Programación Hipermedia I

Práctica 11: PHP 6 (tratamiento de imágenes)

1. Objetivos

- Aprender a utilizar una biblioteca para el tratamiento de imágenes.
- Aprender a realizar una paginación en un listado.

2. Recursos

¿Cómo se procesa una imagen en PHP?

- **Procesamiento de imágenes y GD**[1]: documentación oficial del interfaz de GD en PHP.
- **LibGD**[2]: sitio web oficial de la biblioteca GD.
- **PHP: Dynamic Image Generation**[3]: ejemplos sencillos de uso de GD.

3. ¿Qué tengo que hacer?

En esta práctica tienes que usar una biblioteca para el tratamiento de imágenes que te permita modificar las imágenes que suben los usuarios al servidor.

En concreto, tienes que implementar las siguientes características:

Página principal Muestra un diagrama de barras con el número de fotografías subidas durante los últimos siete días.

Página detalle usuario Al mostrar el listado de todos los álbumes del usuario, además de mostrar el título de cada álbum se debe seleccionar y mostrar una imagen a modo de carátula del álbum. Esta imagen debe ser una miniatura de la imagen original.

Página "Mis álbumes" Al mostrar el listado de todos los álbumes del usuario, además de mostrar el título de cada álbum se debe seleccionar y mostrar una imagen a modo de carátula del álbum. Esta imagen debe ser una miniatura de la imagen original.

Página "Ver álbum" Al mostrar el listado de todas las fotos de un álbum, se muestran miniaturas de las imágenes originales. Este listado no muestra todas las imágenes a la vez, sino que lo realiza mediante una paginación (por ejemplo, de 10 en 10).

Opcional: la carátula del álbum puede ser simplemente una imagen, o puedes realizar un efecto especial como el mostrado en la Figura 1.

4. ¿Cómo lo hago?

Existen múltiples bibliotecas para el tratamiento de imágenes en PHP. Una de las más utilizadas es GD (*Graphics Draw*). Está programada en C, pero se han desarrollado interfaces para otros lenguajes de programación, como por ejemplo PHP.

GD puede crear imágenes a partir de líneas, arcos, texto (usando las fuentes seleccionadas o TrueType), otras imágenes, o múltiples colores. GD puede crear y manipular imágenes en formato GIF, JPEG, PNG, y WBMP.

[1] `http://php.net/manual/es/book.image.php`
[2] `http://www.libgd.org/`
[3] `http://webcheatsheet.com/PHP/dynamic_image_generation.php`

Figura 1: Ejemplo de carátula de álbum

4.1. Creación de imágenes con GD

Con GD se pueden crear imágenes desde cero o se pueden manipular imágenes que ya existen. Para crear una imagen desde cero, dibujándola, se pueden utilizar las siguientes funciones:

- `imagecreatetruecolor()`: crea una imagen nueva de color verdadero.
- `imagecolorallocate()`: define un color para una imagen.
- `imagefill()`: rellena una imagen con un color a partir de una posición.
- `imagearc()`: dibuja un arco.
- `imagefilledarc()`: dibuja un arco y con relleno.
- `imageellipse()`: dibuja una elipse.
- `imagefilledellipse()`: dibuja una elipse con relleno.
- `imagepolygon()`: dibuja un polígono.
- `imagefilledpolygon()`: dibuja un polígono con relleno.
- `imagerectangle()`: dibuja un rectángulo.
- `imagefilledrectangle()`: dibuja un rectángulo con relleno.
- `imagestring()`: dibuja una cadena de texto horizontal.
- `imagegif()`: exporta una imagen al navegador o a un fichero en formato GIF.
- `imagepng()`: exporta una imagen al navegador o a un fichero en formato PNG.
- `imagejpeg()`: exporta una imagen al navegador o a un fichero en formato JPEG.
- `imagedestroy()`: libera toda la memoria asociada con una imagen.

En el siguiente ejemplo, se utilizan algunas de las funciones anteriores para dibujar el diagrama circular en tres dimensiones que se muestra en la Figura 2:

```
<?php
    // Crea una imagen
    $image = imagecreatetruecolor(100, 100);

    // Define los colores que se van a emplear
    $white   = imagecolorallocate($image, 0xFF, 0xFF, 0xFF);
    $gray    = imagecolorallocate($image, 0xC0, 0xC0, 0xC0);
    $darkgray = imagecolorallocate($image, 0x90, 0x90, 0x90);
```

Figura 2: Ejemplo de imagen creada con GD

```
    $navy    = imagecolorallocate($image, 0x00, 0x00, 0x80);
    $darknavy = imagecolorallocate($image, 0x00, 0x00, 0x50);
    $red     = imagecolorallocate($image, 0xFF, 0x00, 0x00);
    $darkred = imagecolorallocate($image, 0x90, 0x00, 0x00);

    // Rellena la imagen de blanco
    imagefill($image, 0, 0, $white);

    // Dibuja unos arcos con efecto 3D
    for ($i = 60; $i > 50; $i--) {
        imagefilledarc($image, 50, $i, 100, 50, 0, 45, $darknavy, IMG_ARC_PIE);
        imagefilledarc($image, 50, $i, 100, 50, 45, 75, $darkgray, IMG_ARC_PIE);
        imagefilledarc($image, 50, $i, 100, 50, 75, 360, $darkred, IMG_ARC_PIE);
    }

    imagefilledarc($image, 50, 50, 100, 50, 0, 45, $navy, IMG_ARC_PIE);
    imagefilledarc($image, 50, 50, 100, 50, 45, 75, $gray, IMG_ARC_PIE);
    imagefilledarc($image, 50, 50, 100, 50, 75, 360, $red, IMG_ARC_PIE);

    // Vuelca la imagen en la salida estándar
    header('Content-type: image/png');
    imagepng($image);

    // Libera los recursos utilizados
    imagedestroy($image);
?>
```

4.2. El esquema data:

El ejemplo anterior es un script PHP que devuelve una imagen, pero no una página web. ¿Cómo se podría devolver una página web con la imagen? Se podría hacer de varias formas:

1. El script podría almacenar la imagen en un fichero temporal al cual se haría referencia desde la etiqueta `<img />`. Pero, ¿cuándo se elimina el fichero temporal?

2. Se podrían tener dos scripts, uno para la página web y otro para la imagen. Pero, ¿qué ocurre si hay una gran dependencia entre el contenido de la página web y la imagen?

Existe una solución a este problema, el esquema `data:`[4], que permite la inclusión de pequeños elementos de datos en línea, como si fueran referenciados hacia una fuente externa. Este formato ofrece algunas ventajas:

- Para ficheros pequeños, puede suponer una transmisión de datos menor.
- Reduce el número de conexiones que se establecen con el servidor: algunos navegadores tienen un límite máximo de conexiones simultáneas.
- Permite gestionar una página web, con todos sus recursos, como un único fichero.

[4]RFC 2397 The "data" URL scheme: `http://tools.ietf.org/html/rfc2397`

- Permite el uso de recursos en aquellas situaciones en las que existen limitaciones para hacer uso de recursos externos referenciados.

No se debe abusar de este formato: para ficheros pequeños puede suponer un ahorro, pero para ficheros grandes aumenta la cantidad de datos transmitidos, ya que las imágenes no se envían en formato binario, sino codificadas como se explica a continuación, y esta codificación supone un incremento en el tamaño de los datos enviados.

En el esquema `data:` se tiene que indicar el tipo MIME de los datos devueltos y su codificación. Para el envío de las imágenes se emplea la codificación `base64`, una codificación que permite representar ficheros binarios con los caracteres imprimibles de ASCII: los caracteres A-Z, a-z y 0-9 (62 caracteres) más dos caracteres adicionales que suelen variar entre diferentes versiones de `base64` (normalmente se suelen usar los caracteres "+" y "/", junto con el carácter "=" como sufijo especial de relleno).

En el siguiente ejemplo se muestra cómo integrar una imagen creada con GD directamente en una página web:

```
<?php
    // Crea una imagen
    $image = imagecreatetruecolor(100, 100);

    // Define los colores que se van a emplear
    $white   = imagecolorallocate($image, 0xFF, 0xFF, 0xFF);
    $gray    = imagecolorallocate($image, 0xC0, 0xC0, 0xC0);
    $darkgray = imagecolorallocate($image, 0x90, 0x90, 0x90);
    $navy    = imagecolorallocate($image, 0x00, 0x00, 0x80);
    $darknavy = imagecolorallocate($image, 0x00, 0x00, 0x50);
    $red     = imagecolorallocate($image, 0xFF, 0x00, 0x00);
    $darkred = imagecolorallocate($image, 0x90, 0x00, 0x00);

    // Rellena la imagen de blanco
    imagefill($image, 0, 0, $white);

    // Dibuja unos arcos con efecto 3D
    for ($i = 60; $i > 50; $i--) {
        imagefilledarc($image, 50, $i, 100, 50, 0, 45, $darknavy, IMG_ARC_PIE);
        imagefilledarc($image, 50, $i, 100, 50, 45, 75, $darkgray, IMG_ARC_PIE);
        imagefilledarc($image, 50, $i, 100, 50, 75, 360, $darkred, IMG_ARC_PIE);
    }

    imagefilledarc($image, 50, 50, 100, 50, 0, 45, $navy, IMG_ARC_PIE);
    imagefilledarc($image, 50, 50, 100, 50, 45, 75, $gray, IMG_ARC_PIE);
    imagefilledarc($image, 50, 50, 100, 50, 75, 360, $red, IMG_ARC_PIE);

    // Activa el almacenamiento en el buffer de salida
    ob_start();
    imagepng($image);
    // ob_get_contents() devuelve el contenido del buffer de salida
    $img_src = "" . base64_encode(ob_get_contents());
    // Limpia y deshabilita el buffer de salida
    ob_end_clean();

    // Libera los recursos utilizados
    imagedestroy($image);
?>
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba del esquema data:</title>
</head>
<body>
<p>
```

```
La siguiente imagen muestra un diagrama circular en tres dimensiones:
<br /><br />
<img src="<?php echo $img_src; ?>" />
</p>
</body>
</html>
```

En este ejemplo se tiene que aplicar un "pequeño truco" para poder convertir la imagen que se ha creado con GD en una cadena. Sorprendentemente, en GD no hay ninguna función para convertir una imagen en una cadena: la función `imagepng()` muestra directamente una imagen en el flujo de salida, es decir, hacia el navegador web o la almacena en un fichero, pero no permite almacenarla en una cadena. Con las funciones de control del buffer de salida[5] se puede capturar la salida y luego recuperarla en forma de cadena. Las funciones que se han utilizado en el ejemplo son:

- `ob_start()`: activa el almacenamiento en el buffer de salida. Cuando el buffer está activo, no se envía ninguna salida al navegador, sino que se almacena en el buffer.
- `ob_get_contents()`: devuelve el contenido del buffer de salida en forma de cadena.
- `ob_end_clean()`: limpia y deshabilita el buffer de salida.

4.3. Trabajar con imágenes que ya existen

GD también permite manipular imágenes que ya existen. Para abrir una imagen que ya existe y manipularla (copiarla, redimensionarla, cortarla), se pueden utilizar las siguientes funciones:

- `imagecreatefromgif()`: crea una imagen a partir de un fichero GIF.
- `imagecreatefromjpeg()`: crea una imagen a partir de un fichero JPEG.
- `imagecreatefrompng()`: crea una imagen a partir de un fichero PNG.
- `imagesx()`: obtiene el ancho de una imagen.
- `imagesy()`: obtiene el alto de una imagen.
- `imagecolorat()`: obtiene el color de un pixel en una imagen.
- `imagesetpixel()`: establece el color de un pixel en una imagen.
- `imagecrop()`: recorta una imagen.
- `imagecopy()`: copia parte de una imagen.
- `imagecopyresampled()`: copia y cambia el tamaño de parte de una imagen.
- `imagescale()`: redimensiona una imagen con un nuevo ancho y alto.
- `imagerotate()`: gira una imagen con un ángulo dado.
- `imageconvolution()`: aplica una matriz de convolución 3x3 a una imagen.

Una imagen se puede manipular pixel a pixel, aunque es un proceso lento. Por ejemplo, el siguiente código transforma una imagen a escala de grises calculando el nivel de gris de cada pixel:

```
<?php
  header("Content-type: image/jpg");

  $foto = imagecreatefromjpeg("cervino.jpg");

  for($x = 0; $x < imagesx($foto); $x++) {
    for($y = 0; $y < imagesy($foto); $y++) {
      $rgb = imagecolorat($foto, $x, $y);
      // Realiza un desplazamiento de bits para obtener cada componente
      $r = ($rgb >> 16) & 0xFF;
```

[5] http://www.php.net/manual/es/book.outcontrol.php

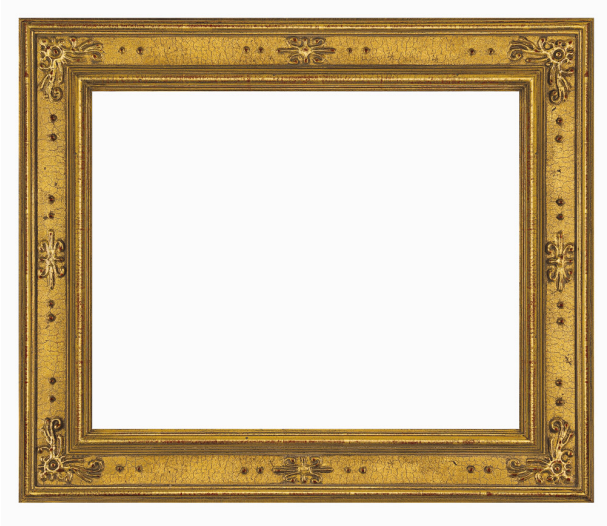

Figura 3: Ejemplo de dos imágenes combinadas

```
      $g = ($rgb >> 8) & 0xFF;
      $b = $rgb & 0xFF;
      $nivel = ($r + $g + $b) / 3;
      $color = imagecolorallocate($foto, $nivel, $nivel, $nivel);
      imagesetpixel($foto, $x, $y, $color);
    }
  }

  // Devuelve la imagen en formato JPEG
  imagejpeg($foto);
?>
```

Por último, el siguiente ejemplo muestra como combinar dos imágenes en una sola. La primera imagen se muestra en la Figura 3 y es un marco de un cuadro. La segunda imagen se muestra en la Figura 4 y es la fotografía de una montaña. Al combinar las dos imágenes se logra la imagen que se muestra en la Figura 5.

```
<?php
  header("Content-type: image/jpg");

  $marco = imagecreatefromjpeg("marco.jpg");
  $foto = imagecreatefromjpeg("cervino.jpg");

  // (150, 152) x (854, 709) son las coordenadas del rectángulo
  // en blanco en el marco
  imagecopyresampled($marco, $foto, 150, 152, 0, 0, (854 - 150 + 1), (709 - 152 + 1),
                     imagesx($foto), imagesy($foto));
  // Devuelve la imagen en formato JPEG
  imagejpeg($marco);
?>
```

5. Recomendaciones

La biblioteca GD ofrece más de 100 funciones: al principio, tantas funciones pueden abrumar. Antes de hacer algo, revisa la biblioteca de funciones y busca las funciones que crees que vas a necesitar.

Para realizar la paginación al mostrar el listado de todas las fotos de un álbum, tendrás que utilizar unos parámetros para indicar la página que se está visualizando. Escribe el código de forma que sea muy fácil configurar el número de resultados que se quieren por página.

A la hora de realizar la consulta SQL con paginación, no es necesario recuperar todo el resultado y realizar un bucle para localizar el resultado deseado, se puede recuperar sólo aquello que se va a mostrar.

Figura 4: Ejemplo de dos imágenes combinadas

Figura 5: Ejemplo de dos imágenes combinadas