

UD3 – Envío de datos y archivos al servidor

2º CFGS
Desarrollo de Aplicaciones Web
2022-23

1.- Variables superglobales

PHP dispone de un conjunto de variables que se pueden usar **en cualquier ámbito** sin necesidad de declararlas.

Estas variables son **arrays** que contienen información interesante del sistema: datos del servidor, datos del entorno, cookies...

`$GLOBALS`

`$_POST`

`$_SESSION`

`$_SERVER`

`$_FILES`

`$_REQUEST`

`$_GET`

`$_COOKIE`

`$_ENV`

Gracias a las funciones **print_r** y **var_dump** se puede mostrar el contenido de una variable e información sobre su contenido.

```
print_r($_SERVER);  
var_dump($_SERVER);
```

1.- Variables superglobales

| Variable | Información |
|-------------------------|---|
| <code>\$GLOBALS</code> | Todas las variables globales. |
| <code>\$_SERVER</code> | Información del servidor y del entorno de ejecución. |
| <code>\$_GET</code> | Datos en la query de la URL o de un formulario con método GET. |
| <code>\$_POST</code> | Datos de un formulario con método POST. |
| <code>\$_FILES</code> | Información de los archivos enviados con un formulario con método POST |
| <code>\$_COOKIE</code> | Información de las cookies de la aplicación web. |
| <code>\$_SESSION</code> | Variables de sesión usadas por la aplicación. |
| <code>\$_REQUEST</code> | Toda la información de <code>\$_GET</code> , <code>\$_POST</code> y <code>\$_COOKIE</code> junta. |
| <code>\$_ENV</code> | Variables del entorno de la aplicación web. |

2.- Peticiones HTTP

Las peticiones HTTP son el medio mediante el cual los clientes web (navegadores) piden recursos a los servidores web.

Las peticiones HTTP existentes son:

GET

HEAD

POST

DELETE

PUT

PATCH

TRACE

CONNECT

OPTIONS

De momento solo se usaran los tipos **GET** y **POST** que son las usadas en los formularios HTML.

Más adelante con los servicios web API REST se verán los tipos PUT, PATCH y DELETE

2.- Peticiones HTTP

- **GET:** Se solicita un recurso específico del repositorio del servidor. Se puede **enviar información** mediante la **query** de la URL.



http://www.dominio.com/pagina.html?variable1=valor1&variable2=valor2

The diagram shows a URL with a bracket above the query part, labeled 'query string'.

- **POST:** Se **envían datos** para que un recurso específico del servidor los **procese**. Estos datos no se pueden ver en la URL.

3.- Envío de datos mediante enlaces y query string

Mediante los enlaces, añadiendo query string se puede indicar al servidor que muestre una información determinada.

```
<a href="index.php?user=alex">Ver publicaciones</a>
```

En el recurso destino se podrá acceder a las variables recibidas por query string mediante la variable superglobal **\$_GET**.

```
<?php
echo 'Mostrando publicaciones de '. $_GET['user'];
// Acceso a la base de datos para obtener las publicaciones del usuario
// Recorrer los resultados y mostrarlos
```

3.- Formularios web

Los formularios web son la herramienta que permite recoger datos introducidos por el usuario y procesarlos en el servidor.

```
<form method="post" action="busqueda.php">  
    <!-- campos del formulario -->  
</form>
```

Es importante elegir el método (**method**) con el que se enviará el formulario web al servidor (**get** o **post**).

En el atributo **action** se indica el recurso (**script php**) que recibirá los datos.

Para que un campo del formulario se envíe tiene que tener definido el atributo **name**.

En el recurso destino se podrá acceder a las variables recibidas desde el formulario mediante las variables superglobales **\$_GET** o **\$_POST** dependiendo del método usado.

3.- Formularios web

```
<form action="procesa.php" method="post">
  Nombre del alumno: <input type="text" name="nombre" id="nombre">
  <br>
  Apellidos del alumno: <input type="text" name="apellidos" id="apellidos">
  <br>

  Ciclo que cursa:<br>
  <input type="radio" name="ciclo" value="DAW"> Des. de ap. Web
  <br>
  <input type="radio" name="ciclo" value="DAM"> Des. de ap. Multiplataforma
  <br><br>
  <input type="submit" value="Enviar">
</form>
```


3.- Formularios web

Archivo **procesa.php**

```
<?php
    echo 'El alumno ';
    echo $_POST['nombre'] . ' ' . $_POST['apellidos'];
    echo '<br>Se encuentra cursando el ciclo: ';
    echo $_POST['ciclo'];
?>
```

3.- Formularios web

En el caso de que un **checkbox** pueda enviar varios valores hay que indicar en el atributo **name** que es un **array**.

```
<form action="procesa.php" method="post">  
  Nombre del alumno: <input type="text" name="nombre"><br>
```

Módulos que cursa:


```
<input type="checkbox" name="modulos[]" value="DWES">
```

Desarrollo web en entorno servidor

```
<br>
```

```
<input type="checkbox" name="modulos[]" value="DVEC">
```

Desarrollo web en entorno cliente

```
<br><br>
```

```
<input type="submit" value="Enviar">
```

```
</form>
```

3.- Formularios web

Es posible enviar mediante un array otras variables:

```
<form action="procesa.php" method="post">
  Nombre: <input type="text" name="propio[nombre]"><br>
  Apellidos: <input type="text" name="propio[apellidos]"><br>

  Nombre: <input type="text" name="conyuge[nombre]"><br>
  Apellidos: <input type="text" name="conyuge[apellidos]"><br>
  <br>

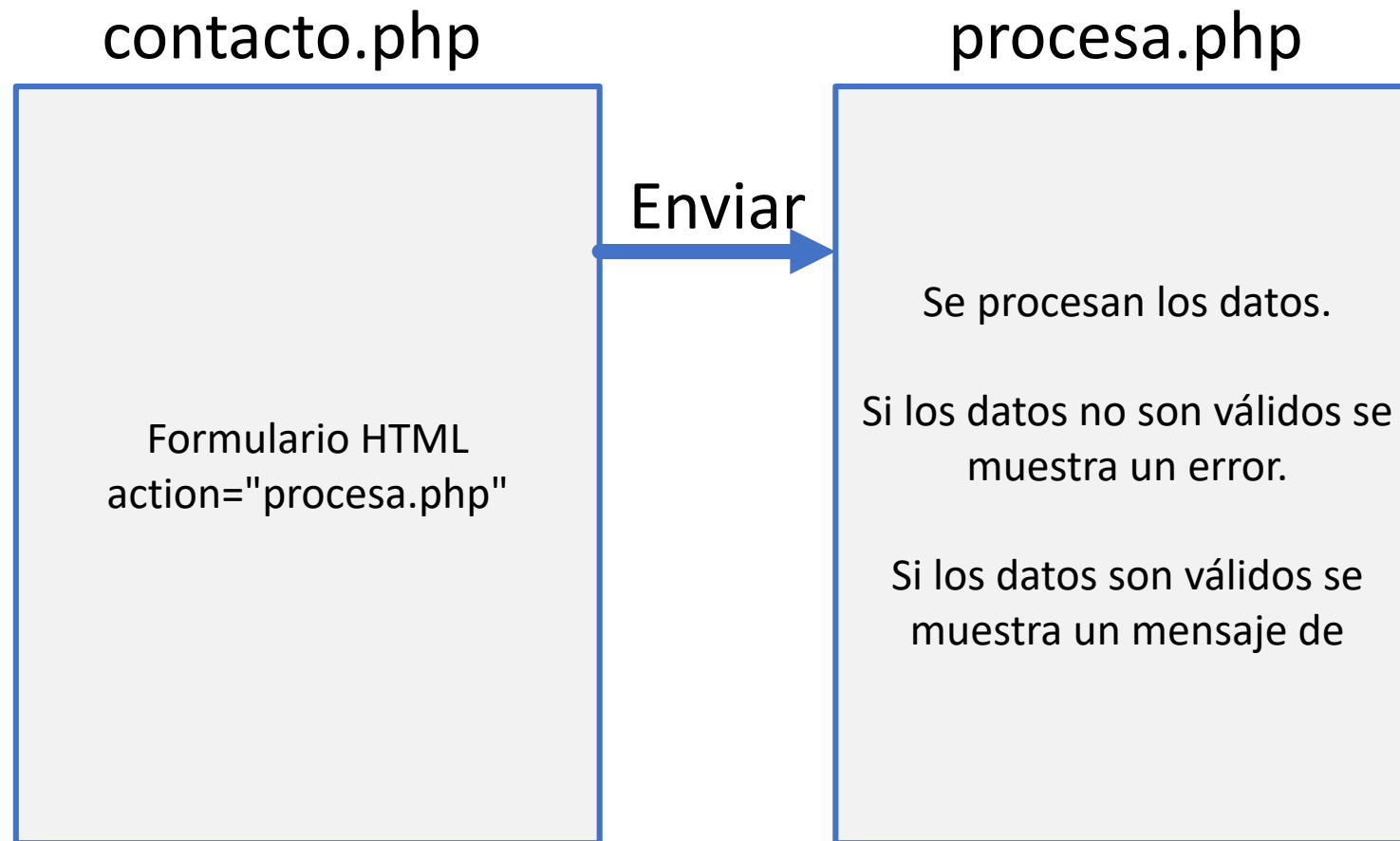
  <input type="submit">
</form>
```

3.- Formularios web

A continuación, se muestran diferentes soluciones al flujo de datos del envío de datos al servidor mediante formularios.

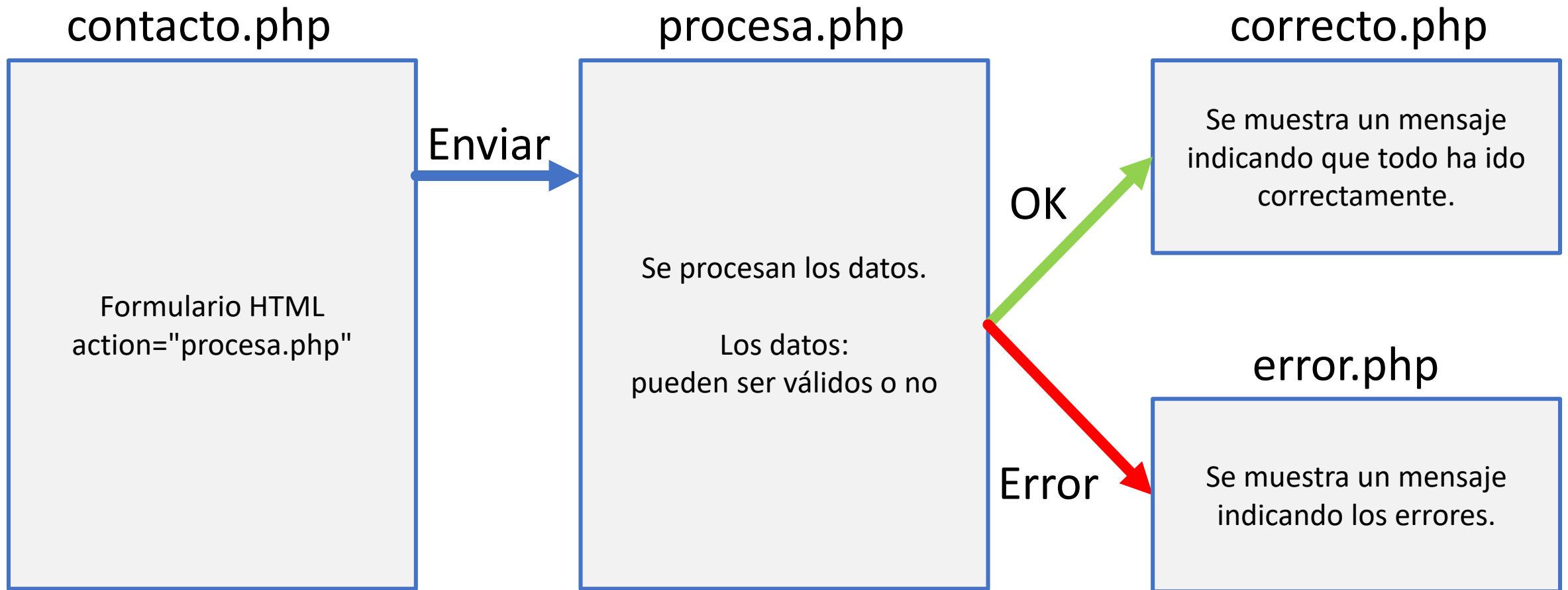
3.- Formularios web

Caso 1



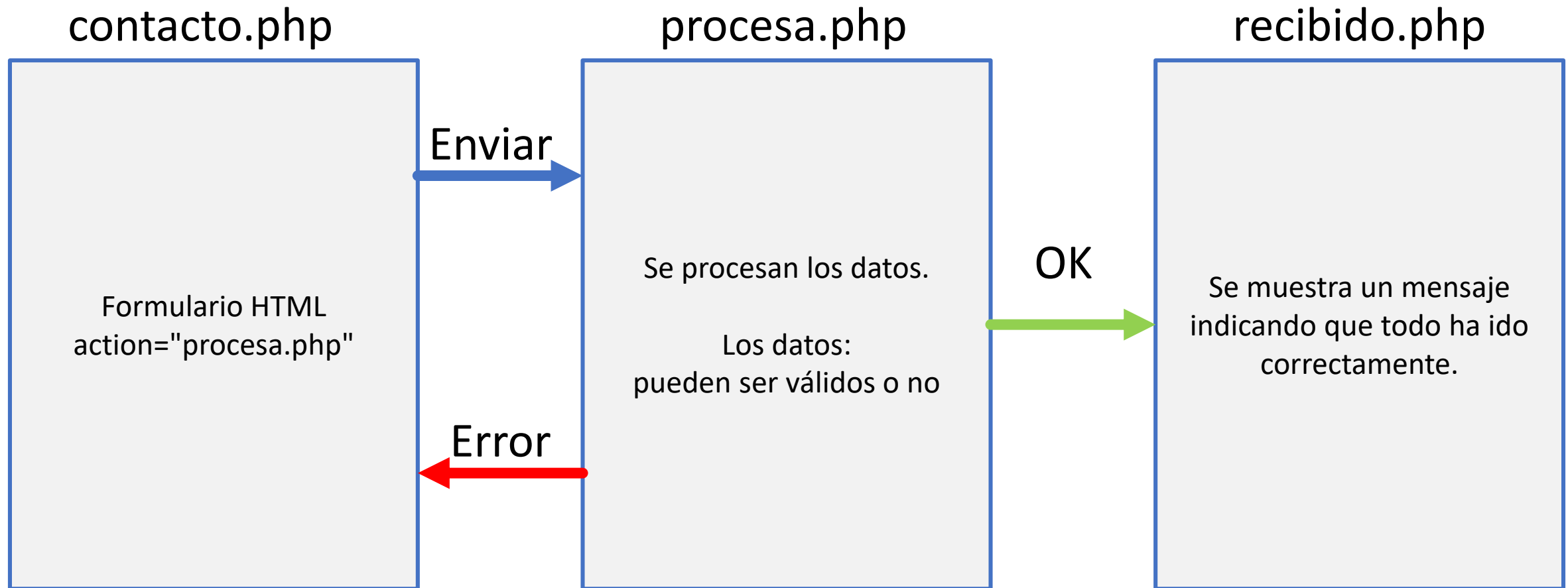
3.- Formularios web

Caso 2



3.- Formularios web

Caso 3



3.- Formularios web

Caso 4

contacto.php

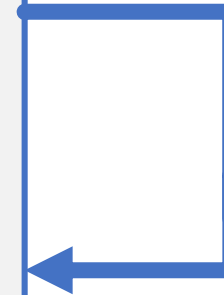
Formulario HTML
action="contacto.php"

Enviar

El mismo script recibe los
datos y los valida.

Si se produce error se
muestran en la propia
página.

Si todo va bien se muestra un
mensaje en la propia página.



Práctica

Actividad 1:

Introduciendo datos en el almacén.

4.- Validación de datos

Si el usuario introduce datos en una aplicación web es importante **realizar la validación** de estos datos en todos los puntos vulnerables.

- **Navegador:**

Mediante el uso de los tipos correctos en los campos **input** y el atributo **required**.

- **Cliente:**

Antes de ser enviados al servidor se validan en el cliente mediante JavaScript.

- **Servidor:**

Antes de realizar acciones, en el script que se reciben los datos se debe validar que estos son correctos.

También hay que evitar la suplantación de identidad y ataques [CSRF](#). Para ello se hace uso de **input type="hidden"** y variables de sesión (esto se verá más adelante)

4.- Validación de datos

Cuando se detecta una vulnerabilidad o ataque es conveniente finalizar el script y redirigir la petición:

```
header('Location: http://localhost.actividades/registro.php');  
exit;
```

Si no se detecta vulnerabilidad o ataque se siguen validando los datos generalmente con funciones **isset**, **is_numeric**, **strlen**, **strcmp**... y/o con **expresiones regulares**.

Se deben validar todos los campos recibidos y posteriormente informar de todos los errores que existan para así solo recargar la página web una vez.

Cuando se detecta algún error en algún campo se debe informar al usuario.

También se deberían rellenar los campos del formulario con los datos introducidos por el usuario para que este pueda comprobar dónde está el error.

4.- Validación de datos

El script php que recibe los datos del formulario puede ser el mismo que muestra el formulario:

action="#"

- Si no recibe variables se mostrará el formulario vacío.
- Si llegan variables se procede a validarlos.
- Si hay error en los datos se muestra el formulario con los datos introducidos por el usuario y además, se muestren los errores en los datos.
- Si todos los datos son correctos se realiza su procesamiento, por ejemplo, guardar en la base de datos.

De esta manera se da más información al usuario sobre los problemas que se han producido.

Práctica

Actividad 2:

Introduciendo datos en el almacén v2.0.

5.- Expresiones regulares

Las expresiones regulares son **patrones** de caracteres que permiten **comprobar** si **una cadena de caracteres** se ajusta al patrón o no.

Ejemplos típicos de comprobaciones con expresiones regulares:

- Que un nombre no tenga números.
- Que tenga una longitud concreta.
- Que se siga un orden en los caracteres (DNI → 8 números y 1 letra).
- Que sea un email.
- ...

Las expresiones regulares existen en muchos lenguajes de programación así que es importante conocer su funcionamiento.

5.- Expresiones regulares

En PHP las expresiones regulares se escriben como cadenas de caracteres y van delimitadas al inicio y al final por el carácter: /

'/expresión_regular/'

El diseño de expresiones regulares es una disciplina cuyo nivel de dificultad aumenta conforme se quieren comparar patrones más complejos.

A continuación, se verá una pequeña guía del uso de las expresiones regulares que servirá para aprender a utilizarlas de una manera básica.

5.- Expresiones regulares

Modificadores

- **i**: no distingue entre mayúsculas y minúsculas (case-insensitive).

'/a/i' → contiene la a o la A.

- **m**: busca el patrón en todas las líneas de la cadena, la cadena debe tener salto de línea: \n
- **u**: incluye caracteres UTF-16
- **g**: no se detiene tras la primera coincidencia.

5.- Expresiones regulares

Agrupación

- `[]` `[abc]` contiene cualquier carácter de entre los indicados.
- `[^]` `[^abc]` contiene cualquier carácter que no sea de los indicados.
- `[-]` `[0-9]` contiene cualquier carácter que se encuentre en el rango.
- `[^ -]` `[^A-B]` contiene cualquier carácter que no esté en el rango.
- `(|)` `(x|y)` contiene uno de los caracteres (separador `|`).

Cantidad de caracteres

- `{ }` `a{3}` contiene exactamente 3 'a' seguidas.
- `{ , }` `a{3,}` contiene 3 o más 'a' seguidas.
- `{ , }` `a{3,5}` contiene 3, 4 o 5 'a' seguidas.
- `*` `a*` contiene 0 o más 'a'. Similar: `a{0, }`
- `+` `a+` contiene 1 o más 'a'. Similar: `a{1, }`
- `?` `a?` contiene 0 o 1 'a'. Similar: `a{0,1}`

5.- Expresiones regulares

Inicio – fin

- `^` `^hola` empieza con "hola".
- `$` `hola$` acaba con "hola".
- `^ $` `^hola$` exactamente "hola".

Otros

- `\s` el carácter espacio en blanco.
- `\S` cualquier carácter que no sea espacio en blanco.
- `\w` una letra.
- `\C` no es una letra.
- `\d` un dígito. Similar: `[0-9]`
- `\D` no es un dígito. Similar: `[^0-9]`

5.- Expresiones regulares

Ejemplos de expresiones regulares en PHP:

```
// tiene 4 minúsculas  
'/[a-z]{4}/'
```

```
// tiene 8 caracteres: letras y/o números  
'/[a-zA-Z0-9]{8}/'
```

```
// 7 u 8 dígitos seguidos de 1 letra (DNI)  
'/^\\d{7,8}\\w{1}$/'
```

```
// conjunto de letras, seguidas de arroba seguida, seguidas de un  
// conjunto de letras, seguidas de un punto y seguido de 2 o 3  
// letras → patrón simple para un email: rick_sanchez@mail.com  
'/^\\[a-z_\\.]+@[a-z]+\\.\\[a-z]{2,3}$/'
```

5.- Expresiones regulares

Ejemplos de expresiones regulares en PHP:

```
// conjunto de letras, seguidas de arroba seguida, seguidas de un  
// conjunto de letras, seguidas de un punto y seguido de 2 o 3  
// letras → patrón simple para un email: rick_sanchez@mail.com  
'/^ [a-z_ . ]+@[a-z]+ \. [a-z]{2,3}$/'
```

Las expresiones regulares se pueden almacenar en variables.

```
$expr4Minusculas = '/[a-z]{4}/';
```

5.- Expresiones regulares

Si se quiere comprobar si una cadena cumple el patrón establecido por una expresión regular se usa la función [preg_match](#).

Aunque esta función puede recibir hasta 5 parámetros bastará con utilizar los dos primeros para realizar las comprobaciones.

El valor devuelto será:

- **1** si la cadena cumple el patrón
- **0** si la cadena **no cumple** el patrón
- **false** si se produce un **error**.

```
preg_match(  
    string $pattern,  
    string $subject,  
    array &$matches = ?,  
    int $flags = 0,  
    int $offset = 0  
): int
```

5.- Expresiones regulares

Ejemplo de uso de `preg_match()`

```
$exprPass = '/[a-zA-Z0-9]{8}/';  
$pass = 'Xje3T8Uk';  
if (!preg_match($exprPass, $pass)) {  
    echo 'La contraseña debe tener al menos 8 letras y/o números';  
}
```

Si se comprueba una variable recibida desde un formulario se usa `$_POST`:

```
$pass = 'Xje3T8Uk';  
if (!preg_match($exprPass, $_POST['pass'])) {  
    echo 'La contraseña debe tener al menos 8 letras y/o números';  
}
```

5.- Expresiones regulares

La sintaxis para definir expresiones regulares en PHP se puede encontrar en la documentación oficial:

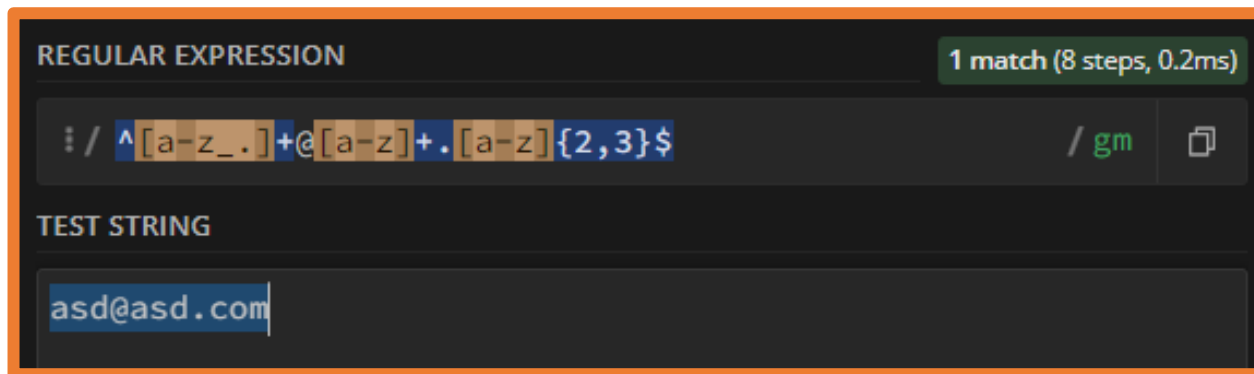
<https://www.php.net/manual/en/reference.pcre.pattern.syntax.php>

También se pueden encontrar Cheat Sheets:

<https://quickref.me/regex>

Existen herramientas para comprobar el funcionamiento de una expresión regular:

<https://regex101.com/>



Práctica

Actividad 3:

Validando la información.

Actividad 4:

Oferta de trabajo.

6.- Tipos MIME

MIME → **M**ultipurpose Internet **M**ail **E**xtensions

En los inicios de internet, para enviar contenido por email se definió el **estándar MIME**.

Este estándar indicaba el tipo de contenido que se enviaba por email.

Hoy en día los tipos MIME se utilizan para cualquier envío de archivos por internet.

Cuando el servidor envía un archivo (html, php, png...) lo primero que hace es enviar las cabeceras en las cuales se indica el tipo MIME.

6.- Tipos MIME

MIME → **M**ultipurpose Internet **M**ail **E**xtensions

En los inicios de internet, para enviar contenido por email se definió el **estándar MIME**.

Este estándar indicaba el tipo de contenido que se enviaba por email.

Hoy en día los tipos MIME se utilizan para cualquier envío de archivos por internet.

Cuando el servidor envía un archivo (html, php, png...) lo primero que hace es enviar las cabeceras en las cuales se indica el tipo MIME.

6.- Tipos MIME

El tipo MIME se indica mediante una cadena que tiene la siguiente estructura: tipo/subtipo

Algunos ejemplos:

| Extensión | Tipo | MIME |
|-----------|-------------------|------------------------|
| .css | Hoja de estilo | text/css |
| .epub | Libro electrónico | application/epub+zip |
| .js | JavaScript | application/javascript |
| .json | Formato JSON | application/json |
| .png | Imagen PNG | Image/png |

[Lista completa de tipos MIME.](#)

6.- Tipos MIME

Gracias a los tipos MIME los servidores y clientes tienen información de los archivos que intercambian.

Por ejemplo, si el servidor ejecuta un script PHP que genera código HTML, al cliente le enviará un archivo de texto con HTML.

La cabecera que enviará el servidor será:

Content-type: text/html

7.- Subir archivos al servidor

Los formularios, además de para enviar información en forma de texto, también permiten enviar ficheros.

Al enviar archivos se debe tener en cuenta lo siguiente:

- El método ha de ser siempre **post**.
- En el formulario se debe indicar el tipo MIME mediante el atributo: **enctype="multipart/form-data"**
- El servidor tiene configurado un **tamaño máximo de archivo** de 2MB (php.ini, se puede cambiar). Otra manera de cambiar el tamaño máximo de archivo es añadiendo al formulario un campo oculto como el siguiente:

```
<input type="hidden" name="MAX_FILE_SIZE" value="20000">
```
- En el servidor los ficheros se almacenan en un **directorio temporal** y cuando acaba la ejecución del script que recibe los datos, los ficheros se **eliminan**.

7.- Subir archivos al servidor

Para enviar archivos se utilizan los **input** tipo **file** en el formulario:

```
<form action="subida.php" method="post" enctype="multipart/form-data">  
  Nombre:  
  <input type="text" name="nombre" id="nombre">  
  <br>  
  Selecciona tu foto de perfil:  
  <input type="file" name="foto" id="foto">  
  <input type="submit" value="Enviar">  
</form>
```

Para enviar más de un archivo en un formulario se puede añadir el parámetro **multiple** al **input file** o añadir varios **input file**.

8.- Recibiendo archivos desde un formulario

El **tratamiento** de archivos enviados desde un formulario se debe realizar en el **script** que se indique en el atributo **action** del formulario.

En ese script se puede usar la variable superglobal **\$_FILES** para acceder a la información de los archivos recibidos.

El array **\$_FILES** es **asociativo** y tendrá tantos elementos como archivos se hayan recibido.

A su vez, **cada elemento de \$_FILES** será un **array asociativo** con toda la información de un archivo.

8.- Recibiendo archivos desde un formulario

A su vez, **cada elemento de `$_FILES`** será un **array asociativo** con toda la información de un archivo.

- **tmp-name**: ruta del archivo temporal en el servidor.
- **name**: nombre del archivo original (en el cliente).
- **size**: tamaño en bytes del archivo.
- **type**: tipo MIME del archivo (image/gif, text/plain...).
- **error**: código de error (UPLOAD_ERR_OK si se ha subido bien).

8.- Recibiendo archivos desde un formulario

Ejemplo, dado el siguiente formulario:

```
<form action="#" method="post" enctype="multipart/form-data">
  Nombre:
  <input type="text" name="nombre" id="nombre"><br>
  Selecciona la foto para subirla (formato .jpg):<br>
  <input type="file" name="foto" id="foto"><br>
  <input type="submit">
</form>
```

En el script **subida.php** se puede acceder a la información del archivo:

```
echo $_FILES['foto']['tmp_name'];
```

8.- Recibiendo archivos desde un formulario

subida.php

```
// Se comprueba si se ha producido algún error al subir el archivo
// La constante UPLOAD_ERR_OK contiene el valor 0
// Si $_FILES['foto']['error'] no es 0 es que ha habido error
if ($_FILES['foto']['error'] != UPLOAD_ERR_OK) {
    echo 'Error: ';
    switch ($_FILES['foto']['error']) {
        case UPLOAD_ERR_INI_SIZE:
        case UPLOAD_ERR_FORM_SIZE:    echo 'El fichero es demasiado grande.';
                                      break;
        case UPLOAD_ERR_PARTIAL:      echo 'El fichero no se ha podido subir entero.';
                                      break;
        case UPLOAD_ERR_NO_FILE:      echo 'No se ha podido subir el fichero.';
                                      break;
        default:                      echo 'Error indeterminado.';
    }
    exit();
}

// Si no ha habido error se comprueba que el archivo sea del tipo requerido
if ($_FILES['foto']['type'] != 'image/jpeg') {
    echo 'Error: No se trata de un fichero .jpg/.jpeg.';
    exit();
}
```

8.- Recibiendo archivos desde un formulario

subida.php

```
// Si no ha habido error se comprueba que el archivo sea del tipo requerido
if ($_FILES['foto']['type'] != 'image/jpeg') {
    echo 'Error: No se trata de un fichero .jpg/.jpeg.';
    exit();
}

// Si no ha habido error y el archivo es del tipo requerido se comprueba si
// si el archivo es uno recién subido al servidor (como medida de seguridad)
if (is_uploaded_file($_FILES['foto']['tmp_name']) === true) {
    // Hay ocasiones en las que si ya existe el archivo no se debe sobrescribir
    // en esos casos se hará la siguiente comprobación.
    $nuevaRuta = './img/fotos-perfil/'. $_FILES['foto']['name'];
    if (is_file($nuevaRuta) === true) {
        echo 'Error: Ya existe un archivo con el mismo nombre.';
        exit();
    }

    // Se procede a mover el fichero desde el directorio temporal al directorio final
    if (!move_uploaded_file($_FILES['foto']['tmp_name'], $nuevaRuta)) {
        echo 'Error: No se puede mover el fichero a su destino';
    }
} else {
    echo 'Error: Posible ataque. Nombre: ' . $_FILES['foto']['nombre'];
}
```

8.- Recibiendo archivos desde un formulario

Para comprobar que se reciben datos del formulario se comprueba si el array `$_POST` tiene elementos.

```
if (!empty($_POST)) {  
    // Acciones  
}
```

Es importante destacar que los archivos recibidos no se encuentran en `$_POST` si no en `$_FILES`.

Si se quiere comprobar si se han recibido archivos se debe usar la variable `$_FILES`:

```
if (!empty($_FILES)) {  
    // Acciones  
}
```

Tip programming

Al guardar en el servidor archivos recibidos mediante formularios es habitual también almacenar el nombre del archivo en la base de datos.

La razón de esta práctica se verá en la unidad 5.

Se requiere de un control para que los archivos no tengan caracteres extraños o contenga información sensible.

Se puede usar la función **uniqid** para generar un nombre aleatorio.

```
$uuid = uniqid("img", true);
```

9.- Recibiendo archivos – SEGURIDAD

Recibir datos desde el cliente puede implicar riesgos de seguridad.

Primero se realiza la comprobación de errores `$_FILES['archivo']['error']`, de esta manera solo se continúa la ejecución del script php si el archivo se ha subido de manera correcta.

A continuación, se debe comprobar que el tipo de archivo recibido sea el esperado `$_FILES['archivo']['type']`.

Por último, se debe comprobar que el archivo realmente se ha subido y no se está accediendo a un archivo ya almacenado en el servidor.

9.- Recibiendo archivos – SEGURIDAD

Hace unos años existía un ataque que consistía en engañar al script que recibía el formulario para que cogiera al archivo local `/etc/passwd` y así acceder a información de acceso de los usuarios.

Se crearon las funciones **`is_uploaded_file`** y **`move_uploaded_file`** para asegurarse que se trabaja con archivos cargados desde HTTP POST.

También es habitual asegurarse que no se está sobrescribiendo un archivo ya existente mediante la función **`is_file`**. Aunque hay casos en los que sí es necesario sobrescribir como por ejemplo al actualizar la foto de perfil.

10.- Fallos típicos

- No tener permiso sobre la ruta indicada en el archivo **php.ini** en la directiva **upload_tmp_dir**.
- La directiva **memory_limit** en **php.ini** tiene un valor muy pequeño o menor al de la directiva **upload_max_filesize**.
- La directiva **max_execution_time** en **php.ini** tiene un valor demasiado bajo y la ejecución del script (que incluye la subida del archivo) excede dicho tiempo.
- La directiva **post_max_size** (tamaño máximo de archivo + resto de datos que se envían en el formulario) de **php.ini** tiene que tener un valor mayor a **upload_max_filesize**.

11.- Funciones para archivos

Además de las funciones para trabajar con archivos que se han visto PHP ofrece algunas más:

- delete
- realpath
- dirname
- is_dir
- rename
- mkdir
- rmdir

<https://www.php.net/manual/es/function.realpath.php>

Práctica

Actividad 5:

Oferta de trabajo currículum.

Actividad 6:

Oferta de trabajo foto.

12.- Tratamiento de imágenes en PHP

El tratamiento de imágenes aplicaciones web suele darse en los siguientes supuestos:

- Crear gráficos de barras.
- Añadir marca de agua a una imagen.
- Crear una imagen a menor resolución para enviarla al cliente.
- Guardar varias versiones de una imagen recibida desde el cliente.
- ...

12.- Tratamiento de imágenes en PHP

PHP incorpora una librería para trabajar con imágenes: **GD v2.x**

Para activarla se debe descomentar la línea siguiente (borrar el ;) del archivo php.ini:
;extensión=gd

Tras reiniciar Apache mediante la función `phpinfo()` se puede ver la información sobre la librería

| gd | | |
|------------------------|----------------------------|--------------|
| GD Support | enabled | |
| GD Version | bundled (2.1.0 compatible) | |
| FreeType Support | enabled | |
| FreeType Linkage | with freetype | |
| FreeType Version | 2.9.1 | |
| GIF Read Support | enabled | |
| GIF Create Support | enabled | |
| JPEG Support | enabled | |
| libJPEG Version | 8 | |
| PNG Support | enabled | |
| libPNG Version | 1.6.34 | |
| WBMP Support | enabled | |
| XPM Support | enabled | |
| libXpm Version | 30512 | |
| XBM Support | enabled | |
| WebP Support | enabled | |
| BMP Support | enabled | |
| AVIF Support | enabled | |
| TGA Read Support | enabled | |
| Directive | Local Value | Master Value |
| gd.jpeg_ignore_warning | 1 | 1 |

12.- Tratamiento de imágenes en PHP

Cargando imágenes con GD

En el código HTML:

```

```

logo.php

```
<?php
header('Content-Type: image/png');
$logo = imagecreatefrompng('logo.png');
imageAlphaBlending($logo, true);
imageSaveAlpha($logo, true);
imagepng($logo);
imagedestroy($logo);
```

12.- Tratamiento de imágenes en PHP

Cargando imágenes con GD

Cargar imágenes mediante la librería GD permite por ejemplo añadir una marca de agua en la imagen.

En el código **HTML**:

```

```

En el archivo **marcaagua.php**:

```
$logo = imagecreatefrompng($_GET['img'] . '.png');
```

Más adelante se verá cómo añadir la marca de agua.

12.- Tratamiento de imágenes en PHP

Generar imágenes con GD

Hay ocasiones en los que es interesante generar imágenes desde cero, este proceso requiere de 4 pasos básicos:

- Crear lienzo sobre el que trabajar.
- Dibujar formas y/o imprimir texto en dicho lienzo.
- Generar la imagen final.
- Liberar los recursos.

12.- Tratamiento de imágenes en PHP

Generar imágenes con GD

Crear lienzo sobre el que trabajar.

Dibujar formas y/o imprimir texto en dicho lienzo.

Generar la imagen final.

Liberar los recursos.

```
<?php
// Creación y configuración del Lienzo
$alto = $ancho = 200;
$imagen = imagecreatetruecolor($ancho, $alto);
$blanco = imagecolorallocate($imagen, 255, 255, 255);
$azul = imagecolorallocate($imagen, 0, 0, 64);

// Se dibuja la imagen
imagefill($imagen, 0, 0, $azul);
imageline($imagen, 0, 0, $ancho, $alto, $blanco);
imagestring($imagen, 4, 50, 150, 'DWES', $blanco);

// Generación de la imagen.
header('content-type: image/png');
imagepng ($imagen);

// Liberamos la imagen de memoria.
imagedestroy($imagen);
```


12.- Tratamiento de imágenes en PHP

Generar imágenes con GD

La función **imagecreatetruecolor** permite crear un lienzo en blanco de las medidas que se indiquen.

Se puede crear un lienzo a partir de una imagen, de esta manera el lienzo tendrá el tamaño de la imagen que se carga:

- `imagecreatefromgif`
- `imagecreatefromjpeg`
- `imagecreatefrompng`
- `imagecreatefromwebp`

12.- Tratamiento de imágenes en PHP

Generar imágenes con GD

La función **imagecolorallocate** permite crear un color, si se quiere transparencia existe la función **imagecolorallocatealpha**.

Una vez creado un lienzo y un color, se puede pintar en dicho lienzo usando las funciones existentes:

- **imagefill**
- **imageline**
- **imagestring**
- **imagearc**

12.- Tratamiento de imágenes en PHP

Generar imágenes con GD

La función **imagestring** permite añadir texto aunque es muy limitada.

La función **imagettftext** que permite usar fuentes (.ttf) descargadas en el servidor.

Ejemplo:

```
imagettftext($imagen, 20, 0, 11, 21, $gris, 'arial.ttf', 'DWES');
```

12.- Tratamiento de imágenes en PHP

Transformar imágenes con GD

Además de cargar y generar imágenes, la biblioteca GD permite transformar imágenes.

Los usos más típicos son:

- Guardar varias versiones/tamaños de la imagen.
- Poner marca de agua en las imágenes.

12.- Tratamiento de imágenes en PHP

Cambio de tamaño

```
<?php
$img_org = imagecreatefromgif("dwes.gif");

$ancho_dst = intval(imagesx($img_org) / 2);
$alto_dst = intval(imagesy($img_org) / 2);

$img_dst = imagecreatetruecolor($ancho_dst, $alto_dst);

imagecopyresized($img_dst, $img_org, 0, 0, 0, 0,
                 $ancho_dst, $alto_dst,
                 imagesx($img_org), imagesy($img_org));

header("Content-type: image/png");
imagepng($img_dst);

imagedestroy($img_org);
imagedestroy($img_dst);
```

12.- Tratamiento de imágenes en PHP

Cambio de tamaño

También existe la función **imagecopyresampled** que aplica un suavizado a los bordes para obtener un resultado de mejor calidad.

Si el cambio de tamaño no respeta la relación de aspecto la imagen final se verá distorsionada.

El eje de coordenadas está en la esquina superior izquierda.

12.- Tratamiento de imágenes en PHP

Cambio de tamaño

Si simplemente se quiere escalar la imagen se usa la función **imagescale**, a la que habrá que proporcionar las nuevas medidas.

Si solo se indica el ancho el escalado será proporcional:

```
$imagen = imagescale($imagenOriginal, ANCHO, ALTO);  
// nuevas dimensiones de la imagen: 200x100  
$imagen = imagescale($imagenOriginal, 200, 100);  
// nuevas dimensiones de la imagen: 75% de la original  
$imagen = imagescale($imagenOriginal, imagesx($imagenOriginal)*0.75);  
// nuevas dimensiones de la imagen: 50% de la original  
$imagen = imagescale($imagenOriginal, imagesx($imagenOriginal)*0.5);
```

12.- Tratamiento de imágenes en PHP

Marca de agua

```
<?php
$marca = imagecreatefrompng('marca.png');
$marca = imagescale($marca, 50);
imagealphablending($marca, false);
imagesavealpha($marca, true);

$marcaX = imagesx($marca);
$marcaY = imagesy($marca);
imagefilter($marca, IMG_FILTER_COLORIZE, 0, 0, 0, 60);

$imagen = imagecreatefromjpeg('logo.jpg');
$destX = imagesx($imagen) - $marcaX - 5;
$destY = imagesy($imagen) - $marcaY - 5;

imagecopy($imagen, $marca, $destX, $destY, 0, 0, $marcaX, $marcaY);
header('content-type: image/jpeg');
imagejpeg($imagen);

imagedestroy($imagen);
imagedestroy($marca);
```


12.- Tratamiento de imágenes en PHP

Guardar imágenes

Hasta ahora se han tratado las imágenes enviándolas al cliente.

En ocasiones se puede trabajar con imágenes sin necesidad de enviar la imagen al cliente, por ejemplo, cuando se sube la foto de perfil.

Para guardar una imagen se utilizan las mismas funciones que para mostrarlas pero añadiendo la ruta destino:

```
imagejpeg($imagen, 'img/foto.jpg');  
imagepng($imagen, 'img/foto.png');
```

Si no se va a enviar la imagen al cliente no será necesario indicar la cabecera:

```
header("Content-type: image/png");
```

Práctica

Actividad 7:

Oferta de trabajo 2 fotos.

Actividad 8:

Catálogo on-line.