

## Bloque C

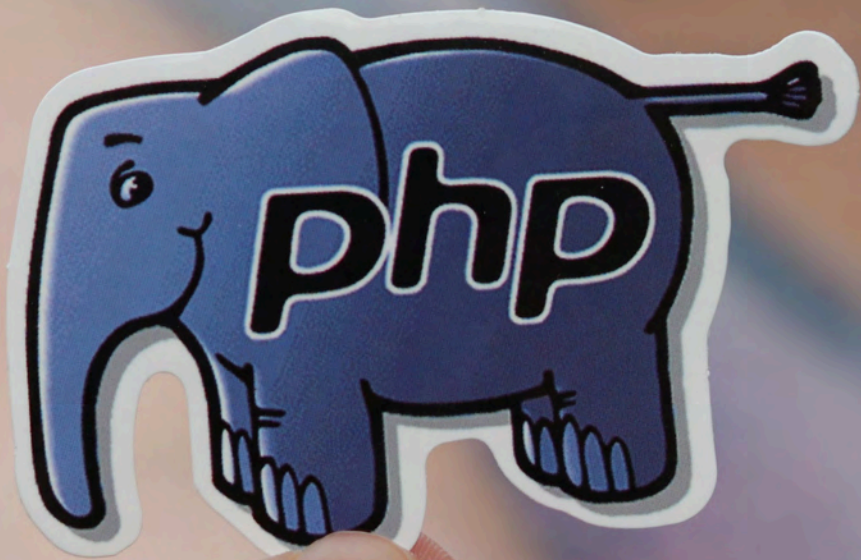
### Sitios web basados en bases de datos

#### C13. Actualizar datos en una base de datos



# Contenidos

1. Introducción
2. Añadiendo datos a una tabla
3. Actualizando datos en una tabla
4. Borrando datos de una tabla
5. Otras acciones
6. Páginas para editar datos en el sitio web de ejemplo
7. Resumen
8. Referencias



# 1. Introducción

# Introducción

Un sitio web puede proporcionar herramientas que **permitan a los usuarios añadir nuevos datos a la base de datos, y actualizar o eliminar los datos existentes** que ya almacena.

# Introducción

Para ello, las páginas PHP necesitan realizar las siguientes tareas:

1. **Recoger los datos:** La unidad 6 mostró cómo obtener datos de formularios y URLs.
2. **Validar los datos:** La unidad 6 también mostró cómo comprobar si se han suministrado los datos requeridos y si los datos están en un formato válido (luego mostrar mensajes a los usuarios si hay errores).
3. **Actualizar la base de datos:** La unidad 11 introdujo las sentencias SQL que crean, actualizan o eliminan datos de la base de datos. La unidad 12 mostró cómo se pueden ejecutar sentencias SQL utilizando PDO.
4. **Proporcionar retroalimentación:** Un mensaje indicará a los usuarios si han tenido éxito o no.

# Introducción

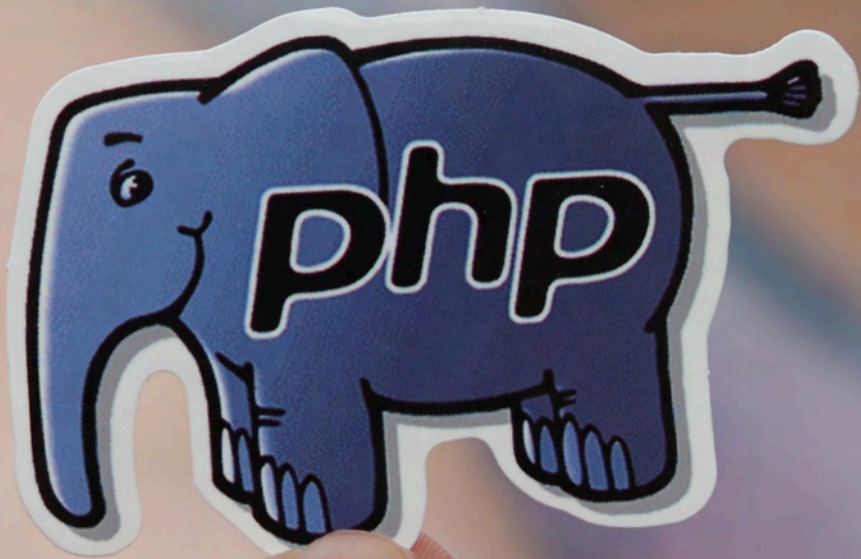
Dado que ya hemos aprendido a realizar la mayoría de estas tareas, esta unidad se centra en cómo controlar cuándo debe ejecutarse cada una de ellas. El orden en que se ejecutan las sentencias se conoce como **flujo de control**.

En esta unidad, una serie de sentencias if le indican al intérprete de PHP cuándo realizar cada tarea. Por ejemplo, si los datos que ha proporcionado un usuario no son válidos, no tiene sentido crear o ejecutar el SQL para actualizar la base de datos. Del mismo modo, sólo es necesario mostrar un mensaje de éxito si los cambios en la base de datos se han realizado correctamente.

# Introducción

También aprenderás cómo ejecutar una serie de sentencias SQL relacionadas utilizando algo llamado **transacciones**, y cómo los cambios sólo se guardan si todas las sentencias SQL tienen éxito (si sólo una de ellas falla, entonces no se guarda ninguno de los cambios en la base de datos).





## 2. Añadiendo datos a una tabla



# Añadiendo datos a una tabla

Para **añadir una nueva fila a una tabla de una base de datos**, utiliza el comando `INSERT` de SQL. Un comando `INSERT` sólo puede añadir datos a una tabla cada vez.

# Añadiendo datos a una tabla

1. La siguiente sentencia SQL **añade una categoría a la tabla de categorías**. Tiene parámetros para las columnas nombre, descripción y navegación. (El valor para la columna id es generado por la base de datos).
2. Los datos que utilizará cada columna se proporcionan en un **array asociativo** con exactamente un elemento por cada parámetro de la sentencia SQL. El array no debe contener elementos adicionales, ya que esto provocaría un error.
3. El método `prepare()` del objeto PDO necesita una sentencia SQL como argumento para poder crear un objeto `PDOStatement`. Luego, el método `execute()` del objeto `PDOStatement` utiliza valores del array para ejecutar el SQL.

# Añadiendo datos a una tabla

```
① [ $sql = "INSERT INTO category (name, description, navigation)
    VALUES (:name, :description, :navigation);" ;

② [ $category = ['name']           = 'News' ;
    $category = ['description']    = 'News about Creative Folk' ;
    $category = ['navigation']     = 1 ;

③ [ $statement = $pdo->prepare($sql) ;
    $statement->execute($category) ;
```

# Añadiendo datos a una tabla

En la siguiente imagen se ha resaltado la nueva fila. La función de autoincremento da a la columna id un valor de 5. En el sitio web de ejemplo, si hubiera un problema, el objeto PDO lanzaría una excepción y esto sería manejado por la función de manejo de excepciones por defecto.

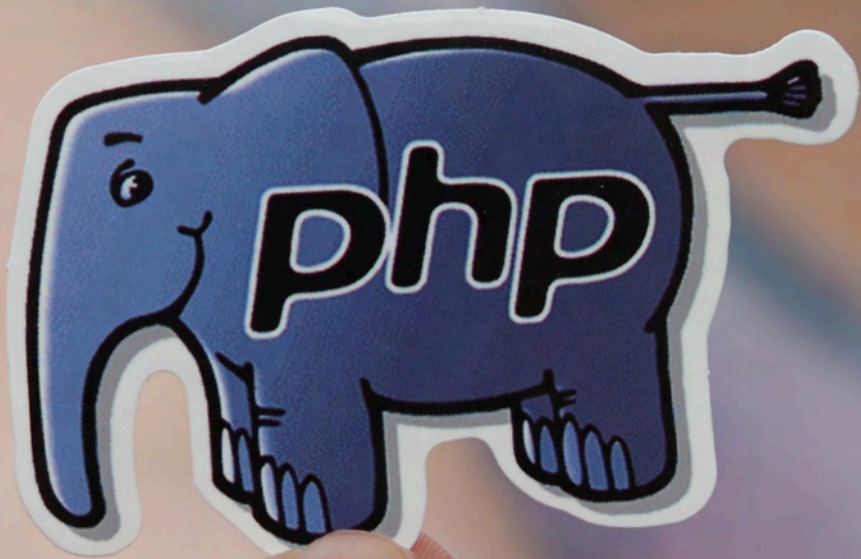
category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	News	News about Creative Folk	1



## Actividad C13.1. Añadiendo datos a una tabla

Crea un script en PHP que permita añadir nuevas filas a la tabla "tbl\_personal" que creaste en la unidad anterior.





### 3. Actualizando datos en una tabla



# Actualizando datos en una tabla

Para **actualizar filas existentes en una tabla de base de datos**, se utiliza el comando `UPDATE` de SQL. Un comando `UPDATE` puede actualizar varias tablas utilizando un `JOIN` .

# Actualizando datos en una tabla

1. Esta sentencia SQL **actualiza una categoría existente**. Los tres primeros parámetros contienen valores para utilizar en las columnas nombre, descripción y navegación. La cláusula `WHERE` utiliza un parámetro para **especificar el id de una fila a actualizar**.
2. Los datos utilizados para sustituir los parámetros **se proporcionan en un array**, que tiene exactamente un elemento por cada parámetro de la sentencia SQL. No debe contener elementos adicionales porque esto provocaría un error.
3. La sentencia se ejecuta como una consulta con parámetros. A continuación, puedes ver que se utiliza la función `pdo()` definida por el usuario (introducida en la unidad anterior) para ejecutar la sentencia SQL. Este enfoque se utilizará en el resto de la unidad.

# Actualizando datos en una tabla

```
① $sql = "UPDATE category
    SET name      = :name,
        description = :description,
        navigation = :navigation
    WHERE id = :id;";

② $category = ['id']           = 5;
   $category = ['name']        = 'News';
   $category = ['description'] = 'Updates from Creative Folk';
   $category = ['navigation']  = 0;

③ pdo($pdo, $sql, $category);
```

# Actualizando datos en una tabla

A continuación se puede ver cómo se ha actualizado la quinta categoría.

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	News	Updates from Creative Folk	0

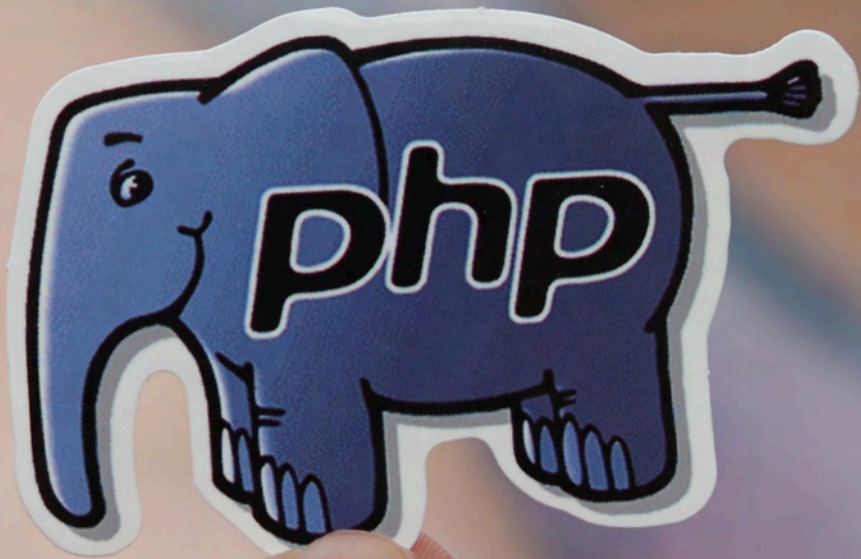
NOTA: Si la condición de búsqueda especificada en la cláusula `WHERE` coincide con más de una fila, el comando SQL `UPDATE` actualizará todas las filas coincidentes.



## Actividad C13.2. Actualizar datos de una tabla

Crea un script en PHP que permita actualizar la edad de una fila concreta de la tabla "tbl\_personal".





## 4. Borrando datos de una tabla



# Borrando datos de una tabla

El comando `DELETE` de SQL **elimina filas de datos de una tabla**. Una condición de búsqueda **restringe las filas que deben eliminarse**. Se puede utilizar un `JOIN` para eliminar datos de varias tablas.

# Borrando datos de una tabla

1. La sentencia SQL que se muestra a continuación utiliza el comando `DELETE`, seguido de la cláusula `FROM` y el nombre de la tabla de la que se eliminarán las filas.
2. A continuación, la condición de búsqueda especifica qué filas deben eliminarse de esta tabla. A continuación, la fila a eliminar se especifica utilizando un valor en la columna id.
3. El id de la fila a borrar se almacena en \$id. Luego el id se proporciona a la función `pdo()` utilizando un array indexado que se crea en el propio argumento.

## Borrando datos de una tabla

```
① $sql = "DELETE FROM category  
②     WHERE id = :id;";  
  
③ [ $id = 5;  
    pdo($pdo, $sql, [$id]);
```

# Borrando datos de una tabla

A continuación se puede comprobar que se ha eliminado la quinta categoría de la tabla.

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1

NOTA: si la condición de búsqueda de la cláusula **WHERE** coincide con más de una fila, el comando **DELETE** de SQL eliminará todas las filas coincidentes.



## Actividad C13.3. Borrar datos de una tabla

Crea un script en PHP que permita borrar una fila concreta de la tabla "tbl\_personal".





## 5. Otras acciones



# Obtener el id de una nueva fila insertada

Cuando una columna de una tabla de base de datos utiliza un id auto-incrementado y se añade una nueva fila de datos a la tabla, el método `lastInsertId()` del objeto PDO devolverá el id que la base de datos creó para la nueva fila.

# Obtener el id de una nueva fila insertada

En el sitio web de ejemplo, la primera columna de cada tabla se llama `id`. Contiene una **clave primaria que se utiliza para identificar de forma única cada fila de la tabla**.

Cuando se añade una nueva fila a la tabla, se utiliza la función de autoincremento de MySQL para crear el valor utilizado en la columna `id` de la nueva fila.

Cuando se ha ejecutado una sentencia SQL que inserta una nueva fila de datos en la tabla, el método `lastInsertId()` del objeto PDO puede **obtener el valor que MySQL generó para la columna id**. Esto puede ser almacenado en una variable y utilizado posteriormente en el código.

# Obtener el id de una nueva fila insertada

Verás que se utiliza esta técnica **cuando se crea un nuevo artículo y se sube una imagen al mismo tiempo.**

- La imagen se añade primero a la tabla de imágenes
- Se utiliza el método `lastInsertId()` para obtener su id
- El artículo se añade a la tabla de artículos en último lugar porque utiliza el id de la nueva imagen en la columna id de imagen de la tabla de artículos

Abajo puedes ver que el método `lastInsertId()` es llamado después de que la función `pdo()` ha sido llamada para añadir una nueva fila a la base de datos.

```
pdo($pdo, $sql, $arguments);  
$new_id = $pdo->lastInsertId();
```

# Averiguando cuantas filas han cambiado

Cuando una sentencia SQL utiliza un comando `UPDATE` o `DELETE`, puede cambiar más de una fila de datos a la vez. El método `rowCount()` del objeto `PDOStatement` **devuelve el número de filas afectadas**.

# Averiguando cuantas filas han cambiado

Cuando se ejecuta un comando `UPDATE` o `DELETE`, puede cambiar cero, una o muchas filas de la base de datos, dependiendo de cuántas filas coincidan con la condición de búsqueda en la cláusula `WHERE`.

Cuando se ejecuta la consulta de abajo, si la tabla de categorías no tiene una categoría con un id de 100, no se borrará nada de la base de datos:

```
DELETE FROM category  
WHERE id = 100;
```

# Averiguando cuantas filas han cambiado

Sin embargo, cuando se ejecuta la consulta siguiente, podría actualizar cero, una o muchas filas, dependiendo de cuántas filas tuvieran un valor 0 en la columna de navegación.

```
UPDATE category  
  SET navigation = 1  
 WHERE navigation = 0;
```



# Averiguando cuantas filas han cambiado

Cuando se llama al método `execute()` del objeto `PDOStatement`, devuelve true si se ejecuta una sentencia SQL y false si no se ejecuta. Pero, como muestran estos ejemplos, esto no indica si ha cambiado alguna fila de la base de datos.

Para determinar cuántas filas cambiaron cuando se ejecutó una sentencia SQL, el objeto `PDOStatement` tiene un método llamado `rowCount()` que **devolverá el número de filas que cambiaron**.

# Averiguando cuantas filas han cambiado

El método `rowCount()` debe ser llamado en la siguiente sentencia después del método `execute()`, y el valor que devuelve puede ser almacenado en una variable.

Si ejecutas una sentencia SQL utilizando la función `pdo()` definida en la última unidad, el método `rowCount()` puede ser llamado en la misma sentencia que llama a la función `pdo()` (utilizando el encadenamiento de métodos) como se puede ver en el siguiente ejemplo.

```
$sql = "UPDATE category  
      SET navigation = 1  
      WHERE navigation = 0;";  
$result = $pdo($pdo, $sql)->rowCount();
```

# Evitando valores duplicados en las columnas

Los valores de algunas columnas deben ser **únicos**. En el sitio web de ejemplo, dos artículos **no pueden tener el mismo título**, dos categorías **no pueden tener el mismo nombre** y dos miembros **no pueden tener la misma dirección de correo electrónico**.

# Evitando valores duplicados en las columnas

En la unidad 11, se añadió una **restricción de unicidad** a las siguientes columnas de la base de datos para garantizar que no hubiera dos filas con el mismo valor en estas columnas, la:

- columna `title` de la tabla de artículos
- columna `forename` de la tabla categoría
- columna `email` de la tabla de miembros

Cuando se añade una nueva fila a estas tablas (o se actualiza una fila existente), si otra fila ya tiene el mismo valor en estas columnas, **PDO lanzará un objeto de excepción** porque no puede guardar los datos (rompería la restricción de unicidad).

# Evitando valores duplicados en las columnas

PDO utiliza la clase `PDOException` para crear un objeto `PDOException`; es como los objetos de excepción que conociste en la unidad 10, pero contiene datos extra que son específicos de PDO. A continuación, se muestra **cómo manejar sentencias SQL que pueden romper una restricción de unicidad**.



# Evitando valores duplicados en las columnas

1. El código para crear una nueva fila en estas tablas, o actualizar una fila existente de ellas, se pone en un bloque try.
2. Si PDO lanza una excepción al ejecutar código en el bloque try, se ejecuta el bloque catch subsiguiente, y el objeto de excepción se almacena en una variable llamada `$e`.

## Evitando valores duplicados en las columnas

3. El objeto `PDOException` tiene una propiedad llamada `errorInfo`. Su valor es un array indexado de datos sobre el error. El segundo elemento del array es un código de error (En el siguiente enlace <https://mariadb.com/kb/en/mariadb-error-code-reference/> encontraréis una lista completa de códigos de error). Si el código de error es `1062`, **una restricción de unicidad está impidiendo que los datos sean guardados**, y el usuario debe ser informado de que el valor ya ha sido utilizado.

# Evitando valores duplicados en las columnas

4. Si es cualquier otro código de error, la excepción se vuelve a lanzar utilizando la palabra clave throw (ver Unidad 10) y será manejada por la función de manejo de excepciones por defecto.

```
① try {  
    pdo($pdo, $sql, $args);  
② } catch (PDOException $e) {  
    ③ if ($e->errorInfo[1] === 1062) {  
        // Tell user a value has already been used  
    } else {  
        ④ throw $e;  
    }  
}
```



## Actividad C13.4. Evitar duplicados

Modifica la actividad C13.1 para que no se permitan nombres duplicados. Tendrás que añadir la restricción de unicidad a la tabla mediante phpmyadmin.





## 6. Páginas para editar datos en el sitio web de ejemplo

# Crear páginas para editar datos de bases de datos

Habiendo visto cómo PDO añade, actualiza y borra datos en la base de datos, el resto de esta unidad te mostrará **cómo crear páginas de administración y formularios que permitan a los usuarios cambiar los datos almacenados en una base de datos.**

Estas son las seis páginas de administración que permiten a los usuarios crear, actualizar y eliminar las categorías y los artículos.

Las páginas de administración están todas en una carpeta llamada *admin*.

# Crear páginas para editar datos de bases de datos

## categories.php

Esta página enumera todas las categorías. También hay enlaces para crear, actualizar y eliminar categorías.

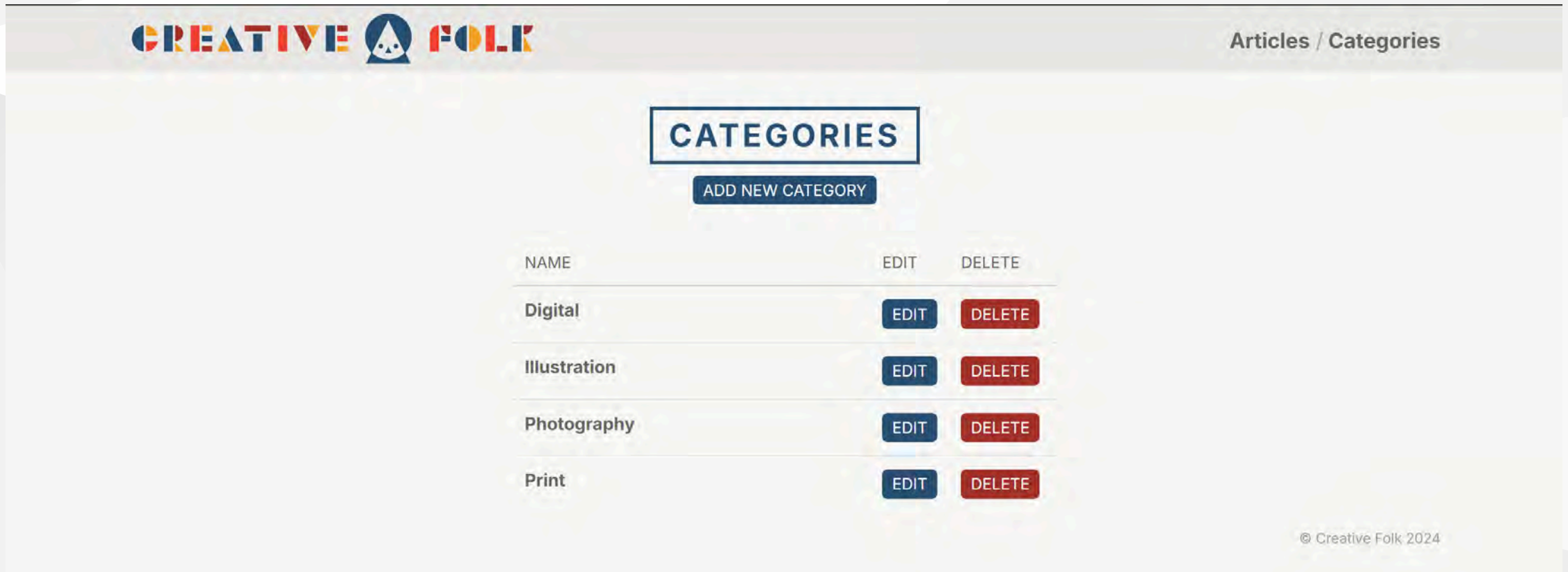
Los enlaces para crear o editar las categorías apuntan a `category.php`.

- Al crear una categoría, no hay cadena de consulta.
- Al editar una categoría, la cadena de consulta contiene el nombre id y su valor es el id de la categoría a editar. Por ejemplo, `category.php?id=2`

Los enlaces para eliminar una categoría apuntan a una página llamada `category-delete.php`, y la cadena de consulta contiene el id de la categoría a eliminar.

# Crear páginas para editar datos de bases de datos

categories.php





# Crear páginas para editar datos de bases de datos

## articles.php

Esta página enumera todos los artículos. Tiene enlaces que permiten a los propietarios del sitio crear, actualizar y eliminar artículos.

Los enlaces para crear o editar los artículos apuntan a `article.php`.

- Al crear un artículo, no hay cadena de consulta.
- Al editar un artículo, la cadena de consulta contiene el nombre `id` y su valor es el id del artículo que se va a editar. Por ejemplo, `article.php?id=2`

Los enlaces para eliminar un artículo apuntan a una página llamada `article-delete.php`, y la cadena de consulta contiene el id del artículo a eliminar.

# Crear páginas para editar datos de bases de datos



articles.php

CREATIVE FOLK

Articles / Categories

ARTICLES

ADD NEW ARTICLE

IMAGE	TITLE	CREATED	PUBLISHED	EDIT	DELETE
	Travel Guide	April 25, 2021	Yes	EDIT	DELETE
	Golden Brown	April 25, 2021	Yes	EDIT	DELETE

# Crear páginas para editar datos de bases de datos

## category.php

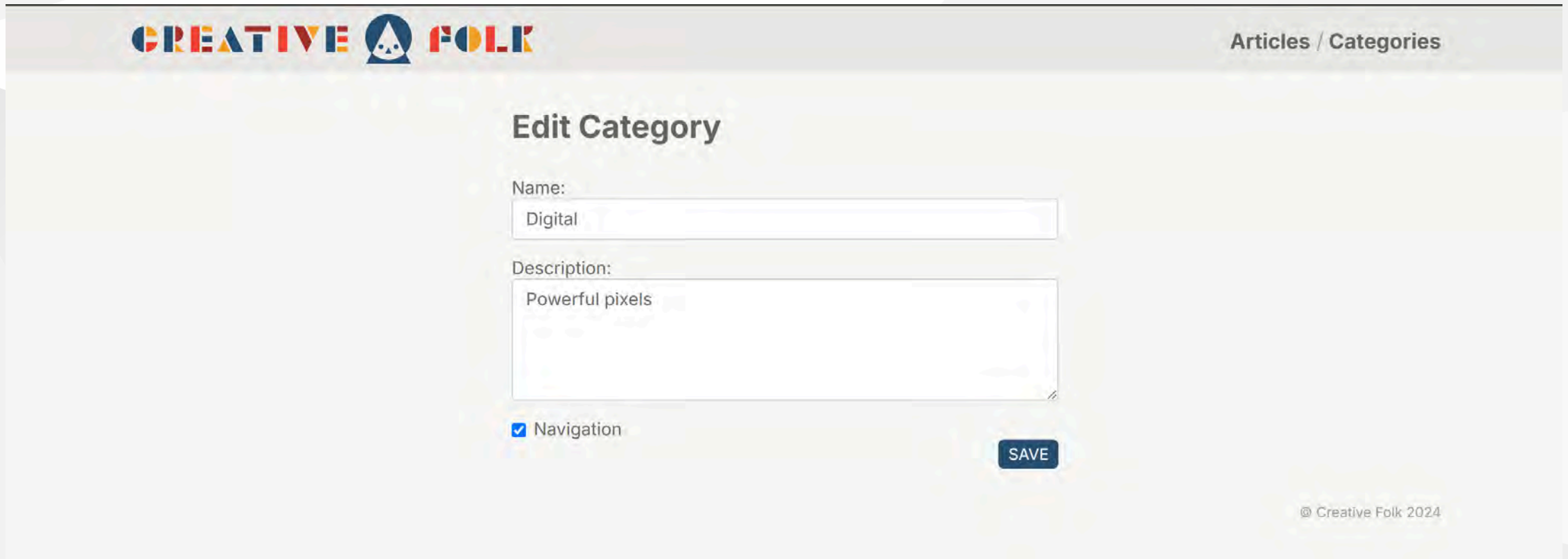
Esta página proporciona un formulario para crear una nueva categoría o actualizar una existente.

Cuando se envía el formulario, se validan los datos:

- **Si son válidos, la página actualizará la base de datos y devolverá al usuario a la página `categories.php`.** Se envía un mensaje en la cadena de consulta para que la página de categorías pueda mostrar que los datos se guardaron.
- **Si no son válidos, el formulario se muestra de nuevo con mensajes debajo de los campos del formulario que necesitan ser corregidos.**

# Crear páginas para editar datos de bases de datos

category.php



The screenshot shows a web interface for editing a category. At the top, there is a header with the 'CREATIVE FOLK' logo on the left and 'Articles / Categories' on the right. The main content area is titled 'Edit Category'. It contains two text input fields: 'Name:' with the value 'Digital' and 'Description:' with the value 'Powerful pixels'. Below these fields is a checked checkbox labeled 'Navigation'. A 'SAVE' button is located at the bottom right of the form area. In the bottom right corner of the page, there is a small copyright notice: '@ Creative Folk 2024'.

CREATIVE FOLK

Articles / Categories

## Edit Category

Name:  
Digital

Description:  
Powerful pixels

☒ Navigation

SAVE

@ Creative Folk 2024



# Crear páginas para editar datos de bases de datos

## article.php


La página de artículos ofrece un formulario para crear un nuevo artículo o actualizar uno existente.

Cuando se envía el formulario, se validan los datos.

- **Si son válidos, la página actualizará la base de datos y devolverá al usuario a la página `articles.php`.** Se envía un mensaje en la cadena de consulta para que la página de artículos pueda mostrar que los datos se guardaron.
- **Si no son válidos, el formulario se muestra de nuevo con mensajes debajo de los campos del formulario que necesitan ser corregidos.**

# Crear páginas para editar datos de bases de datos


article.php

CREATIVE  FOLK

Articles / Categories

## Edit Article

Image:



**Alt text:** Two pages from a travel book showing Nijo Castle

EDIT ALT TEXTDELETE IMAGE

Title:

Summary:

Content:

Author:

Category:

☒ Published

# Crear páginas para editar datos de bases de datos

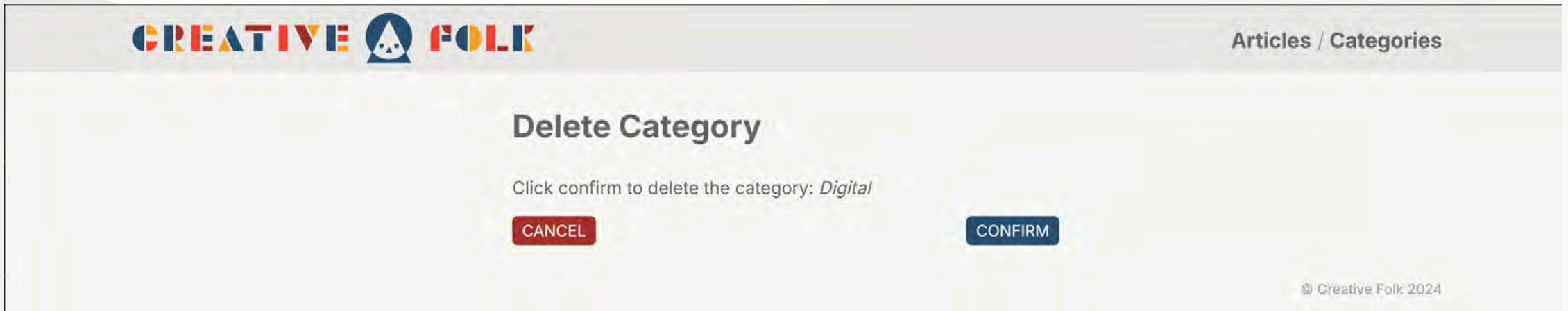
## category-delete.php

Esta página **pide a los usuarios que confirmen que desean eliminar una categoría.**

Si el usuario pulsa el botón confirmar, la categoría será eliminada, y el usuario será enviado de vuelta a la página `categories.php`. Se enviará un mensaje en la cadena de consulta y se mostrará en la página de categorías para informar al usuario de que se ha eliminado la categoría.

# Crear páginas para editar datos de bases de datos

category-delete.php



The screenshot shows a web interface for 'Creative Folk'. At the top, there is a header with the logo 'CREATIVE FOLK' on the left and a navigation link 'Articles / Categories' on the right. The main content area is titled 'Delete Category'. Below the title, a message reads: 'Click confirm to delete the category: *Digital*'. There are two buttons: a red 'CANCEL' button and a dark blue 'CONFIRM' button. At the bottom right of the page, there is a copyright notice: '© Creative Folk 2024'.

# Crear páginas para editar datos de bases de datos

## article-delete.php

Esta página **pide a los usuarios que confirmen que desean eliminar un artículo.**

Si el usuario pulsa el botón de confirmación, el artículo se eliminará y el usuario será devuelto a la página `articles.php`. Se enviará un mensaje en la cadena de consulta y se mostrará en la página de artículos para informar al usuario de que se ha eliminado el artículo.



# Crear páginas para editar datos de bases de datos

article-delete.php



# Ejemplo: Crear, actualizar y eliminar categorías

La página `categories.php` proporciona enlaces para crear, actualizar y eliminar categorías.

# Ejemplo: Crear, actualizar y eliminar categorías

1. Se utilizan tipos estrictos y se incluyen dos archivos: `database-connection.php` crea un objeto PDO y `functions.php` contiene funciones definidas por el usuario, incluyendo la función `pdo()`, funciones para formatear datos y una nueva función que se muestra en los pasos 15-19.
2. Si la cadena de consulta contiene un mensaje de éxito, se almacena en una variable llamada `$sucess`. En caso contrario, `$success` contendrá el valor null.
3. Si la cadena de consulta contiene un mensaje de error, se almacena en `$failure`. En caso contrario, `$failure` tendrá el valor null.

## Ejemplo: Crear, actualizar y eliminar categorías

4. La variable `$sql` contiene la consulta SQL para obtener datos sobre cada una de las categorías almacenadas en la base de datos.
5. La función `pdo()` ejecuta la consulta y luego el método `fetchAll()` del objeto `PDOStatement` obtiene los datos de las categorías y los almacena en `categories`.
6. Se incluye el archivo de cabecera para las páginas de administración.
7. Si la cadena de consulta contenía un mensaje de éxito o fracaso, se muestra en la página.
8. Se añade un enlace para crear una nueva categoría. Cuando un enlace a `category.php` no tiene una cadena de consulta, `category.php` sabe que debe crear una nueva categoría.

## Ejemplo: Crear, actualizar y eliminar categorías

9. Se añade una tabla a la página. La primera fila contiene tres encabezados de columna: nombre, editar y eliminar.
10. Se utiliza un bucle foreach para mostrar datos sobre las categorías existentes con enlaces para editarlas o eliminarlas.
11. La primera columna muestra el nombre de la categoría.



## Ejemplo: Crear, actualizar y eliminar categorías

12. A continuación, se crea un enlace a la página `category.php`. La cadena de consulta contiene el id de la categoría para que, cuando se cargue la página `category.php`, permita al usuario editar los detalles sobre esa categoría. Por ejemplo, `<a href=«category.php?id=1»>`.
13. Se crea un enlace a la página `category-delete.php`, que elimina una categoría de la base de datos. El id de la categoría se almacena en la cadena de consulta.
14. Se incluye el pie de página de las páginas de administración.

## Ejemplo: Crear, actualizar y eliminar categorías

15. Se ha añadido a `functions.php` una nueva función que redirige a los usuarios a otra página. Permite añadir mensajes de éxito o fracaso a la cadena de consulta de la página a la que se envía al usuario. Tiene tres parámetros:

- El nombre del archivo al que se enviará al usuario
- Un array opcional utilizado para crear una cadena de consulta
- Un código de respuesta HTTP opcional (por defecto es 302)

## Ejemplo: Crear, actualizar y eliminar categorías

1. La variable `$qs` se crea para contener una cadena de consulta. su valor se asigna utilizando un operador ternario. Si `$parameters` contiene un array, se añade un signo de interrogación a `$qs` , y entonces la función incorporada de PHP `http_build_query()` crea la cadena de consulta a partir de los valores del array. Para cada elemento del array, la clave se convierte en un nombre en la cadena de consulta, y su valor se añade después de un símbolo igual (y los caracteres que no están permitidos en una URL también se escapan).

## Ejemplo: Crear, actualizar y eliminar categorías

17. El valor en `$qs` se une al final de la URL de la página a la que se envía al usuario.
18. Se llama a la función `header()` de PHP para redirigir al visitante. El primer argumento indica al navegador la página a solicitar; el segundo es el código de respuesta HTTP.
19. `exit` detiene la ejecución de cualquier otro código.

# Ejemplo: Crear, actualizar y eliminar categorías

*cms/admin/categories.php*

```
<?php
① declare(strict_types = 1);           // Use strict types
   include '../includes/database-connection.php'; // Database connection
   include '../includes/functions.php';           // Include functions

② $success = $_GET['success'] ?? null;           // Check for success message
③ $failure = $_GET['failure'] ?? null;           // Check for failure message

④ $sql = "SELECT id, name, navigation FROM category;"; // SQL to get all categories
⑤ $categories = pdo($pdo, $sql)->fetchAll();       // Get all categories
?>

⑥ <?php include '../includes/admin-header.php' ?>
   <main class="container" id="content">
     <section class="header">
       <h1>Categories</h1>
       ⑦ {
         <?php if ($success) { ?><div class="alert alert-success"><?= $success ?></div><?php } ?>
         <?php if ($failure) { ?><div class="alert alert-danger"><?= $failure ?></div><?php } ?>
       }
       ⑧ <p><a href="category.php" class="btn btn-primary">Add new category</a></p>
     </section>
```



# Ejemplo: Crear, actualizar y eliminar categorías

*cms/admin/categories.php*

```

  <table class="categories">
⑨    <tr><th>Name</th><th class="edit">Edit</th><th class="delete">Delete</th></tr>
⑩    <?php foreach ($categories as $category) { ?>
      <tr>
⑪        <td><?= htmlspecialchars($category['name']) ?></td>
⑫        <td><a href="category.php?id=<?= $category['id'] ?>"
⑬        class="btn btn-primary">Edit</a></td>
          <td><a href="category-delete.php?id=<?= $category['id'] ?>"
          class="btn btn-danger">Delete</a></td>
      </tr>
      <?php } ?>
    </table>
  </main>
⑭ <?php include '../includes/admin-footer.php'; ?>

```

# Ejemplo: Crear, actualizar y eliminar categorías

*cms/admin/functions.php*

```
⑮ function redirect(string $location, array $parameters = [], $response_code = 302)
{
⑮     $qs = $parameters ? '?' . http_build_query($parameters) : ''; // Create query string
⑮     $location = $location . $qs; // Create new path
⑮     header('Location: ' . $location, $response_code); // Redirect to new page
⑮     exit; // Stop code
}
```

# Creando y actualizando datos

**El código para crear o actualizar artículos y categorías se divide en cuatro partes.**  
Cada parte utiliza un conjunto de sentencias if que determinan qué código se ejecuta.

# Creando y actualizando datos

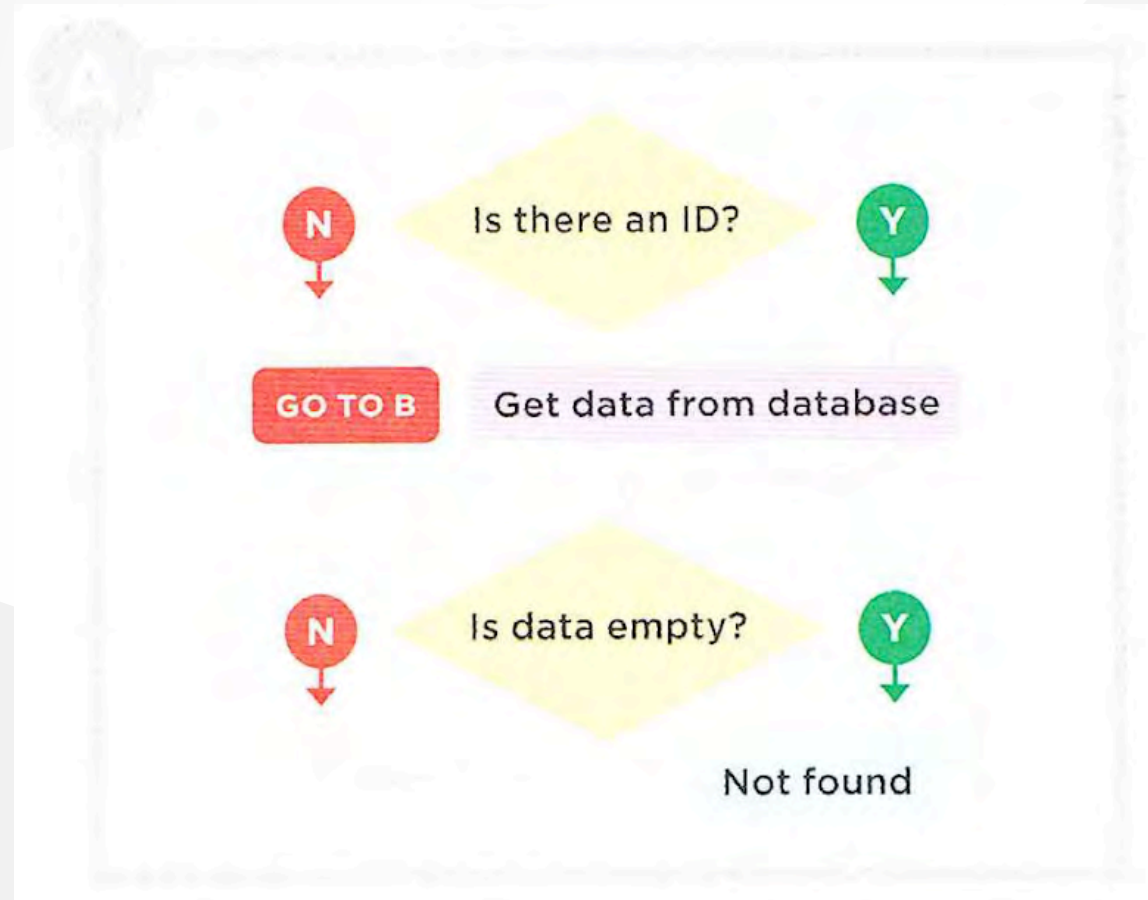
## A: PÁGINA DE CONFIGURACIÓN

En primer lugar, **las páginas comprueban si están creando o actualizando datos**. Para ello, comprueban si la cadena de consulta tiene un nombre llamado `id` con un valor que es un número entero.

- **No:** La página está siendo utilizada para crear una nueva fila en la base de datos. En este punto, el intérprete PHP pasará a la Parte B.
- **Sí:** La página está intentando editar una fila de datos existente y debe cargar esos datos para que puedan ser editados.

Si no se devuelven los datos que el usuario quiere editar, se le dice al usuario que no se encontró el artículo o la categoría.

# Creando y actualizando datos



Los diagramas de flujo ayudan a describir qué código debe ejecutarse en diferentes situaciones. Puedes consultar estos diagramas de flujo cuando trabajes con el

código



# Creando y actualizando datos

## B: OBTENER Y VALIDAR LOS DATOS DEL USUARIO

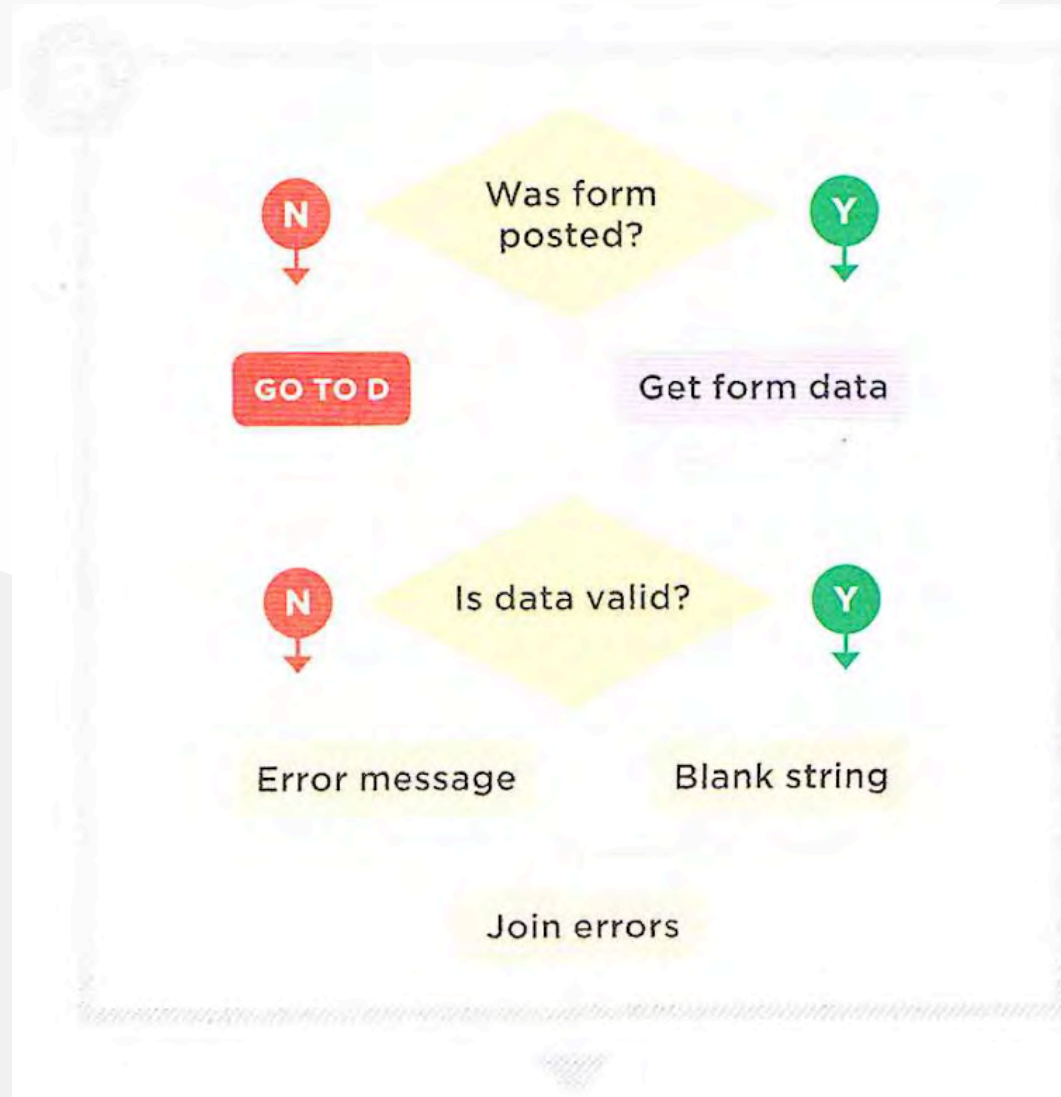
A continuación, la **página comprueba si se ha enviado el formulario.**

- **No:** Pasar a la parte D
- **Sí:** Los datos deben ser recogidos y validados

Se crea un array con un elemento por cada dato que recibe la página. El valor para ese elemento se asigna validando los datos utilizando las funciones que se introdujeron en la unidad 6. Si los datos son

- **Válidos:** ese elemento del array contiene una cadena en blanco
- **No válidos:** el array almacena un mensaje de error indicando qué datos se esperan del control

# Creando y actualizando datos



# Creando y actualizando datos

## C: GUARDAR DATOS DEL USUARIO

En esta parte, **la página comprueba si todos los datos son válidos.**

- **No:** Saltar a la Parte D
- **Sí:** Continúa ejecutando el código de la Parte C

A continuación **comprueba si había un id en la cadena de consulta:**

- **No:** El SQL crea un nuevo artículo o categoría
- **Sí:** El SQL actualiza el artículo o la categoría

# Creando y actualizando datos

## C: GUARDAR DATOS DEL USUARIO

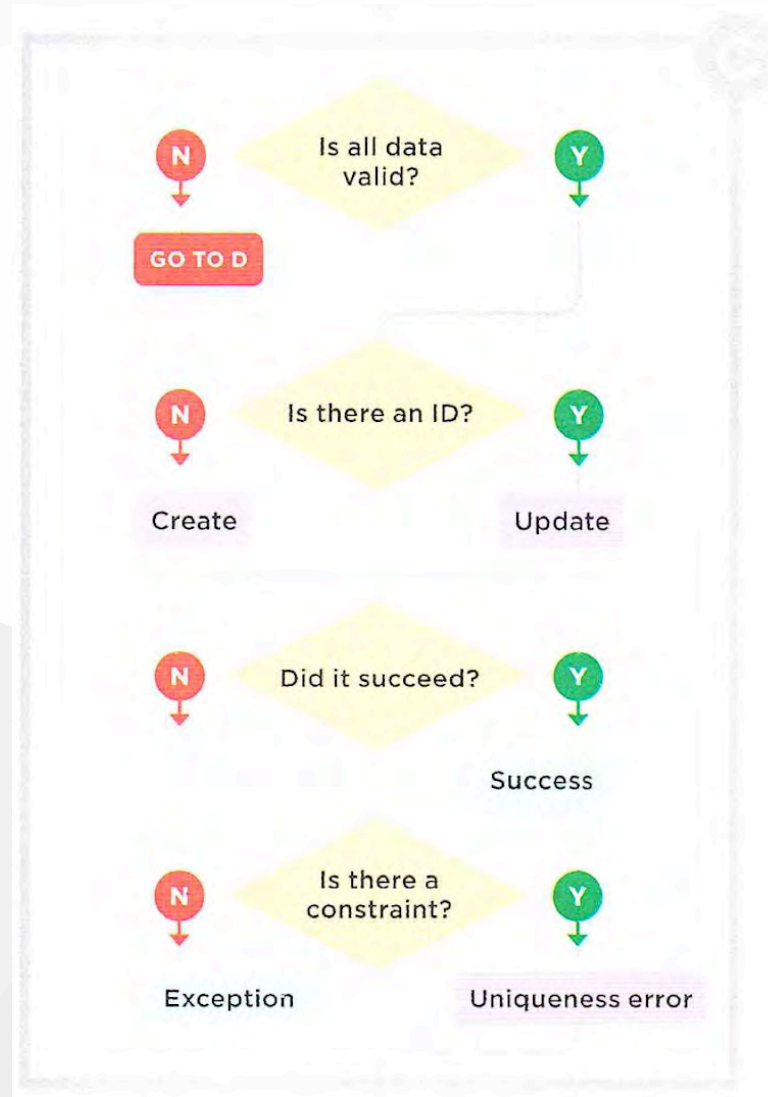
A continuación, comprueba si el SQL se ejecutó correctamente:

- **No:** Comprueba el tipo de excepción lanzada
- **Sí:** Se muestra al usuario un mensaje de éxito

Si se lanzó una excepción, comprueba si fue causada por una restricción de unicidad:

- **No:** se vuelve a lanzar la excepción
- **Sí:** se muestra un mensaje que describe el problema

# Creando y actualizando datos





# Creando y actualizando datos

## D: MOSTRAR FORMULARIO

A continuación se muestra el formulario:

- Si no hay id y el formulario no fue enviado, **el formulario estará en blanco**
- Si hay un identificador pero el formulario no se ha enviado, **el formulario mostrará los datos existentes que deben editarse.**
- Si el formulario fue enviado y los datos no son válidos, **el formulario muestra los datos que el usuario proporcionó con mensajes de error indicando cómo corregirlos.**



## Ejemplo: Obteniendo y validando datos de categoria

El código de `category.php` se detalla en las siguientes diapositivas. Primero, puedes ver el código para las Partes A y B (como se ha descrito anteriormente). **La Parte A configura la página y determina si se está creando una nueva categoría o actualizando una ya existente.**

# Ejemplo: Obteniendo y validando datos de categoria

1. Se habilitan los tipos estrictos y se incluyen los archivos necesarios. Se necesita `database-connection.php`, `functions.php`, y `validate.php` (que contiene las funciones de validación creadas en la unidad 6).
2. Si la página está editando una categoría existente, la URL tendrá una cadena de consulta con el nombre `id`; su valor será el `id` de la categoría a editar.  
La función `filter_input()` de PHP se utiliza para comprobar si un `id` está presente y su valor es un entero. La variable `$id` almacenará
  - el entero si está presente
  - `false` si el valor no es un entero válido
  - `null` si el nombre `id` no está en la cadena de consulta

# Ejemplo: Obteniendo y validando datos de categoría

3. El array `$category` se declara para contener detalles sobre la categoría. Se inicializa con los valores que se pueden mostrar en el formulario de la Parte D (se verá más adelante) cuando se está creando una nueva categoría y el formulario no tiene ningún valor que mostrar.
4. El array `$errors` se inicializa con cadenas en blanco para cada elemento (porque todavía no se ha descubierto ningún error). Los valores de este array se muestran en la Parte D después de cada control de formulario (se verá más adelante).
5. Una sentencia if comprueba si hay un entero válido para el id de categoría en la cadena de consulta.
6. Si es así, el SQL para obtener la categoría que el usuario quiere editar de la base de datos se almacena en `$sql`.

# Ejemplo: Obteniendo y validando datos de categoria

7. La función `pdo()` ejecuta la consulta, y el método `fetch()` recoge los datos de la categoría.

El array devuelto se almacena en la variable `$category` (sobrescribiendo los valores creados en el paso 3).

8. Si había un id en la cadena de consulta, pero la base de datos no encontró una categoría coincidente, la variable `$category` mantendrá false y se ejecutará el bloque de código subsiguiente.

9. La función `redirect()` (vista anteriormente en esta unidad) envía al usuario a la página `categories.php`. El segundo argumento es un array que se utiliza para mostrar un mensaje de error indicando al usuario que no se ha podido encontrar la categoría.



# Ejemplo: Obteniendo y validando datos de categoria

**La parte B recoge y valida los datos del formulario:**

10. Una sentencia if comprueba si el formulario fue enviado.

11. Si es así, los valores del formulario se almacenan en el array \$categoria que se creó en el Paso 4. **Nota:** la opción de navegación (que indica si la categoría debe mostrarse o no en la barra de navegación) sólo se envía al servidor si la casilla está seleccionada. Por lo tanto, se utiliza la función `isset()` de PHP para comprobar si se ha enviado un valor para este control de formulario y, a continuación, un operador de igualdad comprueba si su valor es 1. En caso afirmativo, la opción de navegación mantendrá el valor 1; en caso contrario, mantendrá 0.

## Ejemplo: Obteniendo y validando datos de categoria

12. El nombre y la descripción de la categoría se validan utilizando la función `is_text()` en el archivo include `validate.php` . Si no son válidos, los mensajes de error se almacenan en el array `$errors` .
13. Los valores de la matriz `$errors` se unen y se almacenan en una variable llamada `$invalid` .

# Ejemplo: Obteniendo y validando datos de categoria

*cms/admin/category.php*

```
<?php
// Part A: Setup
declare(strict_types = 1);
① include '../includes/database-connection.php';
include '../includes/functions.php';
include '../includes/validate.php';

// Use strict types
// Database connection
// Include functions
// Include validation

// Initialize variables
② $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // Get id and validate
$category = [
③     'id'          => $id,
    'name'         => '',
    'description'  => '',
    'navigation'   => false,
];
$errors = [
④     'warning'    => '',
    'name'         => '',
    'description'  => '',
];

// Initialize category array
// Initialize errors array

// If there was an id, page is editing the category, so get current category
⑤ if ($id) {
    $sql = "SELECT id, name, description, navigation
⑥         FROM category
        WHERE id = :id;";
    $category = pdo($pdo, $sql, [$id])->fetch();
    if (!$category) {
⑦         redirect('categories.php', ['failure' => 'Category not found']); // Show error
⑧     }
⑨ }
```

# Ejemplo: Obteniendo y validando datos de categoria

*cms/admin/category.php*

```
// Part B: Get and validate form data

⑩ if ($_SERVER['REQUEST_METHOD'] == 'POST') {           // If form submitted
    $category['name'] = $_POST['name'];                 // Get name
    ⑪ $category['description'] = $_POST['description']; // Get description
    $category['navigation'] = (isset($_POST['navigation'])
    and ($_POST['navigation'] == 1)) ? 1 : 0;           // Get navigation

    // Check if all data is valid and create error messages if it is invalid
    ⑫ $errors['name'] = (is_text($category['name'], 1, 24))
        ? '' : 'Name should be 1-24 characters.';      // Validate name
    $errors['description'] = (is_text($category['description'], 1, 254))
        ? '' : 'Description should be 1-254 characters.'; // Validate description

    ⑬ $invalid = implode($errors);                      // Join error messages
```

## Ejemplo: Guardando datos de la categoría

En la parte C de esta página, el archivo `category.php` **determina si la base de datos debe guardar o no los datos y, en caso afirmativo, si se debe añadir una nueva categoría o actualizar una categoría existente.**



# Ejemplo: Guardando datos de la categoría

1. Una sentencia if comprueba si `$invalid` contiene texto. Si lo contiene, la condición se evalúa a true, indicando que hay errores que el usuario necesita corregir. El bloque de código subsiguiente almacena un mensaje de advertencia en el array `$errors`.
2. En caso contrario, los datos son válidos y pueden ser procesados.
3. Los datos del array `$category` se copian en la variable `$arguments`. Esto se hace porque:
  - Cuando la función `pdo()` ejecuta la sentencia SQL, utiliza los valores que se guardaron en `$category` para reemplazar los marcadores de posición.
  - Pero, la sentencia SQL no siempre necesita todos los elementos que están guardados en el array `$category`, y la función `pdo()` no se ejecutará si algunos de los elementos no son eliminados (ver Paso 9).

## Ejemplo: Guardando datos de la categoría

4. Si `$id` contiene un número (que se trata como verdadero), significa que la página está actualizando una categoría existente.
5. La variable `$sql` contiene la sentencia SQL para actualizar la categoría. Comienza con el comando `UPDATE` y el nombre de la tabla a actualizar.
6. La cláusula `SET` es seguida por los nombres de las columnas que serán actualizadas y los marcadores de posición que serán reemplazados por los valores de esas columnas.
7. La cláusula `WHERE` indica el id de la fila de la tabla de categorías que debe actualizarse.

# Ejemplo: Guardando datos de la categoría

8. Si `$id` no contiene un número, significa que la página está añadiendo una nueva categoría a la base de datos.
9. La función `unset()` de PHP elimina el elemento que contiene el id del artículo del array de argumentos. Esto se hace porque el array de datos que contiene los valores utilizados para reemplazar los marcadores de posición en la sentencia SQL no puede contener elementos extra.
10. La variable `$sql` contiene la sentencia SQL para crear una nueva categoría. Comienza con:
  - `INSERT` para añadir una nueva fila a la base de datos
  - `INTO` seguido del nombre de la tabla a la que se van a añadir los datos
  - Los nombres de las columnas a las que se darán valores, escritos entre paréntesis

## Ejemplo: Guardando datos de la categoría

11. El comando `VALUES` va seguido de los nombres de los marcadores de posición que representan los nuevos valores. También se escriben entre paréntesis.
12. La sentencia SQL se ejecuta en un bloque try porque el nombre de la categoría tiene una restricción de unicidad y, si el usuario intentara proporcionar un nombre que ya estuviera utilizado, se lanzaría una excepción.

## Ejemplo: Guardando datos de la categoría

13. La función `pdo()` ejecuta la sentencia SQL.

14. Si el código del bloque try sigue en ejecución, la sentencia SQL se ha ejecutado correctamente, por lo que se llama a la función `redirect()` (ya vista anteriormente en esta unidad). El primer argumento indica que el usuario debe ser devuelto a la página `categories.php`. El segundo argumento es un array que indica que la categoría fue guardada: `['success'=> 'Category saved']`. La función `redirect()` tomará estos datos y le dirá al navegador que solicite la siguiente página: `category.php?success=Category%20saved`



## Ejemplo: Guardando datos de la categoría

Redirigir al visitante al `categories.php` cuando la categoría ha sido guardada evita que el usuario envíe los datos de nuevo o refresque la página.

15. Si los datos de la categoría no han podido ser guardados, se habrá lanzado una excepción, y el intérprete PHP ejecutará el código en el bloque catch. El propósito del bloque catch es comprobar si la excepción fue lanzada porque el nombre de la categoría no era único. Dentro del bloque catch, el objeto de excepción será almacenado en una variable llamada `$e`.

## Ejemplo: Guardando datos de la categoría

16. La condición de una sentencia if comprueba la propiedad `errorInfo` del objeto de excepción, que contiene un array indexado. Se almacena un código de error en el elemento cuya clave es 1. Si el código de error es 1062, indica que se ha roto la restricción de unicidad y que el nombre de la categoría ya está siendo utilizado.
17. Se almacena un mensaje de error en el array `$errors` indicando al usuario que el nombre de la categoría ya está siendo utilizado.
18. Si el objeto de excepción tiene un código de error diferente, la excepción es lanzada de nuevo y será manejada por la función de manejo de excepciones por defecto.

# Ejemplo: Guardando datos de la categoría

*cms/admin/category.php*

```
// Part C: Check if data is valid, if so update database
1  if ($invalid) {                                // If data is invalid
2      $errors['warning'] = 'Please correct errors'; // Create error message
3  } else {                                        // Otherwise
4      $arguments = $category;                    // Set arguments array for SQL
5      if ($id) {                                  // If there is an id
6          $sql = "UPDATE category
7              SET name = :name, description = :description,
8              navigation = :navigation
9              WHERE id = :id;";                    // SQL to update category
10     } else {                                     // If there is no id
11         unset($arguments['id']);                 // Remove id from category array
12         $sql = "INSERT INTO category (name, description, navigation)
13             VALUES (:name, :description, :navigation);"; // Create category
14     }
15
16     // When running the SQL, three things can happen:
17     // Category saved | Name already in use | Exception thrown for other reason
18     try {                                         // Start try block to run SQL
19         pdo($pdo, $sql, $arguments);             // Run SQL
20         redirect('categories.php', ['success' => 'Category saved']); // Redirect
21     } catch (PDOException $e) {                 // If a PDO exception was raised
22         if ($e->errorInfo[1] === 1062) {          // If error is duplicate entry
23             $errors['warning'] = 'Category name already in use'; // Store error message
24         } else {                                  // Otherwise unexpected error
25             throw $e;                            // Re-throw exception
26         }
27     }
28 }
```

## **Ejemplo: Formulario para crear o editar datos de una categoría**

**La parte D del proceso consiste en mostrar al visitante un formulario que puede utilizar para crear o editar la información de la categoría. El mismo formulario se muestra al usuario tanto si está creando como editando una categoría.**

# Ejemplo: Formulario para crear o editar datos de una categoría

1. El atributo action de la etiqueta `<form>` de apertura apunta a la misma página ( `category.php` ). La cadena de consulta contiene una clave llamada id, y su valor es el valor almacenado en el atributo `$id` . Si la categoría ya ha sido creada, contendrá el id de la categoría; si no, será nulo. El formulario se envía utilizando HTTP POST.
2. Si el formulario se hubiera enviado y los datos no fueran válidos o si el nombre de la categoría ya estuviera siendo utilizado, se habría almacenado un mensaje de error en el array `$errors` como valor de la clave *warning*. Una sentencia if comprueba si hay un mensaje de error.

# Ejemplo: Formulario para crear o editar datos de una categoría

3. En caso afirmativo, se mostrará encima del formulario.
4. Una entrada de texto permite al usuario introducir o actualizar el nombre de la categoría. Si ya se ha proporcionado un nombre, se muestra en el atributo value de la entrada de texto. La función `html_escape()` asegura que cualquier carácter HTML reservado en ese valor sea reemplazado por entidades. Esto evita el riesgo de un ataque XSS.



# Ejemplo: Formulario para crear o editar datos de una categoría

Cuando la página se carga por primera vez para crear una nueva categoría, el usuario no habrá proporcionado ningún dato de categoría. Por eso es importante inicializar el array `$category` en la Parte A (especificando los nombres de las claves y estableciendo sus valores como cadenas en blanco) para que la página tenga valores que pueda mostrar en los controles del formulario.

5. El array `$errors` también se inicializó en la Parte A. Tiene un elemento para cada entrada de texto. Si el formulario se enviaba y el nombre de la categoría no era válido, el valor asociado con la clave del nombre contendría un mensaje de error describiendo el problema y se mostraría debajo de la entrada de texto.

## Ejemplo: Formulario para crear o editar datos de una categoría

Si el formulario no fue enviado, o no hubo errores, contendría una cadena en blanco (porque fue inicializada en la Parte A) y la cadena en blanco se mostraría bajo la entrada de texto. (Si el array `$errors` no se hubiera inicializado en la Parte A, al intentar mostrar este mensaje se produciría un error de índice Indefinido).

# Ejemplo: Formulario para crear o editar datos de una categoría

6. Una entrada `<textarea>` permite al usuario proporcionar una descripción para la categoría. Si ya se ha proporcionado un valor, se muestra entre las etiquetas de apertura y cierre.
7. Si hubo un problema al validar la descripción, el mensaje de error se muestra debajo de ella.
8. Una casilla de verificación indica si el nombre de la categoría debe mostrarse o no en la navegación.
9. Un operador ternario comprueba si a la opción de navegación se le ha dado el valor 1. En caso afirmativo, añade el atributo `checked` a la entrada de la casilla de verificación para seleccionarla. En caso contrario, escribe en su lugar una cadena en blanco.
10. Se añade un botón de envío al final del formulario.

# Ejemplo: Formulario para crear o editar datos de una categoría

*cms/admin/category.php*

```
<?php include 'includes/admin-header.php'; ?>
<main class="container admin" id="content">
  ① <form action="category.php?id=<?= $id ?>" method="post" class="narrow">

    <h2>Edit Category</h2>
    ② <?php if ($errors['warning']) { ?>
    ③ <div class="alert alert-danger"><?= $errors['warning'] ?></div>
    <?php } ?>

    <div class="form-group">
      <label for="name">Name: </label>
      ④ <input type="text" name="name" id="name"
      ⑤ value="<?= htmlspecialchars($category['name']) ?>" class="form-control">
      <span class="errors"><?= $errors['name'] ?></span>
    </div>

    <div class="form-group">
      <label for="description">Description: </label>
      ⑥ <textarea name="description" id="description" class="form-control">
      ⑦ <?= htmlspecialchars($category['description']) ?></textarea>
      <span class="errors"><?= $errors['description'] ?></span>
    </div>

    <div class="form-check">
      ⑧ <input type="checkbox" name="navigation" id="navigation"
      ⑨ value="1" class="form-check-input"
      <?= ($category['navigation'] === 1) ? 'checked' : '' ?>
      <label class="form-check-label" for="navigation">Navigation</label>
    </div>

    ⑩ <input type="submit" value="save" class="btn btn-primary btn-save">

  </form>
</main>
<?php include 'includes/admin-footer.php'; ?>
```

## Ejemplo: Borrando una categoría

Cuando los usuarios hacen clic en el enlace para borrar una categoría, **la página obtiene el nombre de la categoría de la base de datos y se lo muestra al usuario, pidiéndole que confirme que quiere borrarla** (esto evita que alguien haga clic accidentalmente en un enlace que borra una categoría).

Si el usuario confirma que quiere borrar la categoría, **la página se recarga e intenta borrarla. Si esto tiene éxito, el usuario es enviado a `categories.php`**, y se le muestra un mensaje diciendo que funcionó.

## Ejemplo: Borrando una categoría

1. Los tipos estrictos están habilitados y los archivos requeridos están incluidos.
2. `filter_input()` busca el nombre `id` en la cadena de consulta. Si está presente y tiene un valor entero válido, se almacena en `$id`. Si su valor no es un entero válido, `$id` almacena `false`. Si no está presente, `$id` almacena `null`.
3. La variable `$category` contendrá el nombre de la categoría; se inicializa para contener una cadena en blanco.



## Ejemplo: Borrando una categoría

4. Una sentencia if comprueba si el valor en `$id` no es equivalente a true (si se estableció como false o null en el paso 2). Si es así, el usuario es enviado a `categories.php` con un mensaje diciendo que la categoría no fue encontrada.
5. Si la página sigue en ejecución, la variable `$sql` almacena una consulta SQL para obtener el nombre de la categoría.
6. La función `pdo()` ejecuta la consulta SQL y el método `fetchColumn()` intenta obtener el nombre de la categoría. Si se encuentra la categoría, su nombre se almacena en `$category`. En caso contrario, `fetchColumn()` devuelve false.
7. Una sentencia if comprueba si el valor de `category` es falso. Si lo es, el usuario es enviado a `categories.php` con un mensaje diciendo que la categoría no fue encontrada.

## Ejemplo: Borrando una categoría

8. Si el formulario ha sido enviado...
9. Se crea un bloque try para eliminar la categoría.
10. El SQL para borrar la categoría se almacena en `$sql`.
11. Se utiliza la función `pdo()` para eliminar la categoría.
12. Si la sentencia SQL se ejecuta sin error, se utiliza la función `redirect()` para enviar al usuario a `categories.php`, con un mensaje confirmando que la categoría ha sido eliminada.

## Ejemplo: Borrando una categoría

13. Si se lanza una excepción al ejecutar la sentencia SQL, se ejecuta el bloque catch.
14. Si el código de error es 1451, una restricción de integridad está impidiendo la eliminación de la categoría (porque la categoría todavía contiene artículos).
15. En este caso, se utiliza la función `redirect()` para enviar al visitante a la página `categories.php` con un mensaje de error indicándole que la categoría contiene artículos que deben ser movidos o borrados primero.
16. De lo contrario, el error se vuelve a lanzar, y será manejado por el manejador de excepciones por defecto.

## Ejemplo: Borrando una categoría

17. El formulario se utiliza para mostrar el nombre de la categoría y pedir al usuario que confirme que la categoría debe ser eliminada. El atributo action en la etiqueta `<form>` envía el formulario a `category-delete.php`. La cadena de consulta contiene el id de la categoría a eliminar.
18. Se muestra el nombre de la categoría a eliminar.
19. El botón *submit* se utiliza para confirmar que la categoría puede ser eliminada.

# Ejemplo: Borrando una categoría

*cms/admin/category-delete.php*

```
<?php
① declare(strict_types = 1); // Use strict types
  include '../includes/database-connection.php'; // Database connection
  include '../includes/functions.php'; // Include functions

② $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // Get and validate id
③ $category = ''; // Initialize category name

  if (!$id) { // If no valid id
④    redirect('categories.php', ['failure' => 'Category not found']); // Redirect + error
  }
```

# Ejemplo: Borrando una categoría

*cms/admin/category-delete.php*

```
⑤ $sql = "SELECT name FROM category WHERE id = :id;";           // SQL to get category name
⑥ $category = pdo($pdo, $sql, [$id])->fetchColumn();           // Get category name
⑦ { if (!$category) {                                           // If no category
    redirect('categories.php', ['failure' => 'Category not found']); // Redirect + error
} }

⑧ if ($_SERVER['REQUEST_METHOD'] == 'POST') {                  // If form was submitted
⑨     try {                                                     // Try to delete data
⑩         $sql = "DELETE FROM category WHERE id = :id;";       // SQL to delete category
⑪         pdo($pdo, $sql, [$id]);                               // Delete category
⑫         redirect('categories.php', ['success' => 'Category deleted']); // Redirect
⑬     } catch (PDOException $e) {                               // Catch exception
⑭         if ($e->errorInfo[1] === 1451) {                       // If integrity constraint
⑮             redirect('categories.php', ['failure' => 'Category contains articles that
⑯             must be moved or deleted before you can delete it']); // Redirect
⑰         } else {                                             // Otherwise
⑱             throw $e;                                         // Re-throw exception
⑲         }
    }
}

}??>
```



# Ejemplo: Borrando una categoría

*cms/admin/category-delete.php*

```
<?php include 'includes/admin-header.php'; ?>

<main class="container admin" id="content">
  <h2>Delete Category</h2>
  17 <form action="category-delete.php?id=<?= $id ?>" method="POST" class="narrow">
  18   <p>Click confirm to delete the category <?= html_escape($category) ?></p>
  19   <input type="submit" name="delete" value="confirm" class="btn btn-primary">
    <a href="categories.php" class="btn btn-danger">cancel</a>
  </form>
</main>

<?php include 'includes/admin-footer.php'; ?>
```

# Creando y editando artículos

El archivo `article.php` que se utiliza para **crear y editar artículos** comparte un flujo de control muy similar con `category.php`, pero tiene que recoger más datos y además permite a los usuarios subir imágenes, lo que añade complejidad.

# Creando y editando artículos

El archivo `article.php` permite a los usuarios:

- Crear o editar el texto de un artículo
- Cargar una imagen para el artículo

Es más complicado que `category.php` porque:

- Se almacenan más datos sobre cada artículo, por lo que hay más datos que recopilar y validar.
- Los datos sobre el artículo están vinculados a datos de otras tablas (la categoría y el miembro que lo escribió).
- Los usuarios pueden subir opcionalmente una imagen para el artículo con un texto alternativo que la describa.

# Creando y editando artículos

Para guardar un artículo y su imagen en la base de datos, el código PHP tiene que trabajar tanto con la tabla de artículos como con la de imágenes.

Dado que SQL sólo puede insertar nuevas filas de datos en una tabla a la vez, las sentencias SQL utilizadas para crear y editar un artículo utilizarán **transacciones**.

Las transacciones permiten a la base de datos comprobar si los cambios en un conjunto de sentencias SQL se ejecutarán correctamente, y sólo guardará los cambios si todos ellos lo hacen. Si hay algún problema con una sola sentencia SQL de la transacción, no se guardará ninguno de los cambios.

# Creando y editando artículos

Como has visto, el flujo de control determina qué sentencias se ejecutan. La posibilidad de subir imágenes puede hacer que el flujo de control sea mucho más complejo. Imagina que un usuario ya ha subido un artículo y una imagen; podría querer editar estos datos y:

- **Actualizar el artículo, pero dejar la imagen como está.** Esto sólo actualizaría la tabla del artículo.
- **Actualizar el artículo y la imagen.** Esto implicaría eliminar primero el archivo de imagen antiguo y los datos correspondientes de las tablas de imágenes y artículos, luego subir el nuevo archivo de imagen y, a continuación, actualizar las tablas de artículos e imágenes con los nuevos datos.
- **Sólo cambiar el texto alternativo de la imagen,** nada más. Esto significaría sólo actualizar la tabla de imágenes.

# Creando y editando artículos

Cuantas más opciones tenga el usuario en una sola página, **más complejo se vuelve el flujo de control y más difícil de seguir es el código.**

Para evitar que el flujo de control se vuelva demasiado complejo, puedes restringir el número de acciones que un usuario puede realizar en una sola página. Por ejemplo, los usuarios deben borrar una imagen antes de subir una nueva, y hay una página separada para editar el texto alternativo.




# Creando y editando artículos

Al crear un artículo, los usuarios pueden suministrar una imagen y texto al mismo tiempo.

NOTA: La mayoría de los navegadores no permiten que el código del servidor rellene el control de carga de archivos HTML. Por lo tanto, si se envía un artículo, pero no pasa la validación, los usuarios tendrán que seleccionar la imagen de nuevo.

Cuando los usuarios tengan que subir la imagen de nuevo, también se verán obligados a volver a introducir el texto alternativo.

# Creando y editando artículos




Articles / Categories

## Edit Article

Upload image:

SELECCIONAR ARCHIVO

Ningún archivo seleccionado



Alt text:

Title:

Summary:

Content:

Author: Ivy Stone

Category: Digital

☐ Published

SAVE

# Creando y editando artículos

Al actualizar un artículo:

- Si el usuario no ha subido una imagen, se muestra la parte del formulario que le permite subir una imagen (como arriba).
- Si ha cargado una imagen, se muestra la imagen y su texto alternativo en lugar del formulario.

# Creando y editando artículos

Una vez que se ha cargado una imagen, hay dos enlaces debajo de la imagen que permiten a los usuarios:

- **Editar el texto alternativo**
- **Suprimir la imagen (y el texto alternativo)**

La página `articles.php`, que muestra todos los artículos, está incluida en la descarga de recursos de la unidad y funciona igual que la página `categories.php`, que hemos visto anteriormente.

# Creando y editando artículos

## Edit Article

Image:



**Alt text:** Photograph of the interior of a cafe

EDIT ALT TEXT

DELETE IMAGE

Title:

Golden Brown

Summary:

Photograph for interior design book

Content:

This photograph is one of a range that appears in a book about interior design called Golden Brown. The interiors featured in the book show the current trend for looking back to the 1970's and the colour treatment of the photography reflects this warm, earthy palette.

Author: Emiko Ito

Category: Photography

☒ Published

SAVE

# Transacciones

Una transacción se utiliza para **agrupar un conjunto de sentencias SQL**. Si todas las sentencias tienen éxito, todos los cambios se guardan en la base de datos. **Si alguna de ellas falla, no se guardará ninguno de los cambios.**



# Transacciones

Algunas tareas requieren más de una sentencia SQL. Por ejemplo, cuando se carga una imagen para un artículo, el código PHP debe:

- Añadir los datos de la imagen a la tabla de imágenes
- Obtener el ID que la base de datos creó para la imagen (esto se crea utilizando la función de auto-incremento)
- Añadir el ID de la nueva imagen a la tabla del artículo en la columna `image_id`.

# Transacciones

Además, la base de datos sólo puede añadir datos a una tabla a la vez; por lo tanto, cada vez que se añade un nuevo artículo, debe añadir datos a las tablas de imágenes y artículos utilizando sentencias SQL independientes.

Cuando se ejecuta cada una de estas sentencias SQL, cada una se denomina **operación**. Una operación representa una tarea que puede implicar varias operaciones de base de datos.

# Transacciones

Al incorporar más de una operación en una transacción, PDO puede comprobar que todas las sentencias SQL de la transacción se ejecutarán correctamente:

- Si no causan una excepción, puede **confirmar (commit)** esos cambios en la base de datos.
- Si hay un problema ejecutando alguna de las sentencias, PDO lanzará una excepción. El código PHP puede entonces decirle a PDO que se asegure de que ninguno de los cambios sea guardado en la base de datos. Esto se conoce como **revertir (roll back)** la transacción.

# Transacciones

Las transacciones se realizan utilizando bloques try y catch.

- El bloque try contiene el código que se desea intentar ejecutar.
- El bloque catch se utiliza para manejar una excepción si se lanza una en el bloque try.

# Transacciones

Las sentencias dentro del bloque try le dicen a PDO que:

- **Inicie una transacción**
- **Ejecute todas las sentencias SQL individuales** que forman la transacción
- **Confirme los cambios en la base de datos**

# Transacciones

Si la ejecución de cualquiera de las sentencias resulta en una **excepción**, el intérprete de PHP ejecutará inmediatamente el código que se encuentra en el bloque catch subsiguiente. El bloque catch necesita:

- Utilizar PDO para asegurar que la base de datos retrocede (*roll back*) cualquier cambio realizado, de forma que contenga los mismos datos que tenía antes de iniciar la transacción.
- Volver a lanzar la excepción para que pueda ser capturada por la función de gestión de excepciones por defecto.



# Transacciones

**Esto garantiza que todas las sentencias SQL serán ejecutadas o, si se lanza una excepción en el bloque try, ninguno de los cambios será almacenado.**

# Transacciones

El objeto PDO tiene tres métodos que le permiten iniciar una transacción, confirmar los cambios en la base de datos o revertir los cambios para que la base de datos contenga los mismos datos que tenía antes de realizar los cambios.

METHOD	DESCRIPTION
<code>beginTransaction()</code>	Start a transaction
<code>commit()</code>	Save changes to the database
<code>rollBack()</code>	Undo changes in transaction

# Transacciones

A continuación se muestra un ejemplo de uso de estos métodos:

1. Un bloque try contiene el código para ejecutar todas las sentencias SQL de la transacción.
2. La primera sentencia llama al método `beginTransaction()` del objeto PDO para iniciar la transacción.
3. Esto es seguido por el código para ejecutar las sentencias SQL involucradas en la transacción.
4. La última sentencia del bloque try llama al método `commit()` del objeto PDO para guardar los cambios.

# Transacciones

5. Si se lanza una excepción mientras se está ejecutando alguna de las sentencias SQL, el código del bloque try deja de ejecutarse y el bloque catch posterior gestiona la excepción.
6. El método rollBack del objeto PDO es llamado para asegurar que ninguno de los cambios en el bloque try son almacenados en la base de datos.
7. El objeto de excepción se vuelve a lanzar para que pueda ser manejado por la función por defecto del manejador de excepciones.

# Transacciones

```
① try {  
②     $pdo->beginTransaction();  
③     // Run SQL statements here  
④     $pdo->commit();  
⑤ } catch (PDOException $e) {  
⑥     $pdo->rollBack();  
⑦     throw $e;  
    }
```

## Artículos: Configurar página (Parte A)

La página `article.php` utiliza un flujo de control similar al de `category.php`. Si la cadena de consulta:

- Tiene un nombre llamado `id`, su valor debe ser el **id de un artículo que el usuario está intentando editar**.
- No tiene un nombre llamado `id`, significa que **la página está siendo utilizada para crear un nuevo artículo**.



## Articulos: Configurar página (Parte A)

1. Se declaran los tipos estrictos y se incluyen los archivos necesarios.
2. La ruta de la carpeta de subida se guarda en `$uploads` . Los tipos de imagen permitidos se almacenan en `$file_types` . Las extensiones de archivo permitidas se guardan en `$file_exts` . Su tamaño máximo de archivo se guarda en `$max_size` .
3. La función `filter_input()` comprueba si hay un nombre llamado id en la cadena de consulta. Si contiene un número entero válido, se guarda en `$id` . Si contiene un valor inválido, `$id` almacena false. Si no está presente, `$id` guarda null.

# Artículos: Configurar página (Parte A)

4. Para comprobar si se ha subido un archivo, la página intenta obtener la ubicación temporal del archivo y almacenarla en `$temp` . Se utiliza el operador de coalescencia nula para almacenar una cadena en blanco en `$temp` si no se ha subido una imagen.
5. Se inicializa la variable `destination`. Si se ha subido una imagen, se actualizará para contener la ruta donde debe guardarse la imagen.
6. El array `$article` es declarado e inicializado con valores por defecto. Estos valores por defecto dan al formulario de la Parte D algo que mostrar cuando un usuario está a punto de crear un nuevo artículo (pero todavía no ha proporcionado valores). Para que el código del ejemplo quepa en la captura, cada línea de código declara dos elementos del array; en la descarga de código, cada uno está en una nueva línea.
7. El array `$errors` se inicializa con cadenas en blanco. Los valores de este array se muestran después de cada control de formulario.

## Artículos: Configurar página (Parte A)

8. Una sentencia `if` comprueba si se ha proporcionado un `id` en la cadena de consulta. Si es así, la página está editando un artículo y los datos del artículo deben ser recogidos de la base de datos.
9. `$sql` almacena la consulta SQL para obtener los datos del artículo.
10. La función `pdo()` ejecuta la sentencia SQL, y el método `fetch()` del objeto `PDOStatement` recoge los datos del artículo. Los valores que devuelve sobrescriben los datos almacenados en el array `$article` creado en el Paso 6. Si no se encuentra el artículo, `article` mantendrá `false`.

# Artículos: Configurar página (Parte A)

11. Si la variable `$article` mantiene el valor false, el usuario es redirigido a `articles.php` con un mensaje de fallo indicando que no se ha podido encontrar el artículo.
12. Si se ha guardado una imagen para el artículo, el elemento `image_file` del array `$article` tendrá un valor, por lo que a `$saved_image` se le da el valor true. Si no hay imagen, se le da el valor false.
13. Todos los autores y categorías se recogen de la base de datos. Se utilizan para crear cuadros desplegables para seleccionar el autor y la categoría, y para validar los valores seleccionados. En primer lugar, `$sql` contiene una consulta SQL para obtener el id, nombre y apellidos de cada miembro.

## Artículos: Configurar página (Parte A)

14. La función `pdo()` ejecuta la consulta, y el método `fetchAll()` del objeto `PDOStatement` recoge los resultados y los almacena en la variable `$authors`. (Se devuelve un array indexado, y el valor de cada elemento es un array asociativo de detalles de los miembros).
15. A continuación la variable `$sql` contiene la consulta SQL para obtener el id y el nombre de todas las categorías.
16. La función `pdo()` ejecuta la consulta, y el método `fetchAll()` del objeto `PDOStatement` obtiene los datos de las categorías; se almacenan en `$categories`.

# Articulos: Configurar página (Parte A)

*cms/admin/article.php*

```
// Part A: Setup
declare(strict_types = 1); // Use strict types
1 include '../includes/database-connection.php'; // Database connection
include '../includes/functions.php'; // Functions
include '../includes/validate.php'; // Validate functions
$uploads = dirname(__DIR__, 1) . DIRECTORY_SEPARATOR . 'uploads' . DIRECTORY_SEPARATOR;
2 $file_types = ['image/jpeg', 'image/png', 'image/gif']; // Allowed types
$file_exts = ['jpg', 'jpeg', 'png', 'gif']; // Allowed extensions
$max_size = 5242880; // Max file size
// Initialize variables that the PHP code needs
3 $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // Get id + validate
4 $temp = $_FILES['image']['tmp_name'] ?? ''; // Temporary image
5 $destination = ''; // Where to save file
// Initialize variables that the HTML page needs
6 $article = [
    'id' => $id, 'title' => '',
    'summary' => '', 'content' => '',
    'member_id' => 0, 'category_id' => 0,
    'image_id' => null, 'published' => false,
    'image_file' => '', 'image_alt' => '',
]; // Article data
$errors = [
7 'warning' => '', 'title' => '', 'summary' => '', 'content' => '',
  'author' => '', 'category' => '', 'image_file' => '', 'image_alt' => '',
]; // Error messages
```



# Articulos: Configurar página (Parte A)

*cms/admin/article.php*

```
// If there was an id, page is editing an article, so get current article data
⑧ if ($id) { // If have id
    $sql = "SELECT a.id, a.title, a.summary, a.content,
                a.category_id, a.member_id, a.image_id, a.published,
                i.file AS image_file,
                i.alt AS image_alt
    ⑨ FROM article AS a
        LEFT JOIN image AS i ON a.image_id = i.id
        WHERE a.id = :id;"; // SQL to get article
    ⑩ $article = pdo($pdo, $sql, [$id])->fetch(); // Get article data
    if (!$article) { // If no article
    ⑪ redirect('articles.php', ['failure' => 'Article not found']); // Redirect
    }
}

⑫ $saved_image = $article['image_file'] ? true : false; // Has an image been uploaded
// Get all members and all categories
⑬ $sql = "SELECT id, forename, surname FROM member;"; // SQL to get all members
⑭ $authors = pdo($pdo, $sql)->fetchAll(); // Get all members
⑮ $sql = "SELECT id, name FROM category;"; // SQL to get all categories
⑯ $categories = pdo($pdo, $sql)->fetchAll(); // Get all categories
```

# Artículos: Obtener y validar datos (Parte B)

Estas diapositivas desglosan el código de la Parte B, que **recoge y valida los datos enviados por el usuario**.

1. Una sentencia if comprueba si el formulario fue enviado.
2. El elemento `image_file` se añade al array `$errors` y un operador ternario le asigna un valor. Si un archivo no pudo ser subido porque era más grande que el tamaño máximo de archivo de subida en `php.ini` o `.htaccess`, almacena un error; de lo contrario, almacena una cadena en blanco.

## Artículos: Obtener y validar datos (Parte B)

3. Una sentencia if comprueba si un archivo fue subido y no hubo errores. Si es así, el archivo será validado.
4. Como se ha subido una imagen, se recoge su texto alternativo y se almacena en el array `$article`.
5. Si el tipo de medio de la imagen es un tipo de archivo permitido (establecido en el Paso 2 de la Parte A), se añade una cadena en blanco al valor del elemento `image_file` de la array `$errors`. En caso contrario, se añade un mensaje de error.
6. Si la extensión del archivo está permitida (establecida en el paso 2 de la Parte A), se añade una cadena en blanco al valor del elemento `image_file` del array `$errors`. Si no, se añade un mensaje de error.

## Artículos: Obtener y validar datos (Parte B)

7. Si el archivo es mayor que el tamaño máximo (establecido en el Paso 2 de la Parte A), se añade un mensaje diciendo que es demasiado grande al valor en `image_file`.
8. Se añade una clave llamada `image_alt` al array `$errors`. Si el texto alt tiene entre 1 y 254 caracteres almacena una cadena en blanco; si no, almacena un mensaje de error.
9. Una sentencia if comprueba si las claves `image_file` e `image_alt` del array `$errors` están ambas vacías. Si es así, la imagen puede ser procesada.

## Artículos: Obtener y validar datos (Parte B)

10. La función `create_filename()` (que se encuentra en `functions.php` y se introdujo en la Unidad 7) elimina los caracteres no deseados del nombre del archivo y se asegura de que sea único. El nombre se almacena en la clave `image_file` del array `$article`.
11. La variable `$destination` contiene la ruta donde se cargará la imagen. Se crea uniéndola a la carpeta uploads (almacenada en la variable `$uploads` en el paso 2 de la página anterior) con el nombre de archivo creado en el paso anterior.

## Artículos: Obtener y validar datos (Parte B)

12. Los datos del artículo se recogen del formulario. Si se está actualizando un artículo, estos valores sobrescriben los existentes que se recogieron de la base de datos en los Pasos 9-10 de la Parte A.
13. La opción de publicar el artículo es una casilla de verificación. Sólo se envía al servidor si se ha marcado. Se asigna un valor de 1 si se ha marcado y de 0 si no. Esto se debe a que 0 y 1 son los valores que almacena la base de datos para representar valores booleanos de verdadero y falso.



## Artículos: Obtener y validar datos (Parte B)

14. Cada trozo de texto se valida con las funciones que se crearon en el Capítulo 6. Si los datos son válidos, el elemento almacena una cadena en blanco. En caso contrario, almacena un mensaje de error para ese control de formulario.
15. Las funciones `is_member_id()` y `is_category_id()` han sido añadidas al archivo `validate.php`. Recorren en bucle el array de miembros y categorías que se recogieron en los Pasos 13-16 de la Parte A para comprobar si el valor proporcionado es válido.
16. Los valores del array `$errors` se unen en una cadena utilizando la función `implode()` de PHP y se guardan en una variable llamada `$invalid`. Esto se utilizará para saber si los datos deben guardarse en la base de datos o no.

# Artículos: Obtener y validar datos (Parte B)

*cms/admin/article.php*

```
// Part B: Get and validate form data
① if ($_SERVER['REQUEST_METHOD'] == 'POST') { // If form submitted
    // If file bigger than limit in php.ini or .htaccess store error message
    ② $errors['image_file'] = ($_FILES['image']['error'] === 1) ? 'File too big ' : '';

    // If image was uploaded, get image data and validate it
    ③ if ($temp and $_FILES['image']['error'] === 0) { // If file uploaded
        ④ $article['image_alt'] = $_POST['image_alt']; // Get alt text
        // Validate image file
        ⑤ {
            $errors['image_file'] .= in_array(mime_content_type($temp), $file_types)
                ? '' : 'Wrong file type. '; // Check file type
            $ext = strtolower(pathinfo($_FILES['image']['name'], PATHINFO_EXTENSION));
            ⑥ {
                $errors['image_file'] .= in_array($ext, $file_extensions)
                    ? '' : 'Wrong file extension. '; // Check extension
            }
            ⑦ {
                $errors['image_file'] .= ($_FILES['image']['size'] <= $max_size)
                    ? '' : 'File too big. '; // Check size
            }
            ⑧ {
                $errors['image_alt'] = (is_text($article['image_alt'], 1, 254))
                    ? '' : 'Alt text must be 1-254 characters.'; // Check alt text
            }
        }
        // If image file is valid, specify the location to save it
        ⑨ if ($errors['image_file'] === '' and $errors['image_alt'] === '') { // If valid
            ⑩ $article['image_file'] = create_filename($_FILES['image']['name'], $uploads);
            ⑪ $destination = $uploads . $article['image_file']; // Destination
        }
    }
}
```

# Artículos: Obtener y validar datos (Parte B)

*cms/admin/article.php*

```
// Get article data
12 {
    $article['title']      = $_POST['title'];           // Title
    $article['summary']    = $_POST['summary'];         // Summary
    $article['content']    = $_POST['content'];         // Content
    $article['member_id']  = $_POST['member_id'];       // Author
    $article['category_id'] = $_POST['category_id'];    // Category
13 {
    $article['published']  = (isset($_POST['published'])
        and ($_POST['published'] == 1)) ? 1 : 0;       // Is it published?

// Validate article data and create error messages if it is invalid
14 {
    $errors['title']      = is_text($article['title'], 1, 80)
        ? '' : 'Title must be 1-80 characters';
    $errors['summary']    = is_text($article['summary'], 1, 254)
        ? '' : 'Summary must be 1-254 characters';
    $errors['content']    = is_text($article['content'], 1, 100000)
        ? '' : 'Article must be 1-100,000 characters';
15 {
    $errors['member']     = is_member_id($article['member_id'], $authors)
        ? '' : 'Please select an author';
    $errors['category']   = is_category_id($article['category_id'], $categories)
        ? '' : 'Please select a category';
16 {
    $invalid = implode($errors);                       // Join errors
```

## Artículos: Guardar cambios (Parte C)

Las siguientes diapositivas muestran el código de la Parte C.

1. Una sentencia if comprueba si `$invalid` contiene algún mensaje de error. Si lo contiene, el array `$errors` almacena un mensaje indicando a los usuarios que corrijan los errores del formulario.
2. Si no es así, los datos eran válidos y pueden ser procesados.
3. Los datos en `$article` se copian a `$arguments`. La función `pdo()` utilizará los valores en `$arguments`. El formulario HTML de la Parte D utiliza los valores de `$article`.

## Artículos: Guardar cambios (Parte C)

4. Un bloque try contiene el código para actualizar la base de datos.
5. Se inicia una transacción porque se necesitan dos sentencias SQL para crear o actualizar un artículo. Si una falla, ambas deben fallar.
6. Si `$destination` contiene un valor (Paso 11 Parte B) se ha cargado una imagen. La imagen debe ser procesada antes que los datos del artículo porque la tabla de artículos necesita almacenar el id que se crea para la imagen.



## Artículos: Guardar cambios (Parte C)

7. Imagick se utiliza para redimensionar y guardar la imagen:

- Se crea un objeto Imagick para representar la imagen subida (su ruta está en `$temp` - Paso 4 Parte A).
- Se redimensiona a 1200 x 700 píxeles.
- La imagen se guarda en la ruta de `$destination`

8. `$sql` contiene la sentencia SQL para añadir el nombre de archivo de la imagen y el texto alt a la tabla de imágenes de la base de datos.



## Artículos: Guardar cambios (Parte C)

9. La función `pdo()` ejecuta la sentencia SQL.

Los argumentos (el archivo de imagen y el texto alt) se pasan a la función `pdo()` como un array indexado.

10. El id de la imagen se recoge utilizando el método `lastInsertId()` del objeto PDO y se almacena en la clave `image_id` del array `$arguments` (creado en el paso 3).

## Artículos: Guardar cambios (Parte C)

11. Los elementos con las claves `image_file` e `image_alt` se eliminan del array de argumentos porque el array sólo debe contener un elemento por cada marcador de posición de la segunda sentencia SQL.
12. Una sentencia if comprueba si se ha especificado un id. Si es así, se está actualizando un artículo y la variable `$sql` contiene una sentencia SQL para actualizar la base de datos.
13. Si no hay id, se está creando un nuevo artículo, por lo que se elimina id del array `$arguments` y `$sql` contiene la sentencia SQL para añadir un nuevo artículo a la base de datos.

## Artículos: Guardar cambios (Parte C)

14. La función `pdo()` ejecuta la sentencia SQL.
15. Se llama al método `commit()` del objeto PDO para guardar ambos cambios de la transacción en la base de datos.
16. Si el código sigue en ejecución, el artículo se guarda y `redirect()` envía al usuario a `articles.php`.
17. Si PDO lanza una excepción en el bloque try, se ejecuta el bloque catch. El objeto de excepción se almacena en `$e`.
18. El método `rollback()` del objeto PDO evita que la base de datos guarde los cambios en la transacción.

## Artículos: Guardar cambios (Parte C)

19. Si se guardó una imagen en el servidor en el Paso 7, se utiliza el método `unlink()` de PHP para borrarla.
20. Si el código de error del objeto `PDOException` es 1062, el título está siendo utilizado, por lo que se almacena un error en `$errors`.
21. En caso contrario se vuelve a lanzar la excepción.
22. Si se ejecuta la siguiente línea, el artículo debe haber contenido datos no válidos. Si el artículo ya tenía una imagen (ver Paso 12 Parte A), se guarda en el array `$article`. Si no, `image_file` se establece en una cadena en blanco.

# Artículos: Guardar cambios (Parte C)

*cms/admin/article.php*

```
// Part C: Check if data is valid, if so update database
1 if ($invalid) { // If invalid
2     $errors['warning'] = 'Please correct the errors below'; // Store message
3 } else { // Otherwise
4     $arguments = $article; // Save article data
5     try { // Try to insert data
6         $pdo->beginTransaction(); // Start transaction
7         if ($destination) { // If have valid image
8             $imagick = new \Imagick($temp); // Create Imagick object
9             $imagick->cropThumbnailImage(1200, 700); // Create cropped image
10            $imagick->writeImage($destination); // Save file
11            $sql = "INSERT INTO image (file, alt)
                VALUES (:file, :alt);"; // SQL to add image
12            pdo($pdo, $sql, [$arguments['image_file'], $arguments['image_alt']],);
13            $arguments['image_id'] = $pdo->lastInsertId(); // Get new image id
14        }
15        unset($arguments['image_file'], $arguments['image_alt']); // Cut image data
```

# Artículos: Guardar cambios (Parte C)

*cms/admin/article.php*

```
12 if ($id) {  
    $sql = "UPDATE article  
        SET title = :title, summary = :summary, content = :content,  
        category_id = :category_id, member_id = :member_id,  
        image_id = :image_id, published = :published  
        WHERE id = :id;"; // SQL to update article  
13 } else {  
    unset($arguments['id']); // Remove id  
    $sql = "INSERT INTO article (title, summary, content, category_id,  
        member_id, image_id, published)  
        VALUES (:title, :summary, :content, :category_id, :member_id,  
        :image_id, :published);"; // SQL to create article  
}
```



# Artículos: Guardar cambios (Parte C)

*cms/admin/article.php*

```
14         pdo($pdo, $sql, $arguments);           // Run SQL to add article
15         $pdo->commit();                         // Commit changes
16         redirect('articles.php', ['success' => 'Article saved']); // Redirect
17     } catch (PDOException $e) {                 // If PDOException thrown
18         $pdo->rollBack();                       // Roll back SQL changes
19     [      if (file_exists($destination)) {      // If image file exists
20         [      unlink($destination);           // Delete image file
21         [      } // If the exception was a PDOException and it was an integrity constraint
22         [      if ($e->errorInfo[1] === 1062)) {
23             $errors['warning'] = 'Article title already used'; // Store warning
24         } else {                                // Otherwise
25             throw $e;                          // Rethrow exception
26         }
27     }
28 } // If a new image uploaded but data is not valid, remove image from $article
29 $article['image_file'] = $saved_image ? $article['image_file'] : '';
```

## Artículos: Formulario/Mensajes (Parte D)

El mismo formulario se muestra al usuario tanto si está creando como editando un artículo.

NOTA: El formulario que encontraréis en los recursos de la unidad tiene más elementos y atributos HTML que se utilizan para etiquetar los controles del formulario y controlar su presentación. Se han eliminado del código de la derecha para que el código importante quepa en las capturas, y para ayudarte a centrarte en lo que está haciendo.

## Artículos: Formulario/Mensajes (Parte D)

1. El atributo *action* de la etiqueta `<form>` apunta a la página `article.php`. La cadena de consulta contiene un nombre llamado `id`; si la página está actualizando un artículo existente, entonces su valor es el `id` del artículo (que habría sido almacenado en el atributo `$id` en la parte superior del archivo). Los datos se envían utilizando HTTP POST.
2. Si hay un valor para la clave *warning* del array `$errors`, se mostrará al usuario.

## Artículos: Formulario/Mensajes (Parte D)

3. El formulario para subir una imagen sólo se muestra si aún no se ha proporcionado una imagen para el artículo.
4. Una entrada de archivo permite a los visitantes subir una imagen.
5. Si el array `$errors` contiene un mensaje de error para este control de formulario, se muestra después de la entrada de archivo.

## Artículos: Formulario/Mensajes (Parte D)

Un paso equivalente se realiza después de todos los controles de formulario excepto la casilla de verificación para publicar el artículo.

6. Una entrada de texto permite al visitante proporcionar texto alternativo.
7. En caso contrario, si se ha cargado una imagen (y los datos eran válidos), la imagen se muestra en la página seguida del texto alternativo.
8. Debajo hay dos enlaces: el primero permite editar el texto alternativo; el segundo, borrar la imagen.

## Artículos: Formulario/Mensajes (Parte D)

9. El título del artículo se introduce mediante una entrada de texto. Si ya se ha proporcionado un valor, se añade al atributo value. Cualquier carácter reservado se sustituye por entidades para evitar un ataque XSS.
10. El resumen utiliza un elemento `<textarea>` . Si se ha proporcionado un resumen, se escribe entre las etiquetas `<textarea>` utilizando la función `html_escape()` para sustituir los caracteres reservados por entidades.



## Artículos: Formulario/Mensajes (Parte D)

11. El contenido del artículo principal utiliza otro elemento `<textarea>` . Si se ha proporcionado un valor, se escribe entre las etiquetas `<textarea>` .
12. El cuadro de selección de autor muestra una lista de todos los miembros que pueden haber escrito el artículo.
13. Se construye utilizando un bucle foreach que trabaja a través del array de todos los miembros (recogidos en la Parte A y almacenados en una variable llamada `$authors` ).
14. Se añade un elemento `<option>` para cada miembro.

## Artículos: Formulario/Mensajes (Parte D)

15. La condición de un operador ternario comprueba si se ha proporcionado un autor y su id coincide con el id del autor actual que se está añadiendo a la caja de selección. Si es así, se añade el atributo `selected` a la opción para seleccionar al miembro actual como autor del artículo.
16. El nombre del miembro se muestra en la opción.
17. El array de categorías almacenado en `$categories` se utiliza para crear la caja de selección de categorías.
18. Una casilla de selección indica si el artículo debe o no ser publicado (mostrado en el sitio). Un operador ternario comprueba si la opción ha sido marcada; en caso afirmativo, el atributo `checked` se añade al elemento.

# Artículos: Formulario/Mensajes (Parte D)

*cms/admin/article.php*

```

    <!-- Part D - Display form -->
    ① <form action="article.php?id=<?= $id ?>" method="post" enctype="multipart/form-data">
        <h2>Edit Articles</h2>
    ② {
        <?php if ($errors['warning']) { ?>
            <div class="alert alert-danger"><?= $errors['warning'] ?></div>
        <?php } ?>

    ③ <?php if (!$article['image_file']) { ?>
    ④     Upload image: <input type="file" name="image" class="form-control-file" id="image">
    ⑤     <span class="errors"><?= $errors['image_file'] ?></span>
    ⑥     Alt text: <input type="text" name="image_alt">
        <span class="errors"><?= $errors['image_alt'] ?></span>
    <?php } else { ?>
    ⑦ {
        <label>Image:</label> "
        <p class="alt"><strong>Alt text:</strong> <?= html_escape($article['image_alt']) ?></p>
    ⑧ {
        <a href="alt-text-edit.php?id=<?= $article['id'] ?>">Edit alt text</a>
        <a href="image-delete.php?id=<?= $id ?>">Delete image</a><br><br>
    <?php } ?>
    <?php } ?>

```

# Artículos: Formulario/Mensajes (Parte D)

*cms/admin/article.php*

```
⑨ Title: <input type="text" name="title" value="<?= html_escape($article['title']) ?>">
<span class="errors"><?= $errors['title'] ?></span>
⑩ { Summary: <textarea name="summary"><?= html_escape($article['summary']) ?></textarea>
<span class="errors"><?= $errors['summary'] ?></span>
⑪ Content: <textarea name="content"><?= html_escape($article['content']) ?></textarea>
<span class="errors"><?= $errors['content'] ?></span>
⑫ Author: <select name="member_id">
⑬ <?php foreach ($authors as $author) { ?>
⑭ <option value="<?= $author['id'] ?>"
⑮ <?= ($article['author_id'] == $author['id']) ? 'selected' : ''; ?>
⑯ <?= html_escape($author['forename'] . ' ' . $author['surname']) ?>
</option>
<?php } ?></select>
```



# Artículos: Formulario/Mensajes (Parte D)

*cms/admin/article.php*

```
17 <span class="errors"><?= $errors['author'] ?></span>
    Category: <select name="category_id">
        <?php foreach ($categories as $category) { ?>
            <option value="<?= $category['id'] ?>"
                <?= ($article['category_id'] == $category['id']) ? 'selected' : ''; ?>>
                <?= html_escape($category['name']) ?>
            </option>
        <?php } ?></select>
    <span class="errors"><?= $errors['category'] ?></span>
18 <input type="checkbox" name="published" value="1"
    <?= ($article['published'] == 1) ? 'checked' : '' ?>> Published
    <input type="submit" name="create" value="save" class="btn btn-primary">
</form>
```

## Borrando un artículo

La página para borrar un artículo es como la de borrar una categoría, con un formulario para confirmar que se debe borrar.



# Borrando un artículo

1. La página declara tipos estrictos e incluye los archivos necesarios.
2. La función `filter_input()` de PHP comprueba si hay un nombre llamado id en la cadena de consulta. Si contiene un entero válido, se almacena en `$id`. Si contiene un valor inválido, `$id` almacena false. Si no está presente, `$id` guarda null.

## Borrando un artículo

3. Si no se encuentra un id, el usuario es redirigido a `articles.php` con un mensaje de error.
4. `$article` se inicializa con un valor de false.
5. `$sql` almacena el SQL para obtener el título del artículo, el archivo de la imagen y el id de la imagen.
6. La función `pdo()` ejecuta el SQL y los datos sobre el artículo se recogen y almacenan en `$article`.

## Borrando un artículo

7. Si no se encuentran los datos del artículo, el usuario es redirigido a `articles.php` con un mensaje de error.
8. Una sentencia if comprueba si el formulario ha sido enviado (para confirmar que el artículo debe ser eliminado).
9. Si el formulario ha sido enviado, un bloque try contiene el código que se utilizará para borrar el artículo.

## Borrando un artículo

10. Se inicia una transacción porque la eliminación del artículo puede implicar la ejecución de tres sentencias SQL.
11. Una sentencia if comprueba si el artículo tiene una imagen.
12. Si la tiene, `$sql` contiene la sentencia SQL para establecer la columna `image_id` de la tabla `article` para ese artículo en null, y la función `pdo()` ejecuta la sentencia.

## Borrando un artículo

13. `$sql` contiene entonces el SQL para borrar la imagen de la tabla de imágenes y la función `pdo()` ejecuta esa sentencia.
14. La variable `$path` almacena la ruta a la imagen.
15. Una sentencia if usa la función `file_exists()` de PHP para comprobar si el archivo puede ser encontrado. Si es así, la función `unlink()` de PHP borra el archivo (ver Unidad 5).
16. La variable `$sql` almacena el SQL para borrar el artículo de la tabla article y la función `pdo()` ejecuta la sentencia.

## Borrando un artículo

17. Si no se lanza una excepción en el bloque try, se llama a la función `commit()` del objeto PDO para guardar todos los cambios realizados por las sentencias SQL.
18. El usuario es enviado a `articles.php`, con un mensaje de éxito indicando que el artículo ha sido eliminado.
19. Si se lanza una excepción mientras se borran los datos, se ejecuta el bloque catch.



## Borrando un artículo

- 20. La función `rollback()` del objeto PDO impide que se guarde cualquier cambio en las sentencias SQL.
- 21. La excepción se vuelve a lanzar para que pueda ser manejada por la función de manejo de excepciones por defecto.
- 22. Cuando la página se carga por primera vez, el formulario muestra el título del artículo y un botón de envío para confirmar que debe ser eliminado. El atributo `action` de la etiqueta `<form>` utiliza el id del artículo en la cadena de consulta.

# Borrando un artículo

*cms/admin/article-delete.php*

```
<?php
declare(strict_types = 1);                                // Use strict types
① require_once '../includes/database-connection.php';      // Database connection
  require_once '../includes/functions.php';                // Functions
② $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // Validate id
  if (!$id) {                                              // If no valid id
③     redirect('articles.php', ['failure' => 'Article not found']); // Redirect with error
  }
④ $article = false;                                       // Initialize $article
⑤ $sql = "SELECT a.title, a.image_id, i.file AS image_file FROM article AS a
        LEFT JOIN image AS i ON a.image_id = i.id WHERE a.id = :id;"; // SQL
⑥ $article = pdo($pdo, $sql, [$id])->fetch();             // Get article data
  if (!$article) {                                         // If $article empty
⑦     redirect('articles.php', ['failure' => 'Article not found']); // Redirect
  }
```

# Borrando un artículo

*cms/admin/article-delete.php*

```

⑧ if ($_SERVER['REQUEST_METHOD'] == 'POST') { // If form was submitted
⑨     try { // Try to delete
⑩         $pdo->beginTransaction(); // Start transaction
⑪         if ($image_id) { // If there was an image
⑫             $sql = "UPDATE article SET image_id = null WHERE id = :article_id;"; // SQL
                pdo($pdo, $sql, [$id]); // Remove image from article
⑬             $sql = "DELETE FROM image WHERE id = :id;"; // SQL to delete image
                pdo($pdo, $sql, [$article['image_id']]); // Delete from image table
⑭             $path = '../uploads/' . $article['image_file']; // Set the image path
                if (file_exists($path)) { // If image file exists
⑮                 $unlink = unlink($path); // Delete image file
                }
            }
⑯         $sql = "DELETE FROM article WHERE id = :id;"; // SQL to delete article
                pdo($pdo, $sql, [$id]); // Delete article
⑰         $pdo->commit(); // Commit transaction
⑱         redirect('articles.php', ['success' => 'Article deleted']); // Redirect + success
⑲     } catch (PDOException $e) { // If exception thrown
⑳         $pdo->rollBack(); // Roll back SQL changes
㉑         throw $e; // Re-throw exception
    }
}
?>
```



# Borrando un artículo

*cms/admin/article-delete.php*

22

```
<?php include '../includes/admin-header.php' ?> ...  
    <h2>Delete Article</h2>  
    <form action="article-delete.php?id=<?= $id ?>" method="POST" class="narrow">  
        <p>Click confirm to delete: <i><?= html_escape($article['title']) ?></i></p>  
        <input type="submit" name="delete" value="Confirm" class="btn btn-primary">  
        <a href="articles.php" class="btn btn-danger">Cancel</a>  
    </form> ...  
<?php include '../includes/admin-footer.php'; ?>
```



## Actividad C13.5: Borrando imagen y editando el texto alternativo

Crea una página (crea un nuevo archivo `image-delete.php` dentro de `cms/admin/`) para eliminar una imagen de un artículo utilizando el mismo enfoque que se muestra en este archivo.

A continuación, crea la página para editar el texto alternativo (archivo `alt-text-edit.php`).





## 7. Resumen



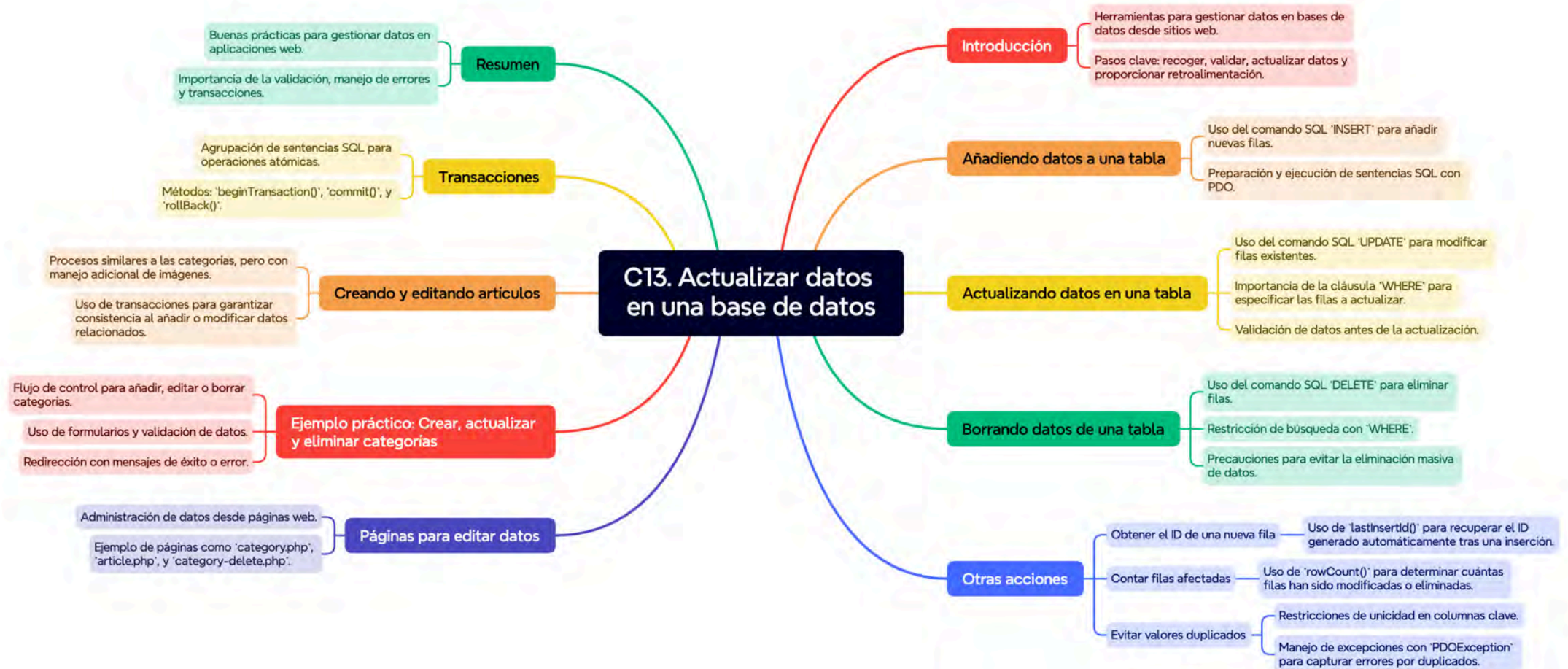
# Resumen

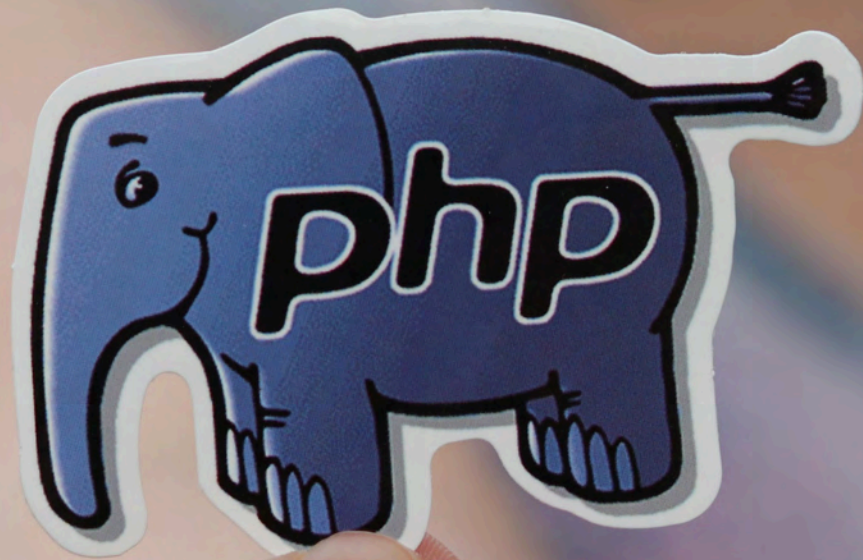
- Los datos de usuario deben recopilarse y validarse antes de añadirse a la base de datos.
- El método `execute()` del objeto `PDOStatement` puede ejecutar sentencias SQL que creen, actualicen o eliminen datos. SQL sólo puede añadir nuevos datos a una tabla cada vez.
- El método `getLastInsertId()` del objeto PDO devuelve el id de una nueva fila cuando se añade a la base de datos.

# Resumen

- El método `rowCount()` del objeto `PDOStatement` devuelve cuántas filas de la tabla se ven afectadas cuando se ejecuta un comando SQL `INSERT`, `UPDATE` o `DELETE`.
- Una transacción ejecuta una serie de sentencias SQL y sólo guarda los cambios si todas ellas se ejecutan sin errores. Si una sentencia SQL rompe una restricción de unicidad, se lanza un objeto `PDOException`. Este contendrá un código de error que describe la causa de la excepción.

# Resumen





## 8. Referencias

# Referencias

- [Actualización de un registro de la base de datos con PHP \(DesarrolloWeb\)](#)
- [Actualizar Datos en una Tabla MySQL con PHP \(Oregoom\)](#)
- [MySQL y PHP con PDO: CRUD \(create, read, update, delete\) \(Parzibyte\)](#)



## Bloque C

### Sitios web basados en bases de datos

#### C13. Actualizar datos en una base de datos

