

CREACIÓN DE UN SITIO WEB

“LIBRERÍA CALTAN”

Proyecto realizado por: Joaquín Barragán Benítez.

Curso: 2º Desarrollo de Aplicaciones Multiplataforma.

ÍNDICE

1. Resumen.....	5
1.1. Motivación.....	5
2. Introducción.....	7
3. Objetivos y características del proyecto.....	9
4. Finalidad del proyecto.....	15
5. Medios materiales usados.....	17
5.1. Especificaciones hardware.....	17
5.2. Visual Studio Code.....	17
5.3. HTML.....	18
5.4. CSS.....	19
5.5. Bootstrap.....	19
5.6. PHP.....	20
5.7. JavaScript.....	20
5.7.1. JQuery.....	21
5.8. XAMPP.....	21
5.9. phpMyAdmin.....	22
5.10. Git.....	22
5.11. GitHub.....	23
6. Planificación del proyecto.....	25
6.1. Estructura del proyecto.....	26
6.2. Base de datos.....	29
6.3. Login.....	31
6.4. CRUD de libros.....	33

6.5.	CRUD de clientes.....	35
6.6.	CRUD de usuarios.....	36
6.6.1.	Roles de usuarios.....	37
6.7.	Muestra de libros en la página principal.....	38
6.8.	Gestión de las reservas de libros.....	39
6.9.	Filtros en la página principal.....	42
6.10.	Nuevo cliente.....	43
6.11.	Exportar datos a Excel.....	45
7.	Fase de pruebas.....	47
8.	Posibles mejoras.....	57
9.	Conclusiones del proyecto.....	61
10.	Apéndices.....	63
10.1.	Inicio de sesión.....	63
10.2.	Cifrado de contraseñas.....	66
10.3.	Ejemplo de CRUD.....	67
11.	Referencias bibliográficas.....	73

1. Resumen.

En este proyecto vamos a realizar un sitio web de una librería en el cual podremos reservar cualquier libro disponible que se encuentre en la red. También tenemos un área de empleados con la que con un usuario y contraseña podremos añadir, modificar y borrar los libros que se muestran en la página web, otro apartado para crear usuarios (empleados), otro apartado para insertar clientes y por ultimo otro apartado más para mostrar las reservas que tenemos de los clientes.

1.1. Motivación.

Las razones por las que se decidió realizar este proyecto son principalmente dos motivos:

- La realización de un sitio web para aprender y complementar la formación del Grado Superior de Desarrollo de Aplicaciones Multiplataforma dado que este ciclo está más orientado al desarrollo de aplicaciones de escritorio y móvil por lo que este proyecto sería todo un reto para mí.
- La lectura es una de mis pasiones por lo que este proyecto se encargaría de gestionar de una manera sencilla cualquier librería y con ello también ayuda a promocionarlas además de contribuir a la cultura y a la afición por la lectura.

2. Introducción.

En este proyecto vamos a realizar un sitio web de una librería física en el cual hay dos perspectivas: para los clientes y para los empleados:

- Los clientes: Los clientes verán los libros disponibles y podrán reservar los libros que quieran.
- Los empleados: Los empleados tienen una sección en el sitio web (Área de empleados) en donde podrán administrar los libros que aparecen en el sitio web, gestionar las reservas de los clientes, gestionar los clientes y la gestión de las cuentas de los empleados. Para este paso vamos a realizar un CRUD de libros, reservas, clientes y empleados.

Para este proyecto mi idea es que una librería pueda tener un sitio web y que ellos mismos puedan gestionarla de una manera simple de esta manera las librerías tendrán un sitio en internet donde mejorar y posicionar la marca de estas librerías y poder mejorar las ventas.

3. Objetivos y características del proyecto.

Los objetivos de este proyecto son los siguientes:

- Ampliar el conocimiento en el desarrollo web y complementar la formación del Grado Superior de Desarrollo de Aplicaciones Multiplataforma.
- Tener un sitio web prototipo para poder utilizarla para cualquier librería.
- Tener un sitio web que está orientada tanto para el cliente como para la gestión de los libros que se muestran en la web, la gestión de las reservas de los libros y la gestión de los clientes.
- Utilizar los conocimientos que se han adquirido con mi formación y con las prácticas de empresa para realizar un sitio web que ayude a una afición personal.

Las características del proyecto podemos resumirlo de la siguiente manera:

- En la página principal de la librería podemos ver los libros que dispone la librería con un botón para reservar el libro, además también podemos buscar libros y podemos filtrar libros por categoría.

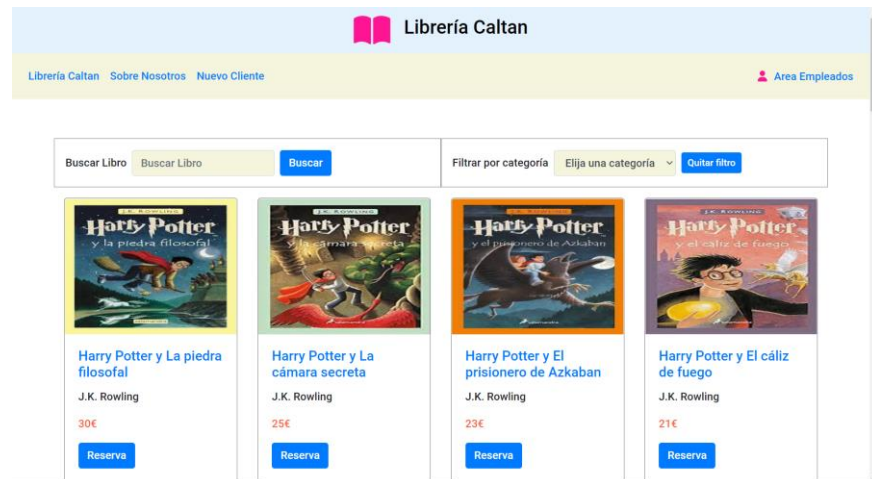


Figura 1: Vista de la página principal del sitio web.

- Cuando pulsamos el botón de reservar nos muestra una página en el que podemos ver más detalles del libro y un formulario para que los clientes nos introduzcan su información de contacto.



Figura 2: Vista de la página para reservar libros

- En la página sobre nosotros vamos a introducir información de la librería y vemos mi motivación para realizar este proyecto.



Figura 3: Vista de la página nosotros en la que viene información

- En la página nuevo introducimos un formulario para que un cliente se pueda inscribir y rellenar sus datos de contacto para que de esta manera podamos enviar noticias o información interesante.

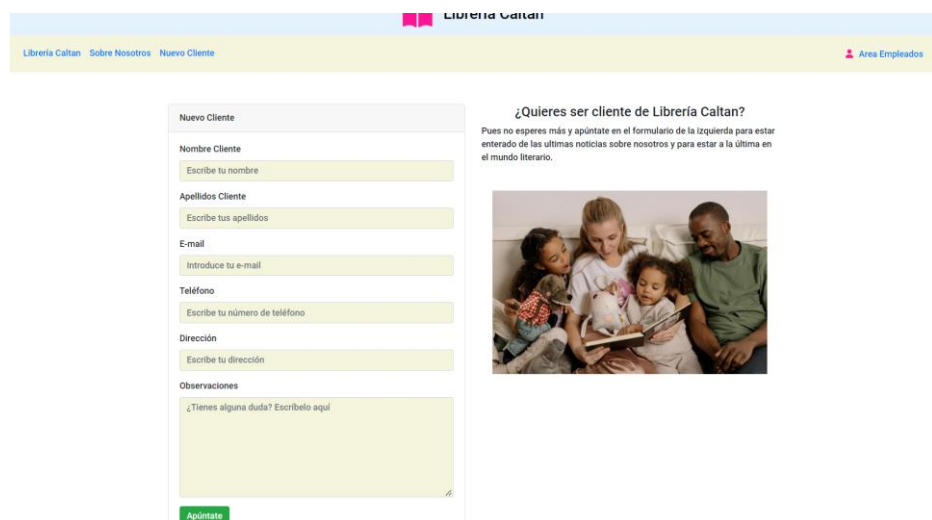


Figura 4: Vista de la página de nuevos clientes para que se puedan inscribir

- En la sección área de empleados se nos muestra una página de inicio de sesión en la que tendremos que introducir el usuario (empleado) y contraseña para acceder a la administración de los libros, usuarios y reservas de la librería.

Formulario de inicio de sesión con el título "Iniciar Sesión". Incluye campos para "Usuario" y "Contraseña", ambos con el placeholder "Introduce tu usuario" y "Introduce tu contraseña" respectivamente. Hay un botón azul "Entrar" y un enlace "Volver" en azul.

Figura 5: Vista de la página de inicio de sesión del área empleados

- En la sección de libros tenemos dos partes bien diferenciadas: un formulario para introducir o modificar los libros que se muestran en el sitio web y una tabla que se muestran todos los libros y donde podremos seleccionar un libro en concreto para después poder modificar o borrar un libro en concreto. También podremos exportar la tabla de libros a Excel.

La página de gestión de libros tiene una barra de navegación superior con enlaces: Libros, Clientes, Usuarios, Reservas, Ir a la web, y Joaquín/Cerrar sesión. A la izquierda hay un formulario para crear o editar libros con campos para ID, Nombre, Autor, Precio, Stock, Categoría (menú desplegable), Descripción y Imagen (botón "Seleccionar archivo" o "Ninguno...o selec."). A la derecha hay un botón "Exportar tabla a excel" y una tabla con los libros.

ID	Nombre	Autor	Precio	Categoría	Stock	Imagen	Acciones
29	Harry Potter y La piedra filosofal	J.K. Rowling	30	ficcion	8		Seleccionar Borrar
35	Harry Potter y La cámara secreta	J.K. Rowling	25	ficcion	3		Seleccionar Borrar
36	Harry Potter y El prisionero de Azkaban	J.K. Rowling	23	ficcion	10		Seleccionar Borrar
37	Harry Potter y El cáliz de fuego	J.K. Rowling	21	ficcion	12		Seleccionar Borrar
38	Harry Potter y La orden del fenix	J.K. Rowling	27	ficcion	11		Seleccionar Borrar
39	Harry Potter y El misterio del principe	J.K. Rowling	25	ficcion	15		Seleccionar Borrar
40	Harry Potter y Las reliquias de la muerte	J.K. Rowling	26	ficcion	10		Seleccionar Borrar
50	África. Tormenta	H.	16	accion	20		Seleccionar

Figura 6: Vista de la página de gestión de libros del área empleados

- En la sección de usuarios tenemos dos partes bien diferenciadas: un formulario para introducir o modificar los empleados y una tabla que se muestran todos los empleados y donde podremos seleccionar un usuario para después poder modificar o borrar un usuario en concreto.

Libros Clientes Usuarios Reservas Ir a la web joaquin(Cerrar sesión)

Usuarios

ID

ID

Usuario

Usuario

Contraseña

Contraseña

Rol

Elige un rol

Agregar

Modificar

Cancelar

ID	Usuario	Rol	Acciones	
11	joaquin	administrador	<div>Seleccionar</div>	<div>Borrar</div>
14	fatima	administrador	<div>Seleccionar</div>	<div>Borrar</div>
16	ana	empleado	<div>Seleccionar</div>	<div>Borrar</div>
18	carmen	empleado	<div>Seleccionar</div>	<div>Borrar</div>

Figura 7: Vista de la página de gestión de usuarios del área empleados

- En la sección de clientes tenemos dos partes bien diferenciadas: un formulario para introducir o modificar los clientes y una tabla que se muestran todos los clientes y donde podremos seleccionar un usuario para después poder modificar o borrar un cliente en concreto. También podremos exportar la tabla de clientes a Excel.

Libros Clientes Usuarios Reservas Ir a la web joaquin(Cerrar sesión)

Clientes

Exportar tabla a excel

ID

ID

Nombre Cliente

Nombre del cliente

Apellidos Cliente

Apellidos del cliente

E-mail

e-mail

Teléfono

Teléfono

Dirección

Dirección

Observaciones

Observaciones

Agregar

Modificar

Cancelar

ID	Nombre	Apellidos	Correo	Teléfono	Dirección	Observaciones	Acciones
1	ana	barragan	joaquin@example.com	987654321	madrid	consulta	<div>Seleccionar</div> <div>Borrar</div>
4	joaquin	barragan	joaquin@example.com	234567890	madrid	pregunta sobre libro	<div>Seleccionar</div> <div>Borrar</div>
8	carmen	blanco	carmen@example.com	890123456	badajoz	sobre algo	<div>Seleccionar</div> <div>Borrar</div>

Figura 8: Vista de la página de gestión de clientes de área de empleados

LIBRERÍA CALTAN

13

- En la sección de reservas tenemos una tabla con todas las reservas que se obtienen de los clientes además si pulsamos en el botón de completado además de borrar la reserva reducimos el stock de ese libro concreto. También podremos exportar la tabla de reservas a Excel.

Libros Clientes Usuarios Reservas Ir a la web joaquin(Cerrar sesión)

Exportar tabla a excel

ID	Nombre Cliente	Apellidos Cliente	e-mail	Teléfono	Dirección	Cantidad	Stock	Libro	Acción
16	prueba	adios	ana@example.com	987654321	madrid	4	3	Harry Potter y La cámara secreta	Borrar
17	hola	barragan	joaquin@example.com	234589876	madrid	3	0	Harry Potter y La piedra filosofal	Borrar
20	prueba	blanco	joaquin@example.com	234567890	madrid	2	10	Harry Potter y El prisionero de Azkaban	Completado Borrar
23	prueba	barragan	joaquin@example.com	987654321	madrid	2	0	Harry Potter y La piedra filosofal	Borrar
24	Juan	González	juan@example.com	654321098	galicia	1	17	El oro de Esparta. Libro de Acción	Completado Borrar

Figura 9: Vista de la página de reservas de los clientes

4. Finalidad del proyecto.

El proyecto tiene como finalidad el realizar un aprendizaje sobre lenguajes para el desarrollo web porque de esta manera podremos complementar la formación profesional de Desarrollo de Aplicaciones Multiplataforma aprendiendo y reforzando lenguajes como HTML, CSS, PHP o SQL (para bases de datos).

Otra finalidad de este proyecto es realizar un sitio web para librerías que sea fácil de utilizar para los clientes y también para los empleados para que estos puedan gestionar toda la información que se muestra en la web de una forma sencilla.

Por último, este proyecto sirve como un buen prototipo para poder realizar sitios web para otras librerías y poder reutilizar esta web para futuros proyectos dado que se podría personalizar mucho.

5. Medios materiales usados.

Los medios materiales usados (tanto hardware como software) que se han utilizado para este proyecto son las siguientes:

5.1. Especificaciones hardware.

Vamos a utilizar para este proyecto mi ordenador personal dado que este sitio web solo se va a ejecutar en local y no en un servidor web real. En la siguiente imagen vemos en una tabla las especificaciones técnicas del portátil.

Especificaciones del PC	
Procesador:	Intel i5-1035G1 1,20 GHz.
Memoria RAM:	12 GB.
Sistema Operativo:	Windows 10 Home
Almacenamiento:	SSD 500 GB.

Figura 10: Tabla de las especificaciones hardware

5.2. Visual Studio Code.

Se ha utilizado esta aplicación para implementar el código del sitio web. *Visual Studio Code* es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Este editor de código se puede personalizar de muchas maneras gracias a la amplia variedad de extensiones. Las extensiones que se han utilizado para este proyecto son las siguientes:

- *Bootstrap 4, Font awesome 4, Font Awesome 5 Free & Pro snippets*: Esta extensión es un snippet que se utiliza para crear fragmentos de código del framework Bootstrap de una forma más sencilla.

- *The Powerful Bootstrap*: Esta extensión es un snippet que se utiliza para crear fragmentos de código del framework Bootstrap de una forma más sencilla como los formularios que vemos en el proyecto.
- *PHP Inteliphense*: Esta extensión se va a utilizar para que cuando escribamos código PHP el editor de código Visual Studio Code nos marque donde nos hemos equivocado y nos muestre una pista para que podamos intuir donde nos hemos equivocado.



Figura 11: Logo de Visual Studio Code

5.3. HTML.

HTML es un lenguaje de marcado para que podamos elaborar páginas web. Este lenguaje se considera una referencia de software para la elaboración de páginas web que define la estructura básica y un código (HTML) para la definición de contenidos para las páginas web como texto, imágenes, videos, entre otros.



Figura 12: Logo de HTML

5.4. CSS.

CSS es un lenguaje de diseño gráfico para que podamos definir y se puedan crear las presentaciones de documentos estructurados escritos en un lenguaje de marcado. Este lenguaje se utiliza para establecer el diseño visual de los documentos web, e interfaces de usuario que se escriben en HTML.



Figura 13: Logo de CSS

5.5. Bootstrap.

Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto que se utiliza para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con el que se consigue realizar la maquetación web de una forma sencilla. Se basa en HTML y CSS, así como extensiones de JavaScript adicionales. En nuestro caso se ha utilizado la versión 4.5 y aunque la versión más moderna es la 5 se ha elegido la anterior porque es más que suficiente para este proyecto.



Figura 14: Logo de Bootstrap

5.6. PHP.

PHP es un lenguaje de programación de uso general que se adapta especialmente al desarrollo web. El código PHP se suele procesar en un servidor web por un intérprete PHP implementado como un módulo, un daemon o como un ejecutable de interfaz de entrada común (CGI). Vamos a utilizar este lenguaje de programación para la back-end de nuestro sitio web.



Figura 15: Logo de PHP

5.7. JavaScript.

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos. Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y en páginas web dinámicas y JavaScript del lado del servidor. En nuestro caso se ha utilizado JavaScript para que podamos utilizar las ventanas modales de la librería Bootstrap.



Figura 16: Logo de JavaScript

5.7.1. JQuery.

JQuery es una biblioteca multiplataforma de JavaScript con el que se puede simplificar la manera con la que interactuamos con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Utilizamos esta biblioteca porque la versión que se utiliza nos pide incluirla y se ha utilizado esta biblioteca para realizar el filtrado de libros por categoría que explicaremos más adelante.



Figura 17: Logo de JQuery

5.8. XAMPP.

XAMPP es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos *MySQL*, el servidor web *Apache* y los intérpretes para lenguajes de script *PHP* y *Perl*. A partir de la versión 5.6.15, *XAMPP* se cambió la base de datos *MySQL* por *MariaDB*, un fork de *MySQL* con licencia *GPL*. Con este programa instalamos el servidor Apache que es el que utilizamos para poder ejecutar el sitio web, el intérprete de *PHP* y el administrador de bases de datos *MySQL* (o *MariaDB*).



Figura 18: Logo de XAMPP

5.9. phpMyAdmin.

phpMyAdmin es una herramienta escrita en *PHP* con la intención de que se pueda manejar la administración de *MySQL* a través de páginas web, utilizando un navegador web. Actualmente se puede crear y eliminar bases de datos, crear, eliminar y alterar tablas, se puede borrar, editar y añadir campos, también se puede ejecutar cualquier sentencia *SQL*, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y podemos tenerlo en 72 idiomas. Se encuentra disponible bajo la licencia *GPL* Versión 2. Este gestor de base de datos se encuentra instalado por defecto en *XAMPP*.



Figura 19: Logo de phpMyAdmin

5.10. Git.

Git es un software de control de versiones, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tenemos un gran número de archivos de código fuente. Su propósito es que llevemos un registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código. Utilizamos *Git* para que tener un respaldo de nuestro proyecto y que además podamos tener varias versiones del proyecto por si tenemos que volver hacia atrás en nuestro proyecto.



Figura 20: Logo de Git

5.11. GitHub.

GitHub es una plataforma de desarrollo colaborativo para que podamos alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. GitHub lo utilizamos para que alojemos este proyecto, la memoria y la base de datos de nuestro proyecto.



Figura 21: Logo de GitHub

6. Planificación del proyecto.

En este punto se va a redactar todo el proyecto en profundidad. Podremos ver cada uno de los puntos más relevantes de este proyecto, que son:

- Estructura del proyecto.
- Base de datos.
- Login.
- CRUD de libros.
- CRUD de clientes.
- CRUD de usuarios.
- Roles de usuarios.
- Muestra de libros en la página principal.
- Gestión de las reservas de libros.
- Filtros en la página principal.
- Nuevo Cliente.
- Exportar datos a Excel.

6.1. Estructura del proyecto.

En este punto vamos a hablar de cómo tenemos estructurado el proyecto. En la siguiente imagen vemos como es la estructura principal del proyecto.

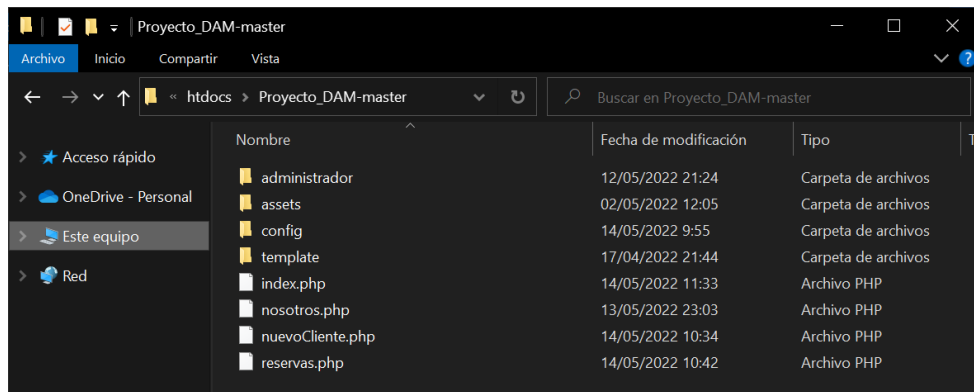


Figura 22: Estructura de carpetas y archivos del proyecto

Los archivos que se ven corresponden con las vistas del sitio web accesibles para cualquier persona, es decir, las que no corresponden al área de empleados. En la imagen anterior se han visto carpetas que se van a explicar a continuación:

- Carpeta *administrador*: En esta carpeta tenemos los archivos a relacionados con las vistas del área de empleados como la pantalla de login, la gestión de libros, clientes, usuarios y reservas. También tenemos las carpetas exports (aquí tenemos los archivos que realizan las tablas para Excel) y template (aquí tenemos las plantillas de header y footer para que solo tengamos que escribir este código una vez dado que esta parte del código se repetirá varias veces en diferentes páginas del área de empleados). En la siguiente imagen vemos la estructura de la carpeta administrador.

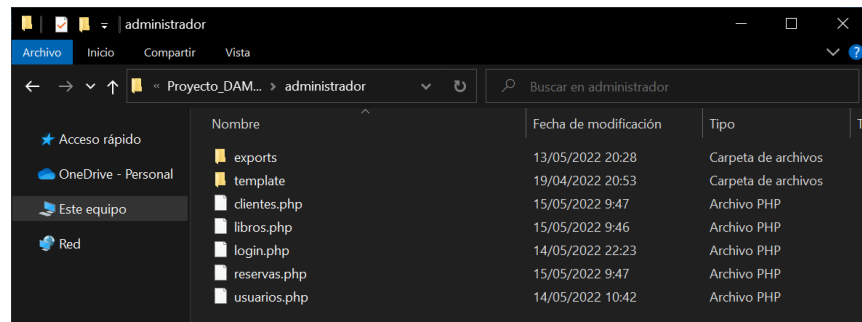


Figura 23: Estructura de carpetas y archivos de la carpeta administrador

- Carpeta *assets*: En esta carpeta tenemos todos los activos que necesitamos para el proyecto. Aquí tendremos dos carpetas: *css* donde tenemos el archivo de estilo de nuestro proyecto que hemos llamado *style* y la carpeta *img* donde alojaremos todas las imágenes que tendrá nuestro proyecto. En la siguiente imagen vemos la estructura de la carpeta *assets*.

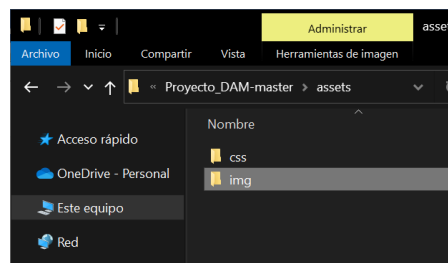


Figura 24: Estructura de carpetas en la carpeta assets

- Carpeta *config*: En esta carpeta tendremos todo lo relacionado con la parte de modelo de base de datos, como la conexión a la base de datos, cerrar la sesión y la inserción, actualización, lectura y borrado de datos en nuestra base de datos. En la siguiente imagen vemos la estructura de la carpeta config.

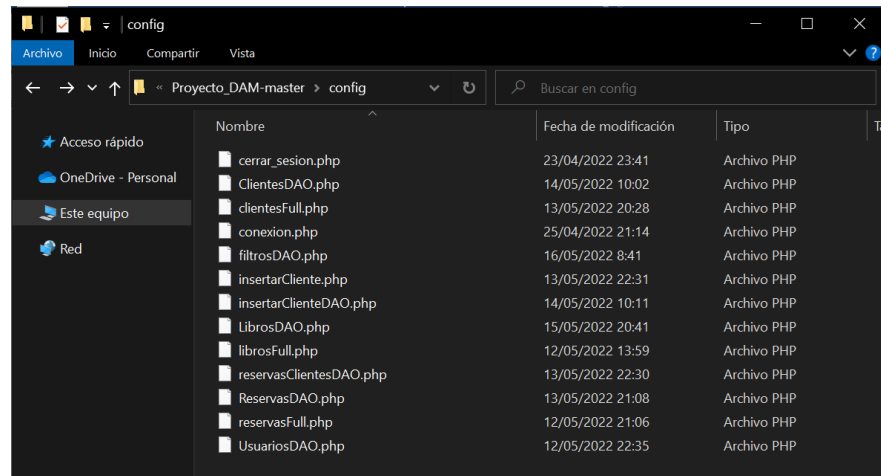


Figura 25: Estructura de archivos de la carpeta config

- Carpeta *template*: En esta carpeta tenemos las plantillas header y footer de las vistas del sitio web, donde excluimos las vistas del área de empleados dado que ya tienen la suya propia. En la siguiente imagen vemos la estructura de la carpeta template.

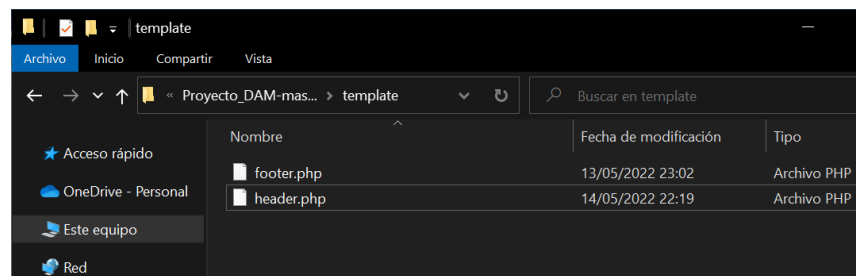


Figura 26: Estructura de archivos de la carpeta template

6.2. Base de datos.

Una base de datos se utiliza para que almacenemos la información necesaria para nuestro proyecto y para que podamos unir la información y relacionarla entre sí. En este proyecto se ha creado una base de datos que se ha llamado libreriacaltan y consta de 4 tablas: libros, clientes, usuarios y reservas. En la siguiente imagen vemos las tablas de la base de datos y la relación entre ellos.

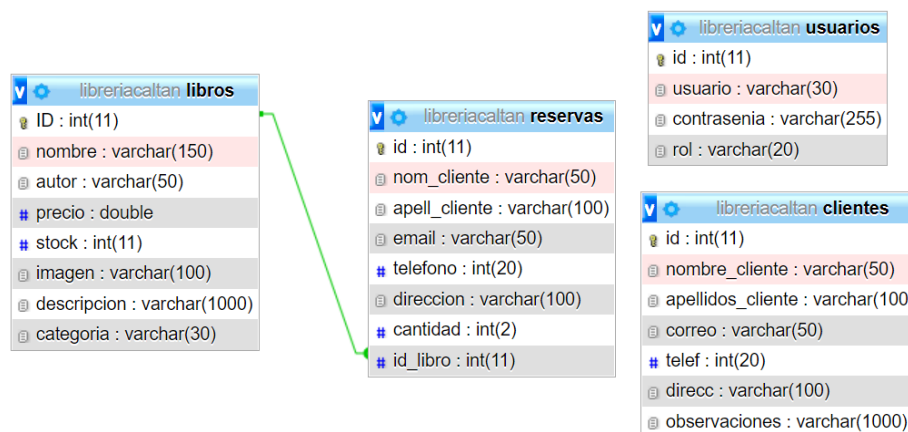


Figura 27: Estructura de tablas de la base de datos de nuestro proyecto

A continuación, se pasa a explicar cada una de las tablas y sus campos:

- **Libros:** En esta tabla se almacena toda la información de los libros. Esta tabla se compone de los siguientes campos:
 - *ID*: Clave primaria de la tabla libros.
 - *nombre*: Título del libro.
 - *autor*: Autor del libro.
 - *precio*: Precio del libro.
 - *stock*: Cantidad disponible del libro en la librería.
 - *imagen*: Se almacena el nombre del archivo de la imagen.
 - *descripcion*: Se almacena la sinopsis del libro.
 - *categoria*: Categoría o género del libro.

- *Reservas*: En esta tabla se almacena toda la información necesaria de la reserva de libros. Esta tabla se compone de los siguientes campos:
 - *id*: Clave primaria de la tabla reservas.
 - *nom_cliente*: Nombre de la persona que reserva el libro.
 - *apell_cliente*: Apellidos de la persona que reserva el libro.
 - *email*: Correo electrónico del que realiza la reserva del libro.
 - *telefono*: Teléfono del que realiza la reserva del libro.
 - *direccion*: Dirección del que realiza la reserva del libro.
 - *cantidad*: Número de libros que se han pedido en la reserva del libro.
 - *id_libro*: Clave foránea de la tabla cliente. De esta manera podremos obtener el título del libro que se ha reservado.

- *Usuarios*: En esta tabla se almacena toda la información de acceso y rol de un usuario. Esta tabla se compone de los siguientes campos:
 - *id*: Clave primaria de la tabla usuarios.
 - *usuario*: Nombre del usuario.
 - *contrasenia*: Contraseña del usuario.
 - *rol*: Rol del usuario (tendrá unos privilegios u otros dependiendo de este campo).

- *Clientes*: En esta tabla se almacena toda la información de los clientes.

Esta tabla se compone de los siguientes campos:

- *id*: Clave primaria de la tabla clientes.
- *nombre_cliente*: Nombre del cliente.
- *apellidos_cliente*: Apellidos del cliente.
- *correo*: Correo electrónico del cliente.
- *telef*: número de teléfono del cliente.
- *direcc*: Dirección del cliente.
- *observaciones*: Campo para que el cliente pueda escribir alguna duda que tenga.

6.3. Login.

Esta vista podemos verla cuando pulsamos en el enlace área de empleados como vimos en el apartado 3 cuando hablamos de las características de este proyecto de esta memoria. En ella tenemos que introducir un usuario y su contraseña para entrar en el área de empleados. Sobre el inicio de sesión mostraremos el código en el apartado de apéndices. Los datos de usuario y contraseña se guardan en la tabla usuarios de nuestra base de datos. Todo el inicio de sesión se gestiona en el archivo de nuestro proyecto llamado login.php aunque antes se realiza la llamada al archivo conexión.php, donde realizamos la conexión a la base de datos introduciendo el host, usuario, contraseña y nombre de la base de datos.

En la siguiente imagen vemos el código de este archivo.

A screenshot of a code editor window titled 'conexion.php'. The code is written in PHP and shows the configuration for a MySQL database connection using PDO. It sets the host to 'localhost', the user to 'root', the password to an empty string, and the database name to 'libreriacaltan'. A try-catch block is used to attempt the connection and handle any exceptions that might occur.

```
<?php
    $host = "localhost";
    $usuario = "root";
    $contraseña = "";
    $bdnombre = "libreriacaltan";

    try {
        $conexion = new PDO("mysql:host=$host;dbname=$bdnombre", $usuario, $contraseña);
        $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {
        echo $e->getMessage();
    }
?>
```

Figura 28: Código PHP que realiza la llamada para conectarse a la base de datos.

Cada vez que queramos realizar una consulta a la base de datos tendremos que llamar a este archivo con la sentencia `include` o `require` de PHP.

Con respecto a la contraseña que se utiliza para el inicio de sesión se ha realizado un cifrado por motivos de seguridad. Para verificar que la contraseña que se pone para el inicio de sesión es la misma que la contraseña cifrada se utiliza la función PHP `password_verify()`. En el apéndice profundizaremos en el código del cifrado de las contraseñas.

6.4. CRUD de libros.

Para la gestión de los libros en el área de empleados se utilizan principalmente dos archivos: `libros.php` (que se encuentra en la carpeta `administrador`) y `librosDAO.php` que se encuentra en la carpeta `config`.

- En *libros.php* nos encontramos con la vista de la gestión de libros. Este archivo se divide en tres partes bien diferenciadas:
 - Un formulario para poder agregar y modificar la información que introducimos. Para crear un formulario utilizamos la etiqueta HTML form, realizaremos el envío de la información a la base de datos por el método post, de esta manera los datos se envían de forma no visible, siendo esta la opción más recomendable y utilizaremos el atributo `enctype` para el codificado de la información que pasamos a través del formulario. El valor de `enctype` debe ser `multipart/form-data` porque de esta manera podremos enviar archivos como por ejemplo imágenes.
 - Haremos un control para que si existen los datos escritos en los inputs del formulario (para ello utilizamos la variable global `POST`) y los almacenemos en una variable esto se realizara con la función `isset`. En la siguiente imagen vemos un ejemplo de esto.

```
$txtID=(isset($_POST['txtID']))?$_POST['txtID']:"";  
$txtNombre=(isset($_POST['txtNombre']))?$_POST['txtNombre']:"";
```

Figura 29: Código para almacenar en una variable el contenido de los inputs del formulario

- Crearemos una tabla con todos los libros que se han almacenado en la tabla libros de la base de datos con dos botones: Seleccionar (para mostrar la información de ese libro en el formulario) y Borrar con el podremos eliminar un libro de la base de datos. El botón de borrado nos redirigirá a una ventana modal en el que se nos preguntará de si estamos seguros de borrar el libro. En la siguiente imagen vemos un ejemplo de esta ventana modal.

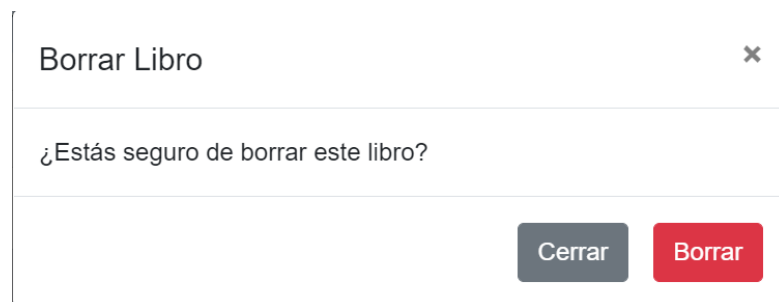


Figura 30: Ventana modal para confirmar el borrado de un libro

- En *librosDAO.php* utilizamos este archivo para la inserción, modificación, lectura y borrado de clientes en la base de datos. Este archivo sirve para que podamos realizar las sentencias SQL necesarias (INSERT, UPDATE, SELECT y DELETE), llamando al archivo para conectarse a la base de datos (conexión.php). En el apéndice profundizaremos en el código para realizar el CRUD.

Por último, para poder tratar con las imágenes del libro necesitamos usar en vez de la variable global POST como en el resto de campos usar la variable global FILES. Para ubicar la imagen utilizamos la función `move_uploaded_file` y para borrar la imagen de la carpeta `img` utilizamos la función `unlink`.

6.5. CRUD de clientes.

Para la gestión de los clientes en el área de empleados se utilizan principalmente dos archivos: `clientes.php` (que se encuentra en la carpeta administrador) y `clientesDAO.php` que se encuentra en la carpeta config.

- En *clientes.php* nos encontramos con la vista de la gestión de clientes.

Este archivo se divide en tres partes bien diferenciadas:

- Un formulario para poder agregar y modificar la información que introducimos. Para crear un formulario utilizamos la etiqueta HTML form, realizaremos el envío de la información a la base de datos por el método post, de esta manera los datos se envían de forma no visible, siendo esta la opción más recomendable.
- Haremos un control para que si existen los datos escritos en los inputs del formulario (para ello utilizamos la variable global POST) y los almacenemos en una variable esto se realizara con la función `isset` como vimos en la gestión de libros.
- Crearemos una tabla con todos los clientes que se han almacenado en la tabla clientes de la base de datos con la peculiaridad de que el campo del correo electrónico aparecerá como un enlace en el que podremos pinchar y se nos redirigirá a nuestro gestor de correo para mandar un correo a ese cliente. Cada cliente cuenta con dos botones: Seleccionar (para mostrar la información de ese cliente en el formulario) y Borrar con el podremos eliminar un cliente de la base de datos. El botón de borrado nos redirigirá a una ventana modal en el que se nos preguntará de si estamos seguros de borrar el cliente como vimos en la gestión de libros.

- En *clientesDAO.php* utilizamos este archivo para la inserción, modificación, lectura y borrado de clientes en la base de datos. Este archivo sirve para que podamos realizar las sentencias SQL necesarias (INSERT, UPDATE, SELECT y DELETE), llamando al archivo para conectarse a la base de datos (*conexión.php*). En el apéndice profundizaremos en el código para realizar el CRUD.

6.6. CRUD de usuarios.

Para la gestión de los usuarios en el área de empleados se utilizan principalmente dos archivos: *usuarios.php* (que se encuentra en la carpeta administrador) y *usuariosDAO.php* que se encuentra en la carpeta config.

- En *usuarios.php* nos encontramos con la vista de la gestión de usuarios. Este archivo se divide en tres partes bien diferenciadas:
 - Un formulario para poder agregar y modificar la información que introducimos. Para crear un formulario utilizamos la etiqueta HTML form, realizaremos el envío de la información a la base de datos por el método post, de esta manera los datos se envían de forma no visible, siendo esta la opción más recomendable.
 - Haremos un control para que si existen los datos escritos en los inputs del formulario (para ello utilizamos la variable global POST) y los almacenemos en una variable esto se realizara con la función *isset* como vimos en la gestión de libros.

- Crearemos una tabla con todos los usuarios que se han almacenado en la tabla usuarios de la base de datos con dos botones: Seleccionar (para mostrar la información de ese usuario en el formulario) y Borrar con el podremos eliminar un usuario de la base de datos. El botón de borrado nos redirigirá a una ventana modal en el que se nos preguntará de si estamos seguros de borrar el usuario como vimos en la gestión de libros.
- En *usuariosDAO.php* utilizamos este archivo para la inserción, modificación, lectura y borrado de clientes en la base de datos. Este archivo sirve para que podamos realizar las sentencias SQL necesarias (INSERT, UPDATE, SELECT y DELETE), llamando al archivo para conectarse a la base de datos (conexión.php). En el apéndice profundizaremos en el código para realizar el CRUD.

6.6.1. Roles de usuario.

En este proyecto tenemos dos tipos o roles de usuarios: administrador y empleado. El administrador se establece para los directivos o dueños de la librería y el rol de empleado se establece para el resto de personal de la librería. Para este proyecto con el rol de administrador tendremos acceso completo del área de empleados mientras que con el rol de empleado no tendremos acceso a la gestión de usuarios ni tampoco podremos exportar ninguna tabla a Excel como veremos más adelante. Para conseguir obtener el rol del usuario conectado almacenaremos el rol con una variable de sesión cuando iniciamos sesión (código que veremos más adelante en el apéndice).

En la siguiente imagen vemos como guardamos el rol del usuario con la variable global SESSION.

```
$_SESSION["rol"] = $resultados["rol"];
```

Figura 31: Código para guardar el rol del usuario conectado

6.7. Muestra de libros en la página principal.

En la página principal de nuestro proyecto (archivo index.php) mostramos los libros con un bucle para mostrar todos los libros. Estos libros se muestran en forma de tarjeta en la que mostramos la imagen del libro, título del libro, autor del libro, precio del libro y un botón para que entremos en la página para reservar este libro en concreto. Este botón nos llevara a la página de reservas en el que cogeremos el id del libro por la URL.

En la siguiente imagen vemos el código para que se muestren todos los libros en la página principal de nuestro proyecto.

```
<!-- Mostrar todos los libros -->
<?php foreach($resultados as $resultado){ ?>
<div class="col-md-3 mb-3 mt-3" >
    <div class="card border-secondary">
        <a href="reservas.php?id=?php echo $resultado['ID']; ?>">
            " alt="imagen"
            height="300" class="card-img-top">
        </a>
        <div class="card-body">
            <a href="reservas.php?id=?php echo $resultado['ID']; ?>">
                <h5 class="card-title"><?php echo $resultado["nombre"]; ?></h5>
            </a>
            <p><?php echo $resultado["autor"]; ?></p>
            <p class="precio"><?php echo $resultado["precio"]; ?>€</p>
            <a href="reservas.php?id=?php echo $resultado['ID']; ?>" class="btn btn-primary"
            role="button">Reserva</a>
        </div>
    </div>
</div>
<?php } ?>
```

Figura 32: Código para mostrar todos los libros en la página principal

En la siguiente imagen vemos como mostramos un libro en forma de tarjeta.



Figura 33: Muestra de un libro en la página principal

6.8. Gestión de las reservas de libros.

En este apartado vamos a tratar dos partes del sitio web: La reserva de un libro y la visualización de las reservas en el área de empleados.

- *Reserva de un libro:* Cuando pulsamos sobre un libro en la página principal (botón reserva que vemos en la última imagen) cogemos el id del libro a través de la URL gracias a la variable global GET. Gracias a esto conseguimos que esta página sea dinámica y se modifique dependiendo del id que le pasamos por URL. En esta página, cuando pulsamos sobre el botón ¡reserva ya! Se nos abrirá una ventana modal con un formulario donde tendremos que escribir la información requerida para realizar la reserva. La inserción de la reserva a la base de datos se realiza con el archivo reservasClientesDAO.php en la carpeta config.

En la siguiente imagen vemos la ventana modal con el formulario para realizar la reserva de un libro.

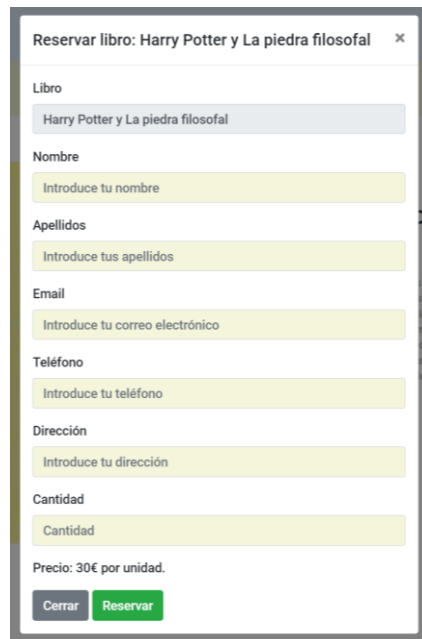
La imagen muestra una ventana modal con el título "Reservar libro: Harry Potter y La piedra filosofal". El formulario contiene los siguientes campos: "Libro" (pre-llenado con "Harry Potter y La piedra filosofal"), "Nombre" (con el placeholder "Introduce tu nombre"), "Apellidos" (con el placeholder "Introduce tus apellidos"), "Email" (con el placeholder "Introduce tu correo electrónico"), "Teléfono" (con el placeholder "Introduce tu teléfono"), "Dirección" (con el placeholder "Introduce tu dirección") y "Cantidad" (con el placeholder "Cantidad"). Debajo de los campos, se indica "Precio: 30€ por unidad." y hay dos botones: "Cerrar" (gris) y "Reservar" (verde).

Figura 34: Ventana modal para reservar un libro

Una vez que pulsemos sobre el botón de reservar que vemos en la anterior imagen la reserva se realizara.

- *Visualización de las reservas:* En el área de empleados, en la sección de reservas (archivo reservas.php en la carpeta administrador), tenemos una tabla con todas las reservas que se han realizado con la peculiaridad de que el campo del correo electrónico aparecerá como un enlace en el que podremos pinchar y se nos redirigirá a nuestro gestor de correo para mandar un correo a la persona que realizó la reserva. También tendremos dos botones para cada reserva:

- *Botón completado*: La finalidad de este botón es de que podamos finalizar el proceso, es decir, que la persona que reservo el libro se ha acercado a por el libro. El botón de completado nos redirigirá a una ventana modal en el que se nos preguntará de si estamos seguros de borrar la reserva y al pulsar en si se realizaran dos funciones: reducir la cantidad de stock por la cantidad de libros que se han llevado y borrar la reserva de la base de datos.

Este proceso de cambios en la base de datos se realiza en el archivo reservasDAO.php de la carpeta config.
- *Botón Borrar*: Este botón se utiliza para que cuando la persona que reservo el libro ya no lo quiere, por lo que se realiza el borrado de la reserva. El botón de borrado nos redirigirá a una ventana modal en el que se nos preguntará de si estamos seguros de borrar la reserva como vimos en la gestión de libros Este cambio en la base de datos se realiza en el archivo reservasDAO.php de la carpeta config.

6.9. Filtros en la página principal.

Volviendo a la página principal de nuestro sitio web (archivo index.php), después del encabezado y de la barra de navegación nos encontramos con dos filtros para que se puedan encontrar más fácil los libros:

- *Filtro por título:* Para realizar este filtro se creado un formulario para que cuando pongamos una palabra en él y le demos al botón buscar, nuestra base de datos busque coincidencias con los títulos de los libros. Para que podamos realizar este filtrado se ha realizado una consulta en la base de datos con el archivo filtrosDAO.php que se encuentra en la carpeta config. En la siguiente imagen vemos como realizamos esta consulta.



```
//para la busqueda de libros por nombre
$busqueda=(isset($_POST['busqueda']))?$_POST['busqueda']:"";
$palabra=(isset($_POST['palabra']))?$_POST['palabra']:"";
if($busqueda=="Buscar"){
    $sentencia=$conexion->prepare("SELECT * FROM libros WHERE nombre LIKE :nombre");
    $sentencia->bindValue(":nombre", "%$palabra%");
    $sentencia->execute();
    $resultados=$sentencia->fetchAll(PDO::FETCH_ASSOC);
}
```

Figura 35: Código que realiza la consulta para realizar el filtrado por título de libro

- *Filtro por categoría:* Para este filtrado tenemos un formulario con un select (lista desplegable) con todas las categorías que disponemos en la librería (Acción, Ciencia Ficción, Cuentos, Infantil, Misterio y Novela Romántica) y un botón para quitar el filtro de categoría. Para realizar este filtrado se ha precisado de dos cosas:
 - un script en JQuery para que cuando elijamos una opción del select se cambie la URL y nos de esa categoría por URL. Este script se realiza en el archivo footer.php de la carpeta template.

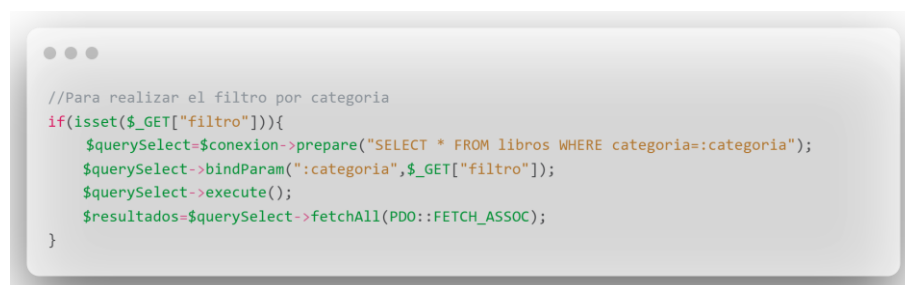
- Una sentencia SQL para seleccionar los libros de una categoría en concreto y que esa categoría se coja por la URL con una variable global GET. Esta sentencia con la que realizamos el filtrado se realiza en el archivo filtrosDAO.php de la carpeta config.

En las siguientes imágenes veremos el script de JQuery y la sentencia SQL.



```
$( "#filtro" ).change(function(){  
    window.location.href="index.php?filtro="+$(this).val();  
});
```

Figura 36: Código JQuery para cambiar la URL al seleccionar una categoría



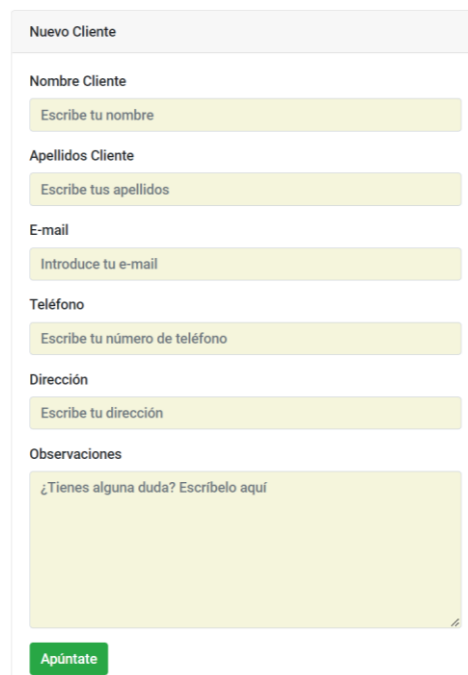
```
//Para realizar el filtro por categoria  
if(isset($_GET["filtro"])){  
    $querySelect=$conexion->prepare("SELECT * FROM libros WHERE categoria=:categoria");  
    $querySelect->bindParam(":categoria",$_GET["filtro"]);  
    $querySelect->execute();  
    $resultados=$querySelect->fetchAll(PDO::FETCH_ASSOC);  
}
```

Figura 37: Código que realiza la consulta para realizar el filtrado por categoría

6.10. Nuevo Cliente.

En nuestro sitio web tenemos una sección llamada nuevo cliente en la que el propio cliente se pueda apuntar para estar informado sobre ultimas noticias. Esta vista se realiza en el archivo *nuevoCliente.php*. En esta vista tenemos un formulario para que se puedan introducir datos por parte del cliente.

En la siguiente imagen vemos el formulario para que se puedan inscribir los clientes.



Formulario de inscripción de nuevo cliente. El formulario está encabezado por 'Nuevo Cliente'. Incluye campos de texto para: Nombre Cliente (con el placeholder 'Escribe tu nombre'), Apellidos Cliente (con el placeholder 'Escribe tus apellidos'), E-mail (con el placeholder 'Introduce tu e-mail'), Teléfono (con el placeholder 'Escribe tu número de teléfono'), Dirección (con el placeholder 'Escribe tu dirección'), y un área de observaciones con el placeholder '¿Tienes alguna duda? Escríbelo aquí'. Al final del formulario hay un botón verde que dice 'Apúntate'.

Figura 38: Formulario para que los clientes se puedan inscribir

Al pulsar en el botón apúntate el cliente se quedará registrado en nuestra base de datos, en la tabla clientes que podremos ver en el área empleados, en la sección de clientes. El proceso de inserción del cliente se realiza a través del archivo insertarClienteDAO.php de la carpeta config.

6.11. Exportar datos a Excel.

Con esta funcionalidad podremos exportar las tablas de libros, clientes y reservas en archivos con formato Excel (extensión .xls). Esta función se podrá visualizar en el área empleados en las secciones libros, clientes y reservas siempre que tengamos un rol de administrador, es decir, que nos aparecerá el botón de exportar tabla si nuestro rol de usuario conectado es administrador. Para poder realizar nuestras exportaciones se han creados archivos como librosExport.php, clientesExport.php y reservasExport.php en la carpeta exports. En estos archivos se han vuelto a recrear las tablas que se ven en las vistas de la gestión de libros, clientes y reservas, pero se tienen que elaborar en archivos separados dado que, al realizar la exportación, esta se realiza sobre el archivo completo por lo que si realizáramos la exportación con las vistas anteriormente mencionadas también se exportaría el formulario. En la siguiente imagen veremos el código en PHP que se necesita para que se pueda realizar la exportación a Excel.

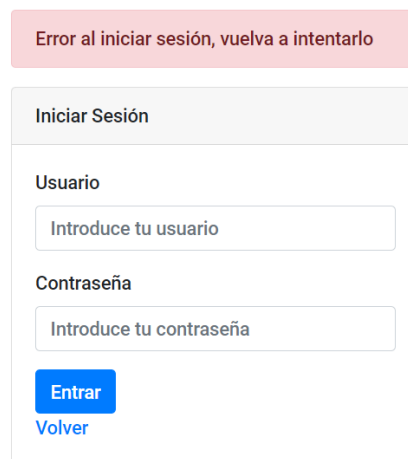
	A	B	C	D	E	F
1	ID	Nombre	Autor	Precio	Categoría	Stock
2	29	Harry Potter y La piedra filosofal	J.K. Rowling	30	ficcion	8
3	35	Harry Potter y La cámara secreta	J.K. Rowling	25	ficcion	3
4	36	Harry Potter y El prisionero de Azkaban	J.K. Rowling	23	ficcion	10
5	37	Harry Potter y El cáliz de fuego	J.K. Rowling	21	ficcion	12
6	38	Harry Potter y La orden del fénix	J.K. Rowling	27	ficcion	11
7	39	Harry Potter y El misterio del príncipe	J.K. Rowling	25	ficcion	15
8	40	Harry Potter y Las reliquias de la muerte	J.K. Rowling	26	ficcion	10
9	50	África. Tormenta de libertad	H. Lanvers	16	accion	20
10	51	Alto riesgo. Libro de Acción	Ken Follett	23	accion	15
11	52	Aguas peligrosas. Libro de Acción	Bernard Cornwell	14	accion	20
12	53	El oro de Esparta. Libro de Acción	Clive Cussler	18	accion	17
13	54	El renacido. Libro de Acción	Michael Punke	17	accion	30
14	55	La tumba perdida. Libro de Acción	Nacho Ares	19	accion	26
15	56	24 besos. Novela Romántica	Caroline March	14	romantica	27
16	57	¿Quién eres?. Novela Romántica	Megan Maxwell	21	romantica	21
17	58	A contracorriente. Novela Romántica	Noe Casado	18	romantica	23
18	59	Cincuenta sombras de Grey	E. L. James	25	romantica	23
19	60	Cincuenta sombras más oscuras	E. L. James	25	romantica	21
20	61	Cincuenta sombras liberadas	E. L. James	25	romantica	26
21	62	Bésame y vente conmigo	Olivia Ardey	17	romantica	14
22	63	Deseo concedido. Novela romántica	Megan Maxwell	17	romantica	12

Figura 39: Tabla libros en Excel

7. Fase de pruebas.

En este apartado vamos a hablar de todas las estructuras de control que tenemos en el proyecto, es decir, las partes de la página web en las que se advierte al usuario de que no podemos realizar alguna tarea, por lo que intentaremos realizar estas acciones y veremos si se obtienen los resultados esperados. A continuación, vamos a ver estas pruebas.

- *Introducir un usuario o contraseña incorrectos:* Esta prueba se va a realizar introduciendo un usuario incorrecto o una contraseña incorrecta. Si se realiza esta acción el resultado esperado es que nos aparezca una alerta en rojo de error al iniciar sesión. En la siguiente imagen vemos lo que ocurre cuando introducimos un usuario incorrecto.



La imagen muestra una interfaz de usuario con una alerta de error y un formulario de inicio de sesión. La alerta, en un recuadro rojo, indica: "Error al iniciar sesión, vuelva a intentarlo". Debajo, el formulario "Iniciar Sesión" contiene campos para "Usuario" y "Contraseña", ambos con el texto "Introduce tu usuario" y "Introduce tu contraseña" respectivamente. Al final del formulario hay un botón azul "Entrar" y un enlace "Volver" en azul.

Figura 40: Alerta cuando introducimos un usuario incorrecto

Como podemos comprobar es el resultado que esperábamos y el código que consigue que aparezca el mensaje de alerta es el que vemos en la siguiente imagen.

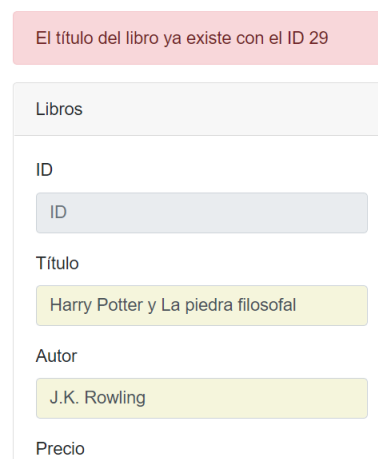


```
<?php if(!empty($mensaje)): ?>
    <div class="alert alert-danger" role="alert">
        <?php echo $mensaje ?>
    </div>
<?php endif; ?>
```

Figura 41: Código para que aparezca el mensaje en rojo

Más adelante, en el apéndice de inicio de sesión veremos cómo conseguimos generar este mensaje.

- *Introducir un título de libro que ya existe:* Esta prueba se va a realizar agregando un libro con un título de libro que ya existe en la sección de gestión de libros. Si se realiza esta acción el resultado esperado es que nos aparezca una alerta en rojo donde se indique que no podemos tener dos títulos de libros iguales y con qué id ya está ese libro. En la siguiente imagen vemos que ocurre cuando introducimos un título que ya existe.



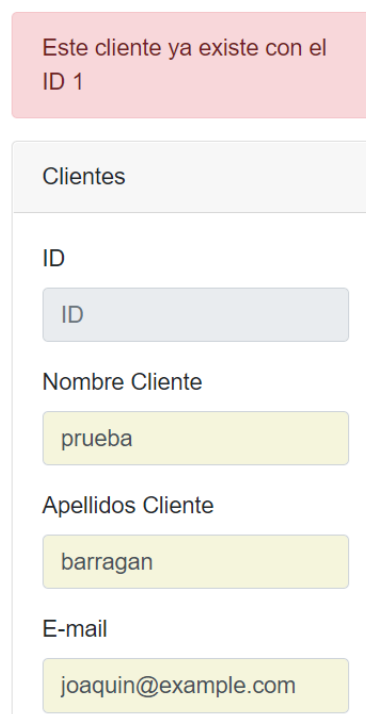
El título del libro ya existe con el ID 29

Libros
ID
<input type="text" value="ID"/>
Título
<input type="text" value="Harry Potter y La piedra filosofal"/>
Autor
<input type="text" value="J.K. Rowling"/>
Precio

Figura 42: Alerta cuando introducimos un título de libro que ya existe

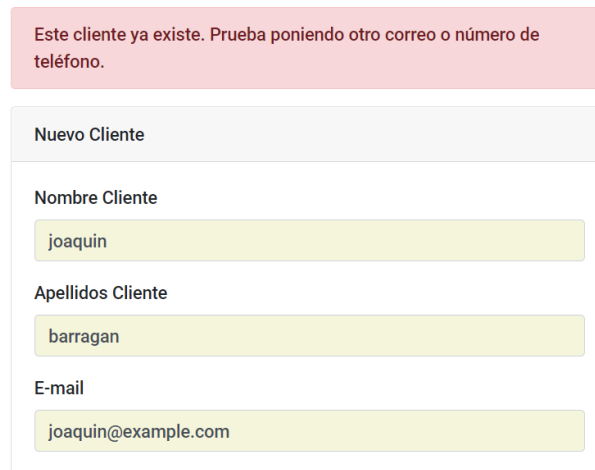
Como podemos comprobar es el resultado que esperábamos y el código que consigue que aparezca el mensaje de alerta es el mismo que vimos en el punto anterior. Más adelante, en el apéndice de ejemplo de CRUD veremos cómo conseguimos generar este mensaje.

- *Introducir un cliente que tenga el mismo número de teléfono o correo electrónico que otro cliente existente.* Esta prueba se va a realizar agregando un cliente con un correo o teléfono que ya existe. En este caso se van a realizar dos pruebas: una se hará en la sección nuevo cliente y la otra se hará en la sección de gestión de clientes en el área empleados. Si se realiza esta acción el resultado esperado es que nos aparezca una alerta en rojo donde se indique que ya existe el cliente. En las siguientes imágenes vemos lo que ocurre en las dos pruebas.



The image shows a web interface for managing clients. At the top, a red alert box contains the text: "Este cliente ya existe con el ID 1". Below this is a form titled "Clientes". The form has several input fields: "ID" (with a light blue placeholder box containing "ID"), "Nombre Cliente" (with a light green placeholder box containing "prueba"), "Apellidos Cliente" (with a light green placeholder box containing "barragan"), and "E-mail" (with a light green placeholder box containing "joaquin@example.com").

Figura 43: Alerta por introducir un cliente que ya existe en la gestión de clientes



Este cliente ya existe. Prueba poniendo otro correo o número de teléfono.

Nuevo Cliente

Nombre Cliente

joaquin

Apellidos Cliente

barragan

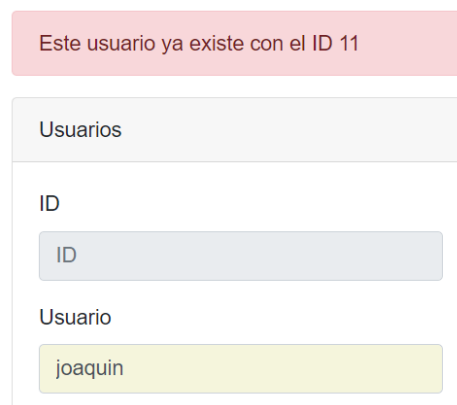
E-mail

joaquin@example.com

Figura 44: Alerta por introducir un cliente que ya existe en la sección nuevo cliente

Como podemos comprobar es el resultado que esperábamos y el código que consigue que aparezca el mensaje de alerta es el mismo que vimos en el primer punto de este apartado. Más adelante, en el apéndice de ejemplo de CRUD veremos cómo conseguimos generar este mensaje.

- *Introducir un usuario que tiene el mismo nombre que otro:* Esta prueba se va a realizar agregando un usuario que tenga el mismo nombre que otro en la sección de gestión de usuarios. Si se realiza esta acción el resultado esperado es que nos aparezca una alerta en rojo donde se indique que este usuario ya existe y con qué id ya está ese usuario. En la siguiente imagen vemos que ocurre cuando introducimos un usuario que ya existe.



Este usuario ya existe con el ID 11

Usuarios

ID

ID

Usuario

joaquin

Figura 45: Alerta por introducir un usuario que ya existe

Como podemos comprobar es el resultado que esperábamos y el código que consigue que aparezca el mensaje de alerta es el mismo que vimos en el primer punto de este apartado. Más adelante, en el apéndice de ejemplo de CRUD veremos cómo conseguimos generar este mensaje.

- *Reservar más libros de los que tenemos en stock:* Esta prueba se va a realizar agregando una cantidad superior al stock de un libro en la página para reservar un libro. Si se realiza esta acción el resultado esperado es que nos aparezca un bocadoillo encima del input de cantidad donde se indique que cantidad máxima de libro es el número de stock que tenemos del libro. En la siguiente imagen vemos que ocurre cuando pulsamos en el botón reservar e introducimos una cantidad mayor que stock.



Harry Potter y El prisionero de Azkaban

Nombre
hola

Apellidos
blanco

Email
joaquin@example.com

Teléfono
987654321

Dirección
madrid

Cantidad
12

Precio: 23€

El valor debe ser inferior o igual a 10

Cerrar Reservar

Figura 46: Alerta (recuadro en rojo) de que la cantidad es superior al stock

Como podemos comprobar es el resultado que esperábamos y el código que consigue que aparezca el mensaje de alerta es indicando en el atributo *max* del input de cantidad que su valor sea el stock de ese libro.

- *No hay stock de un libro:* Esta prueba se va a realizar poniendo el stock de un libro a cero en la gestión de libros del área empleados, después nos iremos a la página para reservar un libro para ver un resultado. Si se realiza esta acción el resultado esperado es que nos aparezca una alerta en rojo donde se indique que no hay stock del libro y que desaparezca el botón de reservar. En la siguiente imagen vemos que ocurre cuando no tenemos stock de un libro.

Harry Potter y La piedra filosofal

30€

Harry Potter se ha quedado huérfano y vive en casa de sus abominables tíos y del insoportable primo Dudley. Harry se siente muy triste y solo, hasta que un buen día recibe una carta que cambiará su vida para siempre. En ella le comunican que ha sido aceptado como alumno en el colegio interno Hogwarts de magia y hechicería. A partir de ese momento, la suerte de Harry da un vuelco espectacular. En esa escuela tan especial aprenderá encantamientos, trucos fabulosos y tácticas de defensa contra las malas artes. Se convertirá en el campeón escolar de quidditch, especie de fútbol aéreo que se juega montado sobre escobas, y se hará un puñado de buenos amigos... aunque también algunos temibles enemigos. Pero sobre todo, conocerá los secretos que le permitirán cumplir con su destino. Pues, aunque no lo parezca a primera vista, Harry no es un chico común y corriente. ¡Es un verdadero mago!

En stock: 0

Actualmente no hay stock del libro Harry Potter y La piedra filosofal. Vuelva a intentarlo en otro momento.

Figura 47: Alerta cuando no hay stock de un libro

Como podemos comprobar es el resultado que esperábamos y el código que consigue que aparezca el mensaje de alerta es el que vemos en la siguiente imagen.

```
<?php if($datosLibro["stock"]>0){ ?>
  <button type="button" class="btn btn-primary" data-toggle="modal"
    data-target="#staticBackdrop">Reserva ya!</button>
<?php }else{ ?>
  <div class="alert alert-danger" role="alert">
    <?php echo "Actualmente no hay stock del libro ".$datosLibro["nombre"].". Vuelva a
    intentarlo en otro momento."; ?>
  </div>
<?php } ?>
```

Figura 48: Código que controla este error

- *Completar una reserva cuando no hay stock suficiente del libro:* Esta prueba se va a realizar porque el stock de un libro no se actualiza hasta que hayamos completado la reserva por lo que se puede dar la situación de tener una reserva con una cantidad mayor que el stock porque en el momento en el que se realizó la reserva si había suficiente stock en la librería. Esta prueba se puede visualizar en la gestión de reservas del área empleados. Si se realiza esta acción el resultado esperado es que desaparezca el botón de completado. En la siguiente imagen vemos lo que ocurre cuando tenemos una reserva con una cantidad reservada de libros superior al stock.

ID	Nombre Cliente	Apellidos Cliente	e-mail	Teléfono	Dirección	Cantidad	Stock	Libro	Acción
16	prueba	adios	ana@example.com	987654321	madrid	4	3	Harry Potter y La cámara secreta	<button>Borrar</button>

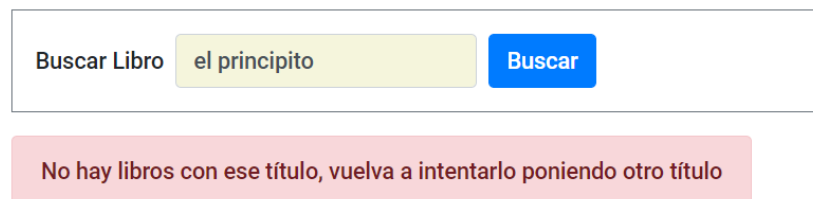
Figura 49: Prueba cuando la cantidad es superior al stock

Como podemos comprobar es el resultado que esperábamos y el código que consigue que el botón completado desaparezca es el que vemos en la siguiente imagen.

```
<!-- no permitir la completar la reserva si hay menos stock que la cantidad de
libros pedida -->
<?php if($resultado["stock"]>$resultado["cantidad"]){ ?>
    <button type="button" class="btn btn-success" data-toggle="modal"
        data-target="#confirmarModal<?php echo $resultado['id']; ?>">Completado</button>
<?php } ?>
```

Figura 50: Código para que desaparezca el botón completado

- *Cuando buscar un libro y no encuentra ninguno:* Esta prueba se va a realizar introduciendo un título de libro en la búsqueda por título en la página principal que no exista en nuestra base de datos. Si se realiza esta acción el resultado esperado es que nos aparezca una alerta en rojo donde se indique que no hay libros con ese título. En la siguiente imagen vemos que ocurre cuando escribimos un libro que no tenemos.



Buscar Libro el principito Buscar

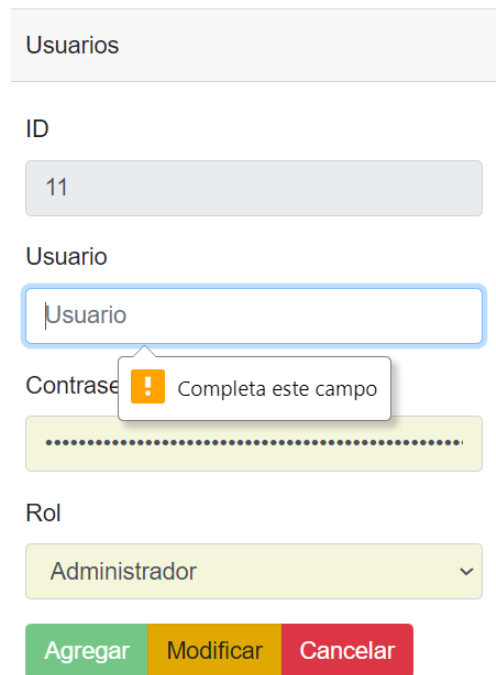
No hay libros con ese título, vuelva a intentarlo poniendo otro título

Figura 51: Alerta de que el buscado no esta

Como podemos comprobar es el resultado que esperábamos y el código con el que conseguimos esta alerta es que en el bucle con el que mostramos los libros crear una variable que contabilice el número de veces que se repite el bucle y que si es cero que nos muestre la alerta.

- *Que los campos obligatorios se tengan que escribir obligatoriamente:* Esta prueba se va a realizar no introduciendo un campo obligatorio. Si se realiza esta acción el resultado esperado es que nos aparezca un bocadoillo encima del input donde nos falte un campo donde se indique que el campo es requerido.

En la siguiente imagen vemos un ejemplo de que ocurre cuando no introducimos un campo obligatorio.



Formulario de Usuarios:

- ID: 11
- Usuario: [Campo vacío con borde azul]
- Contraseña: [Campo oculto con puntos] **Alerta:** Completa este campo
- Rol: Administrador
- Botones: Agregar, Modificar, Cancelar

Figura 52: Alerta para cuando un campo sea obligatorio

Como podemos comprobar es el resultado que esperábamos y el código con el que conseguimos crear esta alerta es introduciendo el atributo *required* en los inputs donde los campos sean obligatorios.

8. Posibles Mejoras.

En este apartado vamos a redactar las posibles mejoras que se podrían realizar en un futuro para nuestro proyecto. Las posibles mejoras pueden ser:

- *Mejorar la gestión de clientes:* Se podría relacionar la tabla de clientes con la de reservas para que cuando un cliente reserve un libro este ya se almacene en la tabla clientes de la base de datos. En este caso en concreto se podría decir que es una cuestión de perspectiva dado que, aunque una persona reserve un libro no necesariamente querría que almacenáramos sus datos para siempre.
- *Que el cliente pueda comprar directamente un libro desde la página web:* Esta mejora se podría complementar con la de reservar un libro y le daría una mejora muy buena a la librería dado que no se necesitaría visitar la librería para que adquiriéramos un libro.
- *Crear una gestión de pedidos de libros:* Realizaríamos una nueva sección del área empleados y sería similar a las secciones de libros, clientes y usuarios dado que podríamos insertar, modificar, leer y borrar pedidos de libros. Esta tabla en la base de datos se relacionaría con la tabla de proveedores (se explicará a continuación) y con la tabla libros. Esta sección podría ser útil dado que se podría gestionar que, si tenemos poco stock de un libro, avisarnos de que tendríamos que realizar el pedido de ese libro.

- *Crear una gestión de proveedores:* Realizaríamos una nueva sección del área empleados y sería similar a las secciones de libros, clientes y usuarios dado que podríamos insertar, modificar, leer y borrar los proveedores que utilizamos para reponer los libros. Esta tabla se relacionaría con la tabla de pedidos y con la tabla de libros para que se puedan visualizar que proveedores venden los libros que necesitamos.
- *Crear una gestión de ventas de libros:* Realizaríamos una nueva sección del área empleados donde tendríamos una tabla con todas las ventas de la librería donde podríamos visualizar la cantidad de veces que se ha vendido un libro y cuánto dinero se ha generado con ello. Esta nueva tabla en la base de datos se relacionaría con la de reservas y libros. También se podría filtrar por fecha (semanal, mensual, anual, etc.)
- *Crear una gestión contable de la librería:* Realizaríamos una nueva sección del área empleados donde tendríamos la posibilidad de insertar, modificar, leer y borrar portes. Esta sección sería útil para que obtengamos el coste total de la librería, los ingresos totales de la librería y con ello tendríamos el beneficio de la librería. Esta sección se relacionaría con las reservas y pedidos, además se podría filtrar por fecha (semanal, mensual, anual, etc.)

- *Añadir más filtros en la página principal:* Se podrían añadir otros tipos de filtros como un filtro por rango de precios (menos de 5€, entre 5 y 10€, entre 10 y 20 €, etc.), también se podría incluir un filtro como los libros más vendidos (al tener una sección de ventas sería ordenar esta tabla por número de libros vendidos). Otro filtro podría ser ordenar los libros por su título en orden alfabético.

9. Conclusiones del proyecto.

El objetivo principal de realizar este proyecto era la realización de un sitio web dado que quería un reto ya que no tenía experiencia en el desarrollo web y quería aprender como poder realizar una desde cero. Gracias a los conocimientos adquiridos en esta formación, aprendizaje autodidacta y gracias a las prácticas de empresa he podido realizar una página web de una librería donde se pueden reservar libros y gestionar los libros, usuarios, reservas y clientes de la librería.

Este proyecto ha sido un reto estimulante y arduo pero muy gratificante al ver como aprendes una gran variedad de cosas para poder realizar lo que quieres.

Para mí, lo más importante es el constante aprendizaje y este proyecto ha tenido grandes dosis de ello, además en esta profesión esta cualidad es una de las más importantes dado que las tecnologías cambian constantemente.

En conclusión, puedo decir que estoy satisfecho con el resultado de mi proyecto y que gracias a los conocimientos que he adquirido con este me veo capacitado para realizar futuros proyectos en el mundo del desarrollo web.

10. Apéndices.

En este vamos a incluir ciertas partes del código del proyecto que consideramos más importantes del sitio web. Vamos a hablar sobre el inicio de sesión, cifrado de la contraseña y un ejemplo de CRUD (Create, Read, Update y Delete).

10.1. Inicio de sesión.

El inicio de sesión se realiza cuando estamos en el área empleados y queremos entrar a la gestión de libros, usuarios, reservas y clientes. Para entrar necesitamos un usuario y contraseña que se generarán en la gestión de usuarios. En la siguiente imagen vamos a ver el código que realiza el inicio de sesión.



Código para el inicio de sesión

```
require '../config/conexion.php';

if (!empty($_POST['usuario']) && !empty($_POST['contrasenia'])) {
    $sentencia = $conexion->prepare('SELECT id, usuario, contrasenia, rol
    FROM usuarios WHERE usuario = :usuario');
    $sentencia->bindParam(':usuario', $_POST['usuario']);
    $sentencia->execute();
    $resultados = $sentencia->fetch(PDO::FETCH_ASSOC);
    $mensaje = '';

    if (is_countable($resultados) > 0 &&
    password_verify($_POST['contrasenia'], $resultados['contrasenia'])) {
        $_SESSION["usuario_id"] = $resultados["usuario"];
        $_SESSION["rol"] = $resultados["rol"];
        header("Location: ../administrador/libros.php");
    } else {
        $mensaje = 'Error al iniciar sesión, vuelva a intentarlo';
    }
}
```

Figura 53: Código que realiza el inicio de sesión

Vamos a explicar con más detalle:

- En primer lugar, se realiza la llamada a la conexión de la base de datos (código que ya se mostró en apartados anteriores) dado que realizaremos una consulta SQL a la base de datos que se ha creado para este proyecto.

- En el siguiente paso se realizará una comprobación de que los campos usuario y contraseña están rellenos. Esta comprobación se obtiene con la función `empty` que comprueba si el campo está vacío, pero al utilizar el símbolo de opuesto (!) se realiza lo contrario que es si está el campo relleno.
- Una vez realizada esta comprobación si se cumple realizamos la consulta SQL con la variable conexión a la base de datos y la función `prepare` para que busca la fila donde este el usuario que hemos puesto en el campo usuario, esto se consigue gracias a la función `bindParam` con el que conseguimos pasar el valor de lo introducido en el campo usuario con la variable global `POST`. Para ejecutar la consulta utilizamos la función `execute` y guardamos los datos de ese usuario en concreto en una variable, aunque para esto se necesita usar la función `fetch` con el modo `FETCH_ASSOC` para que podamos tener los datos utilizando en un array y tener el dato de la columna llamando a esta variable array y poniendo como valor en el array el nombre de la columna. En este caso esta variable se llama `$resultados` y para obtener el campo usuario sería: `$resultados["usuario"]`.
- En el siguiente paso se realiza otras dos comprobaciones: si se ha obtenido resultado en la consulta (para ello se ha utilizado la función `is_countable()` que devuelve falso si en el array no tuviéramos datos) y si la contraseña introducida en la campo coincide con la contraseña que tenemos en la base de datos de ese usuario, para ello utilizamos la función `password_verify` dado que la contraseña que tenemos en la base de datos

está cifrada (sobre el cifrado se explicara con más detalle en el siguiente apartado).

- Si la comprobación se cumple crearemos dos variables globales de sesión: una para indicarnos que hemos iniciado la sesión con un usuario concreto y otra para que almacenemos el rol del usuario conectado y así podamos dar permiso o acceso a varias partes de la página web (esto lo vimos en el apartado de roles de usuarios). Además, con la función header nos redirigirá a la sección de libros. Si la comprobación no se cumpliera almacenamos un mensaje de error que es el que nos aparecerá si no hemos introducido los datos correctamente y que ya vimos en la fase de pruebas.

Una vez visto el código ahora debemos de asegurarnos de que una persona no se pueda meter en las páginas que protege el login desde la URL directamente, para ello empleamos este código.

A screenshot of a code editor window with a title bar that says "Código para no acceder a las páginas web directamente". The code is in PHP and is as follows:

```
session_start();  
if (!isset($_SESSION['usuario_id'])) {  
    header('Location: login.php');  
}
```

Figura 54: Código para no poder a los enlaces directamente

Este código hace la comprobación de que si no existe la variable global SESSION que creamos no existe (!isset) que nos redirija al inicio de sesión (función header). Para realizar esto tenemos que utilizar la función sesión_start para que inicie o reanude la sesión.

10.2. Cifrado de contraseñas.

Para que podamos cifrar las contraseñas de los usuarios y así evitamos lo máximo posible que se puedan hackear, lo realizamos con el código que vemos en la siguiente imagen.

```
$pass=password_hash($txtContrasenia, PASSWORD_BCRYPT);
```

Figura 55: Código para cifrar la contraseña

Este código se encuentra en el archivo UsuariosDAO.php cuando insertamos un usuario a la base de datos. Para el cifrado se utiliza la función password_hash en el cual se introducen dos parámetros: la contraseña que queremos cifrar y el algoritmo que se va a utilizar para el cifrado, en este caso se ha utilizado el algoritmo blowfish con PASSWORD_BCRYPT que es el que se recomienda por la documentación oficial en PHP. En la siguiente imagen vemos la tabla usuarios en la base de datos para mostrar el cifrado de las contraseñas.

id	usuario	contrasenia	rol
11	joaquin	\$2y\$10\$d/QaGLsqCR9IV0EVuN0abuZyQCQ.ECDaBELUFmENqAb...	administrador
14	fatima	\$2y\$10\$uPWndga2o6VtccWv/s7OzuyDFG5.M3RImtcDNQwmvwj...	administrador
16	ana	\$2y\$10\$y4ikBi4czAjZe6YninPmsOSVkXSlshShMzihTu5mHTE...	empleado
18	carmen	\$2y\$10\$6BQ3D8Ga4kXg6snRsewWK.lpvDTu./Pi4t4I43QUcZU...	empleado

Figura 56: Cifrado de las contraseñas en la base de datos

Para desencriptar esta contraseña y poder realizar el inicio de sesión se utiliza la función password_verify que se mostró en el apartado anterior y que se utiliza para este motivo.

10.3. Ejemplo de CRUD

En este apartado vamos a ver el código para ver cómo se realiza el CRUD (Create, Read, Update y Delete) o en español (Insertar, Leer, Modificar y Borrar) para la gestión de clientes.

En primer lugar, dado que recogemos los valores de los inputs de un formulario vamos a crear una variable por cada dato que recojamos de ellos. El código con el que realizamos esta acción es el siguiente:



```

$txtID=(isset($_POST['txtID']))?$_POST['txtID']:"";
$txtNombre=(isset($_POST['txtNombre']))?$_POST['txtNombre']:"";
$txtApellidos=(isset($_POST['txtApellidos']))?$_POST['txtApellidos']:"";
$txtEmail=(isset($_POST['txtEmail']))?$_POST['txtEmail']:"";
$numTelefono=(isset($_POST['numTelefono']))?$_POST['numTelefono']:"";
$txtDireccion=(isset($_POST['txtDireccion']))?$_POST['txtDireccion']:"";
$txtObservaciones=(isset($_POST['txtObservaciones']))?$_POST['txtObservaciones']:"";
$accion=(isset($_POST['accion']))?$_POST['accion']:"";

```

Figura 57: Código para almacenar en una variable los datos introducidos por el usuario

Ahora vamos a pasar a explicar cómo introducir, leer, modificar y borrar datos de los clientes a través del sitio web:

- En primer lugar, se ha de llamar a la conexión a la base de datos y al archivo `clientesFull.php` donde realizamos la consulta para leer a todos los clientes, luego se ha utilizado un switch con la condición de la variable `$accion` que es la variable que almacena los botones (gracias al atributo `name` de HTML y el valor de los botones gracias al atributo `value` de HTML) para agregar, modificar, seleccionar y borrar información.

En la siguiente imagen vemos como se forma el switch.

```
require("conexion.php");
require("clientesFull.php");
switch($accion){
    //case del CRUD...
```

Figura 58: Sentencia switch donde irán alojados en los case el CRUD

- *Insertar un cliente*: El código (uno de los case del switch) con el que realizamos la inserción de un cliente es la siguiente.

```
case "Agregar":
    foreach($resultados as $resultado){
        if($txtEmail==$resultado["correo"] || $numTelefono==$resultado["telef"]){
            $id_cliente=$resultado["id"];
            $contador++;
        }
    }
    if($contador==0){
        $sentencia=$conexion->prepare("INSERT INTO clientes(nombre_cliente, apellidos_cliente,
        correo, telef, direcc, observaciones) VALUES(:nombre_cliente, :apellidos_cliente,
        :correo, :telef, :direcc, :observaciones)");
        $sentencia->bindParam(":nombre_cliente", $txtNombre);
        $sentencia->bindParam(":apellidos_cliente", $txtApellidos);
        $sentencia->bindParam(":correo", $txtEmail);
        $sentencia->bindParam(":telef", $numTelefono);
        $sentencia->bindParam(":direcc", $txtDireccion);
        $sentencia->bindParam(":observaciones", $txtObservaciones);
        $sentencia->execute();
        header("Location:clientes.php");
    }else{
        $mensaje="Este cliente ya existe con el ID ".$id_cliente;
    }
    break;
```

Figura 59: Código para insertar un cliente a la base de datos desde el sitio web

Vamos a explicar ciertas partes del código:

- En el foreach (bucle) iteramos todos los registros de clientes en la base de datos para comprobar que no se repitan ni números de teléfono ni correos (esta parte ya se explica en el apartado de pruebas por lo que no se volverá a explicar aquí).

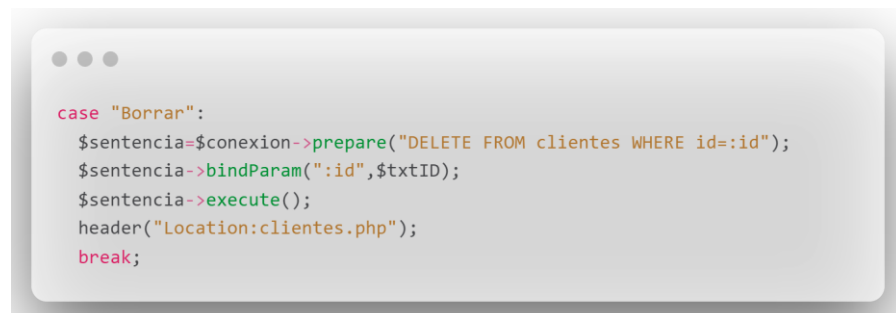
- Para realizar la inserción utilizamos la sentencia SQL INSERT que tiene la siguiente estructura: *INSERT INTO nombre_tabla (campos que se van a utilizar para la inserción) VALUES (valores de esos campos)*. Con la función prepare, prepara la sentencia SQL para su ejecución, luego se utiliza la función bindParam para que introduzcamos los valores que se han escrito en los campos del formulario y la función execute para que se ejecute la sentencia SQL.
- *Modificar un cliente*: El código (uno de los case del switch) con el que realizamos la modificación de los datos de un cliente es la siguiente.

```
case "Modificar":
    foreach($resultados as $resultado){
        if(($txtEmail==$resultado["correo"] || $numTelefono==$resultado["telef"]) &&
            $txtID!=$resultado["id"]){
            $id_cliente=$resultado["id"];
            $contador++;
        }
    }
    if($contador==0){
        $sentencia=$conexion->prepare("UPDATE clientes SET nombre_cliente=:nombre_cliente,
apellidos_cliente=:apellidos_cliente, correo=:correo, telef=:telef, direcc=:direcc,
observaciones=:observaciones WHERE id=:id");
        $sentencia->bindParam("nombre_cliente",$txtNombre);
        $sentencia->bindParam("apellidos_cliente",$txtApellidos);
        $sentencia->bindParam("correo",$txtEmail);
        $sentencia->bindParam("telef",$numTelefono);
        $sentencia->bindParam("direcc",$txtDireccion);
        $sentencia->bindParam("observaciones",$txtObservaciones);
        $sentencia->bindParam("id",$txtID);
        $sentencia->execute();
        header("Location:clientes.php");
    }else{
        $mensaje="Este cliente ya existe con el ID ".$id_cliente;
    }
    break;
```

Figura 60: Código para modificar un cliente de la base de datos desde el sitio web

Vamos a explicar ciertas partes del código:

- En el foreach (bucle) iteramos todos los registros de clientes en la base de datos para comprobar que no se repitan ni números de teléfono ni correos y que el id sea diferente del que estamos modificando porque si no se podría dar el caso de que no se modifique (esta parte ya se explica en el apartado de pruebas por lo que no se volverá a explicar aquí).
- Para realizar la modificación utilizamos la sentencia SQL UPDATE que tiene la siguiente estructura: *UPDATE nombre_tabla SET campo_a_modificar=valor_que_tendra (esto por cada campo)*. Con la función prepare, prepara la sentencia SQL para su ejecución, luego se utiliza la función bindParam para que introduzcamos los valores que se han escrito en los campos del formulario y la función execute para que se ejecute la sentencia SQL.
- *Borrar un cliente*: El código (uno de los case del switch) con el que realizamos el borrado de los datos de un cliente es la siguiente.

A screenshot of a code editor window showing PHP code for deleting a client. The code is enclosed in a light gray box with a subtle drop shadow. It features a 'case' statement for 'Borrar' (Delete) which uses PDO methods: prepare, bindParam, and execute. It also includes a header redirect and a break statement.

```
case "Borrar":  
    $sentencia=$conexion->prepare("DELETE FROM clientes WHERE id=:id");  
    $sentencia->bindParam(":id",$txtID);  
    $sentencia->execute();  
    header("Location:clientes.php");  
    break;
```

Figura 61: Código para el borrado un cliente de la base de datos desde el sitio web

Vamos a explicar ciertas partes del código:

- Para realizar el borrado de un cliente utilizamos la sentencia SQL `DELETE` que tiene la siguiente estructura: *DELETE FROM nombre_tabla WHERE condición*. Esta condición será igual el id (identificador único) con el id del registro que queremos borrar. Con la función `prepare`, prepara la sentencia SQL para su ejecución, luego se utiliza la función `bindParam` para que introducir el id del registro que queremos borrar y la función `execute` para que se ejecute la sentencia SQL.
- *Leer un cliente*: El código (uno de los case del switch) con el que realizamos la lectura de los datos de un cliente es la siguiente.

```
case "Seleccionar":
    $sentencia=$conexion->prepare("SELECT * FROM clientes WHERE id=:id");
    $sentencia->bindParam(":id",$txtID);
    $sentencia->execute();
    $datosCliente=$sentencia->fetch(PDO::FETCH_LAZY);
    $txtNombre=$datosCliente['nombre_cliente'];
    $txtApellidos=$datosCliente['apellidos_cliente'];
    $txtEmail=$datosCliente['correo'];
    $numTelefono=$datosCliente['telef'];
    $txtDireccion=$datosCliente['direcc'];
    $txtObservaciones=$datosCliente['observaciones'];
    break;
```

Figura 62: Código para la lectura de un cliente de la base de datos desde el sitio web

Vamos a explicar ciertas partes del código:

- Para realizar la lectura de un cliente utilizamos la sentencia SQL `SELECT` que tiene la siguiente estructura: *SELECT campos_a_mostrar FROM nombre_tabla WHERE condición*. Esta condición será igual el id (identificador único) con el id del registro que queramos leer. Con la función `prepare`, prepara la sentencia SQL para su ejecución, luego se utiliza la función `bindParam` para que introduzca el id del registro que queremos leer, después la función `execute` para que se ejecute la sentencia SQL y almacenamos en una variable los valores obtenidos de un cliente. Por último, almacenamos cada campo que obtenemos en los inputs del formulario para que cuando pulsemos en el botón seleccionar los datos del cliente se muestren en el formulario.

11. Referencias Bibliográficas.

- Documentación oficial PHP: <https://www.php.net/manual/es/>
- Documentación HTML: <https://www.w3schools.com/html/>
- Documentación CSS: <https://www.w3schools.com/css/>
- Documentación oficial Bootstrap:
<https://getbootstrap.com/docs/4.5/getting-started/introduction/>
- Documentación oficial JQuery: <https://api.jquery.com/>
- Documentación de Git y GitHub: <https://git-scm.com/book/es/v2>
- Página donde se han obtenido la información sobre los libros:
<https://www.lecturalia.com/>