

BOOKSPHERE



Realizado por: Mario Martín Godoy

Curso: 2º Desarrollo Aplicaciones Web

Módulo: Desarrollo de entorno servidor

Tecnologías: HTML, CSS, JavaScript, PHP

ÍNDICE DE CONTENIDOS

1. Explicación del proyecto.....	página	2
2. Demostración web.....	página	4
3. Explicación funciones.....	página	11
4. Tecnologías utilizadas.....	página	16

1. EXPLICACIÓN DEL PROYECTO

BookSphere: Un nuevo concepto de biblioteca online.

Bienvenido a BookSphere, tu nueva experiencia bibliotecaria en línea. Este proyecto es una aplicación web diseñada para conectar a amantes de la lectura con una vasta colección de libros. Tanto los clientes como los administradores disfrutarán de un viaje literario fluido y fácil de gestionar.

Cliente:

Búsqueda Intuitiva: Nuestra interfaz de usuario permite a los lectores buscar libros por título, autor, isbn o género, brindándoles acceso rápido a sus lecturas deseadas. Además, podrán realizar uso de una búsqueda avanzada y más directa.

La aplicación ofrece diversas clasificaciones para los libros en función de su popularidad dentro de los clientes que componen la biblioteca o también en función de su fecha de lanzamiento.

Personalización: ¿Buscas algo nuevo? BookSphere te ofrece recomendaciones personalizadas y la opción de explorar el "Libros más populares", proporcionando variedad y sorpresas a los lectores ávidos.

Administrador:

Gestión de Catálogo: Los administradores tienen el control total del catálogo de libros. Pueden agregar nuevos títulos, actualizar información y retirar libros obsoletos, manteniendo la biblioteca siempre actualizada a las necesidades de los clientes y a las novedades del momento.

Supervisión de Interacciones:

Garantizamos un entorno amigable. Los administradores supervisan las interacciones de los usuarios, pudiendo realizar un seguimiento de el número de lecturas que realizan los usuarios y de los géneros más leídos.

Seguridad:

Protección de Datos: La seguridad de la información es primordial. Implementamos medidas robustas para proteger los datos de los usuarios y garantizar una experiencia segura en línea. Además, la aplicación ofrece un servicio de integridad de datos garantizado en nuestra Base de datos.

Innovación:

¿Qué nos hace únicos?: BookSphere va más allá de ser una simple biblioteca en línea. Buscamos constantemente innovar con características únicas que mejoren la experiencia de lectura, manteniendo a nuestros usuarios enganchados y comprometidos a la lectura.

Con BookSphere, estamos construyendo un refugio virtual donde las páginas de los libros cobran vida. Bienvenido a un mundo de historias ilimitadas, todo en la punta de tus dedos. ¡Sumérgete en la esencia de BookSphere y deja que la aventura literaria comience!

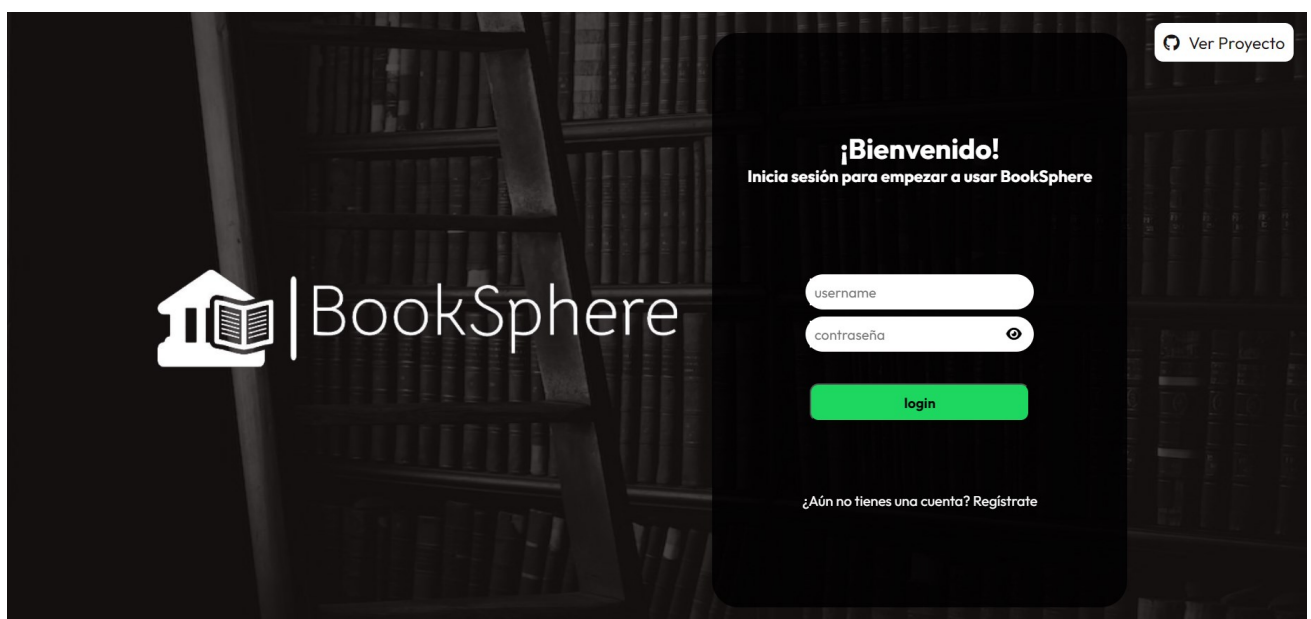
2. DEMOSTRACIÓN DE LA WEB

Autenticación:

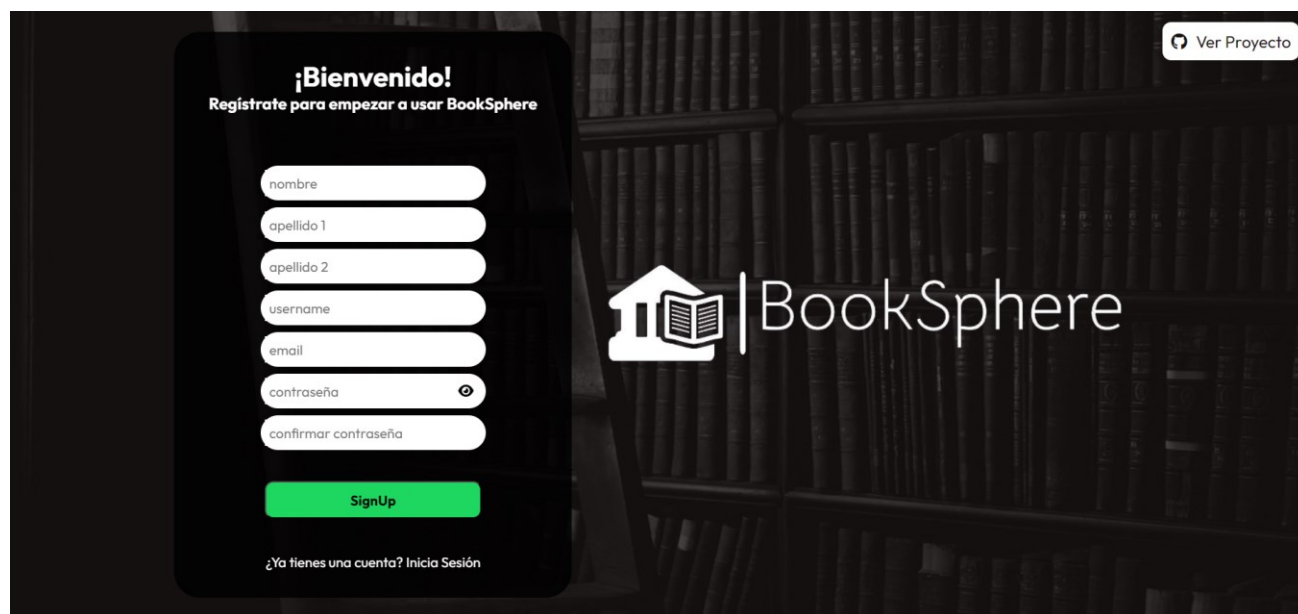
Para empezar, encontramos una página home, que brinda al usuario la posibilidad de iniciar sesión o realizar un registro con una cuenta nueva.



Si el usuario ha elegido la parte de inicio sesión, podrá acceder a la pantalla donde se realizará la comprobación de que el usuario es correcto.



En la parte de registro, el usuario deberá introducir los campos correctamente teniendo en cuenta expresiones regulares, que las contraseñas coincidan, que el nombre de usuario no exista, etc...

The image shows a registration form for 'BookSphere' overlaid on a background of a bookshelf. The form is titled '¡Bienvenido!' and 'Regístrate para empezar a usar BookSphere'. It contains input fields for 'nombre', 'apellido 1', 'apellido 2', 'username', 'email', 'contraseña' (with an eye icon for visibility), and 'confirmar contraseña'. A green 'SignUp' button is at the bottom of the form. Below the button, it says '¿Ya tienes una cuenta? Inicia Sesión'. To the right of the form, the 'BookSphere' logo is displayed, featuring a house icon with an open book inside. In the top right corner, there is a button labeled 'Ver Proyecto'.

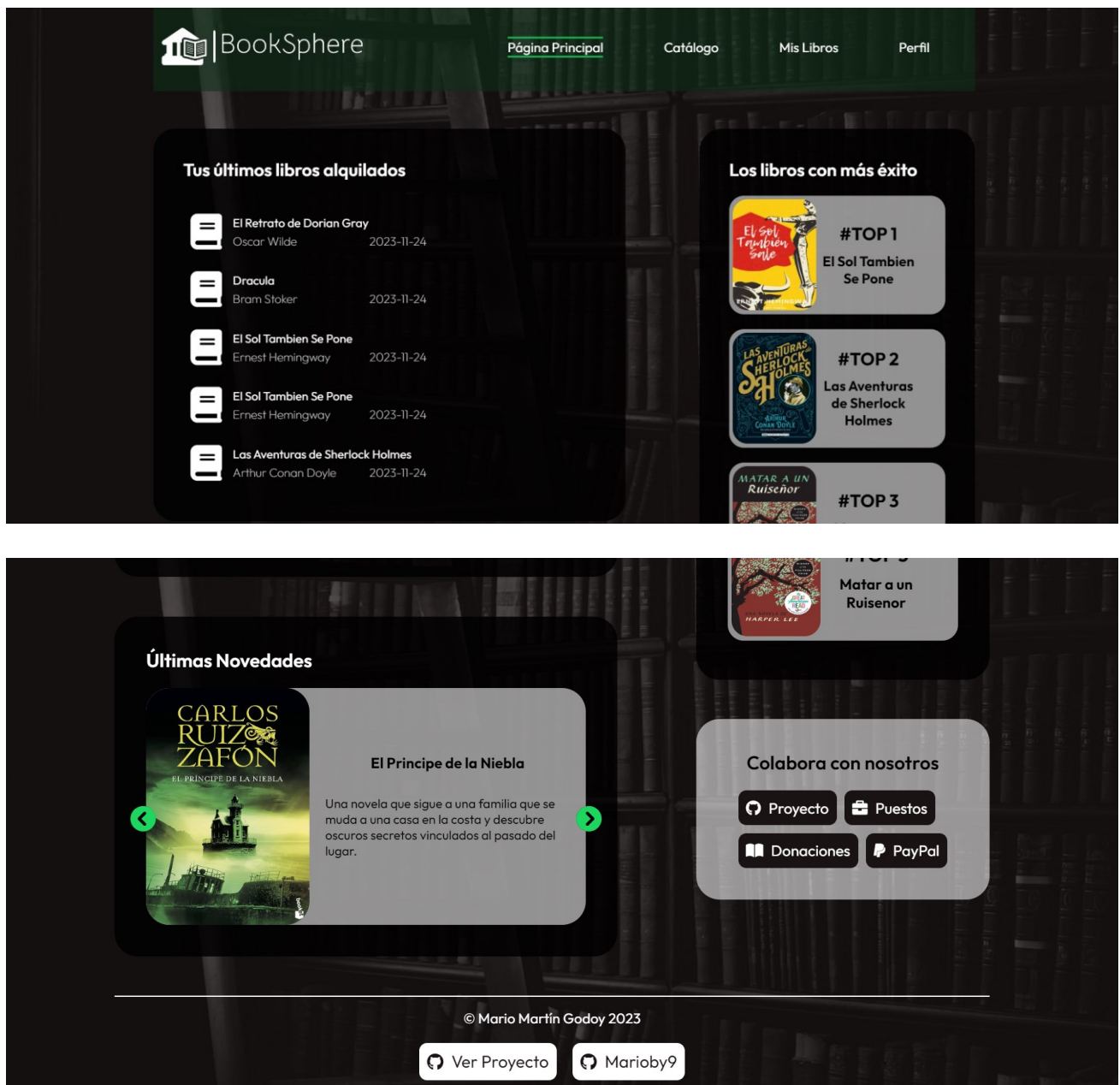
Una vez pasados todos los criterios de comprobación, se realizará una inserción a nuestra base de datos con los datos del usuario añadido. Una vez registrado, el usuario deberá iniciar sesión con sus credenciales para acceder a la aplicación.

Parte del cliente:

Dashboard

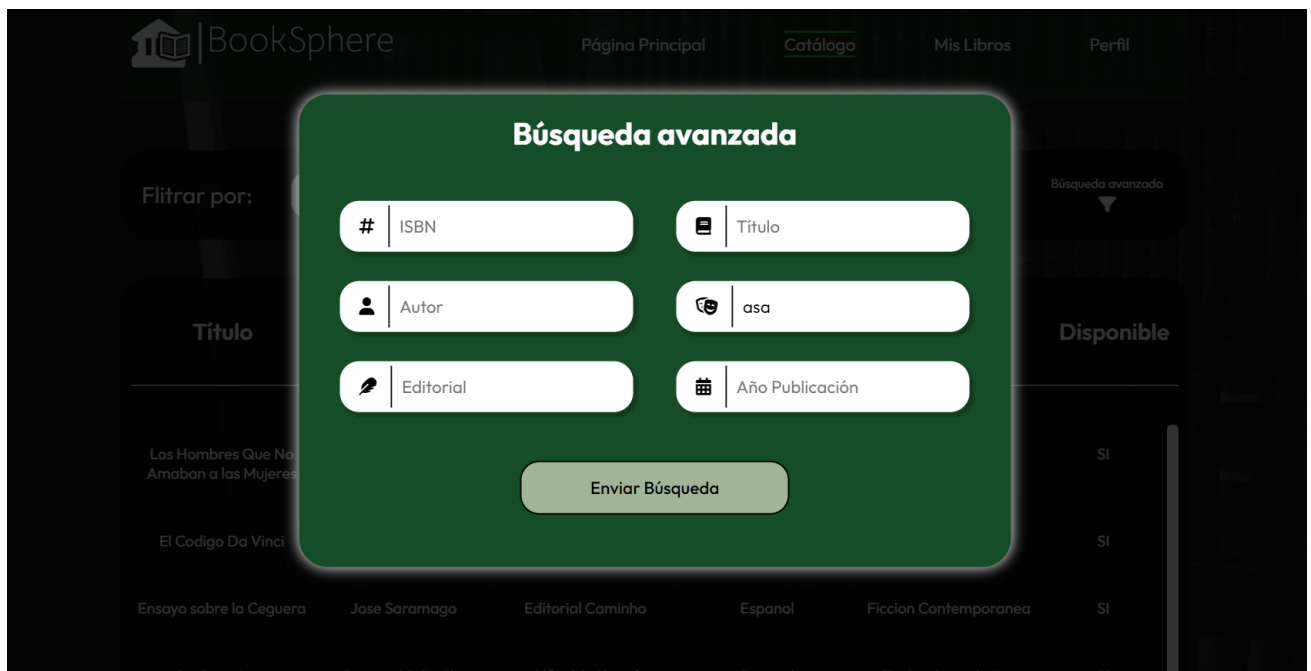
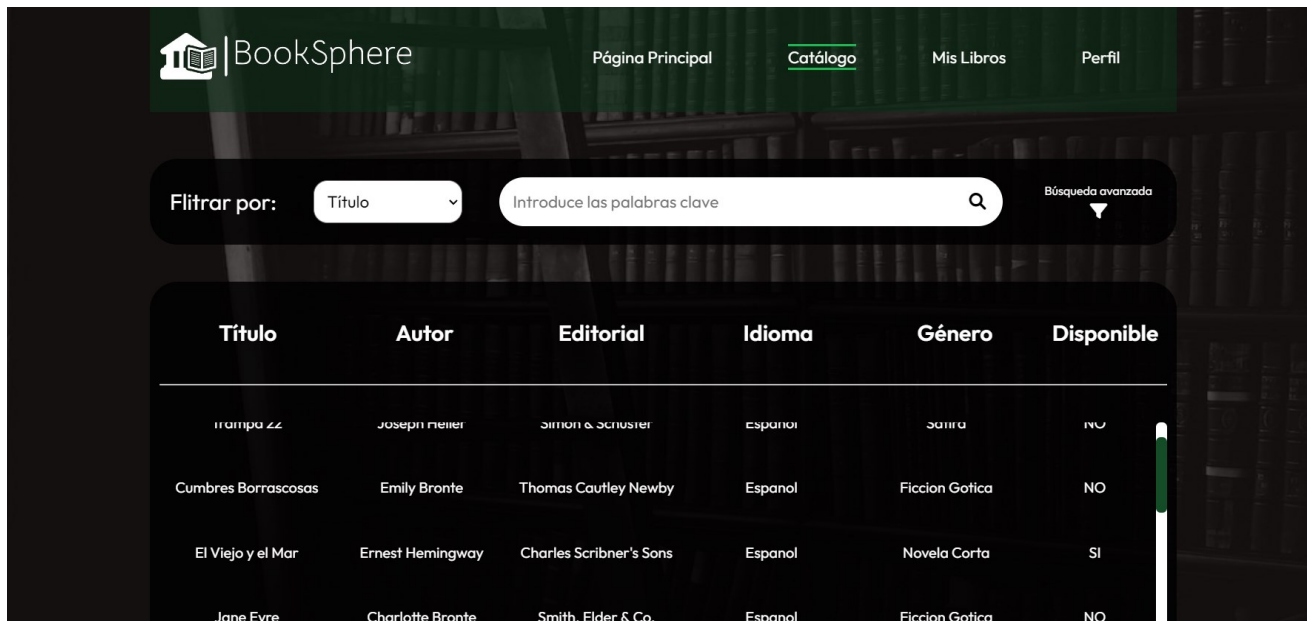
Una vez ha ingresado el cliente con sus datos autenticados, se le redirigirá al controlador automáticamente, y mediante un sistema de rutas de navegación, podrá acceder a todas las funcionalidades de la parte cliente. La navegación se realiza mediante un menú situado en el encabezado.

La primera página a la que accederá será el Dashboard (página principal). En la que encontramos varias secciones: Una sección de los 5 últimos libros que ha alquilado el usuario, una sección con los 3 libros más populares de la web y una sección con slider de los 3 últimos libros que han sido añadidos a nuestra base de datos. Por último, encontramos una sección con enlaces para colaborar con el proyecto, trabajar en la librería o donación de libros.



Catálogo

En la sección catálogo encontramos una lista con todos los libros que tenemos en la base de datos de BookSphere, mostrando los datos más relevantes de cada uno de ellos, además de su disponibilidad. El cliente encontrará una parte de búsqueda simple que le otorga la posibilidad de filtrar los libros en función de alguno de sus datos. Además existe la posibilidad de ejecutar un filtrado más directo a partir de todos sus datos (búsqueda avanzada). Al hacer click sobre uno de los libros, se nos redirigirá a la ficha técnica del libro, donde podremos alquilar o devolver el libro.



Ficha técnica de los libros

Cuando el usuario ha seleccionado un libro, ya sea en catálogo, en alguna de las secciones del dashboard o en la parte de mis libros, accederá a una ficha técnica de ese libro. En esta ficha técnica encontrará todos los datos del mismo junto a su portada y la opción de devolver (en caso de que el libro ya lo tenga alquilado), la opción de alquilar (en caso de que no lo tenga ni él ni ningún otro usuario), o un mensaje de no disponible (en caso de que él no lo tenga, pero sí otro usuario).



Sección Mis libros

En esta sección, el cliente tiene acceso a un seguimiento de los libros que está leyendo o ha leído alguna vez. Hay 3 secciones: Una con todos los libros que está leyendo actualmente, una con tarjetas de los 3 libros que más veces ha leído (alquilado) y una sección que le posibilita devolver o realquilar libros y tiene el mismo formato de búsqueda simple que el catálogo.

Leyendo actualmente: 7

	El Principe de la Niebla Carlos Ruiz Zafon	Novela Juvenil	2023-11-22
	El Retrato de Dorian Gray Oscar Wilde	Ficcion Gotica	2023-11-24
	El Sol Tambien Se Pone Ernest Hemingway	Literatura Modernista	2023-11-24

Tus favoritos



Todos tus libros

Todos tus libros

Filtrar por:

Título

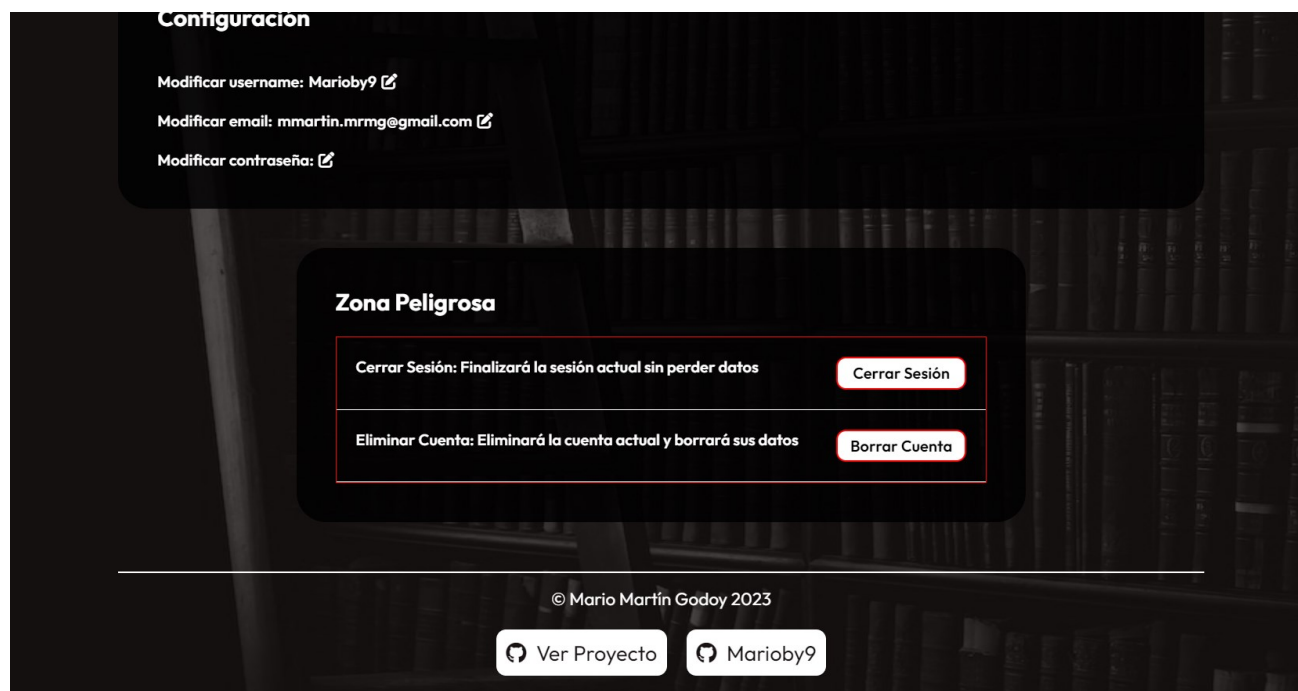
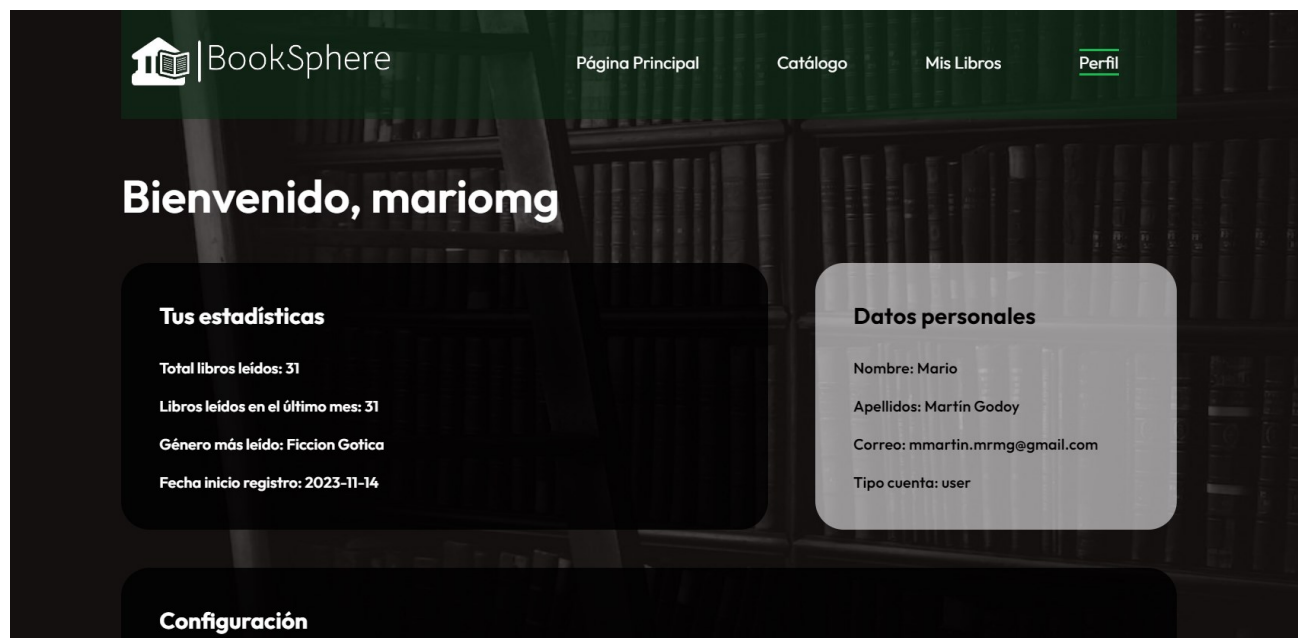
Introduce las palabras clave



Título	Autor	Género	Fecha Inicio	Fecha Final	Devolver
Trampa 22	Joseph Heller	Satira	2023-11-14	2023-11-14	<button>realquilar</button>
El Sol Tambien Se Pone	Ernest Hemingway	Literatura Modernista	2023-11-14	2023-11-14	<button>realquilar</button>
Las Aventuras de Sherlock Holmes	Arthur Conan Doyle	Misterio	2023-11-14	2023-11-14	<button>realquilar</button>
Cien Anos de Soledad	Gabriel Garcia Marquez	Realismo Magico	2023-11-16	2023-11-16	<button>realquilar</button>

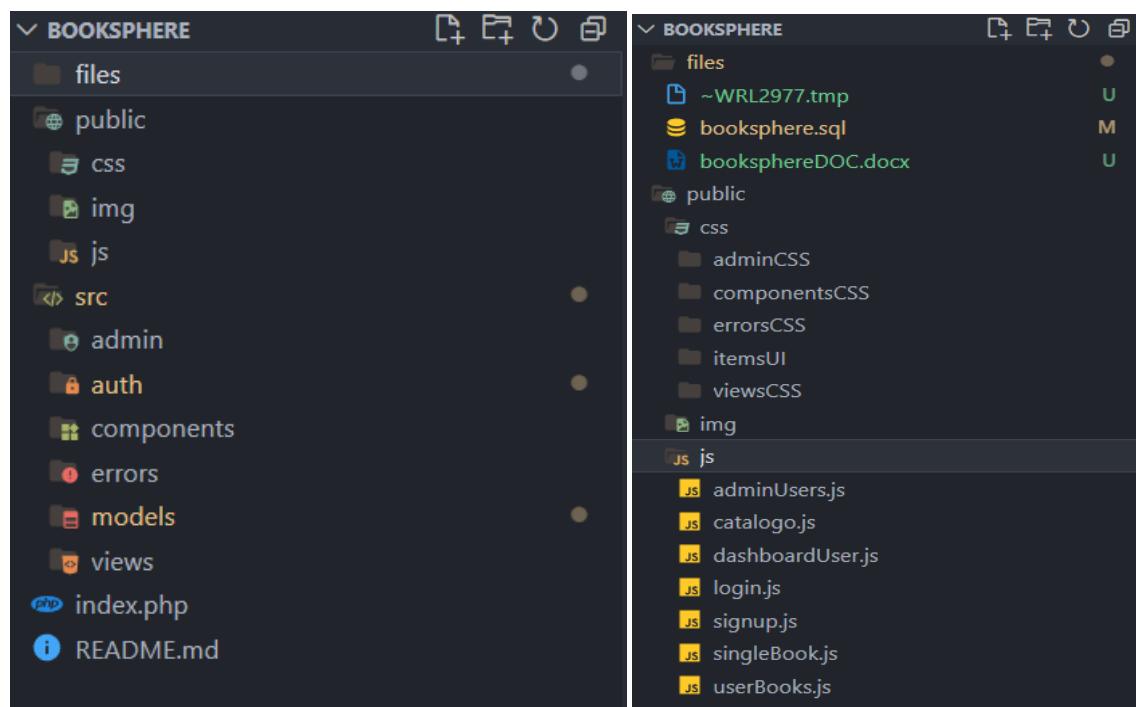
Perfil

En este apartado, el usuario tiene acceso a sus datos y estadísticas dentro de la aplicación. Además, encontrará un apartado configuración donde podrá modificar algunos de sus datos. Por último, tendrá la posibilidad de cerrar sesión o eliminar cuenta (se añade una fecha de baja).



3. EXPLICACIÓN FUNCIONES

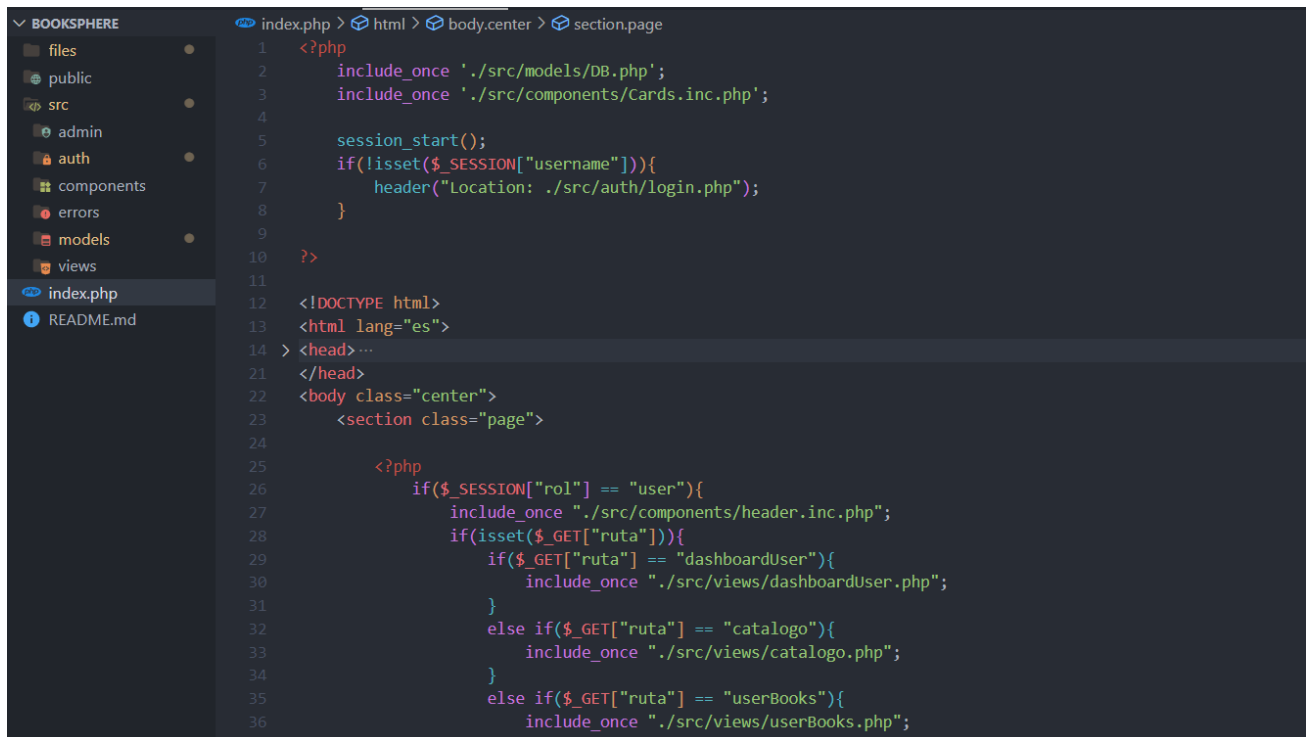
Para empezar, explicaré el sistema de organización elegido para los directorios y ficheros del proyecto:



Encontramos el controlador: index.php, a su altura un directorio src en el que están todas las subcarpetas de desarrollo del proyecto. Una carpeta files en la que están el script de SQL a importar y el fichero con la documentación. Además, encontramos la carpeta public con los archivos CSS (divididos en subcarpetas), la carpeta img con los logos e iconos utilizados en la web y la carpeta JS con todos los archivos JavaScript utilizados. Por último, el README.md del proyecto con toda la información para el despliegue del proyecto.

Carpeta SRC: Contiene las subcarpetas admin (con todas las vistas de la parte administrador), auth (login, home, signup), components (con los componentes independientes utilizados: header, footer y overlay), además de una clase Cards que contiene librerías propias para manejar los mensajes de error. Errors contiene las páginas a las que se dirige en caso de que la ruta de acceso no exista (404) o en caso de que haya acceso restringido. Models contiene la clase DB con todas las funciones de base de datos, la clase regex con todas las funciones de expresiones regulares, y por último, un controlador de logout y de deleteAccount.

INDEX.PHP

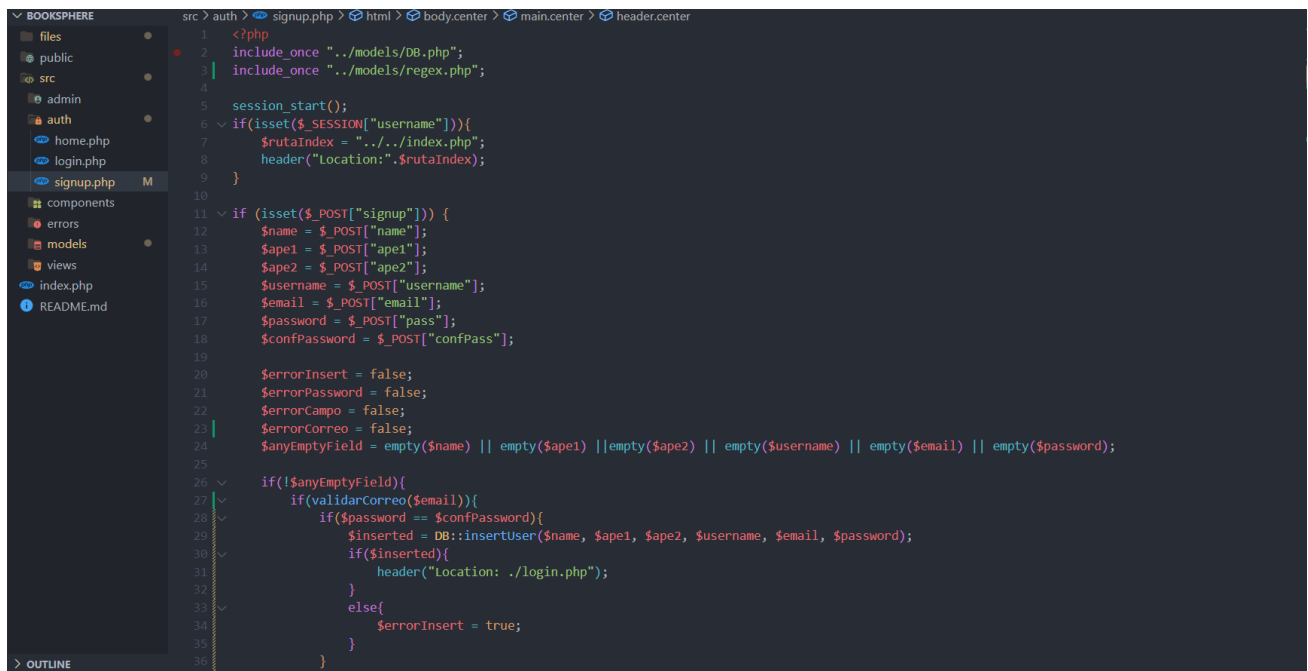


```
1 <?php
2 include_once './src/models/DB.php';
3 include_once './src/components/Cards.inc.php';
4
5 session_start();
6 if(!isset($_SESSION["username"])){
7     header("Location: ./src/auth/login.php");
8 }
9
10 ?>
11
12 <!DOCTYPE html>
13 <html lang="es">
14 <head>...
15 </head>
16 <body class="center">
17     <section class="page">
18
19         <?php
20             if($_SESSION["rol"] == "user"){
21                 include_once './src/components/header.inc.php';
22                 if(isset($_GET["ruta"])){
23                     if($_GET["ruta"] == "dashboardUser"){
24                         include_once './src/views/dashboardUser.php';
25                     }
26                     else if($_GET["ruta"] == "catalogo"){
27                         include_once './src/views/catalogo.php';
28                     }
29                     else if($_GET["ruta"] == "userBooks"){
30                         include_once './src/views/userBooks.php';
31                     }
32                 }
33             }
34         ?>
35     </section>
36 </body>
37 </html>
```

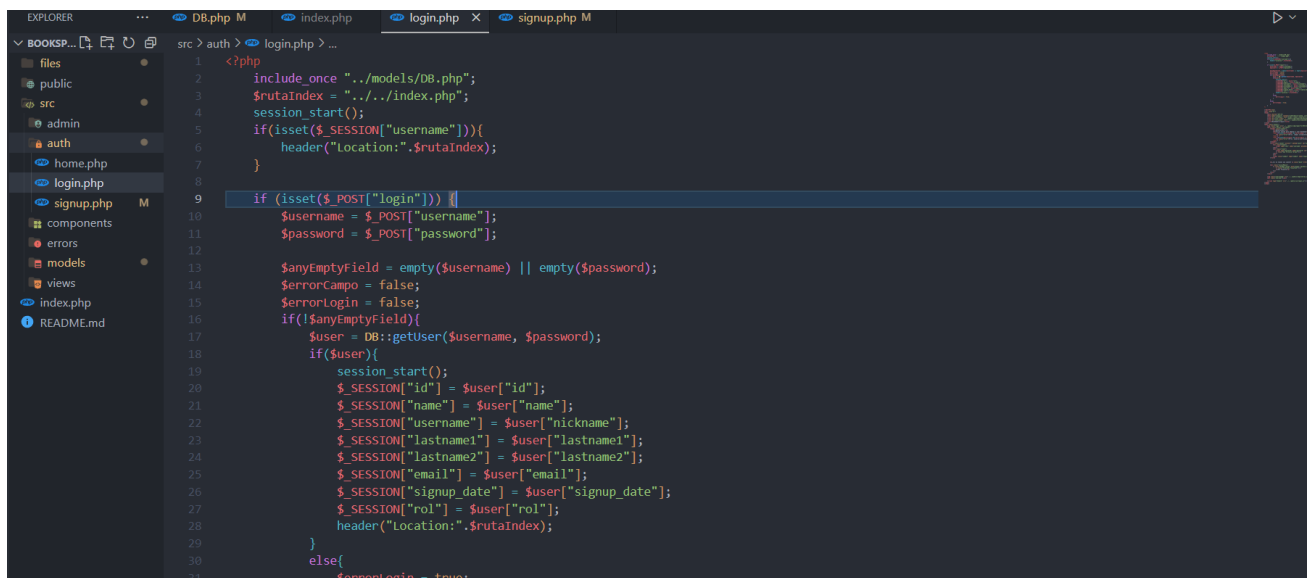
Encontramos el controlador de rutas que comprueba si hay sesión iniciada o no. En caso negativo, redirige al login para iniciar sesión. Si la sesión ya está iniciada, comprueba si el tipo de cuenta es usuario o admin. Si es, usuario la aplicación maneja las posibilidades de todas las rutas a las que puede acceder el usuario, lo mismo para el admin. Si uno de los dos intenta acceder a una ruta del otro, salta error de acceso y no se permite.



AUTH



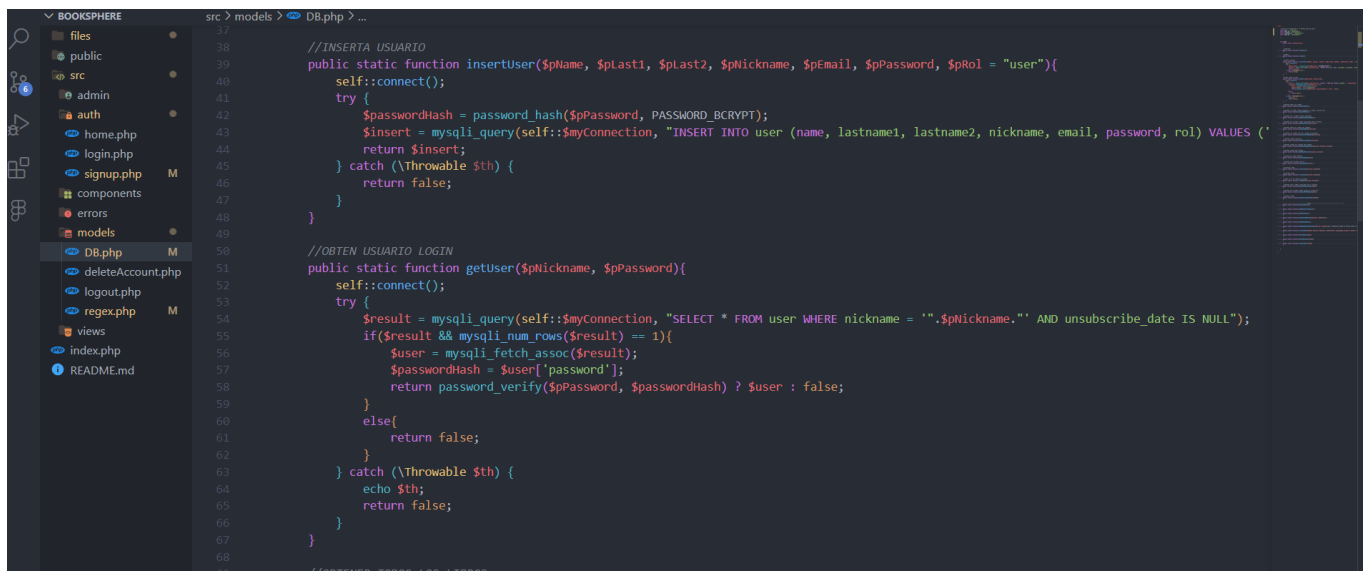
```
1 <?php
2 include_once "../models/DB.php";
3 include_once "../models/regex.php";
4
5 session_start();
6 if(isset($_SESSION["username"])){
7     $rutaIndex = "../index.php";
8     header("Location:".$rutaIndex);
9 }
10
11 if (isset($_POST["signup"])) {
12     $name = $_POST["name"];
13     $ape1 = $_POST["ape1"];
14     $ape2 = $_POST["ape2"];
15     $username = $_POST["username"];
16     $email = $_POST["email"];
17     $password = $_POST["pass"];
18     $confPassword = $_POST["confPass"];
19
20     $errorInsert = false;
21     $errorPassword = false;
22     $errorCampo = false;
23     $errorCorreo = false;
24     $anyEmptyField = empty($name) || empty($ape1) || empty($ape2) || empty($username) || empty($email) || empty($password);
25
26     if(!$anyEmptyField){
27         if(validarCorreo($email)){
28             if($password == $confPassword){
29                 $inserted = DB::insertUser($name, $ape1, $ape2, $username, $email, $password);
30                 if($inserted){
31                     header("Location: ../login.php");
32                 }
33             } else{
34                 $errorInsert = true;
35             }
36         }
37     }
```



```
1 <?php
2 include_once "../models/DB.php";
3 $rutaIndex = "../index.php";
4 session_start();
5 if(isset($_SESSION["username"])){
6     header("Location:".$rutaIndex);
7 }
8
9 if (isset($_POST["login"])) {
10     $username = $_POST["username"];
11     $password = $_POST["password"];
12
13     $anyEmptyField = empty($username) || empty($password);
14     $errorCampo = false;
15     $errorLogin = false;
16     if(!$anyEmptyField){
17         $user = DB::getUser($username, $password);
18         if($user){
19             session_start();
20             $_SESSION["id"] = $user["id"];
21             $_SESSION["name"] = $user["name"];
22             $_SESSION["username"] = $user["nickname"];
23             $_SESSION["lastname1"] = $user["lastname1"];
24             $_SESSION["lastname2"] = $user["lastname2"];
25             $_SESSION["email"] = $user["email"];
26             $_SESSION["signup_date"] = $user["signup_date"];
27             $_SESSION["rol"] = $user["rol"];
28             header("Location:".$rutaIndex);
29         }
30     } else{
31         $errorLogin = true;
32     }
```

En signup comprobamos que todos los campos estén rellenos y que el email cumple con la expresión regular de la clase REGEX.php (validarCorreo). Si todo se cumple correctamente, se registra. En login comprobamos que los campos están rellenos, hacemos una consulta a la base de datos para comprobar que los datos coinciden con el usuario y dejamos pasar con un header al index ya con la sesión creada.

Además, es necesario utilizar la función passwordhash para el registro y cifrar la contraseña. Y también es necesaria la función passwordVerify para hacer la comprobación del login.



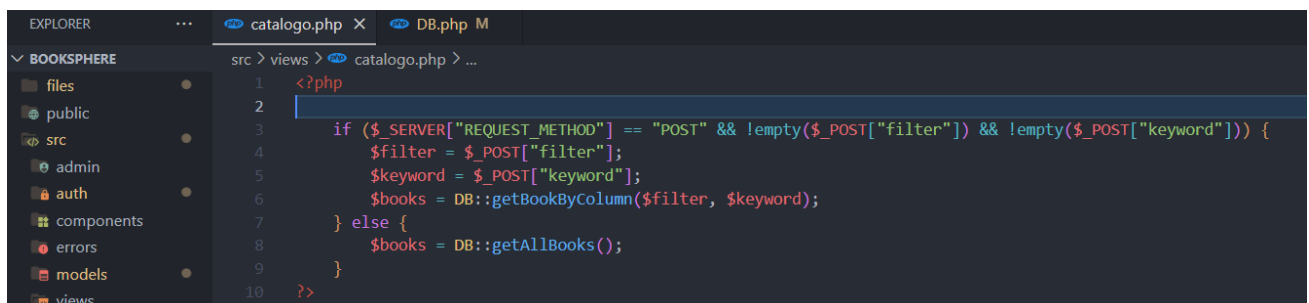
```
37
38 //INSERTA USUARIO
39 public static function insertUser($pName, $pLast1, $pLast2, $pNickname, $pEmail, $pPassword, $pRol = "user"){
40     self::connect();
41     try {
42         $passwordHash = password_hash($pPassword, PASSWORD_BCRYPT);
43         $insert = mysqli_query(self::$myConnection, "INSERT INTO user (name, lastname1, lastname2, nickname, email, password, rol) VALUES ('
44         return $insert;
45     } catch (\Throwable $th) {
46         return false;
47     }
48 }
49
50 //OBTEN USUARIO LOGIN
51 public static function getUser($pNickname, $pPassword){
52     self::connect();
53     try {
54         $result = mysqli_query(self::$myConnection, "SELECT * FROM user WHERE nickname = '" . $pNickname . "' AND unsubscribe_date IS NULL");
55         if($result && mysqli_num_rows($result) == 1){
56             $user = mysqli_fetch_assoc($result);
57             $passwordHash = $user['password'];
58             return password_verify($pPassword, $passwordHash) ? $user : false;
59         }
60         else{
61             return false;
62         }
63     } catch (\Throwable $th) {
64         echo $th;
65         return false;
66     }
67 }
68
69 //OBTENER TODOS LOS LIBROS
```

Búsqueda simple y avanzada

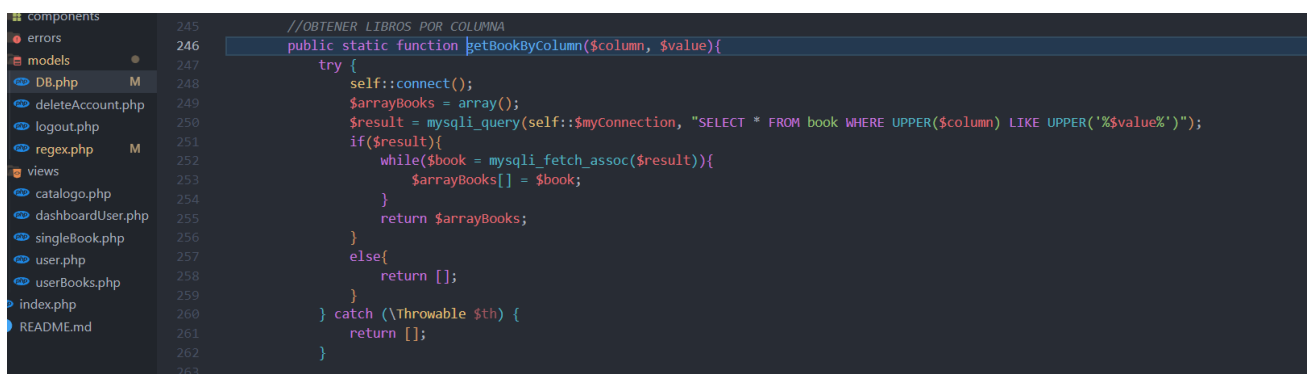
En BookSphere, hemos implementado dos tipos de búsqueda:

Una búsqueda simple que permite al usuario filtrar los libros del catálogo según uno de sus campos.

Una búsqueda avanzada más directa en la que se especifican todos los campos.



```
1 <?php
2
3 if ($_SERVER["REQUEST_METHOD"] == "POST" && !empty($_POST["filter"]) && !empty($_POST["keyword"])) {
4     $filter = $_POST["filter"];
5     $keyword = $_POST["keyword"];
6     $books = DB::getBookByColumn($filter, $keyword);
7 } else {
8     $books = DB::getAllBooks();
9 }
10 ?>
```



```
245 //OBTENER LIBROS POR COLUMNA
246 public static function getBookByColumn($column, $value){
247     try {
248         self::connect();
249         $arrayBooks = array();
250         $result = mysqli_query(self::$myConnection, "SELECT * FROM book WHERE UPPER($column) LIKE UPPER('%$value%')");
251         if($result){
252             while($book = mysqli_fetch_assoc($result)){
253                 $arrayBooks[] = $book;
254             }
255             return $arrayBooks;
256         }
257         else{
258             return [];
259         }
260     } catch (\Throwable $th) {
261         return [];
262     }
263 }
```


Librerías extra añadidas

Clase Cards

Esta clase sirve para manejar el sistema de mensajes de error. Encontramos 3 funciones que devuelven un componente HTML que se renderizará al llamar a dicha función. Las 3 funciones reciben un parámetro mensaje, que será el que se muestre en la tarjeta renderizada. Correcto (verde), Error (rojo), Alerta (naranja).

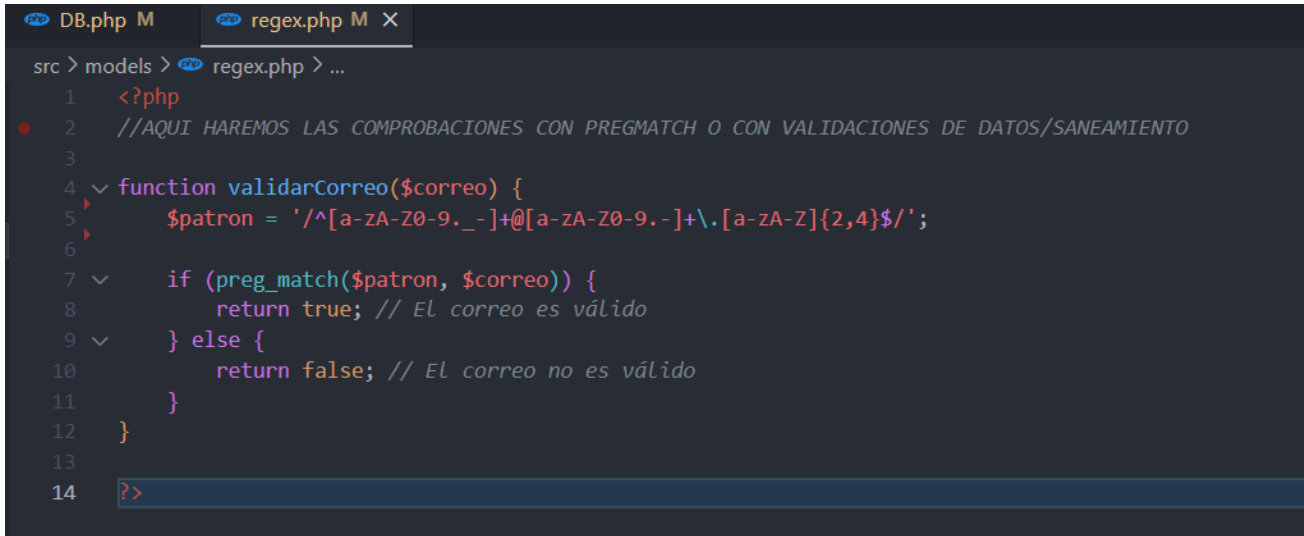
```
src > components > Cards.inc.php > Cards > errorCard
1  <?php
2      class Cards{
3
4          public static function errorCard($message){
5              return '
6                  <link rel="stylesheet" href="./public/css/componentsCSS/cards.css">
7                  <div class="CardsCard errorCard">
8                      <p><strong>Error:</strong> ' . $message . '</p>
9                  </div>';
10             }
11
12             public static function correctCard($message){
13                 return '
14                     <link rel="stylesheet" href="./public/css/componentsCSS/cards.css">
15                     <div class="CardsCard correctCard">
16                         <p><strong>Correcto:</strong> ' . $message . '</p>
17                     </div>';
18                 }
19
20             public static function alertCard($message){
21                 return '
22                     <link rel="stylesheet" href="./public/css/componentsCSS/cards.css">
23                     <div class="CardsCard alertCard">
24                         <p><strong>Alerta:</strong> ' . $message . '</p>
25                     </div>';
26                 }
27
28             }
29
30
31  ?>
```

```
<main>
    <link rel="stylesheet" href="./public/css/adminCSS/adminSingleBook.css">
    <?php if(isset($deleted)){
        if($deleted){
            echo Cards::correctCard("Libro eliminado correctamente.");
        }
        else{
            echo Cards::errorCard("No se ha conseguido borrar el libro.");
        }
    } ?>
```

Ejemplo de llamada a Cards para manejar los errores posibles cuando un administrador trata de borrar un libro de la base de datos.

REGEX.PHP

Esta librería guarda todas las funciones que contienen verificación de expresiones regulares, por ejemplo, la verificación de correo al registrar usuario.



```
src > models > regex.php > ...
1  <?php
2  //AQUI HAREMOS LAS COMPROBACIONES CON PREGMATCH O CON VALIDACIONES DE DATOS/SANEAMIENTO
3
4  function validarCorreo($correo) {
5      $patron = '/^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/' ;
6
7      if (preg_match($patron, $correo)) {
8          return true; // El correo es válido
9      } else {
10         return false; // El correo no es válido
11     }
12 }
13
14 ?>
```

4. TECNOLOGÍAS UTILIZADAS

En el desarrollo de esta aplicación web, he utilizado las siguientes herramientas:

Un entorno de desarrollo de código: Visual Studio Code con diversos plugins para facilitar y agilizar el trabajo en los lenguajes utilizados.

4 Lenguajes de programación: HTML para la estructura base de la web, CSS para estilar los diseños, JavaScript para aplicar funcionalidades superficiales a botones, animaciones y demás. Y, por último, PHP, para aplicar la lógica al sistema de control del servidor.

Una aplicación para el control de versiones: GIT y GITHUB.

Git lo he utilizado para subir todo el proyecto en su respectiva versión mediante comandos por consola. Y en GitHub se encuentra el repositorio en la nube al que se suben los archivos.