Desarrollo de Aplicaciones Web

Práctica 10: PHP 4 (acceso a una base de datos)

1. Objetivos

- Aprender a acceder a una base de datos desde PHP.
- Aprender a realizar una consulta INSERT, UPDATE y DELETE.
- Aprender a validar los datos recibidos desde un formulario web en PHP.

2. Recursos

¿Qué funciones se emplean en PHP para acceder a MySQL?

- **PHP Mysql**[1]: ext/mysql, la API original para usar MySQL desde PHP.
- **PHP Mysqli**[2]: ext/mysqli, la extensión mejorada para usar MySQL desde PHP.
- **PHP MySQL Introduction**[3]: tutorial de W3Schools sobre el uso de una base de datos de MySQL desde PHP.
- **ext/mysqli: Part I - Overview and Prepared Statements**[4] y **Using ext/mysqli: Part II - Extending mysqli**[5]: explica de forma detallada y clara el empleo de mysqli.

¿Existen funciones en PHP para filtrar los datos recibidos?

- **Data Filtering**[6]: documentación oficial del uso de las funciones de filtrado de PHP.
- **PHP Filter**[7]: tutorial de W3Schools sobre el uso de las funciones de filtrado de PHP.
- **PHP Filter Functions**[8]: referencia en W3Schools sobre el uso de las funciones de filtrado de PHP.

¿Cómo se emplean las expresiones regulares con PHP?

- **Regular Expressions (Perl-Compatible)**[9]: documentación oficial sobre el uso de la expresiones regulares con la sintaxis compatible con Perl.
- **PCRE - Perl Compatible Regular Expressions**[10]: documentación oficial sobre la biblioteca de funciones PCRE.
- **Regular Expression (POSIX Extended)**[11]: documentación oficial sobre el uso de expresiones regulares con la sintaxis POSIX.

[1] http://www.php.net/manual/es/book.mysql.php
[2] http://www.php.net/manual/es/book.mysqli.php
[3] http://www.w3schools.com/php/php_mysql_intro.asp
[4] http://devzone.zend.com/239/ext-mysqli-part-i_overview-and-prepared-statements/
[5] http://devzone.zend.com/255/using-ext-mysqli-part-ii_extending-mysqli/
[6] http://es.php.net/manual/es/book.filter.php
[7] http://www.w3schools.com/PHP/php_filter.asp
[8] http://www.w3schools.com/PHP/php_ref_filter.asp
[9] http://es.php.net/manual/es/book.pcre.php
[10] http://www.pcre.org/
[11] http://es.php.net/manual/es/book.regex.php

3. ¿Qué tengo que hacer?

En está práctica debes programar el registro de un nuevo usuario, es decir, la página de respuesta a la página "registro nuevo usuario". El registro se realiza mediante la inserción de los datos del nuevo usuario en la tabla que corresponda. En el lado del servidor, tienes que usar PHP para validar los datos introducidos por el usuario[12]; en concreto, el formulario para registrarse como un nuevo usuario contiene los siguientes campos y restricciones (son las mismas que se definieron en una práctica anterior mediante JavaScript, pero ahora se deben programar en el lado del servidor):

Respuesta "Pagína registro nuevo usuario" La página de registro como nuevo usuario contiene un formulario con los datos necesarios para registrarse (nombre de usuario, contraseña, repetir contraseña, dirección de email, sexo, fecha de nacimiento, ciudad y país de residencia, foto). Una vez enviado el formulario, debes realizar las siguientes comprobaciones en el servidor:

- **nombre de usuario**: sólo puede contener letras del alfabeto inglés (en mayúsculas y minúsculas) y números; no puede comenzar con un número; longitud mínima 3 caracteres y máxima 15.
- **contraseña**: sólo puede contener letras del alfabeto inglés (en mayúsculas y minúsculas), números, el guion y el guion bajo (subrayado); al menos debe contener una letra en mayúscula, una letra en minúscula y un número; longitud mínima 6 caracteres y máxima 15.
- **repetir contraseña**: su valor debe coincidir con el escrito en el campo **contraseña**.
- **dirección de email**: no puede estar vacío, hay que comprobar que cumple el siguiente patrón de una dirección de email (es una simplificación del estándar):
  - El formato de un correo electrónico es `parte-local@dominio`.
  - La longitud máxima de `parte-local` es 64 caracteres. La longitud máxima de `dominio` es 255 caracteres. Cada una de las partes tiene una longitud mínima de 1 carácter.
  - La `parte-local` es una combinación de las letras mayúsculas y minúsculas del alfabeto inglés, los dígitos del 0 al 9, los caracteres imprimibles `!#$%&'*+-/=?^_`{|}~` y el punto. El punto no puede aparecer ni al principio ni al final y tampoco pueden aparecer dos o más puntos seguidos.
  - El `dominio` es una secuencia de uno o más `subdominio` separados por un punto.
  - La longitud máxima de `subdominio` es 63 caracteres.
  - Un `subdominio` es una combinación de las letras mayúsculas y minúsculas del alfabeto inglés, los dígitos del 0 al 9 y el guion. El guion no puede aparecer ni al principio ni al final.
  - La longitud máxima de una dirección de correo electrónico es 254 caracteres[13].
- **sexo**: se debe elegir un valor.
- **fecha de nacimiento**: comprobar que es una fecha válida.
- El resto de campos no indicados se pueden quedar vacíos.
- Por ahora, no se tiene que gestionar el almacenamiento de la foto subida, eso se hará en **una próxima práctica**.

En esta práctica debes aplicar una combinación de los diferentes métodos de validación que existen en PHP: expresiones regulares, funciones de filtrado y validación programada por ti mismo.

Importante: las validaciones de HTML y JS se deben desactivar para que se puedan comprobar correctamente las validaciones de PHP.

Además, tienes que añadir las siguientes páginas a la parte privada de un usuario registrado, algunas de las páginas también pueden incluir algún tipo de validación mediante PHP:

[12] La validación de los datos de entrada también se puede realizar en el cliente mediante JavaScript para mejorar la usabilidad de una aplicación web, pero siempre se debe realizar en el servidor como medida de seguridad.

[13] Evidentemente, no tiene sentido que se diga que la longitud máxima de dominio es 255 caracteres, pero la longitud total es 254 caracteres. Esta incongruencia se debe a un error que existe en el estándar que impide que se cumplan otros estándares relacionados. La explicación está en RFC Errata to 3696: `https://www.rfc-editor.org/errata/eid1690`

Respuesta Página "Configurar" Permite al usuario seleccionar el estilo alternativo que desea[14]. La selección se debe conservar entre diferentes visitas al sitio web, por lo que se debe almacenar en la base de datos. La página de respuesta que confirma la selección de un nuevo estilo ya se debe mostrar con el estilo seleccionado.

Respuesta Página "Mis datos" Tanto en esta página como en [Respuesta "Pagína registro nuevo usuario"] se tienen que realizar los filtrados definidos. Por tanto, lo más conveniente es aislar el código de filtrado en un fichero independiente e incluirlo donde sea necesario. Para confirmar la modificación de los datos, el usuario debe introducir su contraseña actual como medida de seguridad.

Aunque [Pagína registro nuevo usuario] y [Página "Mis datos"] sean muy similares y [Respuesta "Pagína registro nuevo usuario"] y [Respuesta Página "Mis datos"] también lo sean, tienen sus diferencias que debes tener en cuenta. Por ejemplo, en [Pagína registro nuevo usuario], el usuario debe introducir todos los datos que sean obligatorios, pero en [Página "Mis datos"] el usuario puede modificar lo que quiera. Otro detalle importante es que en [Pagína registro nuevo usuario] deberías tener los campos "Contraseña" y "Repetir contraseña", pero en [Página "Mis datos"] deberías tener los campos "Nueva contraseña" y "Repetir nueva contraseña", que deben aparecer vacíos y que el usuario solo debe rellenar si desea modificar su contraseña.

Página "Darme de baja" Elimina al usuario de la base datos (el usuario y todos los datos asociados con el usuario). Antes de realizar el borrado, la operación debe solicitar la confirmación del usuario para evitar que se produzca la operación de forma accidental. Para ello, muestra una página de confirmación con un resumen de la información que almacena la aplicación web: un listado de los álbumes con el número de fotos que contiene cada álbum y el número total de fotos que tiene el usuario. Para confirmar la solicitud de baja, el usuario debe introducir su contraseña actual.

Por ahora, el borrado de las fotos solo se tiene que realizar en la base de datos, el borrado de los ficheros de las fotos en el sistema de archivos se realizará en **una próxima práctica**. Por tanto, al darse de baja un usuario, en el sistema quedará "basura" (recuerda, por ahora, pero en una próxima práctica lo tendrás que evitar).

Respuesta Página "Crear álbum" Realiza la inserción de un álbum nuevo en la base de datos. El título de un álbum es obligatorio, no puede estar vacío. Una vez realizada la inserción, se debe invitar al usuario a introducir su primera fotografía en el álbum que se acaba de crear.

Respuesta Página "Añadir foto a álbum" Realiza la inserción de una foto nueva en la base de datos. El título de una foto es obligatorio, no puede estar vacío. Además, se debe comprobar que el texto alternativo tenga una longitud mínima de 10 caracteres y que no empiece por un texto redundante como, por ejemplo, "foto" o "imagen"[15]. Y por supuesto, no generes un texto alternativo automático como "Texto alternativo para la imagen"[16] si el usuario no introduce un texto alternativo.

Por ahora, no se tiene que gestionar el almacenamiento de la foto subida, eso se hará en **una práctica posterior**. Por tanto, el fichero de la foto se debe subir "a mano" al servidor web.

Y tienes que modificar las siguientes páginas:

Respuesta "Solicitar álbum" Realiza la inserción de una solicitud de un álbum impreso.

En concreto, tienes que modificar o crear las páginas que se indican con un color de relleno claro u oscuro en la Figura 1.

Importante: la práctica no puede fallar porque el usuario introduzca datos erróneos, se tienen que realizar todas las validaciones de los datos de entrada para asegurar el correcto funcionamiento de la práctica, aunque no estén indicadas explícitamente. Por ejemplo, PHP o MySQL no pueden fallar porque el usuario haya introducido un texto en un campo en el que se espere un número o una fecha.

[14] Los estilos alternativos que deben aparecer son, como mínimo, los que desarrollaste en prácticas anteriores, pero si quieres, puedes añadir más estilos alternativos.

[15] En realidad, el texto alternativo debería ser más largo que 10 caracteres en la mayoría de las situaciones. El texto alternativo no debe empezar por "texto", "imagen", "imagen de...", "foto", "foto de..." porque no es necesario, ya que un lector de pantalla que anuncie el texto alternativo, lo primero que dice es que se trata de una imagen o un gráfico.

[16] Existen aplicaciones que hacen esto para "engañar" a las herramientas de evaluación automática de la accesibilidad web que comprueban si las imágenes poseen un texto alternativo.

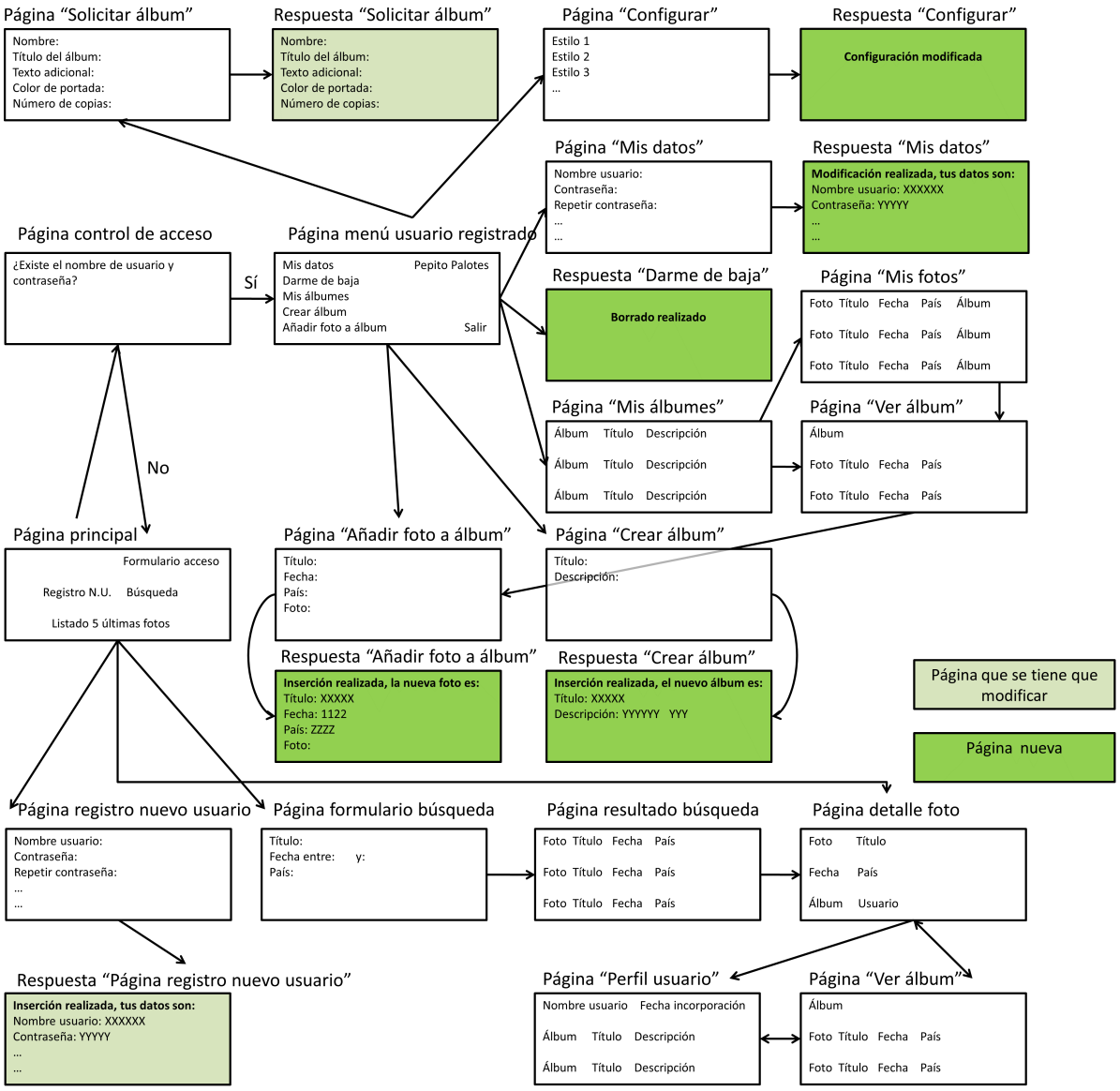

Figura 1: Diagrama de páginas que componen el sitio web

4. ¿Cómo lo hago?

4.1. Funciones de filtrado

Los datos que se reciben a través de Internet **siempre se tienen que validar y filtrar en el servidor**, ya que pueden provenir de una fuente no segura. Aunque los datos provengan de un formulario donde se han verificado previamente con JavaScript, **no hay que confiar en ello**: la ejecución de JavaScript se puede desactivar en el navegador, el código JavaScript puede fallar, el navegador web puede ser que no ejecute el código JavaScript porque no sea compatible o puede que un usuario malintencionado con conocimientos envíe directamente datos incorrectos al servidor.

Por ejemplo, con las herramientas para el desarrollador que poseen la mayoría de los navegadores web, se pueden cambiar o desactivar todas las comprobaciones que se realicen desde HTML o JavaScript.

PHP proporciona un conjunto de funciones básicas para validar y sanear los datos de entrada que pueden ayudar a filtrar los datos de entrada. Para validaciones más complejas será necesario desarrollar un conjunto propio de funciones de filtrado para los distintos casos que se pueden dar.

Las funciones de filtrado de datos en PHP son:

- `filter_has_var()`: verifica si la variable del tipo especificado existe.
- `filter_id()`: devuelve el ID del filtro con el nombre dado.
- `filter_input_array()`: obtiene múltiples variables desde afuera de PHP y opcionalmente las filtra.
- `filter_input()`: obtiene una variable desde afuera de PHP y opcionalmente la filtra.
- `filter_list()`: devuelve una lista de todos los filtros soportados.
- `filter_var_array()`: filtra múltiples variables con el mismo o con diferentes filtros.
- `filter_var()`: filtra una variable con un filtro específico.

Estas funciones se pueden emplear para realizar dos tipos de filtrados:

- Validación:
  - Aplica reglas estrictas de formato (por ejemplo, URL o email).
  - Devuelve el dato original o `false` si no pasa el filtro.
- Saneamiento:
  - Se emplean para permitir o no permitir ciertos caracteres en los datos.
  - No emplea reglas de formato.
  - Devuelve la cadena original tras eliminar los caracteres no permitidos.

Para indicar el tipo de filtrado a realizar (validación o saneamiento), se emplean unas constantes que se utilizan al llamar a las funciones de filtrado. Las principales constantes de filtros de validación son:

- `FILTER_VALIDATE_BOOLEAN`: valida un booleano.
- `FILTER_VALIDATE_INT`: valida un entero.
- `FILTER_VALIDATE_FLOAT`: valida un número real.
- `FILTER_VALIDATE_IP`: valida una dirección IP.
- `FILTER_VALIDATE_MAC`: valida una dirección MAC.
- `FILTER_VALIDATE_URL`: valida un URL.
- `FILTER_VALIDATE_EMAIL`: valida una dirección de correo electrónico.

Las principales constantes de filtros de saneamiento son:

- `FILTER_SANITIZE_STRING`: sanea una cadena.

- `FILTER_SANITIZE_NUMBER_INT`: sanea un entero.
- `FILTER_SANITIZE_NUMBER_FLOAT`: sanea un número real.
- `FILTER_SANITIZE_STRING`: sanea una cadena.
- `FILTER_SANITIZE_URL`: sanea una URL.
- `FILTER_SANITIZE_EMAIL`: sanea una dirección de correo electrónico.

Además, existen una serie de opciones y *flags* para especificar información adicional en el proceso de filtrado, como las opciones `min_range` y `max_range` para definir intervalos numéricos o el *flag* `FILTER_FLAG_ALLOW_THOUSAND` para permitir el empleo de la coma como separador de miles en un número real.

Por ejemplo, en el siguiente código se realizan dos validaciones, una de número entero y otra de dirección IP, además se sanea un número real y un correo electrónico eliminando los caracteres que no son válidos:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de funciones de filtrado y saneamiento</title>
</head>
<body>
<p>
<?php
  // Filtra un número entero en un intervalo
  $int_options = array("options" => array("min_range" => 0, "max_range" => 256));
  $var = 123;
  if(filter_var($var, FILTER_VALIDATE_INT, $int_options))
    echo "$var: Correcto<br />";
  else
    echo "$var: Incorrecto<br />";
  $var = 333;
  if(filter_var($var, FILTER_VALIDATE_INT, $int_options))
    echo "$var: Correcto<br />";
  else
    echo "$var: Incorrecto<br />";

  // Filtra dirección IP
  $var = "125.125.125.125";
  if(filter_var($var, FILTER_VALIDATE_IP))
    echo "$var: Correcto<br />";
  else
    echo "$var: Incorrecto";
  $var = "265.125.125.125";
  if(filter_var($var, FILTER_VALIDATE_IP))
    echo "$var: Correcto<br />";
  else
    echo "$var: Incorrecto<br />";

  // Sanea un número real
  $var = "1,345kg";
  echo "$var: " . filter_var($var, FILTER_SANITIZE_NUMBER_FLOAT) . "<br />";
  echo "$var: " . filter_var($var, FILTER_SANITIZE_NUMBER_FLOAT,
                          FILTER_FLAG_ALLOW_THOUSAND) . "<br />";

  // Sanea un correo electrónico
  $var = "sergio\.\lujan( @ )ua\.\es";
  echo "$var: " . filter_var($var, FILTER_SANITIZE_EMAIL) . "<br />";
?>
</p>
</body>
</html>
```

El ejemplo anterior produce la siguiente salida:

```
123: Correcto
333: Incorrecto
125.125.125.125: Correcto
265.125.125.125: Incorrecto
1,345kg: 1345
1,345kg: 1,345
sergio\.\lujan( @ )ua\.\es: sergio.lujan@ua.es
```

4.2. Expresiones regulares

Una expresión regular es un patrón que define un conjunto de cadenas sin enumerar todos sus elementos. Una expresión regular está formada de caracteres (letras, números y signos) y metacaracteres que tienen una función definida (representar otros caracteres o indicar la forma de combinar los caracteres). Los metacaracteres reciben este nombre porque no se representan a ellos mismos, sino que son interpretados de una manera especial.

Los principales metacaracteres que se emplean en las expresiones regulares son:

- `^`: indica que el patrón que lo acompaña está al principio de la cadena.
- `$`: indica que el patrón que lo acompaña está al final de la cadena.
- `.`: representa cualquier carácter.
- `*`: el patrón que lo precede se repite 0 o más veces.
- `?`: el patrón que lo precede se repite 0 o 1 vez.
- `+`: el patrón que lo precede se repite 1 o más veces.
- `{x,y}`: el patrón que lo precede se repite un mínimo de `x` veces y un máximo de `y` veces.
- `|`: permite alternar expresiones.
- `[]`: indica un rango de caracteres válidos.
- `[^]`: indica un rango de caracteres no válidos.
- `()`: permite agrupar expresiones.
- \: permite escapar los metacaracteres para que aparezcan como literales.

En PHP existen dos conjuntos distintos de funciones que permiten utilizar las expresiones regulares, las funciones POSIX y las funciones PCRE. En algunos aspectos ambas funciones comparten la misma sintaxis, aunque por otro lado tienen sus particularidades propias.

Las funciones POSIX son `ereg()`, `eregi()`, `ereg_replace()`, `eregi_replace()`, `split()` y `spliti()`.

Las funciones PCRE (*Perl Compatible Regular Expressions*) implementan las expresiones regulares con la misma sintaxis y semántica que Perl 5. Estas funciones son más potentes (y por tanto, más complejas) y más rápidas que las funciones POSIX. Las funciones PCRE en PHP son: `preg_match()`, `preg_match_all()`, `preg_replace()`, `preg_split()` y `preg_grep()`. En las funciones PCRE la expresión regular debe empezar y acabar con una barra inclinada "/".

El siguiente ejemplo genera la página web que se muestra en la Figura 2; este ejemplo permite introducir una expresión regular, un texto y la función que se desea emplear (POSIX o Perl) y utiliza la función seleccionada para verificar si el texto introducido cumple la expresión regular:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de expresiones regulares</title>
</head>
<body>
<form action="expreg.php" method="post">
<p>
```

```
Expresión regular: <input type="text" name="expreg" size="20" /><br />
Ejemplo: ^[0-9]{1,4}$ o ^[^@ ]+@[^@ ]+\.[^@ .]+$<br />
Texto: <input type="text" name="texto" size="20" /><br />
Función:
<input type="radio" name="func" value="perl" /> Perl
<input type="radio" name="func" value="posix" /> Posix<br />
<input type="submit" value="Enviar" />
<input type="reset" value="Borrar" />
</p>
</form>
<p>
<?php
  if(isset($_POST['expreg']))
  {
    $expreg = stripslashes($_POST['expreg']);
    $texto = stripslashes($_POST['texto']);
    echo 'Expresión regular: ' . $expreg . '<br />';
    echo 'Texto: ' . $texto . '<br />';
    if($_POST['func'] == 'perl')
    {
      if(preg_match('/' . $expreg . '/', $texto) == 0)
        echo 'No';
      else
        echo 'Ok';
    }
    else if($_POST['func'] == 'posix')
    {
      if(ereg($expreg, $texto) == false)
        echo 'No';
      else
        echo 'Ok';
    }
    else
      echo 'Debe seleccionar una función correcta';
  }
?>
</p>
</body>
</html>
```

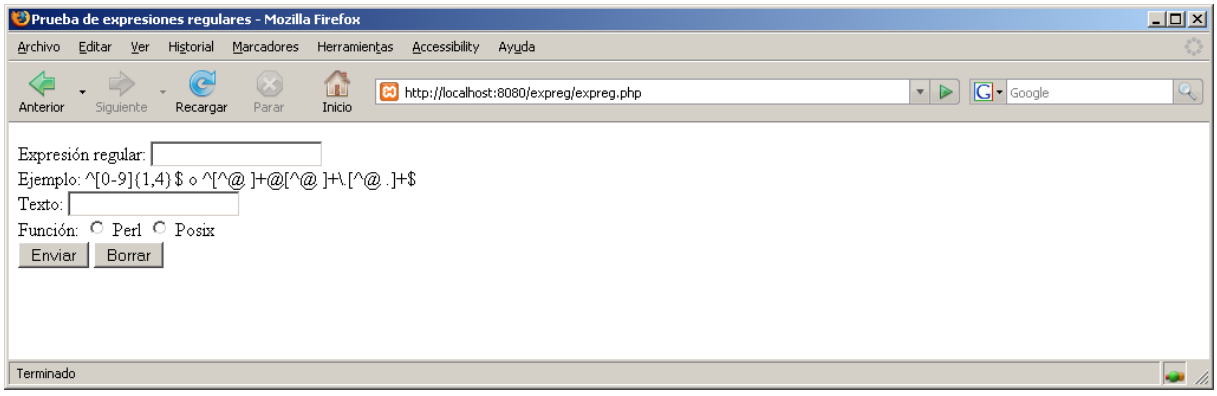

Figura 2: Ejemplo de uso de expresiones regulares en PHP

4.3. Inserción, modificación y borrado en una base de datos

MUY IMPORTANTE: no se debe usar la antigua extensión mysql para nuevos desarrollos, ya que se considera obsoleta y no se desarrollan nuevas funcionalidades para ella, sólo se mantiene para corregir los fallos que aparecen.

4.3.1. Inserción, modificación y borrado

La inserción (INSERT), la modificación (UPDATE) y el borrado (DELETE) de datos mediante SQL en una base de datos creada en MySQL (o MariaDB) se realiza de la misma forma que se ejecuta una consulta (SELECT), excepto que mientras una consulta devuelve un resultado, la inserción, la modificación y el borrado no devuelven un resultado. Si se quiere saber cuántas filas (tuplas o registros) se han visto afectadas por una inserción, una modificación o un borrado se puede emplear la función `mysqli_affected_rows()`.

Cuando se construya la sentencia SQL de inserción o modificación hay que recordar que los valores de tipo cadena y de tipo fecha deben estar encerrados entre comillas y que el orden del tipo fecha puede ser año-mes-día (depende del SGBD y de su configuración).

A continuación se muestra un ejemplo compuesto por dos páginas. La primera página muestra un formulario web que permite al usuario insertar datos en una tabla de una base de datos; la segunda página, programada en PHP, realiza la inserción de los datos recibidos desde el formulario en la base de datos.

El código de la página con el formulario es:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de INSERT y MySQL</title>
</head>
<body>
<h1>Insertar un libro nuevo</h1>
<form action="insert.php" method="post">
<p>
Título: <input type="text" name="titulo" /><br />
Resumen: <textarea name="resumen" rows="5" cols="80"></textarea><br />
Autor: <select name="autor">
<option value="3">Bob Clancy</option>
<option value="2">Fran London</option>
<option value="1">Luis Verne</option>
</select><br />
Categoría: <select name="categoria">
<option value="3">Aventuras</option>
<option value="1">Ciencia ficción</option>
<option value="2">Terror</option>
</select><br />
Editorial: <select name="editorial">
<option value="1">Escribe y publica</option>
<option value="2">Libros a GoGo</option>
<option value="3">Todo libros</option>
</select><br />
Año: <input type="text" name="anyo" size="4" maxlength="4" /><br />
<input type="submit" value="Enviar" />
<input type="reset" value="Borrar" />
</p>
</form>
</body>
</html>
```

El código de la página PHP (`insert.php`) que realiza la inserción en la base de datos es:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de INSERT y MySQL</title>
</head>
<body>
<p>
<?php
  // Recoge los datos del formulario
```

```
  $titulo = $_POST['titulo'];     // text
  $resumen = $_POST['resumen'];   // text
  $autor = $_POST['autor'];       // int
  $categoria = $_POST['categoria']; // int
  $editorial = $_POST['editorial']; // int
  $anyo = $_POST['anyo'];         // int

  // Se conecta al SGBD
  if(!($iden = mysqli_connect("localhost", "wwwdata", "")))
    die("Error: No se pudo conectar");

  // Selecciona la base de datos
  if(!mysqli_select_db($iden, "biblioteca"))
    die("Error: No existe la base de datos");

  // Sentencia SQL: inserta un nuevo libro
  $sentencia = "INSERT INTO libros VALUES (null, '$titulo', '$resumen', $autor, ";
  $sentencia .= "$categoria, $editorial, $anyo)";
  // Ejecuta la sentencia SQL
  if(!mysqli_query($iden, $sentencia))
    die("Error: no se pudo realizar la inserción");

  echo 'Se ha insertado un nuevo libro en la base de datos. ';
  echo '¿Filas insertadas? ' . mysqli_affected_rows($sentencia);

  // Cierra la conexión con la base de datos
  mysqli_close($iden);
?>
</p>
</body>
</html>
```

Nota: el ejemplo anterior no está mal, pero no es la mejor forma de combinar en una página web el código HTML y el código PHP de acceso a una base de datos. Lo correcto es emplear el patrón de arquitectura de software modelo-vista-controlador[17] (MVC). Si lo conoces y sabes cómo usarlo, lo puedes emplear en la práctica.

Después de cada operación con la base de datos hay que comprobar si se ha producido un error. Para ello, se puede comprobar el valor devuelto por las funciones de acceso a MySQL, ya que devuelven `false` si la operación ha fallado. Ante una situación de error, al usuario se le puede mostrar un mensaje de error propio o se puede emplear la función `mysqli_error()` que devuelve una descripción del último error producido.

La función `mysqli_insert_id()` te puede ser muy útil en ciertas situaciones. En la documentación oficial de PHP[18] se puede leer sobre esta función:

> La función `mysqli_insert_id()` devuelve el ID generado por una query (normalmente INSERT) en una tabla con una columna que tenga el atributo AUTO_INCREMENT. Si no se enviaron declaraciones INSERT o UPDATE a través de esta conexión, o si la tabla modificada no tiene una columna con el atributo AUTO_INCREMENT, esta función devolverá cero.

4.3.2. Sentencia preparada

El ejemplo anterior no está mal, pero presenta un agujero de seguridad, ya que no comprueba que en los datos introducidos pueda haber valores incorrectos que produzcan que la sentencia SQL sea errónea. Para evitarlo, se tendrían que escapar los caracteres de entrada, lo cual puede ser tedioso.

Una forma más cómoda y mucho más segura es emplear las sentencias preparadas. Una sentencia preparada (*prepared statement*), también llamada a veces sentencia parametrizada, se usa para ejecutar la misma sentencia repetidamente con gran eficiencia. La ejecución de una sentencia preparada consta de dos partes:

[17] `https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador`
[18] `https://www.php.net/manual/es/mysqli.insert-id.php`

1. La preparación: se define una plantilla con marcadores (parámetros) que se envía al servidor para que realice una comprobación de su sintaxis e inicialice los recursos necesarios para su posterior ejecución.

2. La ejecución: se vinculan los valores a los parámetros de la sentencia y se envía al servidor para su ejecución.

mysqli emplea marcadores de posición anónimos (`?`) para la sustitución de los valores. Otras tecnologías como PDO ofrecen los marcadores de posición con nombre, pero por ahora en mysqli no existen.

Los marcadores de posición se sustituyen por su valor real mediante el método `mysqli_bind_param()`. El segundo parámetro de esta función es una cadena que contiene uno o más caracteres que especifican los tipos para el correspondiente enlazado de las variables:

- `i`: la variable correspondiente es de tipo entero
- `d`: la variable correspondiente es de tipo double
- `s`: la variable correspondiente es de tipo string (una fecha o una hora se tratan como una cadena)
- `b`: la variable correspondiente es un blob y se envía en paquetes

Por supuesto, el número de variables y la longitud de la cadena de tipos debe coincidir con los parámetros en la sentencia.

El siguiente ejemplo muestra cómo se utilizan las sentencias preparadas con marcadores anónimos:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de INSERT y MySQL</title>
</head>
<body>
<p>
<?php
  // Recoge los datos del formulario
  $titulo = $_POST['titulo']; // text
  $resumen = $_POST['resumen']; // text
  $autor = $_POST['autor']; // int
  $categoria = $_POST['categoria']; // int
  $editorial = $_POST['editorial']; // int
  $anyo = $_POST['anyo']; // int

  // Se conecta al SGBD
  if(!($iden = mysqli_connect("localhost", "wwwdata", "")))
    die("Error: No se pudo conectar");

  // Selecciona la base de datos
  if(!mysqli_select_db($iden, "biblioteca"))
    die("Error: No existe la base de datos");

  // Sentencia SQL: inserta un nuevo libro
  $sentencia = mysqli_prepare($iden, "INSERT INTO libros VALUES (null, ?, ?, ?, ?, ?, ?)");

  // Asocia los parámetros
  mysqli_stmt_bind_param($sentencia, 'ssiiii', $titulo, $resumen, $autor, $categoria, $editorial
      , $anyo);

  // Ejecuta la sentencia SQL
  if(!mysqli_stmt_execute($sentencia))
    die("Error: no se pudo realizar la inserción");

  echo 'Se ha insertado un nuevo libro en la base de datos. ';
  echo '¿Filas insertadas? ' . mysqli_stmt_affected_rows($sentencia);
```

```
  // Cierra la sentencia preparada
  mysqli_stmt_close($sentencia);

  // Cierra la conexión con la base de datos
  mysqli_close($iden);
?>
</p>
</body>
</html>
```

El uso de las sentencias preparadas ofrece varias ventajas, pero dos ventajas muy importantes son:

1. Ofrece un aumento de velocidad en la ejecución de una sentencia cuando se ejecuta varias veces: la sentencia no se tiene que analizar e interpretar cada vez, sino que está lista ("está preparada") para ejecutarse tantas veces como se quiera.
2. Ofrece protección frente a la inyección SQL.

Nota: en los ejemplos anteriores se ha empleado la interfaz procedural, pero también es posible utilizar la interfaz orientada a objetos.

4.3.3. Borrado en cascada

Si se usa el motor InnoDB en MySQL y se definen correctamente las claves ajenas, entonces se puede hacer uso del borrado en cascada. En "13.1.20.5 FOREIGN KEY Constraints"[19] de la documentación oficial de MySQL se explica:

> MySQL supports foreign keys, which permit cross-referencing related data across tables, and foreign key constraints, which help keep the related data consistent.
>
> A foreign key relationship involves a parent table that holds the initial column values, and a child table with column values that reference the parent column values. A foreign key constraint is defined on the child table.
>
> [...]
>
> When an UPDATE or DELETE operation affects a key value in the parent table that has matching rows in the child table, the result depends on the referential action specified by ON UPDATE and ON DELETE subclauses of the FOREIGN KEY clause. Referential actions include:
>
> CASCADE: Delete or update the row from the parent table and automatically delete or update the matching rows in the child table. Both ON DELETE CASCADE and ON UPDATE CASCADE are supported. Between two tables, do not define several ON UPDATE CASCADE clauses that act on the same column in the parent table or in the child table.

Por ejemplo, con el borrado en cascada, cuando se elimine un usuario, automáticamente:

1. Se eliminarán todos sus álbumes, que automáticamente...
2. ...forzará a que se eliminen todas las fotos de los álbumes del usuario.

Si no se hace uso del motor InnoDB o no se han definido las claves ajenas correctamente, los borrados se tendrán que hacer manualmente.

4.3.4. Borrado lógico

Aunque no se pide que lo implementes en el proyecto de esta asignatura, es importante que conozcas el concepto de borrado lógico, que en inglés se suele conocer como *soft delete*. Este tipo de borrado permite que la información no desaparezca realmente de la base de datos y así se pueda recuperar en el futuro si es necesario.

El borrado lógico consiste en usar en cada tabla un campo adicional, por ejemplo `es_borrado`, de tipo centinela (*flag*) para señalar que la fila ha sido borrada. Cuando se quiera borrar una fila, no se

[19] `https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html`

emplea la sentencia DELETE, sino la sentencia UPDATE para cambiar el valor del *flag* `es_borrado`. Además, también podría existir un campo adicional de tipo fecha para conservar la fecha en la que la fila fue borrada.

Por supuesto, el uso del *flag* implica que se deba tener en cuenta en todas las consultas que se vayan a realizar. Por ejemplo, si se desea contar cuántos registros existen en una tabla, se tendrá que contar todos los que no tengan en `es_borrado` el valor cierto.

En la implementación más sencilla el valor del *flag* `es_borrado` puede ser de tipo booleano, para indicar "no borrado" (*false*) o "borrado" (*true*). Sin embargo, también se pueden realizar implementaciones más complejas, por ejemplo, cuando existan diferentes tipos de usuarios que puedan consultar diferentes contenidos del sistema de información. En este caso se podría implementar un conjunto de posibles valores para `es_borrado`:

0. No borrado.
1. Borrado lógico, todos los usuarios pueden ver que está borrado.
2. Borrado lógico, solo los usuarios con perfil administrador lo pueden ver.
3. Borrado lógico, solo los desarrolladores lo pueden ver.

5. Recomendaciones

En una aplicación web se deben aplicar muchas medidas de seguridad porque los usuarios suelen ser "despistados" y se pueden dejar su sesión abierta en ordenadores ajenos. Por ello, en aquellas operaciones donde un mal uso pueda implicar que el usuario legítimo pierda su derecho de acceso (eliminar la cuenta, cambiar la contraseña, cambiar el correo electrónico de recuperación y otros datos personales) se debe incluir algún sistema adicional de seguridad. Por ejemplo, para cambiar la contraseña se suele pedir la contraseña antigua al usuario para impedir que sea modificada por otra persona.

Recuerda que las páginas que contengan código PHP tienen que tener la extensión `.php`. Si modificas alguna página web que ya tengas hecha de prácticas anteriores para añadirle código PHP, tendrás que cambiarle la extensión y corregir todos los enlaces que apunten a esa página.

Recuerda que para que el código PHP se ejecute, la página web tiene que pasar por el servidor web para que éste se la pase al intérprete de PHP. Para ello, tienes que cargar la página a través del servidor web: la URL de conexión tiene que tener la forma `http://localhost`. Si en la barra de direcciones del navegador ves escrito algo del estilo `file:///D:/...`, el código PHP no se ejecutará.

El manual de PHP te lo puedes descargar en diferentes formatos de su sitio web[20] para tenerlo siempre a mano y poder hacer las búsquedas de información rápidamente. También puedes acceder a través de Internet a la ayuda de cualquier función de PHP escribiendo el nombre de la función a continuación de la URL `http://php.net/`. Por ejemplo, `http://php.net/header` muestra la ayuda de la función `header()`.

El manual de MySQL te lo puedes descargar en diferentes formatos de su sitio web[21] para tenerlo siempre a mano y poder hacer las búsquedas de información rápidamente.

No filtrar correctamente los datos de entrada es uno de los principales problemas de seguridad de las aplicaciones. Todos los datos que provengan de una fuente externa tienen que ser validados. En una aplicación web los datos pueden provenir principalmente de:

- Datos de un formulario.
- Cookies.
- Variables de entorno.
- Resultado de una llamada a un servicio web.
- Resultado de una consulta a una base de datos.
- Resultado de la ejecución de un programa o comando en el mismo servidor.
- Ficheros.

[20] `http://www.php.net/download-docs.php`
[21] `http://dev.mysql.com/doc/`

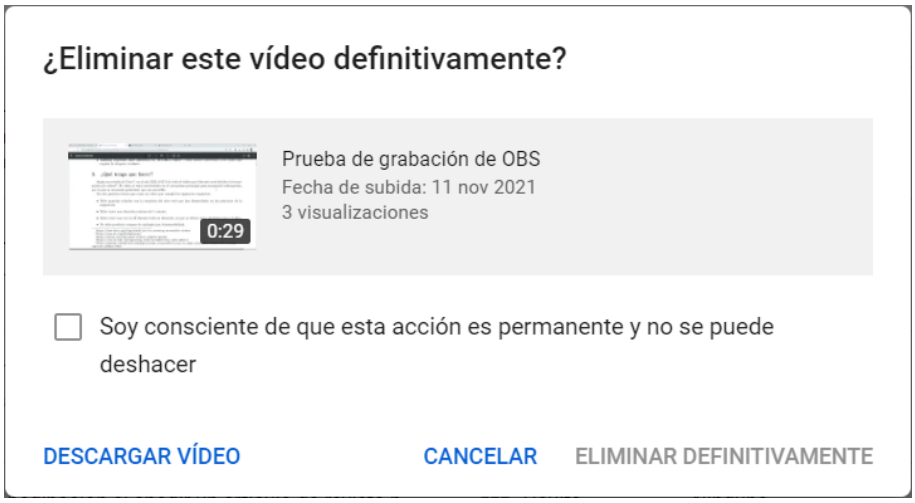

Figura 3: Confirmación de eliminación en YouTube

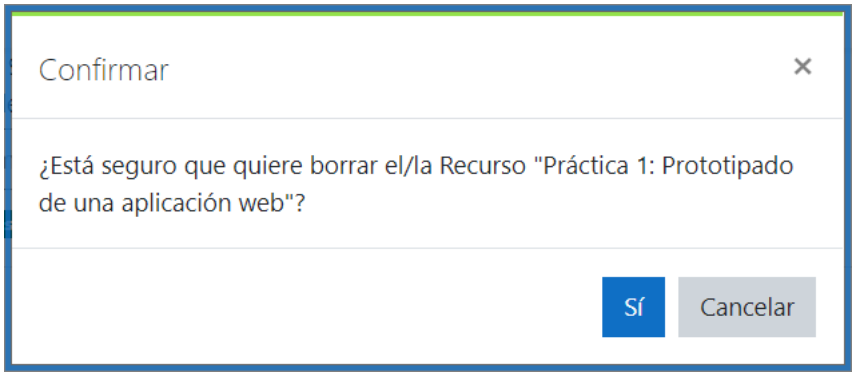

Figura 4: Confirmación de eliminación en Moodle

Las expresiones regulares son un mecanismo de programación muy potente, pero esa potencia origina que sean complejas de utilizar. Comienza a trabajar con expresiones regulares sencillas y poco a poco intenta escribir expresiones más complejas.

Es recomendable solicitar siempre una confirmación del usuario en aquellas operaciones que puedan modificar o eliminar los datos ya existentes en la base de datos. De este modo se evitará la pérdida accidental de información. Por ejemplo, en la Figura 3 se puede ver la confirmación que muestra YouTube cuando se desea eliminar un vídeo. Y en la Figura 4 se puede ver la confirmación que muestra Moodle cuando se desea eliminar un recurso de un curso.

Recuerda que algunas funciones de PHP muestran directamente mensajes de advertencia o de error cuando se produce alguna situación errónea. Para evitar que suceda esto puedes emplear el operador de control de errores “`@`”.