

17-11-2023

# Trabajar con bases de datos en PHP

## Tarea 03

ROBERTO RODRÍGUEZ JIMÉNEZ  
[roberto.rodjim.1@educa.jcyl.es](mailto:roberto.rodjim.1@educa.jcyl.es)

## Contenido

Tarea online DWES03 .....	3
1.- Descripción de la tarea .....	3
<b>Caso práctico</b> .....	3
Enunciado.....	3
2.- Información de interés.....	6
Recursos necesarios .....	6
Consejos y recomendaciones .....	6
3.- Evaluación de la tarea .....	6
Criterios de evaluación implicados.....	6
¿Cómo valoramos y puntuamos tu tarea? .....	7
RESOLUCIÓN.....	8
Estructura del sitio.....	8
Conexión con la base de datos .....	9
Bootstrap.....	10
listado.php.....	10
Errores .....	12
crear.php .....	14
detalle.php .....	17
update.php.....	18
borrar.php.....	21
includes .....	23
conexion.php .....	23
utilidades.php .....	24
CÓDIGO (se omite Bootstrap y los archivos *.sql) .....	25
listado.php.....	25
crear.php .....	27
detalle.php .....	29
update.php.....	30
borrar.php.....	32
includes .....	33
conexion.php .....	33
utilidades.php .....	35
vistas.....	38
head.php .....	38
footer.php.....	38



## Tarea online DWES03

**Título de la tarea:** Cesta de la compra.

**Unidad 3:** Trabajar con bases de datos en PHP.

**Ciclo formativo y módulo:** Desarrollo de Aplicaciones Web. Módulo DWES.

**Curso académico:** ...

### 1.- Descripción de la tarea

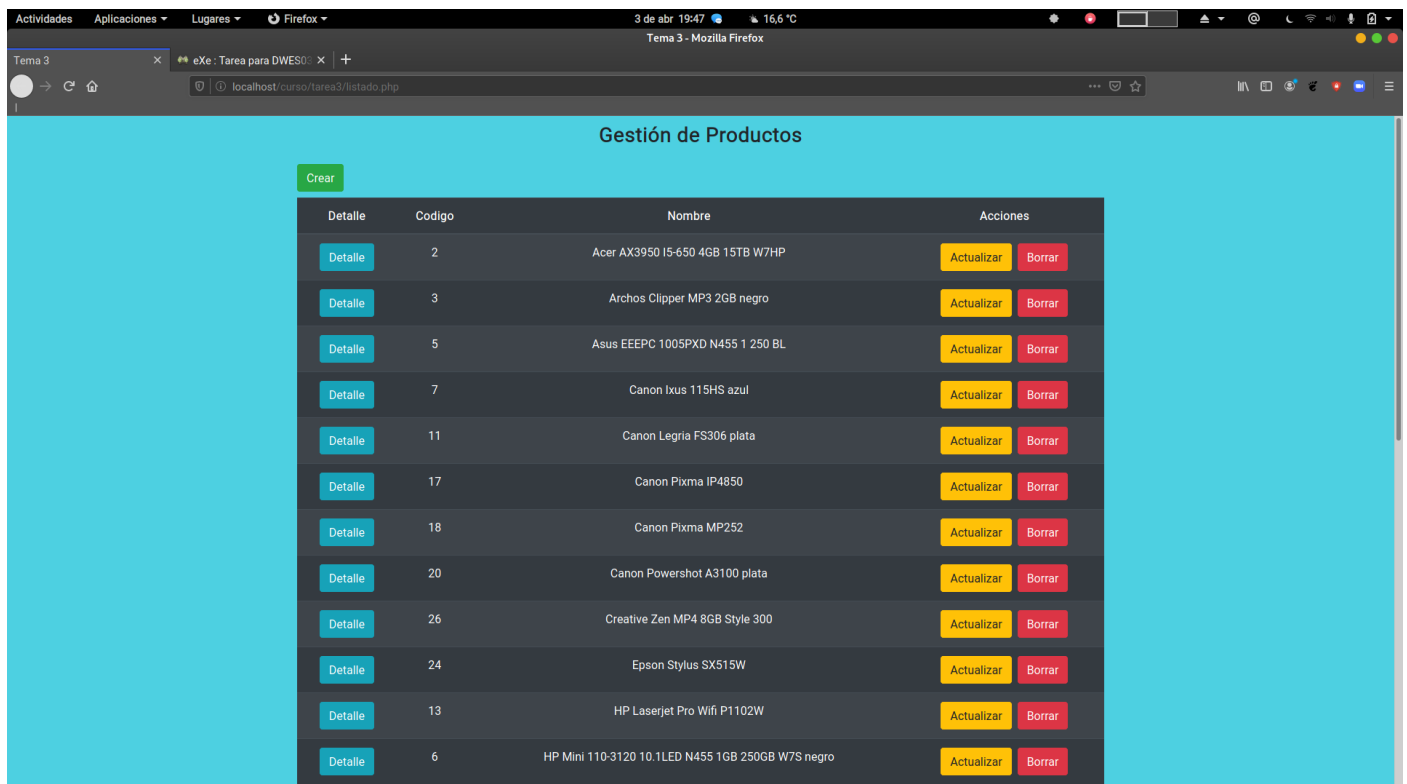
#### Caso práctico

Carlos tiene un amigo con una tienda de informática, que ha decidido vender sus productos por Internet. Carlos piensa que con sus conocimientos de acceder a una base de datos desde la Web podría echarle una mano, y aunque es consciente de que aún le faltan herramientas para que la aplicación sea completamente funcional, se pone a hacer una primera implementación de esta.

#### Enunciado

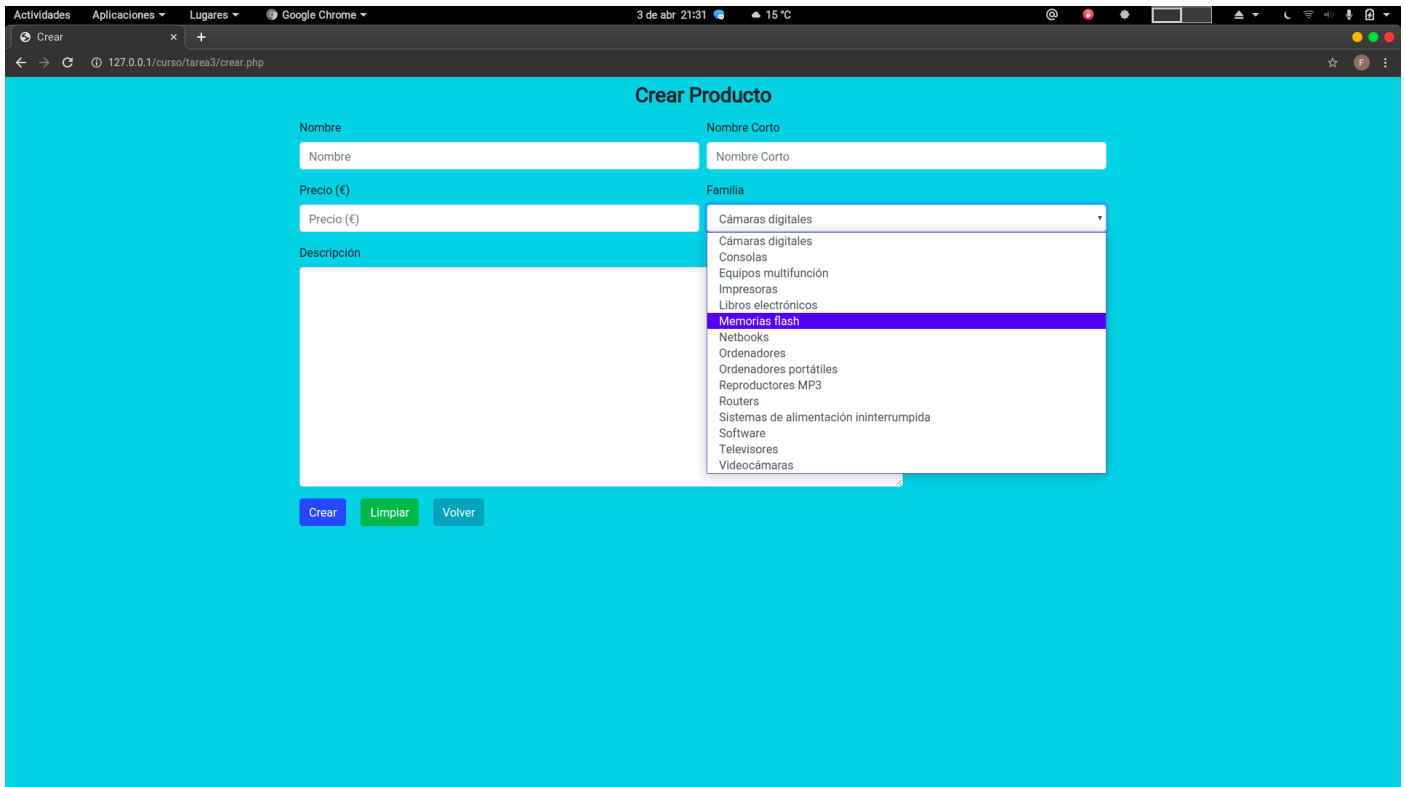
Partiendo de la base de datos 'proyecto' usada en los ejemplos y ejercicios de la unidad, se trata de programar un CRUD (create, read, update, delete) que permita gestionar los registros de la tabla 'productos'. La aplicación se dividirá en 5 páginas:

- ✓ `listado.php`. Mostrará en una tabla los datos código y nombre y los botones para crear un nuevo registro, actualizar uno existente, borrarlo o ver todos sus detalles.



Detalle	Codigo	Nombre	Acciones
<a href="#">Detalle</a>	2	Acer AX3950 I5-650 4GB 15TB W7HP	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	3	Archos Clipper MP3 2GB negro	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	5	Asus EEEPC 1005PXD N455 1 250 BL	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	7	Canon Ixus 115HS azul	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	11	Canon Legria FS306 plata	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	17	Canon Pixma IP4850	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	18	Canon Pixma MP252	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	20	Canon Powershot A3100 plata	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	26	Creative Zen MP4 8GB Style 300	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	24	Epson Stylus SX515W	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	13	HP Laserjet Pro Wifi P1102W	<a href="#">Actualizar</a> <a href="#">Borrar</a>
<a href="#">Detalle</a>	6	HP Mini 110-3120 10.1LED N455 1GB 250GB W7S negro	<a href="#">Actualizar</a> <a href="#">Borrar</a>

- ✓ `crear.php`. Será un formulario para rellenar todos los campos de productos (a excepción del id). Para la familia nos aparecerá un "select" con los nombres de las familias de los productos para elegir uno (lógicamente, aunque mostremos los nombres por formulario enviaremos el código).



**Crear Producto**

Nombre:

Nombre Corto:

Precio (€):

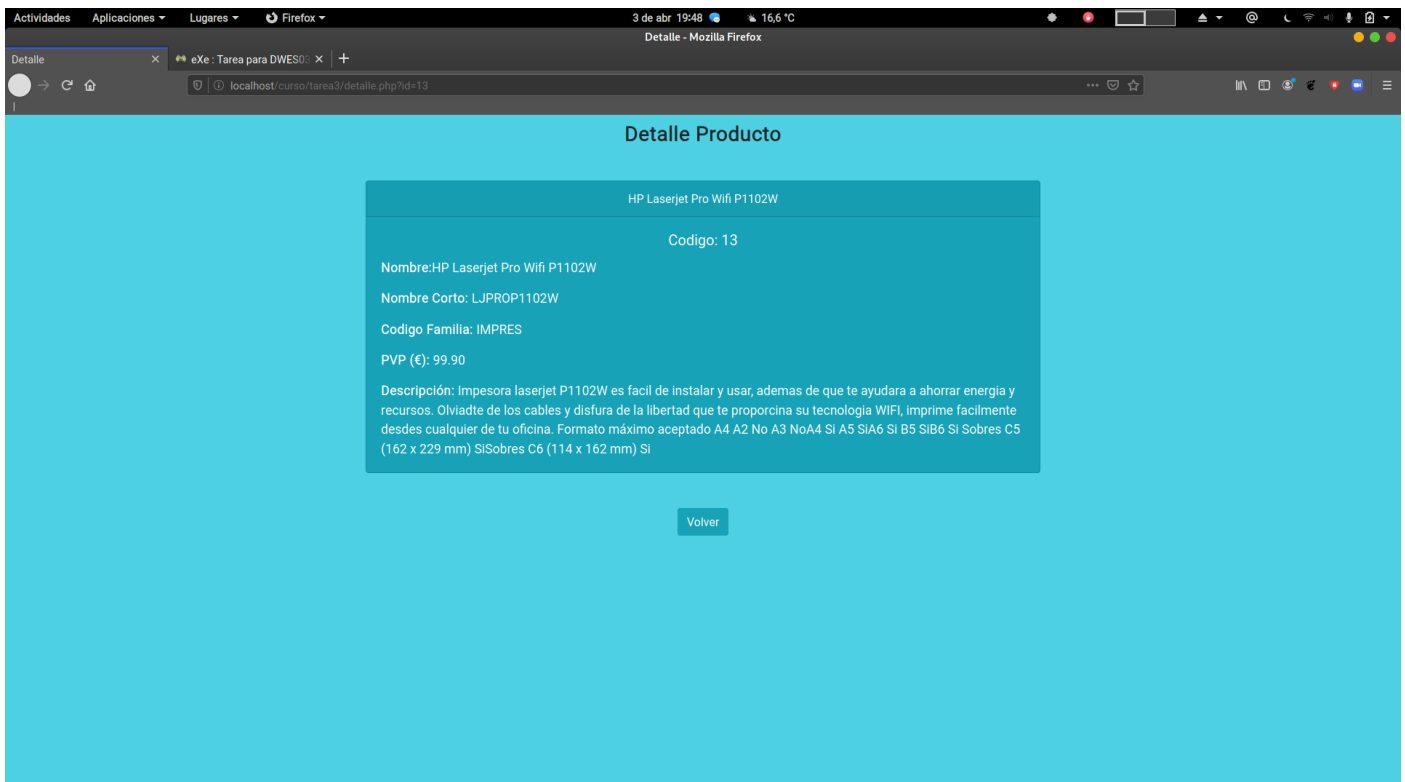
Descripción:

Familia: 

- Cámaras digitales
- Consolas
- Equipos multifunción
- Impresoras
- Libros electrónicos
- Memorias flash**
- Netbooks
- Ordenadores
- Ordenadores portátiles
- Reproductores MP3
- Routers
- Sistemas de alimentación ininterrumpida
- Software
- Televisores
- Videocámaras

[Crear](#) [Limpiar](#) [Volver](#)

- ✓ `detalle.php`. Mostrará todos los detalles del producto seleccionado.



**Detalle Producto**

HP Laserjet Pro Wifi P1102W

Codigo: 13

Nombre: HP Laserjet Pro Wifi P1102W

Nombre Corto: LJP1102W

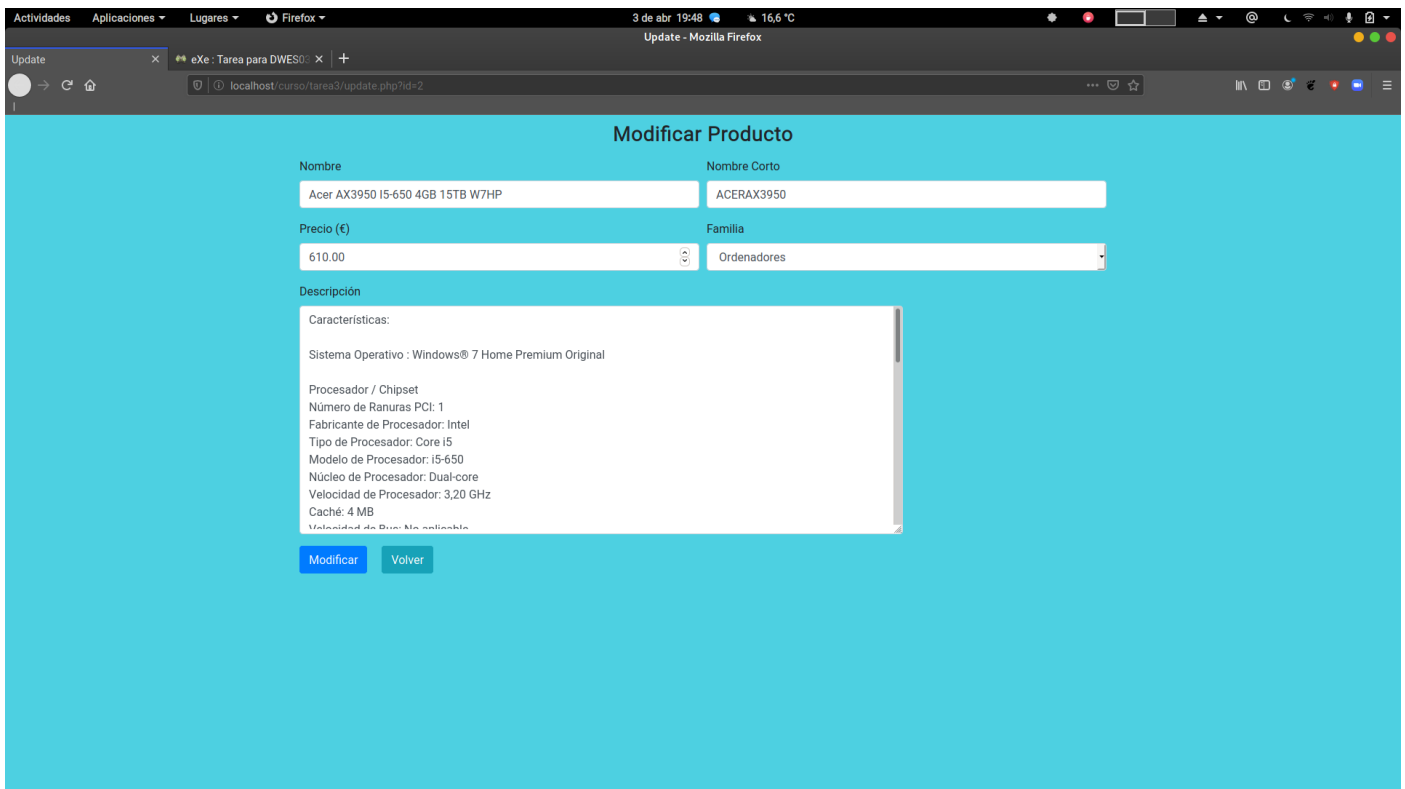
Codigo Familia: IMPRES

PVP (€): 99.90

Descripción: Impresora laserjet P1102W es facil de instalar y usar, ademas de que te ayudara a ahorrar energia y recursos. Olvidate de los cables y disfruta de la libertad que te proporciona su tecnologia WIFI, imprime facilmente desde cualquier de tu oficina. Formato máximo aceptado A4 A2 No A3 NoA4 Si A5 SiA6 Si B5 SiB6 Si Sobres C5 (162 x 229 mm) Si Sobres C6 (114 x 162 mm) Si

[Volver](#)

- ✓ `update.php`. Nos aparecerá un formulario con los campos rellenos con los valores del producto seleccionado desde "`listado.php`" incluido el select donde seleccionamos la familia.

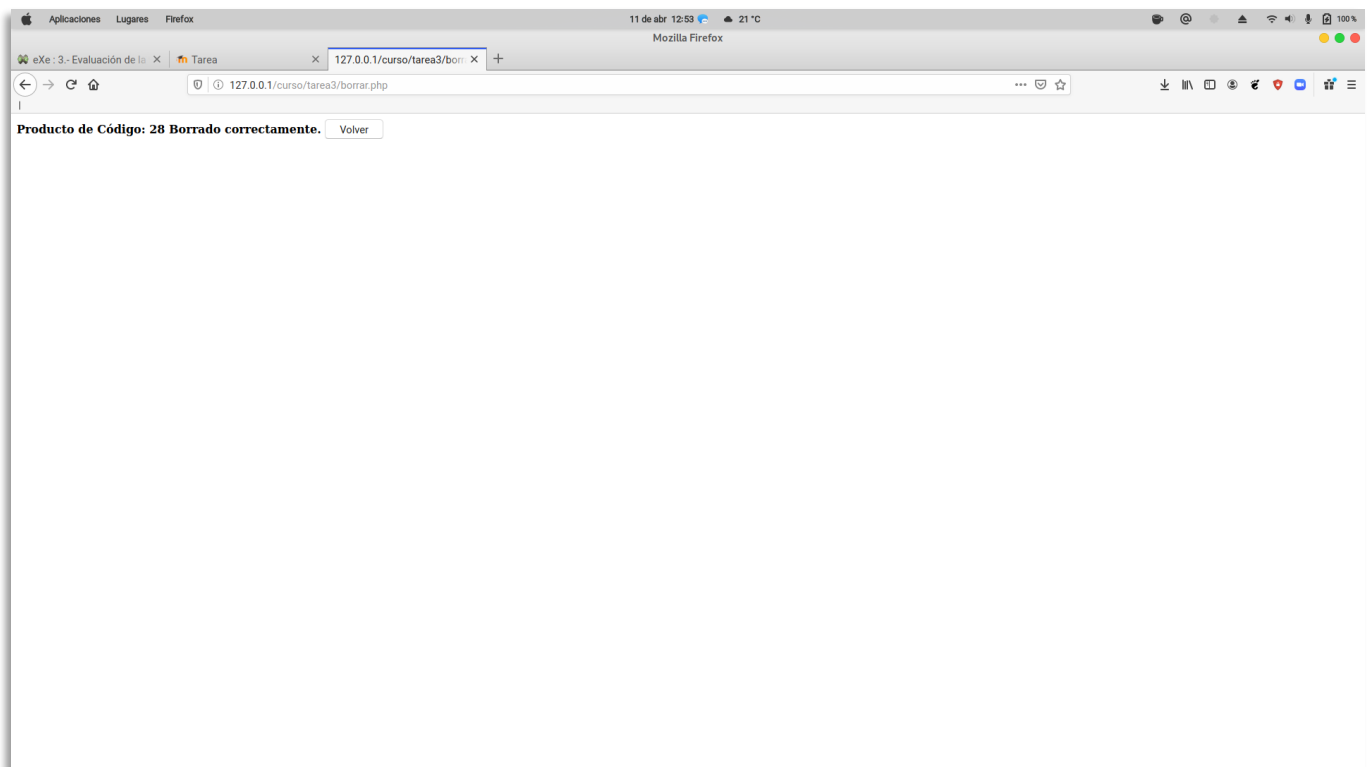


The screenshot shows a web browser window with the address `localhost/cursos/tarea3/update.php?id=2`. The page has a light blue background and is titled "Modificar Producto". It contains a form with the following fields:

- Nombre**: Input field with the value "Acer AX3950 i5-650 4GB 15TB W7HP".
- Nombre Corto**: Input field with the value "ACERAX3950".
- Precio (€)**: Input field with the value "610.00".
- Familia**: A dropdown menu currently showing "Ordenadores".
- Descripción**: A text area containing the following text:  
Características:  
Sistema Operativo : Windows® 7 Home Premium Original  
Procesador / Chipset  
Número de Ranuras PCI: 1  
Fabricante de Procesador: Intel  
Tipo de Procesador: Core i5  
Modelo de Procesador: i5-650  
Núcleo de Procesador: Dual-core  
Velocidad de Procesador: 3,20 GHz  
Caché: 4 MB  
Velocidad de Búsqueda: No seleccionable

At the bottom of the form are two buttons: "Modificar" (blue) and "Volver" (green).

- ✓ `borrar.php`. Será una página php con el código necesario para borrar el producto seleccionado desde "`listado.php`" un mensaje de información y un botón volver para volver a "`listado.php`".



Para acceder a la base de datos se debe usar PDO. Controlaremos y mostraremos los posibles errores. Para los estilos se recomienda usar *Bootstrap*.

Pasaremos el código de producto por "get" tanto para "detalle.php" como para "update.php". Utilizando en el enlace "detalle.php?id=cod". En ambas páginas comprobaremos que esta variable existe, si no redireccionaremos a "listado.php" para esto podemos usar "header('Location:listado.php')"

## 2.- Información de interés

### Recursos necesarios

Ordenador con un entorno AMP, instalado y configurado, un entorno de desarrollo como VSC con las extensiones adecuadas. Navegador con acceso a Internet, para poder consultar el manual online de PHP y poder trabajar online Bootstrap.

### Consejos y recomendaciones

- ✓ Además del manual online de PHP, se recomienda dar libre acceso a Internet para la búsqueda de información.
- ✓ Para no repetir el código de la conexión a la base de datos en cada archivo, se recomienda crear el archivo `conexion.php` y utilizar `require` o `require_once` cada vez que lo necesitamos.
- ✓ Para borrar un producto, crea un formulario con el action apuntando a "borrar.php" y pasa por un campo oculto el código del producto a borrar.
- ✓ Recuerda que en el "option" del HTML "`<option value="v">Nombre</option>`", "v" es el valor que pasamos, "Nombre" es lo que mostramos en la lista desplegable.

## 3.- Evaluación de la tarea

### Criterios de evaluación implicados

- ✓ Se han analizado las tecnologías que permiten el acceso mediante programación a la información disponible en almacenes de datos.
- ✓ Se han creado aplicaciones que establezcan conexiones con bases de datos.
- ✓ Se ha recuperado información almacenada en bases de datos.
- ✓ Se ha publicado en aplicaciones Web la información recuperada.
- ✓ Se han utilizado conjuntos de datos para almacenar la información.
- ✓ Se han creado aplicaciones Web que permitan la actualización y la eliminación de información disponible en una base de datos.
- ✓ Se han utilizado transacciones para mantener la consistencia de la información.
- ✓ Se han probado y documentado las aplicaciones.

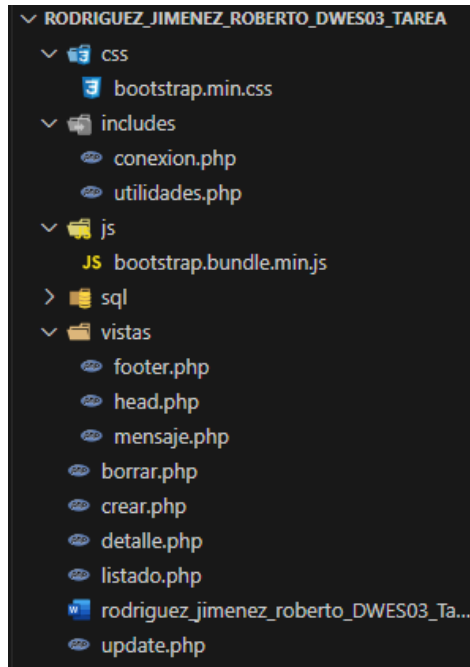
¿Cómo valoramos y puntuamos tu tarea?

Rúbrica de la tarea	
En "listado.php", generar el listado de los productos y los botones funcionando, el botón borrar será el submit de un formulario cuyo action debe ser el indicado	2 puntos
En "crear.php", generar el formulario funcional de creación del producto	2 puntos
En "update.php" cargar el formulario y que sea funcional con todos los campos del producto en cuestión rellenos.	2 puntos
Página "borrar.php" funcional con mensaje y botón para volver.	1 Punto
Página "detalle.php", mostrando el detalle del producto seleccionado y el botón volver funcionando	1 Punto
La lista de selección <b>familia</b> de la página "update.php" tiene seleccionada la familia del producto.	0.75 Puntos
Se utilizan correctamente las excepciones para controlar los posibles errores	0.5 Puntos
Introducir comentarios, legibilidad del código y diseño	0.75 Puntos



## RESOLUCIÓN

### Estructura del sitio



Estructura de archivos y directorios

- **css**
  - *bootstrap.min.css*: archivo css de Bootstrap, para tenerlo en local.
- **includes**: directorio para guardar los archivos php que serán llamados desde otros.
  - *conexion.php*: guarda los datos de la conexión con la base de datos.
  - *utilidades.php*: contiene funciones que son usadas en las páginas.
- **js**
  - *bootstrap.bundle.min.js*: archivo javascript de Bootstrap.
- **sql**
  - *proyecto\_datos.sql*: backup con los datos.
  - *proyecto.sql*: backup con la estructura de la base de datos.
- **vistas**: para los archivos que contienen partes de html.
  - *footer.php*: footer común para cada una de las páginas.
  - *head.php*: header para las páginas.
  - *mensaje.php* muestra un mensaje a modo de confirmación en las operaciones o errores.
- *borrar.php*
- *crear.php*
- *detalle.php*
- *listado.php*
- *update.php*

Para la conexión con la base de datos se usa *PDO* y el driver para *mysql*.

Los datos de configuración y conexión se guardan en el archivo *includes/conexión.php*. En este archivo se guardan los datos y se inicializa el *dsn*.

```
<?php

// La conexión con la base de datos se hace mediante PDO

// Datos de conexión a la base de datos
$host = "localhost";
$port = 3306;
$db = "proyecto";
$user = "gestor";
$pass = "secreto";

// Data Source Name (Nombre de Origen de los Datos)
$dsn = "mysql:host=$host;dbname=$db;charset=utf8mb4";
```

Código con los datos de conexión a la base de datos.

La conexión se realiza desde cada uno de los archivos llamando a *conexión.php/getConexion()*

```
$cnx = getConexion();
```

Llamada a *getConexion()* desde una página.

```
function getConexion($dsn, $user, $pass){

    $dsn = $GLOBALS['dsn'];
    $user = $GLOBALS['user'];
    $pass = $GLOBALS['pass'];

    // Intentar la conexión a la base de datos
    try {
        $cnx = @new PDO($dsn, $user, $pass);
        $cnx->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $cnx;
    } catch (PDOException $e) {
        die(
            . . .
        );
    }
}
```

Código de la función *getConexion()* que establece una conexión con la base de datos.

Para la presentación de las páginas se usa el *framework Bootstrap* en su versión 5.3.

He decidido incorporar los archivos css y js en local para no depender de una conexión a internet, dado que durante el desarrollo de la tarea no siempre tendré acceso a internet.

listado.php

Muestra el listado de los productos, tal y como se ve en la propuesta del ejercicio, en una tabla.

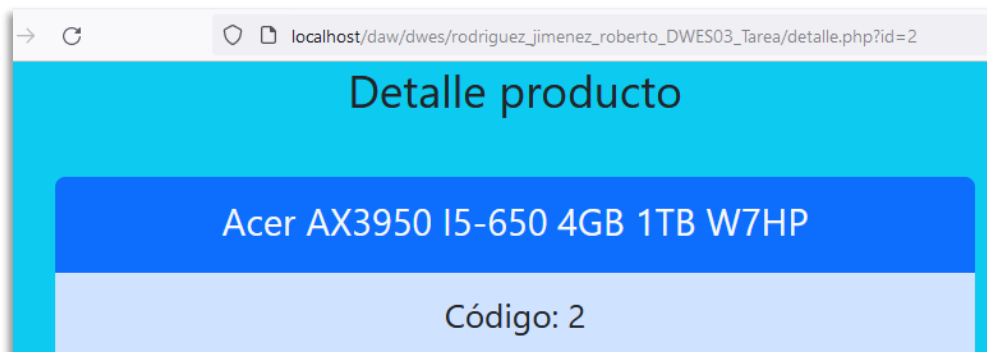
Los datos se cargan en el momento de mostrar las filas, envolviendo el código en un *try-catch* que capture los errores.

```
<tbody class="table-group-divider">
<?php
try {
    $listado = $cnx->query("SELECT id, nombre FROM productos ORDER BY nombre");

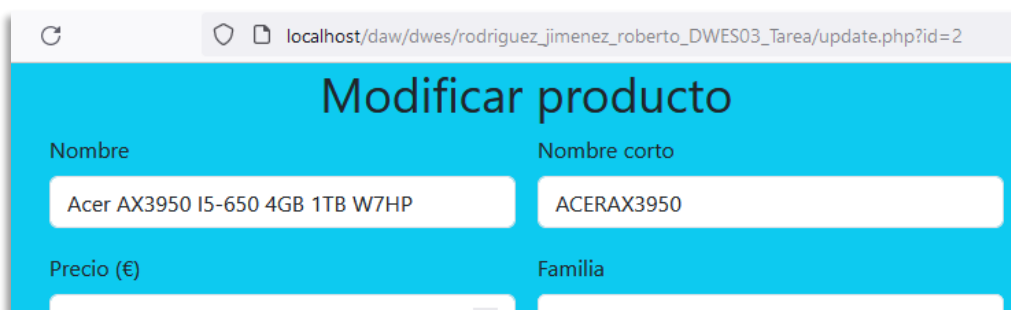
    $producto = $listado->fetch(PDO::FETCH_OBJ);
    while ($producto != null):
?>
        <tr>
            <?php set_error_handler("gestionarError"); ?>
            <td>
                <a href="detalle.php?id=<?=$producto->id?" class="btn btn-success bg-info">Detalle</a>
            </td>
            <td><?=$producto->id ?></td>
            <td><?=$producto->nombre ?></td>
            <td class="d-flex justify-content-around">
                <a href="update.php?id=<?=$producto->id?" class="btn btn-warning">Actualizar</a>
                <form action="borrar.php" method="post">
                    <input type="hidden" name="id" id="id" value="<?=$producto->id?">
                    <button type="submit" class="btn btn-danger">Borrar</button>
                </form>
            </td>
            <?php restore_error_handler(); ?>
        </tr>
    </tr>
<?php
    $producto = $listado->fetch(PDO::FETCH_OBJ);
    endwhile;
} catch (PDOException $e) {
    $mensaje = 'Ocurrió algo inesperado y no se han podido recuperar los articulos';
    require_once('vistas/mensaje.php');
}
?>
</tbody>
```

Código que muestra el resultado de una consulta capturando las excepciones.

Para ver el detalle o actualizar un producto, se pasa el id por GET y, para borrar, usa un formulario que llama a *borrar.php* pasándole el id del producto por POST.



*Detalle del producto*



*Actualizar un producto*

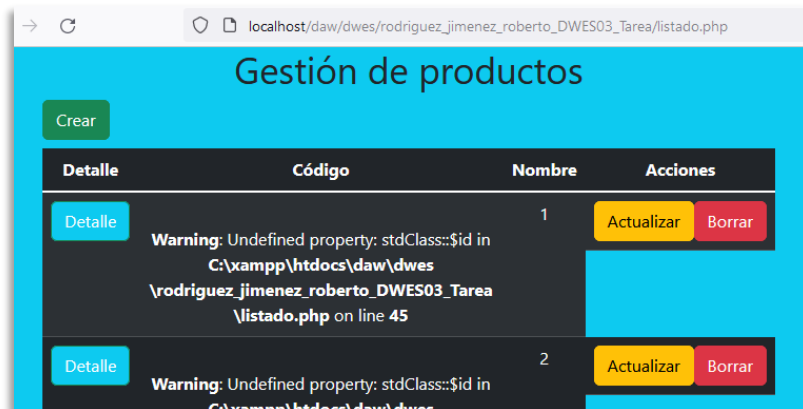


*Detalle del listado*

En el listado puede darse el caso de que la consulta no esté bien formada y se arroje una excepción o un *warning*. Forzamos un error en la declaración de la consulta sql.

```
$listado = $cnx->query("SELECT id nombre FROM productos ORDER BY nombre");
```

Y vemos el resultado



Especificamos el mensaje con `set_error_handler()` para lo cual he creado una función `gestionarError()` en `includes/utilidades.php`



*Personalización de errores mediante `set_error_handler`*

```
$producto = $listado->fetch(PDO::FETCH_OBJ);
while ($producto != null): ?>
<tr>
<?php set_error_handler("gestionarError"); ?>
. . .
<?php restore_error_handler(); ?>
</tr>
<?php
    $producto = $listado->fetch(PDO::FETCH_OBJ);
endwhile;
```

```
$listado = $cnx->query("SELECT id, nombre_ FROM productos ORDER BY nombre");
```



*Error generado en la consulta sql*

Y la gestionamos con *try-catch*



*El mismo error gestionando la excepción*

crear.php

A crear se accede mediante un botón en *listado.php*.



Botón para ir a "crear.php" en la página que muestra el listado

Al ejecutarse el archivo se comprueba si se ha recibido algún dato por POST.

Si POST está vacío, simplemente se muestra el formulario.

El formulario se valida primero en el propio cliente.

El *select* que recoge las familias de productos se obtiene mediante una consulta, capturando los posibles errores mediante *try-catch*.

The screenshot shows a web browser window with the URL 'localhost/daw/dwes/rodriguez\_jimenez\_roberto\_DWES03\_Tarea/crear.php'. The page has a blue header with the title 'Crear producto'. Below the header is a form with several input fields: 'Nombre' (text), 'Nombre corto' (text), 'Precio (€)' (number, value 0), and 'Descripción' (text area). There is a 'Familia' dropdown menu with a list of product families: Cámaras digitales, Consolas, Equipos multifunción, Impresoras, Libros electrónicos, Memorias flash, Netbooks, Ordenadores, Ordenadores portátiles, Reproductores MP3, Routers, Sistemas de alimentación ininterrumpida, Software, Televisores, and Videocámaras. At the bottom of the form are three buttons: 'Crear' (blue), 'Limpiar' (green), and 'Volver' (grey).

Formulario para crear productos

En caso de recibir datos por POST, el primer paso es comprobar que estén todos los datos y que no estén en blanco. Esta operación se realiza en la función *validarDatosCompletos()* dentro del archivo *includes/utilidades.php*.

En caso de que la función encuentre errores, sale de la aplicación mostrando un mensaje:



Crear producto

Ocurrió algún error: Falta algún dato.

Revisa los datos

Pepefunken 13 blanco y negro (máx. 200 caracteres)

PEPE13BN (máx. 50 caracteres)

0

TV

ghghdfghdfghfghghh

Volver

*Error al guardar un dato, en este caso se ha dejado el PVP en 0.*

Si los datos se han recibido correctamente, el siguiente paso es validar el formato.

Igualmente, la función que lo comprueba está en *includes/utilidades.php* y es la función *validarFormato()*.



Crear producto

Ocurrió algún error: Algún nombre no cumple con el patrón.

Revisa los datos

Pepefunken 13 blanco y negro (máx. 200 caracteres)

PEPE-13-BN (máx. 50 caracteres)

49.00

TV

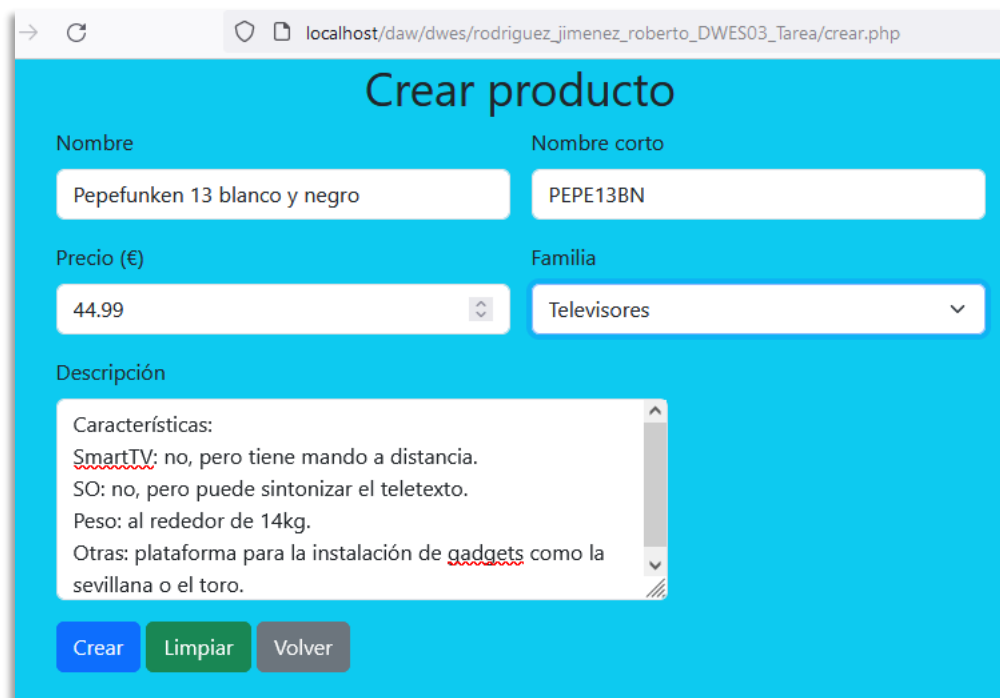
sfgsdgsdfg gsfsdgsg

Volver

*El error está en los guiones del nombre corto.*



Finalmente, si todo es correcto, se ejecuta la consulta y se muestra un feedback de confirmación.



A screenshot of a web browser showing a form titled "Crear producto". The form is set against a blue background. It contains several input fields: "Nombre" with the value "Pepelfunken 13 blanco y negro", "Nombre corto" with "PEPE13BN", "Precio (€)" with "44.99", and a "Familia" dropdown menu set to "Televisores". A "Descripción" text area contains the following text: "Características: SmartTV: no, pero tiene mando a distancia. SO: no, pero puede sintonizar el teletexto. Peso: al rededor de 14kg. Otras: plataforma para la instalación de gadgets como la sevillana o el toro." At the bottom of the form are three buttons: "Crear" (blue), "Limpiar" (green), and "Volver" (grey).

*Datos correctos*



A screenshot of the same "Crear producto" form, but now it displays a success message: "El producto se ha guardado correctamente". Above the input fields, there is a blue button labeled "Volver al listado". The input fields are now empty: "Nombre" and "Nombre corto" are empty text boxes, "Precio (€)" shows "0", and the "Familia" dropdown is also empty. The "Descripción" text area is also empty. The "Crear", "Limpiar", and "Volver" buttons remain at the bottom.

*Pantalla de confirmación*

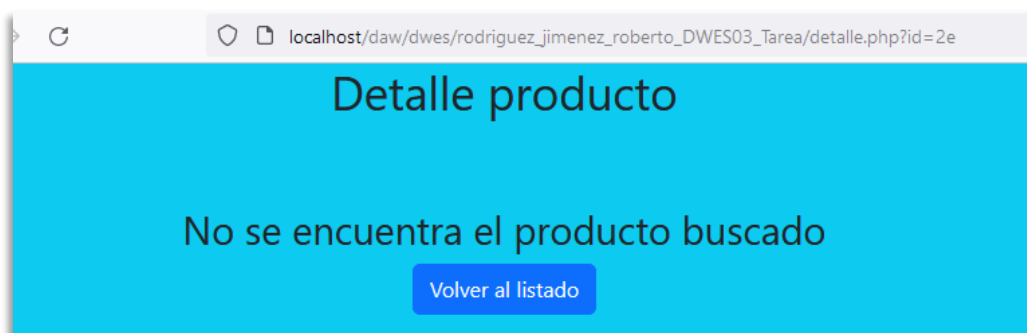
detalle.php

*Pantalla con los detalles de un producto*

El funcionamiento de esta pantalla es bastante simple: se realiza la consulta a la base de datos y se muestran.

La consulta y su tratamiento están envueltas en la cláusula *try-catch* para capturar excepciones. Además, se evita que muestren errores en el caso de que algún campo no esté disponible:

```
<strong>Nombre:</strong> <?= @$caracteristicas->nombre ?>
```

*Error al buscar un id no válido*

La gestión de la excepción se implementa en *mensaje()*, en el archivo *vistas/vistas.php*.

En el caso de que no se indique un id, se redirige a *listado.php*.

```
if (!isset($_GET['id'])) {  
    header('Location: listado.php');  
}
```

update.php



Modificar producto

Nombre: Pepefunken 13 blanco y negro

Nombre corto: PEPE13BN

Precio (€): 44,99

Familia: Televisores

Descripción: Características: SmartTV: no, pero tiene mando a distancia. SO: no, pero puede sintonizar el teletexto. Peso: al rededor de 14kg. Otras: plataforma para la instalación de gadgets como la sevillana o el toro.

Actualizar Volver

*Pantalla de actualización de un producto*

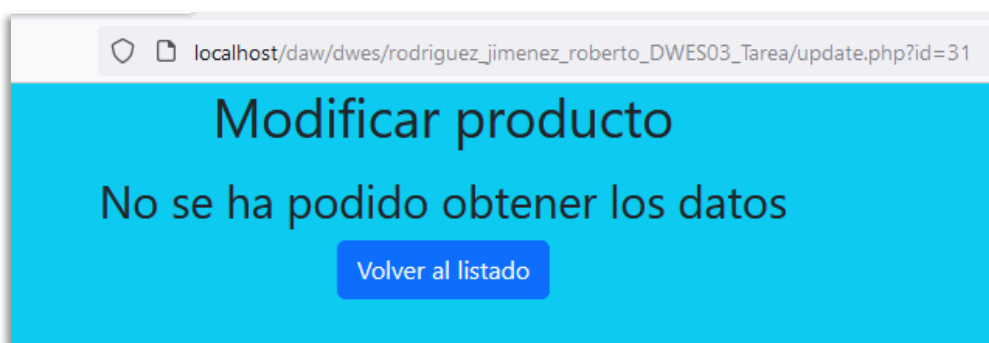
La pantalla que se muestra es la misma que para *crear.php* pero, en este caso, se realiza previamente una consulta a la base de datos para obtener el producto recibido por GET.

Si no se recibe un id, se redirecciona hacia *listado.php*.

Al cargarse el script, se comprueba si existe un id recibido por GET y hay datos en POST.

En caso de cumplirse las condiciones, se siguen los mismos pasos que se dan en *crear.php*

Todo el bloque que muestra la información está incluido en un bloque *try-catch* para capturar las excepciones.

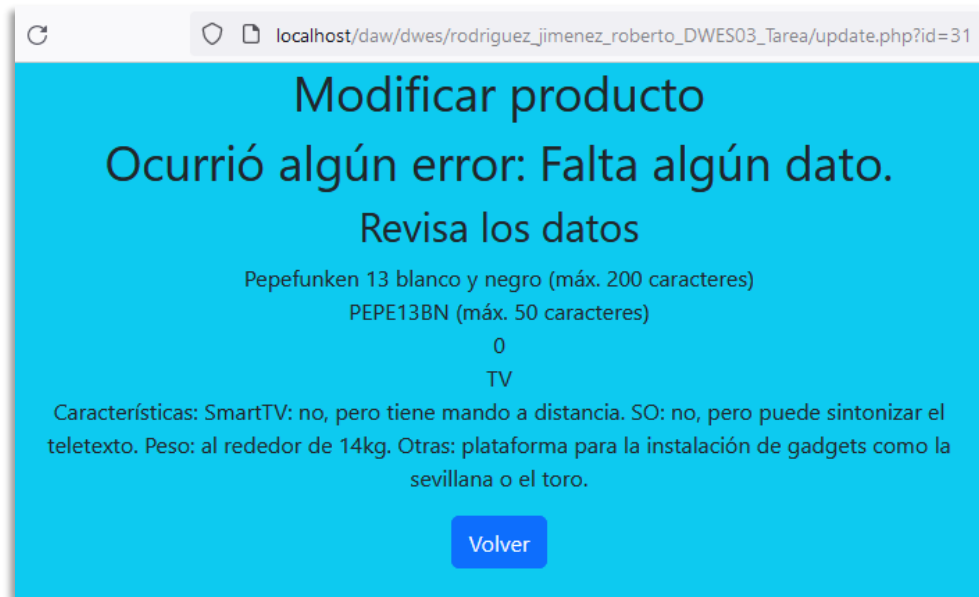


Modificar producto

No se ha podido obtener los datos

Volver al listado

*Feedback tras un error en la conexión a la base de datos*



*Mensaje de error para el formato de los datos: 0€*

El *select* del formulario ejecuta su propia consulta, también dentro de un *try-catch*, y compara el id del producto leído con el recogido en el campo *id* del producto detallado.

```
<select name="familia" id="familia" class="form-select" required>
  <?php
  try {
    // Consulta a la base de datos
    $listado = $cnx->query("SELECT * FROM familias ORDER BY nombre");

    // Obtenemos los productos como objetos
    $familia = $listado->fetch(PDO::FETCH_OBJ);
    while ($familia != null):
      ?>
      <option value="<?= $familia->cod ?>" <?php if ($familia->cod == $caracteristicas->familia)
        echo 'selected'; ?>>
        <?= $familia->nombre ?>
      </option>
    <?php
    $familia = $listado->fetch(PDO::FETCH_OBJ);
    endwhile;
  } catch (PDOException $e) { echo '<option value="">Error</option>'; }
  ?>
</select>
```

Código para cargar el *select* con los datos de las familias de productos.

En el caso de producirse un error, se muestra una única opción en el selector con el texto “Error” y un valor vacío. Forzamos un error en la consulta a la base de datos:

```
$listado = $cnx->query("SELECT * FROM familias_ ORDER BY nombre");
```

The screenshot shows a web browser window with the URL `localhost/daw/dwes/rodriguez_jimenez_roberto_DWES03_Tarea/update.php?id=20`. The page title is "Modificar producto". The form contains the following fields:

- Nombre:** Canon Powershot A3100 plata
- Nombre corto:** PWSHTA3100PT
- Precio (€):** 101,4
- Familia:** A dropdown menu showing "Error" and a placeholder "Seleccione un elemento de la lista."
- Descripción:** La cámara PowerShot A3100 IS, inteligente y compacta, presenta la calidad de imagen de Canon en un cuerpo compacto y ligero para capturar fotografías sin esfuerzo; es tan fácil como apuntar y disparar. Características: 12,1 MP

At the bottom, there are two buttons: "Actualizar" (highlighted in blue) and "Volver".

*Error al cargar los datos de las familias de producto*

Si todo es correcto, se actualiza el producto.

The screenshot shows the same web browser window, but the URL is `localhost/daw/dwes/rodriguez_jimenez_roberto_DWES03_Tarea/update.php?id=31`. The page title is "Modificar producto". A large message "El producto se ha guardado correctamente" is displayed at the top, with a blue button "Volver al listado" below it. The form fields are now:

- Nombre:** Pepefunken 13 Blanco y Negro
- Nombre corto:** PEPE13BN
- Precio (€):** 39,99
- Familia:** Televisores
- Descripción:** Características: SmartTV: no, pero tiene mando a distancia. SO: no, pero puede sintonizar el teletexto. Peso: al rededor de 14kg. Otras: plataforma para la instalación de gadgets como la sevillana o el toro.

At the bottom, there are two buttons: "Actualizar" (highlighted in blue) and "Volver".

*Producto actualizado correctamente.*

borrar.php

Cuando se carga el script *borrar.php* lo primero que se comprueba es que exista un id que se pueda borrar:

```
if (empty($_POST) {  
    header("Location: listado.php");  
}
```

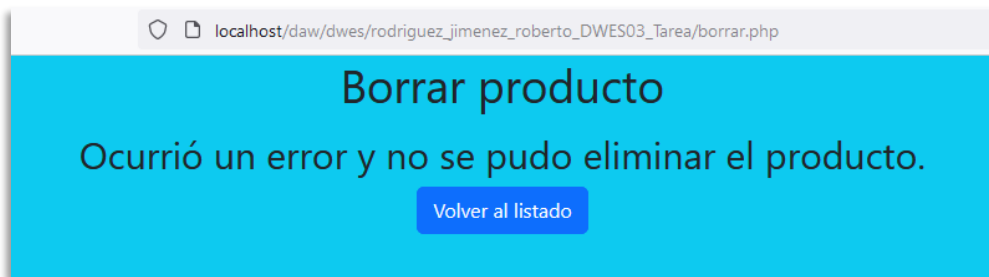
Si se produjese cualquier error, se muestra este y se sale de la aplicación.

Para ello se hace uso de la función *ejecutarConsulta()* en *includes/conexion.php*, que a su vez muestra el mensaje de error a través de *mostrarExcepcion()* que está en el archivo *includes/utilidades.php*.

Forzamos el error:

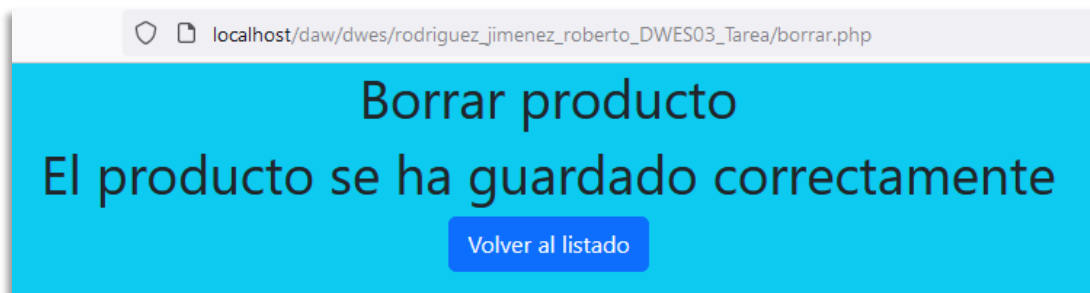
```
$consulta = 'DELETE FROM productos WHERE id = AB';
```

Y tratamos de borrar cualquier producto:



*Mensaje de error al tratar de eliminar un producto incorrecto.*

Recuperamos el código correcto y repetimos la operación:




*Producto borrado correctamente.*

*ejecutarConsulta()* devuelve o *false* dependiendo de si se realizó o no la operación.

A continuación se carga la vista mostrando el mensaje.

```
$consulta = 'DELETE FROM productos WHERE id = ' . $_POST['id'];  
$mensajeOk = 'Producto borrado correctamente.';  
$mensajeKo = 'Ocurrió un error y no se pudo eliminar el producto.';  
$mensaje = (ejecutarConsulta($cnx, $consulta)) ? $mensajeOk : $mensajeKo;  
require_once('vistas/mensaje.php');
```



```
function ejecutarConsulta($cnx, $consulta)  
{  
    try {  
        $cnx->beginTransaction();  
        if ($cnx->exec($consulta)) {  
            $cnx->commit();  
            return true;  
        } else {  
            $cnx->rollBack();  
            return false;  
        }  
    } catch (PDOException $e) {  
        return false;  
    }  
}
```

includes

*conexion.php*

Contiene los datos de conexión con la base de datos y funciones que trabajan directamente con ella.

Variables

```
// Datos de conexión a la base de datos
$host = "localhost";
$port = 3306;
$db = "proyecto";
$user = "gestor";
$pass = "secreto";

// Data Source Name (Nombre de Origen de los Datos)
$dsn = "mysql:host=$host;dbname=$db;charset=utf8mb4";
```

Funciones

```
function getConexion() : PDO
Conecta con la base de datos mediante PDO y devuelve el objeto.

Si la operación falla, se sale de la aplicación.
Usa variables globales para obtener los datos de conexión a la BD.

@global string dsn es nombre del origen de los datos.
@global string user autorizado en la base de datos.
@global string pass clave del usuario.
@return PDO objeto con la conexión a la base de datos.

function ejecutarConsulta($consulta) : boolean
Ejecuta la consulta recibida como parámetro.

@see getConexion() obtiene la conexión con la base de datos.
@param string $consulta que se debe ejecutar.
@return boolean resultado de la transacción
```



*utilidades.php*

Contiene funciones para procesar o validar algunos datos que pueden ser comunes a varias páginas.

```
function mostrarElError($nombre, $nombre_corto, $precio, $familia, $descripcion, $mensaje)
Muestra un mensaje avisando de que ocurrió algún error.
```

Esta función se implementa en utilidades.php en lugar de crear una vista porque puede depender de otras funciones.

```
@param string $nombre del producto.
@param string $nombre_corto nombre corto del producto.
@param string $precio del producto, con formato d*.dd
@param string $familia a la que pertenece el producto. Es texto.
@param string $descripcion del producto.
@param string $mensaje que se muestra.
```

```
function validarDatosCompletos()
Valida que los datos del formulario estén completos.
```

Si falta algún dato o está en blanco, se sale de la aplicación.  
Se asume que conocemos los campos del formulario.

```
function validarFormato() : array[string]
Chequea los datos para asegurarse de que no se cuela algún carácter que no deba estar.

@return array[string] con los datos formateados.
```

```
function gestionarError($nivel, $mensaje)
Personaliza los errores capturados con set_error_handler().
```

```
@param int $nivel de error.
@param string $mensaje que se quiere mostrar.
```

## CÓDIGO (se omite Bootstrap y los archivos \*.sql)

## listado.php

```

1. <?php
2. $titulo = "Gestión de productos";
3. include_once ("vistas/head.php") ;
4. include_once ("includes/conexion.php");
5. include_once ("includes/utilidades.php");
6.
7. $cnx = getConnection();
8. ?>
9.
10. <main class="container">
11.
12.     <nav class="row my-2">
13.         <div class="col">
14.             <a href="crear.php" class="btn btn-success">Crear</a>
15.         </div>
16.     </nav>
17.
18.     <section class="row my-2">
19.         <div class="col">
20.
21.             <table class="table table-dark table-striped table-hover text-center">
22.                 <thead>
23.                     <tr>
24.                         <th scope="col">Detalle</th>
25.                         <th scope="col">Código</th>
26.                         <th scope="col">Nombre</th>
27.                         <th scope="col">Acciones</th>
28.                     </tr>
29.                 </thead>
30.                 <tbody class="table-group-divider">
31.                     <?php
32.                     try {
33.                         // Consulta a la base de datos
34.                         $listado = $cnx->query("SELECT id, nombre FROM productos ORDER BY
35. nombre");
36.
37.                         // Obtenemos los productos como objetos
38.                         $producto = $listado->fetch(PDO::FETCH_OBJ);
39.                         while ($producto != null):
40.                             ?>
41.                             <tr>
42.                                 <?php set_error_handler("gestionarError"); ?>
43.                                 <td>
44.                                     <a href="detalle.php?id=<?= $producto->id ?>"
45.                                         class="btn btn-success bg-info">Detalle</a>
46.                                 </td>
47.                                 <td>
48.                                     <?= $producto->id ?>
49.                                 </td>
50.                                 <td>
51.                                     <?= $producto->nombre ?>
52.                                 </td>
53.                                 <td class="d-flex justify-content-around">
54.                                     <a href="update.php?id=<?= $producto->id ?>"
55.                                         class="btn btn-warning">Actualizar</a>
56.                                     <form action="borrar.php" method="post">
57.                                         <input type="hidden" name="id" id="id"
58.                                             value="<?= $producto->id ?>">
59.                                         <button type="submit" class="btn btn-
60.                                             danger">Borrar</button>
61.                                     </form>
62.                                 </td>

```

```
60.         <?php restore_error_handler(); ?>
61.     </tr>
62.     <?php
63.         $producto = $listado->fetch(PDO::FETCH_OBJ);
64.     endwhile;
65.     } catch (PDOException $e) {
66.         $mensaje = 'Ocurrió algo inesperado y no se han podido recuperar
        los articulos';
67.         require_once('vistas/mensaje.php');
68.     }
69.     ?>
70. </tbody>
71. </class>
72. </table>
73.
74. </div>
75. </section>
76.
77. </main>
78.
79. <?php
80. $cnx = null; // Eliminar la conexión con la base de datos
81. include_once("vistas/footer.php");
82. ?>
```

## crear.php

```

1. <?php
2. $titulo = "Crear producto";
3. include_once("vistas/head.php");
4. include_once("includes/conexion.php");
5. include_once("includes/utilidades.php");
6.
7. $cnx = getConexion();
8.
9. // -- Intentamos crear un producto -----
10. if (!empty($_POST)) {
11.
12.     // -- Verificar los datos -----
13.     // Si no se valida, se sale de la aplicación.
14.     validarDatosCompletos();
15.
16.     // Tenemos todos los campos
17.
18.     // Validamos algunos campos y salimos de la aplicación
19.     // si hay alguno que cumpla con el patrón
20.     $datos = validarFormato();
21.     $nombre = $datos['nombre'];
22.     $nombre_corto = $datos['nombre_corto'];
23.     $pvp = $datos['pvp'];
24.     $familia = $datos['familia'];
25.     $descripcion = $datos['descripcion'];
26.     // -- Fin de verificar los datos -----
27.
28.     // Si se encontró algún error se habría salido de la aplicación, por lo que si
29.     // estamos aquí, es porque es correcto. Guardamos los datos.
30.
31.     $consulta = "INSERT INTO productos (nombre, nombre_corto, descripcion, pvp, familia)
VALUES ('$nombre', '$nombre_corto', '$descripcion', '$pvp', '$familia')";
32.
33.     $mensajeOk = 'Producto guardado correctamente.';
34.     $mensajeKo = 'Ocurrió un error y no se pudo guardar el producto.';
35.     $mensaje = (ejecutarConsulta($consulta)) ? $mensajeOk : $mensajeKo;
36.     require_once('vistas/mensaje.php');
37.
38. }
39. // -- Fin de insertar un producto -----
40. ?>
41.
42. <main class="container">
43.     <form method="post" class="row g-3">
44.         <div class="col-6">
45.             <label for="nombre" class="form-label">Nombre</label>
46.             <input type="text" name="nombre" id="nombre" class="form-control"
placeholder="Nombre" maxlength="200"
required>
47.         </div>
48.         <div class="col-6">
49.             <label for="nombre_corto" class="form-label">Nombre corto</label>
50.             <input type="text" name="nombre_corto" id="nombre_corto" class="form-control"
placeholder="Nombre corto"
maxlength="50" required>
51.         </div>
52.         <div class="col-6">
53.             <label for="precio" class="form-label">Precio (€)</label>
54.             <input type="number" name="precio" id="precio" class="form-control" min="0"
value="0" step=".01" required>
55.         </div>
56.         <div class="col-6">
57.             <label for="familia" class="form-label">Familia</label>
58.             <select name="familia" id="familia" class="form-select" required>
59.                 <option value="" selected></option>
60.             </select>
61.         </div>
62.     </form>

```

```
63.         try {
64.             // Consulta a la base de datos
65.             $listado = $cnx->query("SELECT * FROM familias ORDER BY nombre");
66.
67.             // Obtenemos los productos como objetos
68.             $familia = $listado->fetch(PDO::FETCH_OBJ);
69.             while ($familia != null):
70.                 ?>
71.                 <option value="<?= $familia->cod ?>">
72.                     <?= $familia->nombre ?>
73.                 </option>
74.                 <?php
75.                     $familia = $listado->fetch(PDO::FETCH_OBJ);
76.                 endwhile;
77.             } catch (PDOException $e) {
78.                 echo 'Ocurrió algo inesperado y no se han podido recuperar las familias:
79.                 \n' . $e->getMessage();
80.                 ?>
81.             </select>
82.         </div>
83.         <div class="col-8">
84.             <label for="descripcion" class="form-label">Descripción</label>
85.             <textarea class="form-control" id="descripcion" name="descripcion" rows="5"
86.                 required></textarea>
87.         </div>
88.         <div class="col-12">
89.             <button type="submit" class="btn btn-primary">Crear</button>
90.             <button type="reset" class="btn btn-success">Limpiar</button>
91.             <a href="listado.php" class="btn btn-secondary">Volver</a>
92.         </div>
93.     </form>
94. </main>
95. <?php
96. $cnx = null; // Eliminar el objeto con la conexión a la bd.
97. include_once("vistas/footer.php");
98. ?>
```

## detalle.php

```

1. <?php

83. if (!isset($_GET['id'])) {
84.     header('Location: listado.php');
85. }
86.
87. $titulo = "Detalle producto";
88. include_once("vistas/head.php");
89. include_once("includes/conexion.php");
90. include_once("includes/utilidades.php");
91.
92. $cnx = getConnection();
93. ?>
94.
95. <main class="container">
96.     <section class="row my-5">
97.
98.         <?php
99.
100.             // Intentamos mostrar la información
101.             try {
102.                 // La consulta nos devuelve un solo resultado
103.                 $producto = $cnx->query("SELECT * FROM productos WHERE id = " . $_GET['id']);
104.
105.                 // Al devolver un solo registro, no es necesario recorrer con while
106.                 $caracteristicas = $producto->fetch(PDO::FETCH_OBJ);
107.                 ?>
108.                 <div class="col-12 text-center bg-primary text-light rounded-top-3 m-0 p-3">
109.                     <h2>
110.                         <?= @$caracteristicas->nombre ?>
111.                     </h2>
112.                 </div>
113.                 <div class="col-12 bg-primary-subtle rounded-bottom-3 m-0 p-3">
114.                     <h3 class="text-center">
115.                         Código:
116.                         <?= @$caracteristicas->id ?>
117.                     </h3>
118.                     <br>
119.                     <strong>Nombre:</strong> <?= @$caracteristicas->nombre ?>
120.                     <br>
121.                     <strong>Nombre corto:</strong> <?= @$caracteristicas->nombre_corto ?>
122.                     <br>
123.                     <strong>Familia:</strong> <?= @$caracteristicas->familia ?>
124.                     <br>
125.                     <strong>PVP (€):</strong> <?= @$caracteristicas->pvp ?>
126.                     <br>
127.                     <strong>Descripción:</strong>
128.                     <?= @$caracteristicas->descripcion ?>
129.                 </div>
130.
131.                 <div class="col-12 text-center my-3">
132.                     <a href='listado.php' class='btn btn-primary'>Volver</a>
133.                 </div>
134.
135.                 <?php
136.             } catch (PDOException $e) {
137.                 $mensaje = 'No se encuentra el producto buscado';
138.                 require_once('vistas/mensaje.php');
139.             }
140.             ?>
141.         </section>
142.     </main>
143.
144.     <?php
145.     $cnx = null; // Eliminar la conexión con la base de datos
146.     include_once("vistas/footer.php");
147.     ?>

```

## update.php

```

1. <?php
2. if (!isset($_GET['id'])) {
3.     header('Location: listado.php');
4. }
5.
6. $titulo = "Modificar producto";
7. include_once("vistas/head.php");
8. include_once("includes/conexion.php");
9. include_once("includes/utilidades.php");
10.
11. $cnx = getConexion();
12.
13. // -- Intentamos actualizar un producto -----
14. if (!empty($_POST) && isset($_GET['id'])) {
15.
16.     $id = $_GET['id']; // Guardar el id del producto para usarlo en la consulta
17.
18.     // -- Verificar los datos -----
19.     // Si no se valida, se sale de la aplicación.
20.     validarDatosCompletos();
21.
22.     // Tenemos todos los campos
23.
24.     // Validamos algunos campos y salimos de la aplicación
25.     // si hay alguno que cumpla con el patrón
26.     $datos = validarFormato();
27.     $nombre = $datos['nombre'];
28.     $nombre_corto = $datos['nombre_corto'];
29.     $pvp = $datos['pvp'];
30.     $familia = $datos['familia'];
31.     $descripcion = $datos['descripcion'];
32.
33.     // -- Fin de verificar los datos -----
34.
35.     // Si se encontró algún error se habría salido de la aplicación, por lo que si
36.     // estamos aquí, es porque es correcto. Guardamos los datos.
37.
38.     $consulta = @"UPDATE productos SET nombre = '$nombre', nombre_corto = '$nombre_corto',
descripcion = '$descripcion', pvp = '$pvp', familia = '$familia' WHERE productos.id =
$id";
39.
40.     $mensajeOk = 'Producto actualizado correctamente.';
41.     $mensajeKo = 'Ocurrió un error y no se pudo actualizar el producto.';
42.     $mensaje = (ejecutarConsulta($consulta)) ? $mensajeOk : $mensajeKo;
43.     require_once('vistas/mensaje.php');
44.
45. }
46.
47. // -- Fin de insertar un producto -----
48.
49. // -- Formulario -----
50. // El formulario se muestra solo si tenemos los datos
51. try {
52.
53.     // Obtener los datos del producto a partir del id
54.     // La consulta nos devuelve un solo resultado
55.     $producto = @$cnx->query("SELECT * FROM productos WHERE id = " . $_GET['id']);
56.
57.     // Al devolver un solo registro, no es necesario recorrer con while
58.     $caracteristicas = $producto->fetch(PDO::FETCH_OBJ);
59.
60.     ?>
61.     <main class="container">
62.         <form method="post" class="row g-3">
63.             <div class="col-6">
64.                 <label for="nombre" class="form-label">Nombre</label>

```

```

65.         <input type="text" name="nombre" id="nombre" value="<?= @$caracteristicas-
>nombre ?>" class="form-control"
66.             placeholder="Nombre" maxlength="200" required>
67.     </div>
68.     <div class="col-6">
69.         <label for="nombre_corto" class="form-label">Nombre corto</label>
70.         <input type="text" name="nombre_corto" id="nombre_corto" value="<?=
@$caracteristicas->nombre_corto ?>"
71.             class="form-control" placeholder="Nombre corto" maxlength="50"
required>
72.     </div>
73.     <div class="col-6">
74.         <label for="precio" class="form-label">Precio (€)</label>
75.         <input type="number" name="precio" id="precio" value="<?=
@$caracteristicas->pvp ?>" class="form-control"
76.             min="0" value="0" step=".01" required>
77.     </div>
78.     <div class="col-6">
79.         <label for="familia" class="form-label">Familia</label>
80.         <select name="familia" id="familia" class="form-select" required>
81.             <?php
82.                 try {
83.                     // Consulta a la base de datos
84.                     $listado = $cnx->query("SELECT * FROM familias ORDER BY nombre");
85.
86.                     // Obtenemos los productos como objetos
87.                     $familia = $listado->fetch(PDO::FETCH_OBJ);
88.                     while ($familia != null):
89.                         ?>
90.                         <option value="<?= $familia->cod ?>" <?php if ($familia->cod
== @$caracteristicas->familia)
91.                             echo 'selected'; ?>>
92.                             <?= $familia->nombre ?>
93.                         </option>
94.                         <?php
95.                             $familia = $listado->fetch(PDO::FETCH_OBJ);
96.                         endwhile;
97.                     } catch (PDOException $e) {
98.                         echo '<option value="">Error</option>';
99.                     }
100.                    ?>
101.                </select>
102.            </div>
103.            <div class="col-8">
104.                <label for="descripcion" class="form-label">Descripción</label>
105.                <textarea class="form-control" id="descripcion" name="descripcion"
rows="5"
106.                    required><?= @$caracteristicas->descripcion ?></textarea>
107.            </div>
108.            <div class="col-12">
109.                <input type="hidden" name="id" id="id" value="<?= $_GET['id'] ?>">
110.                <button type="submit" class="btn btn-primary">Actualizar</button>
111.                <a href="listado.php" class="btn btn-secondary">Volver</a>
112.            </div>
113.        </form>
114.    </main>
115.
116.    <?php
117.    } catch (PDOException $e) {
118.        $mensaje = 'No se ha podido obtener los datos';
119.        require_once('vistas/mensaje.php');
120.    } ?>
121.
122.    <?php
123.    $cnx = null; // Eliminar el objeto con la conexión a la bd.
124.    include_once("vistas/footer.php");
125.    ?>

```



## borrar.php

```
2. <?php
3. if (empty($_POST)) {
4.     header("Location: listado.php");
5. }
6.
7. $titulo = "Borrar producto";
8. include_once("vistas/head.php");
9. include_once("includes/conexion.php");
10. include_once("includes/utilidades.php");
11.
12. $cnx = getConexion();
13.
14. // Intentar ejecutar la consulta
15. $consulta = 'DELETE FROM productos WHERE id = ' . $_POST['id'];
16. $mensajeOk = 'Producto borrado correctamente.';
17. $mensajeKo = 'Ocurrió un error y no se pudo eliminar el producto.';
18. $mensaje = (ejecutarConsulta($consulta)) ? $mensajeOk : $mensajeKo;
19. require_once('vistas/mensaje.php');
20.
21. $cnx = null; // Eliminar la conexión con la base de datos
22. include_once("vistas/footer.php");
23. ?>
```

includes

*conexion.php*

```

148.  <?php
149.
150.  // La conexión con la base de datos se hace mediante PDO
151.
152.  // Datos de conexión a la base de datos
153.  $host = "localhost";
154.  $port = 3306;
155.  $db = "proyecto";
156.  $user = "gestor";
157.  $pass = "secreto";
158.
159.  // Data Source Name (Nombre de Origen de los Datos)
160.  $dsn = "mysql:host=$host;dbname=$db;charset=utf8mb4";
161.
162.  /**
163.   * Conecta con la base de datos mediante PDO y devuelve el objeto.
164.   *
165.   * Si la operación falla, se sale de la aplicación.
166.   * Usa variables globales para obtener los datos de conexión a la BD.
167.   *
168.   * @author Roberto Rodríguez <roberto.rodjim.1@educa.jcyl.es>
169.   * @since 2023.11.30
170.   *
171.   * @global string dsn es nombre del origen de los datos.
172.   * @global string user autorizado en la base de datos.
173.   * @global string pass clave del usuario.
174.   * @return PDO objeto con la conexión a la base de datos.
175.   */
176.  function getConexion()
177.  {
178.      // Intentar la conexión a la base de datos
179.      $dsn = $GLOBALS['dsn'];
180.      $user = $GLOBALS['user'];
181.      $pass = $GLOBALS['pass'];
182.
183.      try {
184.          $cnx = @new PDO($dsn, $user, $pass);
185.          $cnx->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
186.          return $cnx;
187.      } catch (PDOException $e) {
188.          die(
189.              "<div class='container my-3'>
190.              <div class='row justify-content-center'>
191.                  <div class='col text-center'>
192.                      <h2>No se ha podido conectar con la base de datos</h2>
193.                  </div>
194.              </div>
195.              </div>"
196.          );
197.      }
198.  }
199.
200.  /**
201.   * Ejecuta la consulta recibida como parámetro.
202.   *
203.   * @author Roberto Rodríguez <roberto.rodjim.1@educa.jcyl.es>
204.   * @since 2023.11.30
205.   * @see getConexion() obtiene la conexión con la base de datos.
206.   *
207.   * @param string $consulta que se debe ejecutar.
208.   * @return boolean resultado de la transacción
209.   */
210.  function ejecutarConsulta($consulta)
211.  {

```

## DESARROLLO DE APLICACIONES WEB

```
212.         // Obtener la conexión
213.         $cnx = getConnection();
214.
215.         try {
216.             $cnx->beginTransaction();
217.             if ($cnx->exec($consulta)) {
218.                 $cnx->commit();
219.                 return true;
220.             } else {
221.                 $cnx->rollBack();
222.                 return false;
223.             }
224.         } catch (PDOException $e) {
225.             return false;
226.         }
227.     }
```

## DESARROLLO WEB EN ENTORNO SERVIDOR

*utilidades.php*

```

1. <?php
2.
3. /**
4.  * Muestra un mensaje avisando de que ocurrió algún error.
5.  *
6.  * Esta función se implementa en utilidades.php en lugar de crear
7.  * una vista porque puede depender de otras funciones.
8.  *
9.  * @author Roberto Rodríguez <roberto.rodjim.1@educa.jcyl.es>
10.  * @since 2023.11.29
11.  *
12.  * @param string $nombre nombre del producto.
13.  * @param string $nombre_corto nombre corto del producto.
14.  * @param string $precio del producto, con formato d*.dd
15.  * @param string $familia a la que pertenece el producto. Es texto.
16.  * @param string $descripcion del producto.
17.  * @param string $mensaje que se muestra.
18.  */
19. function mostrarElError($nombre, $nombre_corto, $precio, $familia,
    $descripcion, $mensaje)
20. {
21.     // Mostramos un mensaje de error si lo hubiera
22.     echo "<div class='container'>
23.         <div class='row justify-content-center'>
24.             <div class='col text-center'>
25.                 <h1>Ocurrió algún error: $mensaje</h1>
26.                 <h2>Revisa los datos</h2>
27.                 <ul style='list-style: none' class='mx-0 px-0'>
28.                     <li>$nombre (máx. 200 caracteres)</li>
29.                     <li>$nombre_corto (máx. 50 caracteres)</li>
30.                     <li>$precio</li>
31.                     <li>$familia</li>
32.                     <li>$descripcion</li>
33.                 </ul>
34.                 <a href='crear.php' class='btn btn-primary'>Volver</a>
35.             </div>
36.         </div>
37.     </div>";
38. }
39.
40. /**
41.  * Valida que los datos del formulario estén completos.
42.  *
43.  * Si falta algún dato o está en blanco, se sale de la aplicación.
44.  * Se asume que conocemos los campos del formulario.
45.  *
46.  * @author Roberto Rodríguez <roberto.rodjim.1@educa.jcyl.es>
47.  * @since 2023.11.30
48.  */
49. function validarDatosCompletos()
50. {
51.     if (
52.         !((isset($_POST['nombre']) && !empty($_POST['nombre'])) &&
53.           (isset($_POST['nombre_corto']) &&
54.            !empty($_POST['nombre_corto'])) &&
55.           (isset($_POST['precio']) && !empty($_POST['precio'])) &&
56.           (isset($_POST['familia']) && !empty($_POST['familia'])) &&
57.           (isset($_POST['descripcion']) && !empty($_POST['descripcion'])))

```

```

58.         ) {
59.             die(
60.                 mostrarElError(
61.                     $_POST['nombre'],
62.                     $_POST['nombre_corto'],
63.                     $_POST['precio'],
64.                     $_POST['familia'],
65.                     $_POST['descripcion'],
66.                     "Falta algún dato."
67.                 )
68.             );
69.         }
70.     }
71.
72. /**
73.  * Chequea los datos para asegurarse de que no se cuele algún
74.  * carácter que no deba estar.
75.  *
76.  * @author Roberto Rodríguez <roberto.rodjim.1@educa.jcyl.es>
77.  * @since 2023.11.30
78.  * @return array[string] con los datos formateados.
79.  */
80. function validarFormato()
81. {
82.     // Tenemos todos los campos
83.     // El nombre corto lo convertimos en mayúsculas y eliminamos los
    espacios en blanco
84.     $nombre = stripslashes(trim($_POST['nombre']));
85.     $nombre_corto = stripslashes(
86.         trim(
87.             strtoupper(
88.                 str_replace(" ", "", $_POST['nombre_corto'])
89.             )
90.         )
91.     );
92.     $precio = stripslashes(trim($_POST['precio']));
93.     $familia = stripslashes(trim($_POST['familia']));
94.     $descripcion = stripslashes(trim($_POST['descripcion']));
95.
96.     // Comprobamos que el precio tenga el punto
97.     if (!preg_match('/\./', $precio)) {
98.         $precio .= ".00";
99.     }
100.
101.     if (
102.         !(
103.             preg_match("/^[A-ZÁÉÍÓÚá-záéíóú0-9 ]{1,200}$/", $nombre) &&
104.             preg_match("/^[A-Z0-9]{1,50}$/", $nombre_corto) &&
105.             preg_match('/^[0-9]*\.[0-9]{2}$/', $precio)
106.         )
107.     ) {
108.         die(
109.             mostrarElError(
110.                 $nombre,
111.                 $nombre_corto,
112.                 $precio,
113.                 $familia,
114.                 $descripcion,
115.                 "Algún nombre no cumple con el patrón."
116.             )
117.         );
118.     }
119.
120.     return array(
121.         'nombre' => $nombre,

```

```
122.         'nombre_corto' => $nombre_corto,
123.         'pvp' => $precio,
124.         'familia' => $familia,
125.         'descripcion' => $descripcion
126.     );
127. }
128.
129. /**
130.  * Personaliza los errores capturados con set_error_handler().
131.  *
132.  * @author Roberto Rodríguez <roberto.rodjim.1@educa.jcyl.es>
133.  * @since 2023.11.30
134.  * @param int $nivel de error.
135.  * @param string $mensaje que se quiere mostrar.
136.  */
137. function gestionarError($nivel, $mensaje)
138. {
139.
140.     switch ($nivel) {
141.         case E_WARNING:
142.             echo 'Error';
143.             break;
144.         default:
145.             echo 'Error no especificado.';
146.     }
147.
148. }
```

vistas

*head.php*

```
1. <!DOCTYPE html>
2. <html lang="es">
3. <head>
4.     <meta charset="UTF-8">
5.     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6.     <title><?=$titulo?></title>
7.     <link rel="stylesheet" href="./css/bootstrap.min.css">
8. </head>
9. <body class="bg-info">
10.
11. <h1 class="text-center"><?=$titulo?></h1>
```

*footer.php*

```
1. <script src="./js/bootstrap.bundle.min.js"></script>
2. </body>
3. </html>
```

*mensaje.php*

```
2. <div class='container'>
3.     <div class='row justify-content-center'>
4.         <div class='col text-center'>
5.             <h1><?=$mensaje?></h1>
6.             <a href='listado.php' class='btn btn-primary'>Volver al listado</a>
7.         </div>
8.     </div>
9. </div>
```