

DESENVOLVIMENTO DE SISTEMAS PHP



Mauricio Saraiva
Jeanine Barreto

Linguagem PHP com upload de arquivos

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Descrever a preparação de formulários para envio e recebimento de arquivos.
- Aplicar validação de tamanho e extensão e renomeação de arquivos.
- Construir a movimentação de arquivos e a gravação de endereço em banco de dados.

Introdução

O PHP é uma das linguagens de programação mais utilizadas atualmente. Em razão disso, existem funcionalidades fundamentais, e uma delas é o upload de arquivos com suas validações, que apresenta uma ampla utilização, desde sistemas de cadastro até sites de compra para a internet.

Neste capítulo, você irá estudar a preparação de formulários para o upload de arquivos, a validação de tamanho, extensão e renomeação de arquivos e, também, a movimentação de arquivos e a gravação do endereço em banco de dados.

Preparação de formulários para envio e recebimento de arquivos

O PHP é uma das linguagens de programação mais utilizadas atualmente, portanto, entender como realizar o upload de um arquivo por meio dela é fundamental. Para esse aprendizado vamos considerar um exemplo em que o usuário escolhe um arquivo e esse arquivo é copiado para uma pasta no servidor, utilizando o método post (PHP, c2001-2018b).

Neste caso, você pode iniciar criando uma pasta dentro de c:\xampp\htdocs, que é o local padrão para armazenar os arquivos em PHP utilizando o XAMPP,

chamada ua12. Dentro da pasta ua12, você criará uma subpasta, chamada imagens. Por meio desse programa um arquivo de imagem será copiado de algum local do computador para dentro dessa subpasta imagens, que está no servidor localhost.

O primeiro passo para a elaboração de um upload de arquivo simples é escrever o código de um arquivo em HTML que servirá para selecionar o arquivo que deverá ser copiado de algum lugar do computador para a pasta imagens no servidor e, também, para exibir um botão, que deverá ser clicado pelo usuário quando tiver escolhido o arquivo e desejar iniciar o upload do arquivo. Nesse caso, o arquivo em HTML será chamado de UPLOAD.HTML e deverá ter o seguinte código:

```
1  <html>
2      <meta charset="utf-8">
3      <head>
4          <title>Upload de Arquivos</title>
5      </head>
6      <body>
7          <h1>Upload de Arquivos com PHP</h1>
8          <form action="faz_upload.php" enctype="multipart/form-
data" method="POST">
9              Selecione o arquivo para fazer o upload:
10             <br>
11             <input name="arquivo" size="20" type="file"/>
12             <br><br>
13             Quando estiver pronto, clique no botão abaixo:
14             <br>
15             <input type="submit" value="Enviar arquivo"/>
16         </form>
17     </body>
18 </html>
```

É importante frisar que, para fazer o upload de arquivos utilizando o PHP, necessariamente você deve informar o encoding de dados. Essa informação está na linha 8 do código apresentado (`enctype="multipart/form-
-data" method="POST"`). É esse comando que vai informar ao navegador que está sendo enviado um arquivo, e não o texto que está no formulário (PHP, c2001-2018b).

Depois de criado o arquivo em HTML, que servirá de interface para o usuário, é o momento de criar o arquivo em PHP, que servirá para fazer o upload do arquivo propriamente dito. Para esse exemplo, conforme consta na linha 8 do código do arquivo upload.html, o arquivo em PHP se chamará FAZ_UPLOAD.PHP.

O arquivo faz_upload.php deverá ter o seguinte código:

```
1 <html>
2     <body>
3     <?php
4         /** NA LINHA ABAIXO DEVERÁ ESTAR INFORMADA A PASTA ONDE O
5         ARQUIVO SERÁ SALVO. NESSE CASO SERÁ NA PASTA IMAGENS **/
6         $uploaddir = 'imagens/';
7         $uploadfile = $uploaddir . $_FILES['arquivo']['name'];
8         if (move_uploaded_file($_FILES['arquivo']['tmp_name'],
9             $uploadfile)) {
10            echo "Upload efetuado com sucesso!";
11        } else {echo "Ops! Houve uma falha no processo de upload do
12 arquivo!"}
13    ?>
14    </body>
15 </html>
```

Note que, na linha 5 do código apresentado, foi informada a pasta IMAGENS como conteúdo para a variável \$uploaddir, que guarda a pasta de destino para o arquivo. Na linha 6, a variável \$uploadfile, que armazena o nome do arquivo que será copiado, conterá a pasta de destino e o nome do arquivo.

O código das linhas 7 a 9 efetua um teste, em cima do comando que efetivamente realiza o upload do arquivo (move_uploaded_file), e verifica se o upload foi realizado com sucesso ou não. Em ambos os casos, é apresentada uma mensagem para o usuário.

Para executar o programa recém-criado nesse exemplo, basta abrir o navegador de internet e digitar o seguinte endereço: <http://localhost/ua12/upload.html>. A tela que aparecerá para o usuário está demonstrada na Figura 1.



Figura 1. Tela principal do exemplo de upload de arquivos.

Quando o usuário clicar em **Escolher arquivo**, deverá escolher o arquivo para upload, utilizando a janela apresentada na Figura 2.

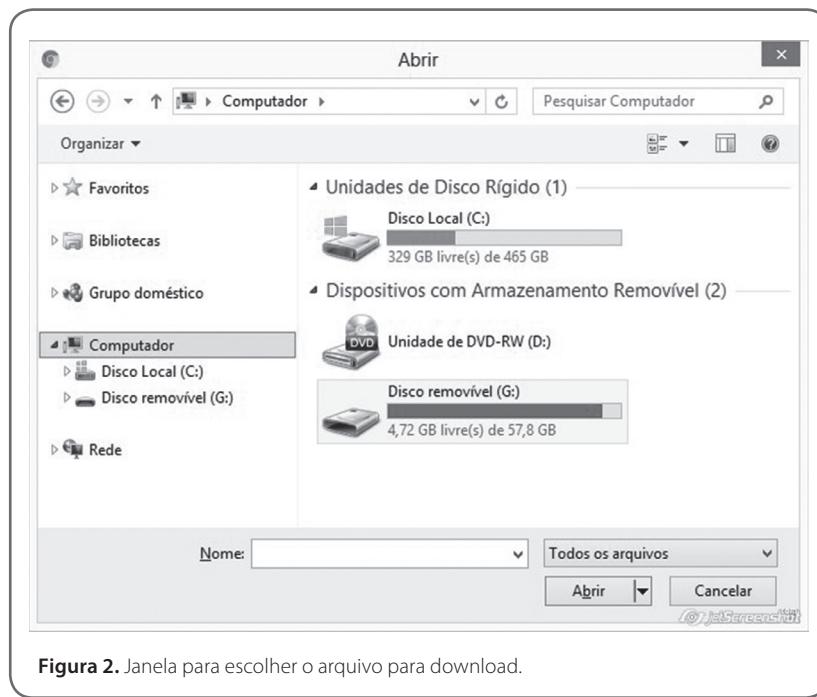


Figura 2. Janela para escolher o arquivo para download.

Quando o usuário clicar em **Enviar arquivo**, o arquivo indicado no programa UPLOAD.HTML vai ser copiado para a pasta definida no arquivo FAZ_UPLOAD.PHP.

Validação de tamanho e extensão e renomeação de arquivos

A linguagem PHP proporciona fazer o upload de qualquer tipo de arquivo, seja de texto ou binário. Na tentativa de fazer o upload de um arquivo, imediatamente a variável global `$_FILES` armazena todas as informações sobre o arquivo que foi enviado, conforme detalhado a seguir (PHP, c2001-2018b). Você deve considerar que a expressão ‘arquivo’ contém o nome do arquivo do qual se quer saber as informações.

- `$_FILES['arquivo']['name']`: contém o nome do arquivo original, na máquina do usuário.
- `$_FILES['arquivo']['type']`: contém o tipo multipurpose internet mail extensions (MIME) do arquivo, caso o navegador forneça essa informação.
- `$_FILES['arquivo']['size']`: contém o tamanho do arquivo que foi enviado, em bytes.
- `$_FILES['arquivo']['tmp_name']`: contém o endereço e o nome de arquivo temporários que foram utilizados para armazenar o arquivo enviado no servidor. Por padrão, durante o upload, os arquivos serão guardados na pasta padrão do servidor, que é `C:\xampp\tmp\`, para quem utiliza o XAMPP.
- `$_FILES['arquivo']['error']`: contém o código de erro associado ao upload do arquivo, caso exista. Os valores possíveis para o código de erro são os seguintes (PHP, c2001-2018a):
 - `UPLOAD_ERR_OK`: retorna valor 0. Significa que não houve erro e o upload foi executado com sucesso.
 - `UPLOAD_ERR_INI_SIZE`: retorna valor 1. Significa que o arquivo enviado para upload excede o limite definido na diretiva `UPLOAD_MAX_FILE_SIZE` do `php.ini`.
 - `UPLOAD_ERR_FORM_SIZE`: retorna valor 2. Significa que o arquivo excede o limite definido em `MAX_FILE_SIZE`, no formulário HTML.
 - `UPLOAD_ERR_PARTIAL`: retorna valor 3. Significa que o upload do arquivo foi executado parcialmente.
 - `UPLOAD_ERR_NO_FILE`: retorna valor 4. Significa que não foi enviado arquivo para upload.
 - `UPLOAD_ERR_NO_TMP_DIR`: retorna valor 6. Significa que a pasta temporária para armazenar o arquivo não existe.
 - `UPLOAD_ERR_CANT_WRITE`: retorna valor 7. Significa que houve falha ao escrever o arquivo no servidor.
 - `UPLOAD_ERR_EXTENSION`: retorna valor 8. Significa que uma extensão qualquer do PHP causou a interrupção do upload do arquivo. O PHP não fornece informações sobre qual extensão causou a interrupção.



Fique atento

A expressão MIME, do inglês *Multipurpose Internet Mail Extensions*, significa extensões multifunção de correio de internet (MDN WEB DOCS, c2005-2018).

O tipo MIME é um padrão, proposto em 1991, que possibilita a inserção de documentos em uma mensagem. Esses documentos podem ser sons, imagens, texto, entre outros tipos de arquivos. Quando existe uma transação entre um servidor de internet e um navegador de internet, a primeira ação do servidor é enviar o tipo MIME do arquivo que foi enviado para o navegador, para que ele entenda como tal documento deve ser exibido.

Alguns tipos de MIME são: **application/acad**, para arquivos de AutoCAD; **application/pdf**, para arquivos Adobe Acrobat; **image/gif**, para imagens GIF; **text/html**, para arquivos HTML; entre outros.

Para fazer a exibição do conteúdo da variável superglobal `$_FILES`, mostrando tamanho, extensão, nome do arquivo e outras informações, podem ser utilizados os arquivos criados anteriormente. O primeiro arquivo é o upload.html, em que o usuário escolhe um arquivo para ser copiado de um lugar no computador para a pasta imagens do servidor.

Lembrando que o arquivo UPLOAD.HTML deverá ter o seguinte código:

```
1 <html>
2     <meta charset="utf-8">
3     <head>
4         <title>Upload de Arquivos</title>
5     </head>
6     <body>
7         <h1>Upload de Arquivos com PHP</h1>
8         <form action="faz_upload.php" enctype="multipart/form-
9             data" method="POST">
10            Selecione o arquivo para fazer o upload:
11            <br>
12            <input name="arquivo" type="file"/>
13            <br><br>
14            Quando estiver pronto, clique no botão abaixo:
15            <br>
16            <input type="submit" value="Enviar arquivo"/>
17        </form>
18    </body>
19 </html>
```

O segundo arquivo que será utilizado é o faz_upload.php. É ideal que sejam feitas alterações no código criado anteriormente, para que as informações do arquivo sejam mostradas na tela após a tentativa de upload feita pelo usuário.

Nesse sentido, o conteúdo do arquivo FAZ_UPLOAD.PHP deve ser alterado para o seguinte:

```
1 <html>
2   <body>
3   <?php
4     /** NA LINHA ABAIXO DEVERÁ ESTAR INFORMADA A PASTA ONDE O
5      ARQUIVO SERÁ SALVO. NESSE CASO SERÁ NA PASTA IMAGENS **/
6     $uploaddir = 'imagens/';
7     $uploadfile = $uploaddir . $_FILES['arquivo']['name'];
8     if (move_uploaded_file($_FILES['arquivo']['tmp_name'],
9       $uploadfile))
10       echo "Upload efetuado com sucesso!";
11     else
12       echo "Ops! Houve uma falha no processo de upload do
13 arquivo!";
14     echo "<br><br>";
15     $nome = $_FILES['arquivo']['name'];
16     echo "Nome original do arquivo: " . $nome;
17     echo "<br>";
18     $mime = $_FILES['arquivo']['type'];
19     echo "Tipo MIME do arquivo: " . $mime;
20     echo "<br>";
21     $stamanho = $_FILES['arquivo']['size'];
22     echo "Tamanho do arquivo em bytes: " . $stamanho;
23     echo "<br>";
24     $stemp = $_FILES['arquivo']['tmp_name'];
25     echo "Nome temporário do arquivo: " . $stemp;
26     echo "<br>";
27     $errores = $_FILES['arquivo']['error'];
28     echo "Código do erro ocorrido durante o upload arquivo: "
. $errores;
29   ?>
30   </body>
31 </html>
```

As linhas 11 a 25 do código apresentado são responsáveis por exibir na tela todas as informações do arquivo, guardadas na variável superglobal `$_FILES`, incluindo o código do erro ocorrido durante o upload, se existir. Nesse momento, a execução do programa fica conforme representação da Figura 3.

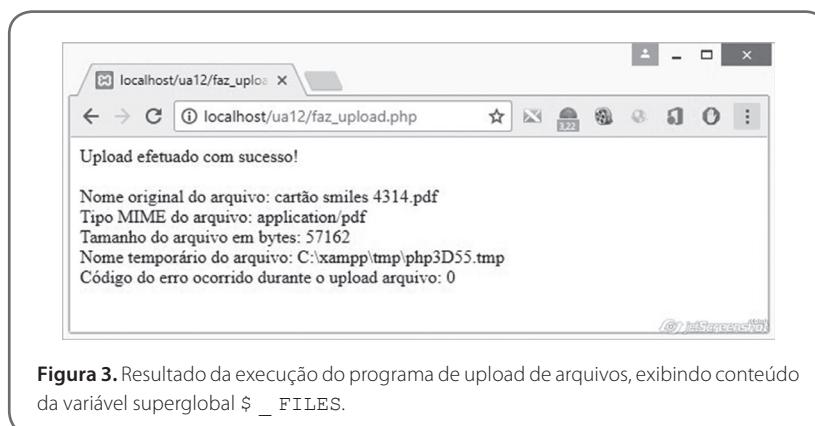


Figura 3. Resultado da execução do programa de upload de arquivos, exibindo conteúdo da variável superglobal `$_FILES`.

Para fazer a validação do arquivo, em termos de tamanho, extensão e nome, por exemplo, devem ser feitas mais algumas alterações no código do arquivo FAZ_UPLOAD.PHP, que deverá ficar minimamente conforme segue:

```

1  <html>
2      <body>
3          <?php
4              /** VERIFICA SE O ARQUIVO EXISTE **/
5              $arquivo = isset($_FILES['arquivo']) ? $_FILES['arquivo']
6 : FALSE;
7              /** VARIÁVEIS QUE GUARDAM O CONTEÚDO DA VARIÁVEL
8 SUPERGLOBAL $_FILES **/
9              $nome = $_FILES['arquivo']['name'];
10             $mime = $_FILES['arquivo']['type'];
11             $tamanho = $_FILES['arquivo']['size'];
12             $temp = $_FILES['arquivo']['tmp_name'];
13             $erros = $_FILES['arquivo']['error'];
14             /** DETERMINA O TAMANHO MÁXIMO DE ARQUIVO QUE SERÁ
15 ACEITO**/
16             $max_tamanho = 30000;
17             /** DETERMINA O TIPO MIME DO ARQUIVO QUE SERÁ ACEITO **/
18             $tipo = "image/png";
19             /** COLOCA O NOME DO ARQUIVO EM MINÚSCULO **/
20             $nome = strtolower($nome);
21             /** TROCA ESPAÇOS POR SUBLINHADO **/
22             $nome = str_replace(" ", "_", $arquivo["name"]);
23             /** SUBPASTA PARA ONDE SERÁ FEITO O UPLOAD **/
24             $uploaddir = 'upload/';
25             /** GUARDA O NOME DO ARQUIVO COM O SEU ENDEREÇO **/
26             $uploadfile = $uploaddir . $nome;
27             /** TESTES DE ARQUIVO REPETIDO, TAMANHO E TIPO MIME**/
28             if (file_exists($uploadfile)){
29                 echo "Ops! Já existe um arquivo chamado " . $nome . "
30 na pasta " . $uploaddir . " Tente de novo.";
31                 exit;
32             elseif ($tamanho > $max_tamanho){
33                 echo "Ops! O arquivo enviado ultrapassa o tamanho
34 máximo de " . $max_tamanho . " bytes . Tente de novo.";
35                 exit;
36             }
37         }
38     }
39 
```

```
31     elseif ($mime !== $tipo) {
32         echo "Ops! O tipo MIME do arquivo enviado não é " .
33             $tipo . ". Tente de novo.";
34         exit;}
35     /** SE O ARQUIVO PASSOU PELAS VALIDAÇÕES, FAZ O UPLOAD**/
36     else{
37         if (move_uploaded_file($temp, $uploadfile))
38             echo "Upload efetuado com sucesso!";
39         else
40             echo "Ops! Houve uma falha no processo de upload do
41         arquivo!";
42         echo "<br><br>";
43         echo "Nome original do arquivo: " . $nome;
44         echo "<br>";
45         echo "Tipo MIME do arquivo: " . $mime;
46         echo "<br>";
47         echo "Tamanho do arquivo em bytes: " . $tamanho;
48         echo "<br>";
49         echo "Nome temporário do arquivo: " . $temp;
50         echo "<br>";
51         echo "Código do erro ocorrido durante o upload arquivo:
52         " . $erro;
53     echo "<br><br>";}
54     ?>
55     </body>
56 </html>
```

É importante frisar que a ordem de algumas linhas foi alterada para melhor visualização do código e para o bom funcionamento das validações necessárias, e que os blocos de código estão comentados para melhor entendimento.

Das linhas 6 a 11 estão as variáveis que recebem o conteúdo da variável superglobal `$_FILES`, contendo as informações referentes ao arquivo do upload, como o nome, o tipo MIME, o tamanho, a pasta temporária em que ele foi guardado até o upload ser concluído e o código do erro que aconteceu durante o upload.

As linhas 12 a 15 contêm informações que podem ser retiradas das regras de negócio, quando um programador recebe um sistema para desenvolvimento. São as informações sobre o tamanho do arquivo que será permitido para o download (em bytes) e o tipo MIME que será permitido. A validação do tipo MIME é importante para os casos em que o usuário tenta burlar a validação alterando a extensão do arquivo, pois o tipo MIME reconhece o arquivo pelo seu conteúdo.

As linhas 16 a 19 fazem alguns dos tipos de renomeação de arquivos mais comuns, que são a colocação de todo o nome do arquivo em minúsculo e a troca de espaços no nome por sublinhado. As linhas 20 e 21 guardam o nome da pasta para a qual o upload será feito, e as linhas 22 e 23 guardam o nome do endereço completo com o arquivo.

Em seguida, nas linhas 24 a 33, são feitos os testes de arquivo repetido na pasta de destino, arquivo que ultrapassa o tamanho limite e tipo MIME não permitido. Caso algum desses testes se confirme, o programa não vai adiante devido ao comando EXIT.

Para finalizar, se o upload não puder ser feito, o usuário receberá uma mensagem de falha na execução, e, se o upload for concluído, o usuário receberá a exibição das informações do arquivo constantes na variável superglobal `$_FILES`.

A execução do programa, por meio da digitação do endereço `http://localhost/ua12/upload.html` no navegador, deverá ficar semelhante ao apresentado nas Figuras 4 a 7.



Figura 4. Exemplo de tentativa de upload para arquivo que já existe na pasta destino.

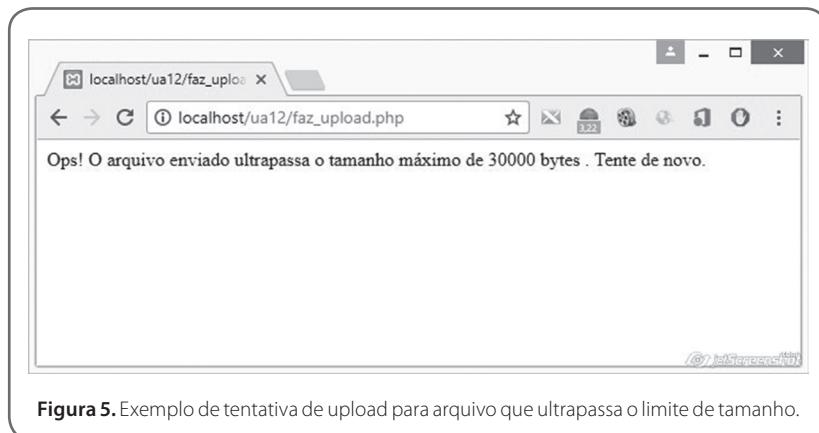


Figura 5. Exemplo de tentativa de upload para arquivo que ultrapassa o limite de tamanho.

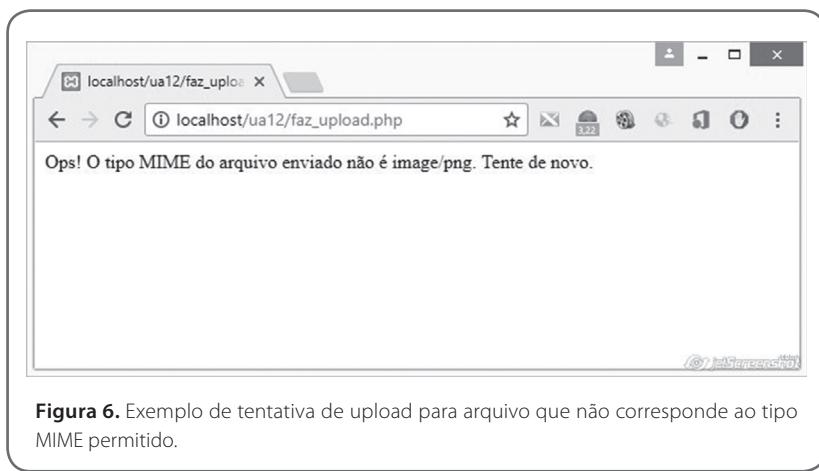


Figura 6. Exemplo de tentativa de upload para arquivo que não corresponde ao tipo MIME permitido.

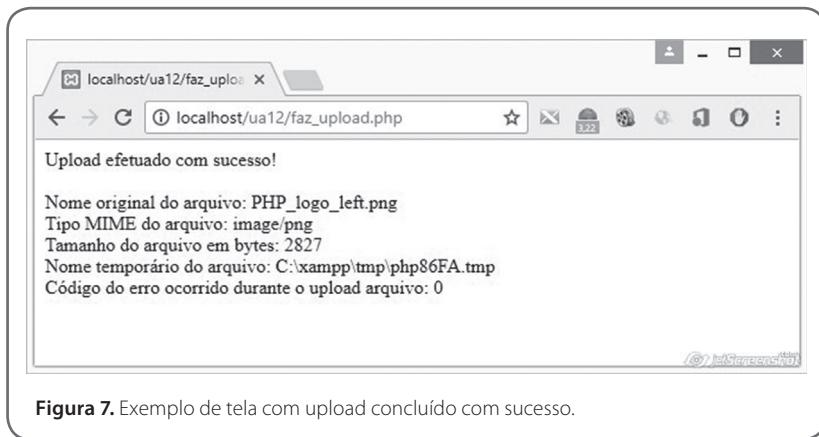


Figura 7. Exemplo de tela com upload concluído com sucesso.

Movimentação de arquivos e gravação de endereço em banco de dados

Para compreender a movimentação de arquivos e a gravação do endereço do upload no banco de dados, é necessário reunir os conhecimentos de upload de arquivos e, também, de funções de create read update delete (CRUD) (GOULARTE, 2016). Vamos utilizar um exemplo com uma tabela, chamada funcionario, na qual será gravado o nome do funcionário e o local onde a imagem da sua foto está armazenada.



Fique atento

O módulo CRUD, comumente chamado de módulo de cadastro, envolve as ações de incluir, consultar, alterar e excluir dados de tabelas de bancos de dados.

Para começar, você deve criar um banco de dados chamado ua12. Nesse banco de dados, deve criar uma tabela chamada funcionario, que conterá os seguintes campos:

- **cod_func:** código do funcionário, chave principal, auto incrementável.
- **nome_func:** nome do funcionário, varchar (50).
- **foto_func_endereco:** endereço da foto do funcionário no servidor, varchar (500).

Depois de criada a tabela, pode-se iniciar a implementação dos programas que vão possibilitar incluir nela o nome e o endereço da foto do funcionário. Serão utilizados ao todo cinco arquivos.

O primeiro arquivo a ser criado possui as informações para acesso ao banco de dados. Para isso, o arquivo INICIA.PHP deve conter o seguinte código:

```
1 <?php
2 require_once 'funcoes.php';
3 /** CONSTANTES COM AS INFORMAÇÕES PARA ACESSO AO BANCO DE DADOS
4 MySQL ***/
5 define('DB_HOST', 'localhost');
6 define('DB_NAME', 'ua12');
7 define('DB_USER', 'root');
8 define('DB_PASS', '');
9 ?>
```

O segundo arquivo será chamado de FUNCOES.PHP e terá a função de conexão com o banco de dados, que servirá para a gravação dos registros. Deverá conter o seguinte código:

```
1 <?php
2 /** O CÓDIGO ABAIXO FAZ A CONEXÃO COM O MYSQL ATRAVÉS DE PDO ***/
3 function conecta_bd(){
4     $PDO = new PDO('mysql:host=' . DB_HOST . ';dbname=' . DB_NAME
5     . ';charset=utf8', DB_USER, DB_PASS);
6     return $PDO;
7 }
```

O terceiro arquivo será a tela principal do programa de cadastro de funcionário. Ele será chamado de INDEX.PHP e vai conter o seguinte código:

```
1  <?php
2  require_once 'inicia.php';
3  $PDO = conecta_bd();
4  ?>
5  <!doctype html>
6  <html>
7      <head>
8          <meta charset="utf-8">
9          <title>Cadastro de Funcionário</title>
10     </head>
11     <body>
12         <h1>Cadastro de Funcionário</h1>
13         <p><a href="form_inclui.php">Adicionar
funcionário</a></p>
14         <h2>Lista de funcionários cadastrados</h2>
15         <?php
16             $stmt_count = $PDO->prepare("SELECT COUNT(*) AS total
FROM funcionario");
17             $stmt_count->execute();
18             $stmt = $PDO->prepare("SELECT
cod_func,nome_func,foto_func_endereco FROM funcionario ORDER BY
nome_func");
19             $stmt->execute();
20             $total = $stmt_count->fetchColumn();
21             if ($total > 0):
22                 <table border="1">
23                     <thead>
24                         <tr>
25                             <th>Nome</th>
26                             <th>Endereço foto</th>
27                             <th>Foto</th>
28                         </tr>
29                     </thead>
30
31                     <tbody>
32                         <?php while ($resultado = $stmt-
>fetch(PDO::FETCH_ASSOC)) : ?>
33                             <tr>
34                                 <td><?php echo $resultado['nome_func'] ?></td>
35                                 <td><?php echo $resultado['foto_func_endereco'] ?>
36                                     <td><?php echo "<img src='".
$resultado['foto_func_endereco'] . "'>" ?></td>
37                                     </tr>
38                         <?php endwhile; ?>
39                     </tbody>
40                 </table>
41                 <p>Total de funcionários cadastrados: <?php echo $total
?></p>
42                 <?php else: ?>
43                 <p>Não há funcionário cadastrado</p>
44                 <?php endif; ?>
45             </body>
46         </html>
```

Na primeira parte do código, você pode perceber que há a conexão com o banco de dados. Depois disso, das linhas 16 a 20, são criadas variáveis que vão armazenar o total de funcionários já inseridos na tabela e todos os funcionários inseridos, para que isso seja mostrado em uma tabela nessa página.

Depois disso, da linha 22 a 39, a tabela que vai mostrar os registros é elaborada contendo três colunas: nome do funcionário, endereço da foto do funcionário e foto, que é a coluna que vai exibir a foto armazenada no endereço. O código finaliza exibindo, logo abaixo da tabela dos registros, o total de funcionários cadastrados.

O quarto arquivo será chamado de FORM_INCLUI.PHP e será responsável por exibir um formulário, no qual o usuário poderá digitar o nome do funcionário e ainda escolher o local em que a foto do funcionário está armazenada. Depois disso, clicando em **Enviar arquivo e Incluir registro**, será possível fazer o upload do arquivo da foto para o servidor e, ainda, inserir o registro na tabela funcionario. Esse arquivo deve conter o seguinte código:

```
1  <!doctype html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title>Cadastro de Funcionários</title>
6      </head>
7      <body>
8          <h2>Cadastro de Funcionários - Inclusão</h2>
9          <form action="incluir.php" enctype="multipart/form-data"
method="post">
10             <label for="nome_func">Nome do funcionário:</label>
11             <input type="text" name="nome_func"
id="nome_func"><br><br>
12             <label for="foto_func_endereco">Foto do funcionário:
13             <input type="file" name="foto_func_endereco"
id="foto_func_endereco"><br><br>
14             <input type="submit" value="Enviar arquivo e incluir
registro"/>
15         </form>
16     </body>
17 </html>
```

A linha 9 do código apresentado contém o comando `enctype="multipart/form-data"` `method="post"`, responsável por informar que será feito um upload de arquivo e que será utilizado o método post (PHP, c2001-2018b).

Das linhas 9 a 15 são exibidos campos para que o usuário digite o nome do funcionário e escolha o local do computador em que o arquivo que contém a foto do funcionário está armazenado. Na linha 14 está o comando que faz o desvio do fluxo do programa para o próximo arquivo, concluindo o upload da foto e também a inclusão do registro na tabela funcionario.

Dessa forma, o quinto e último arquivo, responsável pelo upload e pela gravação do registro na tabela, é chamado de INCLUI.PHP e deve conter o seguinte código:

```
1  <?php
2  require_once 'inicia.php';
3  /** COLETA AS INFORMAÇÕES DIGITADAS NO FORMULÁRIO
4  FORM_INCLUI.PHP **/
5  $nome_func = isset($_POST['nome_func']) ? $_POST['nome_func'] :
6  null;
7  $foto_func_endereco = isset($_POST['foto_func_endereco']) ?
8  $_POST['foto_func_endereco'] : null;
9  /** VARIÁVEIS QUE GUARDAM O CONTEÚDO DA VARIÁVEL SUPERGLOBAL
10 $FILES **/
11 $nome = $FILES['foto_func_endereco']['name'];
12 $temp = $FILES['foto_func_endereco']['tmp_name'];
13 /** SUBPASTA PARA ONDE SERÁ FEITO O UPLOAD **/
14 $uploadaddir = 'upload/';
15 /** GUARDA O NOME DO ARQUIVO COM O SEU ENDEREÇO **/
16 $uploadfile = $uploadaddir . $nome;
17 if (move_uploaded_file($temp, $uploadfile))
18     echo "Upload efetuado com sucesso!";
19 else
20     echo "Ops! Houve uma falha no processo de upload do
21 arquivo!";
22 /** INSERE AS INFORMAÇÕES NA TABELA FUNCIONARIO DO BANCO DE
23 DADOS UA12 **/
24 $PDO = conecta_bd();
25 $sql = "INSERT INTO funcionario(nome_func,foto_func_endereco)
26 VALUES(:nome_func,:uploadfile)";
27 $stmt = $PDO->prepare($sql);
28 $stmt->bindParam(':nome_func', $nome_func);
29 $stmt->bindParam(':uploadfile', $uploadfile);
30 if ($stmt->execute()){
31     header('Location: form_inclui.php');
32 }
33 else{
34     echo "Ocorreu um erro na inclusão de registro";
35     print_r($stmt->errorInfo());
36 }
```

O código inicia verificando o conteúdo das variáveis que receberão o nome e o endereço da foto do funcionário, inicializando-as com nulo caso estejam

vazias. Depois disso, entre as linhas 6 e 8, variáveis recebem o conteúdo da variável superglobal `$_FILES` para armazenar o nome do arquivo para upload, e a pasta temporária em que ele será armazenado até a conclusão do upload. As linhas 9 a 12 armazenam em variáveis a pasta de destino do arquivo e o endereço completo do upload, incluindo o nome do arquivo.

O comando da linha 13 é o responsável por efetivar o upload do arquivo para a pasta informada na linha 10. Será exibida uma mensagem para o usuário após a tentativa de upload, obtendo sucesso ou não. Das linhas 17 a 22 o novo registro é inserido na tabela, e o código finaliza com um teste para a gravação do registro, informando o usuário em caso de insucesso e retornando para o `form_inclui.php`, em caso de sucesso.

A execução do programa se dá pela digitação do endereço `http://localhost/ua12/index.php` no navegador, e as telas deverão ser semelhantes às apresentadas nas Figuras 8 a 10. A Figura 11 traz um esquema de upload.

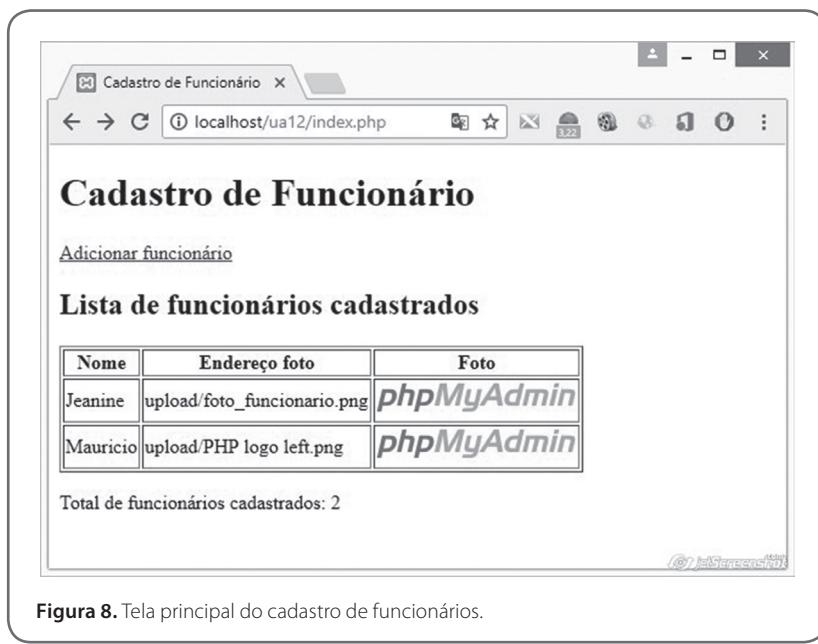


Figura 8. Tela principal do cadastro de funcionários.

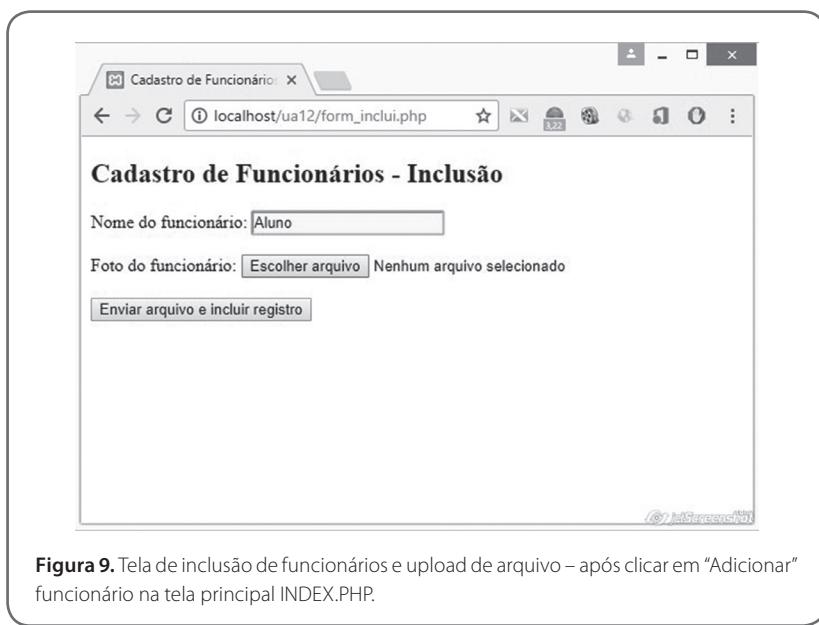


Figura 9. Tela de inclusão de funcionários e upload de arquivo – após clicar em “Adicionar” funcionário na tela principal INDEX.PHP.

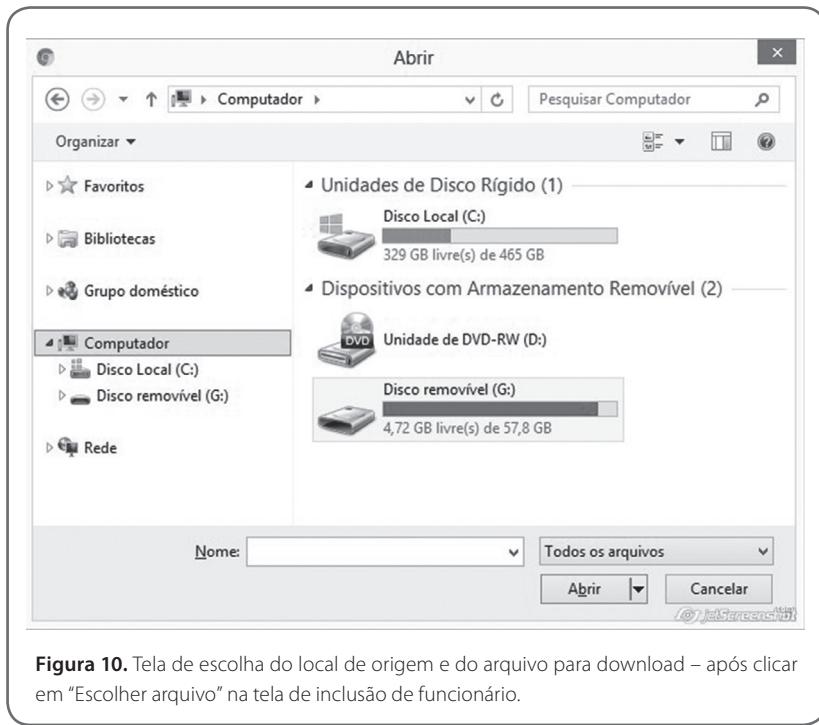


Figura 10. Tela de escolha do local de origem e do arquivo para download – após clicar em “Escolher arquivo” na tela de inclusão de funcionário.

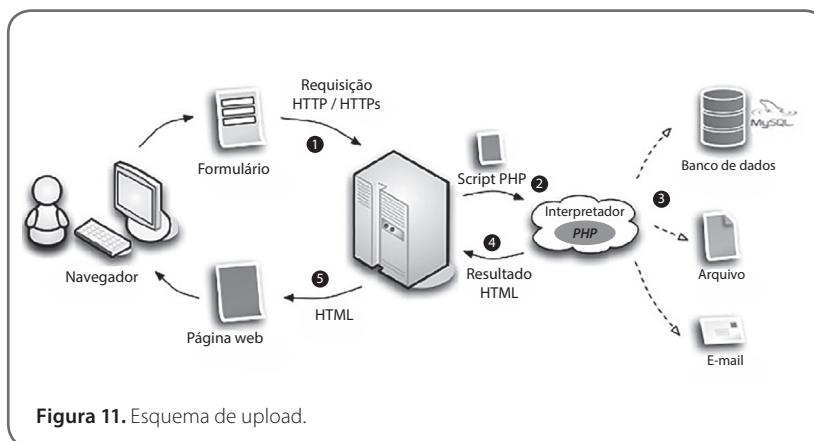


Figura 11. Esquema de upload.



Referências

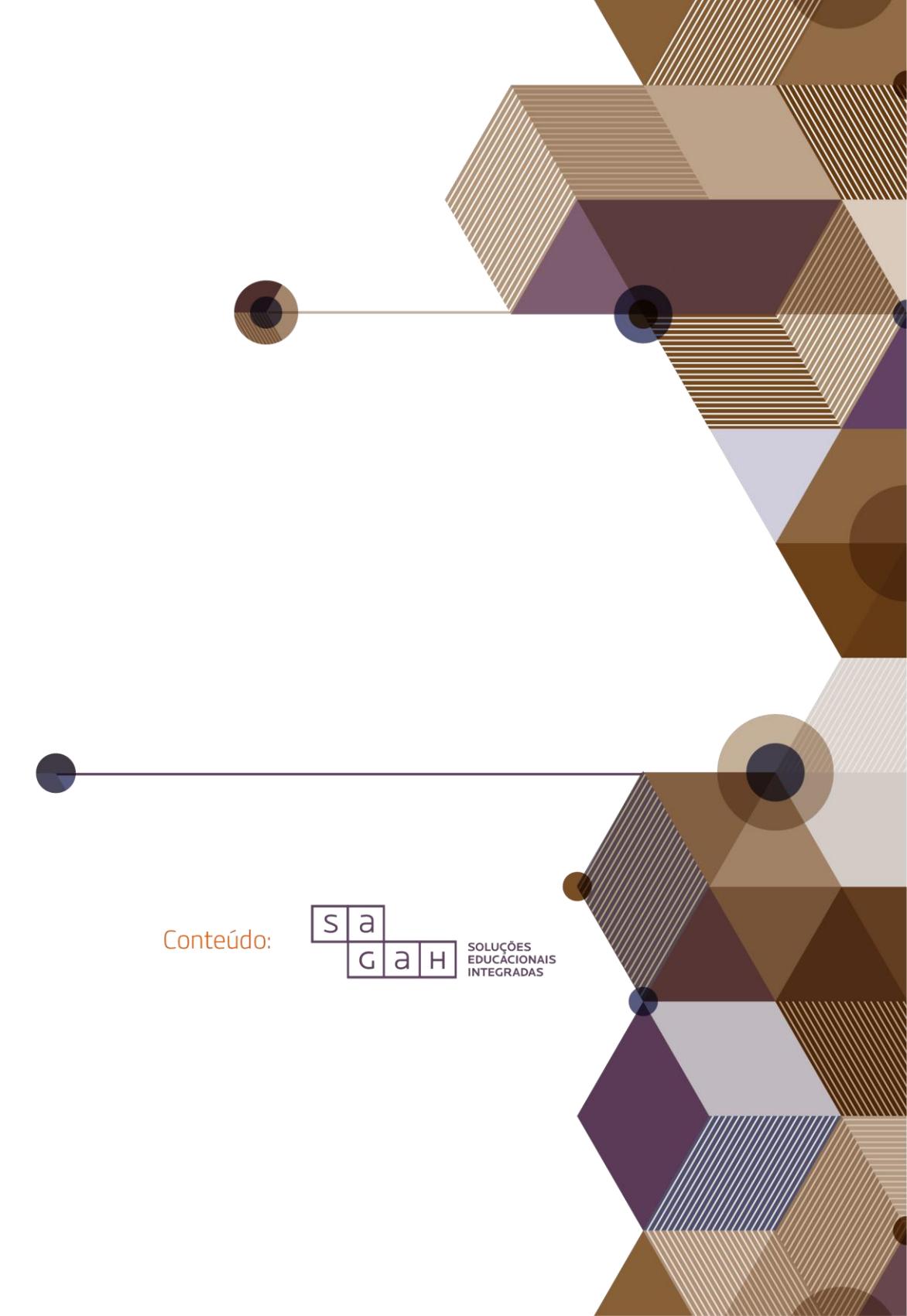
GOULARTE, C. *CRUD com PHP, de forma simples e fácil*. [S.I.]: Código Master, 2016. Disponível em: <<http://www.codigomaster.com.br/desenvolvimento/crud-com-php-de-forma-simples-e-facil>>. Acesso em: 17 jan. 2018.

MDN WEB DOCS. *MIME types*. Mountain View: Mozilla, c2005-2018. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Basico_sobre_HTTP/MIME_types>. Acesso em: 16 jan. 2018.

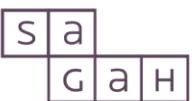
PHP. *Explicando mensagens de erro*. [S.I.]: The PHP Group, c2001-2018a. Disponível em: <http://php.net/manual/pt_BR/features.file-upload.errors.php>. Acesso em: 15 jan. 2018.

PHP. *Upload de arquivos com o método POST*. [S.I.]: The PHP Group, c2001-2018b. Disponível em: <https://secure.php.net/manual/pt_BR/features.file-upload.post-method.php>. Acesso em: 15 jan. 2018.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.



Conteúdo:



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS