.

Criação de um diretório no servidor para armazenar os arquivos enviados

- Para criar um diretório no servidor para armazenar os arquivos enviados, você pode utilizar a função **`mkdir()`** do PHP. Essa função cria um novo diretório com o nome especificado em um caminho fornecido.
- Aqui está um exemplo de como criar um diretório chamado "uploads" na raiz do seu servidor web:

```
if (!file_exists('/var/www/html/uploads')) {
    mkdir('/var/www/html/uploads', 0777, true);
}
```

Criação de diretório no servidor

- Neste exemplo, o diretório "uploads" é criado no caminho "/var/www/html" com permissões de leitura, gravação e execução para todos os usuários (0777). O terceiro parâmetro **`true`** especifica que a função deve criar diretórios pais recursivamente se eles não existirem.
- Depois de criar o diretório, você pode usar a função **`move_uploaded_file()`** para mover o arquivo enviado para o diretório recém-criado:

```
if (move_uploaded_file($_FILES['arquivo']['tmp_name'], '/var/www/html/uploads/' . $nome_arquivo)) {
    echo "Arquivo enviado com sucesso!";
} else {
    echo "Ocorreu um erro ao enviar o arquivo.";
}
```

Nome único para cada arquivo

- Para evitar conflitos entre arquivos com o mesmo nome que são enviados para o servidor, é comum adotar uma estratégia para criar nomes únicos para cada arquivo. Isso pode ser feito adicionando um prefixo ou sufixo ao nome original do arquivo, ou gerando um nome aleatório.
- Uma maneira comum de criar um nome único é usar a função `uniqid()`, que gera um identificador único baseado no horário atual e em um número aleatório. Por exemplo:

```
$nome_arquivo = uniqid() . '_' . $_FILES['arquivo']['name'];
```

Nome único para cada arquivo

- Isso irá concatenar o identificador único gerado pela função **`uniqid()`** com o nome original do arquivo enviado, separados por um sublinhado. Dessa forma, cada arquivo enviado terá um nome único e não haverá conflitos no servidor.

Permissões de acesso

- As permissões de acesso ao diretório criado para armazenar os arquivos enviados devem ser configuradas para garantir a segurança do sistema. Geralmente, as permissões recomendadas para um diretório de upload são:
  - Permissão de leitura e gravação para o proprietário (usuário que criou o diretório);
  - Permissão de leitura e gravação para um grupo específico de usuários (por exemplo, o grupo de usuários do servidor web);
  - Permissão de leitura para todos os usuários.
- Para configurar as permissões de acesso, você pode usar um programa de FTP ou um gerenciador de arquivos no painel de controle do servidor. O comando **`chmod`** também pode ser usado em um terminal para alterar as permissões de acesso.

Permissões de acesso

- Por exemplo, para definir as permissões do diretório para 755 (leitura, gravação e execução para o proprietário e leitura e execução para os demais usuários), você pode usar o seguinte comando no terminal:

```
chmod 755 /caminho/para/o/diretorio
```

Exibição dos arquivos enviados pelos usuários

- Em seguida, podemos percorrer esse array para obter apenas os arquivos (e não os diretórios). Podemos usar a função **`is_file`** para verificar se cada item do array é um arquivo ou não.

```
$files = scandir('caminho/do/diretorio/upload');

foreach ($files as $file) {
    if (is_file('caminho/do/diretorio/upload/' . $file)) {
        // $file é um arquivo
    }
}
```

Exibição e gerenciamento de arquivos enviados

Exibição dos arquivos enviados pelos usuários

- Para exibir a lista de arquivos enviados pelos usuários em uma página, podemos seguir os seguintes passos:
  - Primeiro, precisamos obter a lista de arquivos no diretório de upload. Podemos usar a função **`scandir`** do PHP para isso.
  - Essa função retorna um array com todos os arquivos e diretórios no caminho especificado.

```
$files = scandir('caminho/do/diretorio/upload');
```

Como excluir arquivos

- Para criar um formulário para exclusão de arquivos, você pode seguir os seguintes passos:
  - Crie um formulário HTML com um campo **`select`** que liste os arquivos disponíveis para exclusão, e um botão de **`submit`** para enviar o formulário:

```
<form method="post" action="excluir_arquivo.php">
  <label for="arquivo">Selecione o arquivo para excluir:</label>
  <select name="arquivo" id="arquivo">
    <option value="">Selecione...</option>
    <?php
      // Lista os arquivos do diretório de uploads
      $diretorio = "uploads/";
      $arquivos = glob($diretorio . "*");
      foreach ($arquivos as $arquivo) {
        // Exibe os nomes dos arquivos como opções do select
        echo '<option value="' . $arquivo . '">' . basename($arquivo) . '</option>';
      }
    ?>
  </select>
  <br>
  <button type="submit">Excluir</button>
</form>
```

Exibição dos arquivos enviados pelos usuários

- Com esses passos, podemos exibir a lista de arquivos enviados pelos usuários em uma página da web. É importante lembrar de tomar medidas de segurança para evitar que arquivos maliciosos sejam enviados e exibidos na lista, como verificar a extensão e o tipo do arquivo e configurar corretamente as permissões de acesso ao diretório de upload.

Exibição dos arquivos enviados pelos usuários

- Podemos exibir a lista de arquivos em uma tabela HTML. Podemos usar a tag **`<table>`** para criar a tabela, e dentro dela, usar a tag **`<tr>`** para cada linha e a tag **`<td>`** para cada coluna. Podemos usar a função **`filesize`** para obter o tamanho de cada arquivo em bytes.

```
$files = scandir('caminho/do/diretorio/upload');

echo '<table>';
echo '<tr><th>Arquivo</th><th>Tamanho</th></tr>';

foreach ($files as $file) {
    if (is_file('caminho/do/diretorio/upload/' . $file)) {
        echo '<tr>';
        echo '<td>' . $file . '</td>';
        echo '<td>' . filesize('caminho/do/diretorio/upload/' . $file) . ' bytes</td>';
        echo '</tr>';
    }
}

echo '</table>';
```

Excluir arquivos

- Crie um arquivo PHP (por exemplo, **`excluir_arquivo.php`**) para processar o formulário. Nesse arquivo, você pode verificar se o arquivo selecionado existe, e se o usuário tem permissão para excluí-lo. Em seguida, use a função **`unlink`** para excluir o arquivo:

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $arquivo = $_POST["arquivo"];

  // Verifica se o arquivo selecionado existe
  if (file_exists($arquivo)) {
    // Verifica se o usuário tem permissão para excluir o arquivo
    if (is_writable($arquivo)) {
      // Exclui o arquivo
      unlink($arquivo);
      echo "Arquivo excluído com sucesso!";
    } else {
      echo "Você não tem permissão para excluir este arquivo.";
    }
  } else {
    echo "O arquivo selecionado não existe.";
  }
}
?>
```

Excluir arquivos

- Ajuste as permissões do diretório de uploads para permitir a exclusão de arquivos pelo servidor web. Para isso, você pode usar o comando **`chmod`** no terminal:
- chmod 777 uploads/
- Esse comando dá permissão de leitura, escrita e execução para todos os usuários no diretório **`uploads`**.
- É importante tomar cuidado ao ajustar as permissões de um diretório, pois isso pode afetar a segurança do seu sistema.

Tratamento de erros na exclusão

- A seguir, um exemplo de código para exclusão de arquivos em PHP com tratamento de erros e confirmação do usuário:

```php
<?php
  // Diretório onde os arquivos foram armazenados
  $diretorio = 'uploads/';

  // Verifica se foi enviado um arquivo para exclusão
  if(isset($_POST['arquivo']) && !empty($_POST['arquivo'])) {
    $arquivo = $_POST['arquivo'];

    // Verifica se o arquivo existe
    if(file_exists($diretorio . $arquivo)) {
      // Verifica se o arquivo tem permissão de escrita
      if(is_writable($diretorio . $arquivo)) {
        // Pede uma confirmação do usuário antes de excluir o arquivo
        echo "Tem certeza que deseja excluir o arquivo '$arquivo'?";
        echo "<form method='post' action='excluir.php'>";
        echo "<input type='hidden' name='arquivo' value='$arquivo'>";
        echo "<input type='submit' name='confirmar' value='Sim'>";
        echo "<input type='button' value='Não' onclick='window.location=\"index.php\"'>";
        echo "</form>";
      } else {
        echo "O arquivo '$arquivo' não pode ser excluído porque não tem permissão de escrita.";
      }
    } else {
      echo "O arquivo '$arquivo' não existe.";
    }
  } elseif(isset($_POST['confirmar']) && !empty($_POST['arquivo'])) {
    // Exclui o arquivo se o usuário confirmar
    $arquivo = $_POST['arquivo'];
    unlink($diretorio . $arquivo);
    echo "O arquivo '$arquivo' foi excluído com sucesso.";
  } else {
    // Lista os arquivos no diretório
    $arquivos = array_diff(scandir($diretorio), array('..', '.'));

    echo "<ul>";
    foreach($arquivos as $arquivo) {
      echo "<li>$arquivo <a href='excluir.php?arquivo=$arquivo'>[excluir]</a></li>";
    }
    echo "</ul>";
  }
?>
```

Tratamento de erros na exclusão

- Para tratar erros na exclusão de arquivos em PHP, é importante realizar as seguintes ações:
  - Verificar se o arquivo a ser excluído realmente existe antes de tentar excluí-lo. Isso pode ser feito usando a função **`file_exists()`** do PHP.
  - Garantir que o arquivo a ser excluído tenha as permissões de escrita necessárias. Se o arquivo não tiver permissões de escrita, a função **`unlink()`** (usada para excluir arquivos) irá falhar. Isso pode ser feito usando a função **`chmod()`** do PHP.
  - Pedir uma confirmação do usuário antes de excluir o arquivo. Isso pode ser feito usando JavaScript ou uma caixa de diálogo de confirmação do PHP.

Download de arquivos

- Para permitir o download dos arquivos enviados pelos usuários, é necessário disponibilizá-los no servidor web e criar um link para download.
- Supondo que os arquivos tenham sido salvos no diretório "uploads" na raiz do servidor web, podemos criar um link para download com a tag **<a>** do HTML:

```
<a href="/uploads/nome-do-arquivo" download>Download</a>
```

Download de arquivos

- No exemplo acima, o atributo **`href`** contém o caminho do arquivo a ser baixado e o atributo **`download`** indica que o arquivo deve ser baixado em vez de ser aberto no navegador.
- Note que é importante garantir que apenas os arquivos permitidos sejam baixados, evitando assim que usuários mal-intencionados possam baixar arquivos com conteúdo malicioso ou danoso.
- Para isso, é necessário realizar a validação do tipo e tamanho do arquivo no momento do upload, como explicado anteriormente.