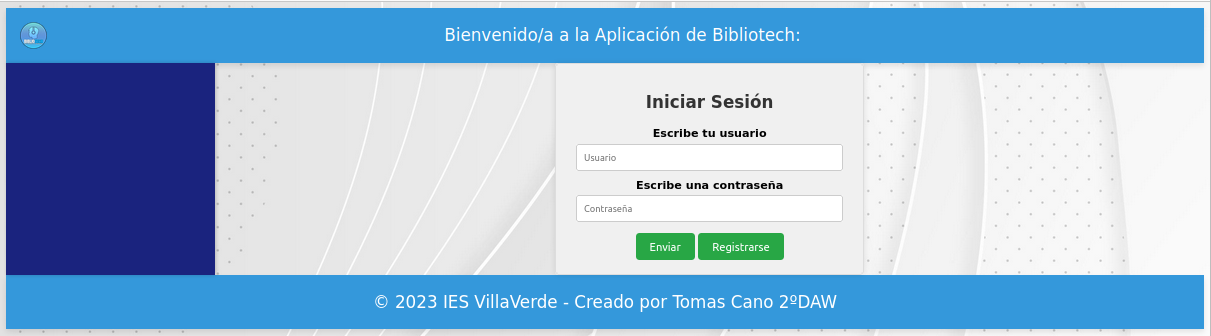
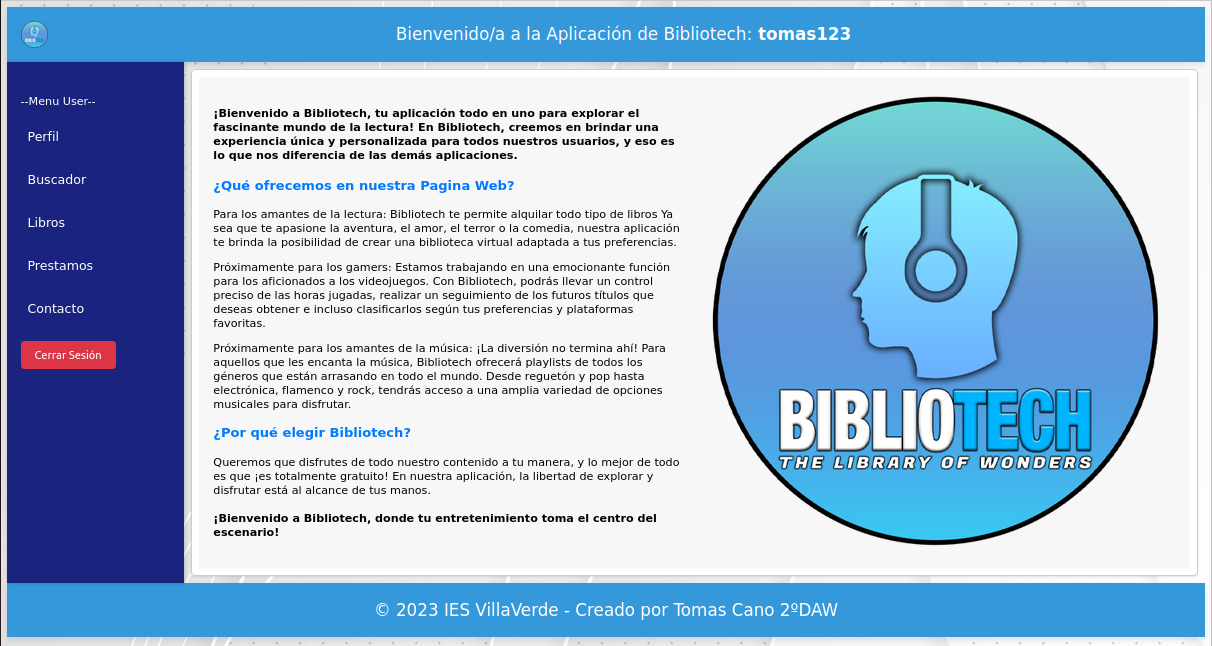
  
  
**PROYECTO BIBLIOTECH**

**ÍNDICE**

1. **¿Qué es y cómo funciona Bibliotech?**
2. **Ficheros importantes de la aplicación:**
   1. **Funciones.php**
   2. **Index.php**
   3. **Registrar.inc.php**
   4. **Login.inc.php**
   5. **Menu.inc.php**
3. **Resto de ficheros de la aplicación:**
   1. **Buscador.inc.php**
   2. **Crearlibros.inc.php**
   3. **Editarlibro.inc.php**
   4. **Eliminarlibro.inc.php**
   5. **Restaurarlibro.inc.php**
   6. **Prestamos.inc.php**
4. **Fichero CSS**
5. **Fichero BBDD**
6. **Conclusión y Justificación**
7. **Biografía**

**1- ¿Qué es y cómo funciona Bibliotech?**Bibliotech es una aplicación creada con los lenguajes de programación de PHP, HTML, CSS y JavaScript la cual tiene su login y registro para poder acceder a ella. Dentro de la aplicación tenemos muchas pestañas con las que podemos interactuar y realizar muchas operaciones. Es una aplicación sencilla, básica y fácil de moverse por la aplicación web.  
  
A continuación, una pequeña vista de lo que tiene Bibliotech en el entorno cliente:  


**2- Ficheros importantes de la aplicación**  
Funciones.php:  
Function HashPassword($password) {}  
→ Esta función en PHP, llamada `hashPassword`, utiliza la función `password\_hash` que de manera resumida, devuelve un hash seguro de la contraseña proporcionada.

Function verificarInicioSesion($usuario, $password) {}  
→ Preparación de la consulta SQL: La función prepara una consulta SQL para seleccionar información de usuario (id, usuario, contrasena, rol) basada en el nombre de usuario proporcionado.

→ Ejecución de la consulta: La función ejecuta la consulta, vincula los resultados a variables y recupera la información del usuario desde la base de datos.

→ Verificación de la contraseña: Utiliza la función `password\_verify` para verificar si la contraseña proporcionada coincide con el hash almacenado en la base de datos.

→ Retorno de resultados: Si la verificación de la contraseña es exitosa, la función devuelve un arreglo asociativo con la información del usuario. En caso contrario, devuelve false.

→ Cierre de la declaración preparada: La función cierra la declaración preparada después de su ejecución.

Function registrarUsuario($nombre, $apellido1, $apellido2, $correo, $usuario, $contrasena, $rol) {}

→ Hashing de la contraseña: La función utiliza la función hashPassword para generar un hash seguro de la contraseña antes de almacenarla en la base de datos.

→ Preparación de la consulta SQL: La función prepara una consulta SQL para insertar nuevos registros en la tabla de usuarios con la información proporcionada.

→ Vinculación de parámetros y ejecución: La función vincula los parámetros de la consulta a las variables proporcionadas como argumentos, ejecuta la consulta y devuelve el resultado de la ejecución.

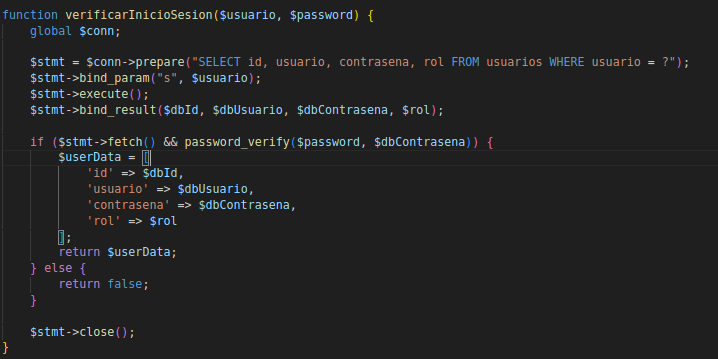
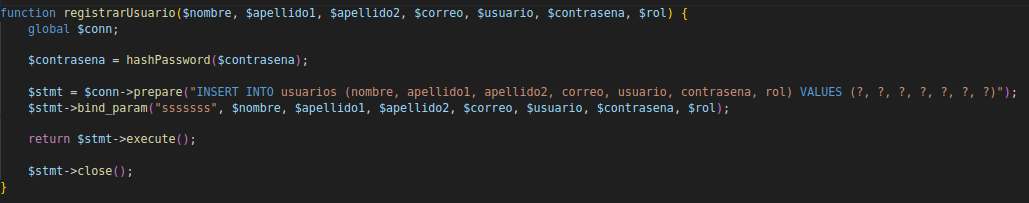
→ Retorno del resultado de la ejecución: La función devuelve true si la inserción en la base de datos fue exitosa, y false en caso contrario.

→ Cierre de la declaración preparada: Es importante señalar que el cierre de la declaración preparada ($stmt->close()) nunca se alcanzará debido al return que precede a esa línea. El código después del return no se ejecutará.

HassPassword:



VerificarInicioSesion:

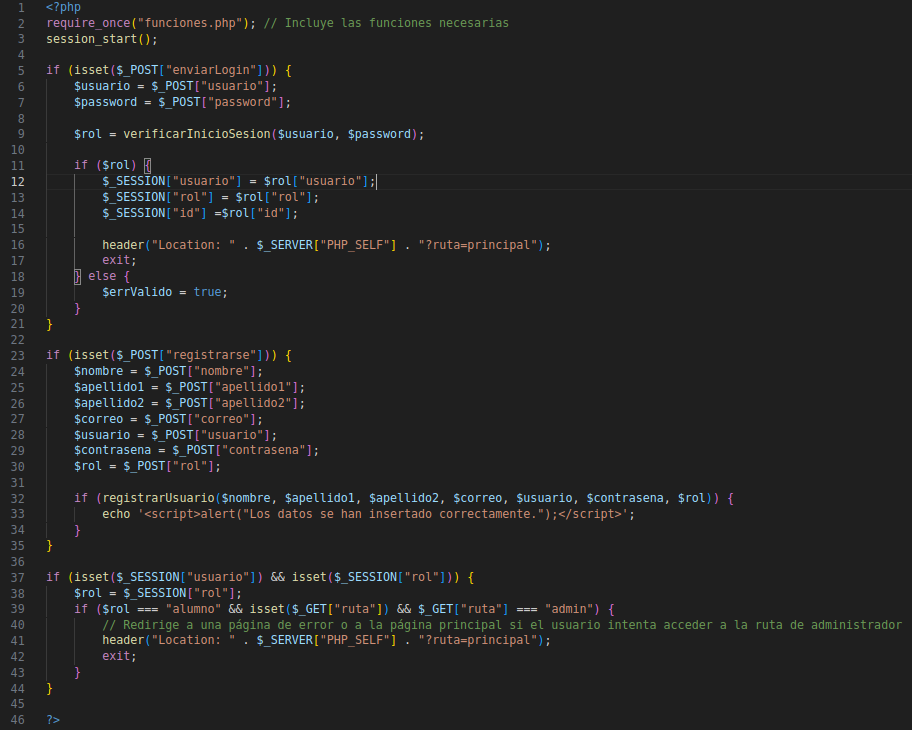
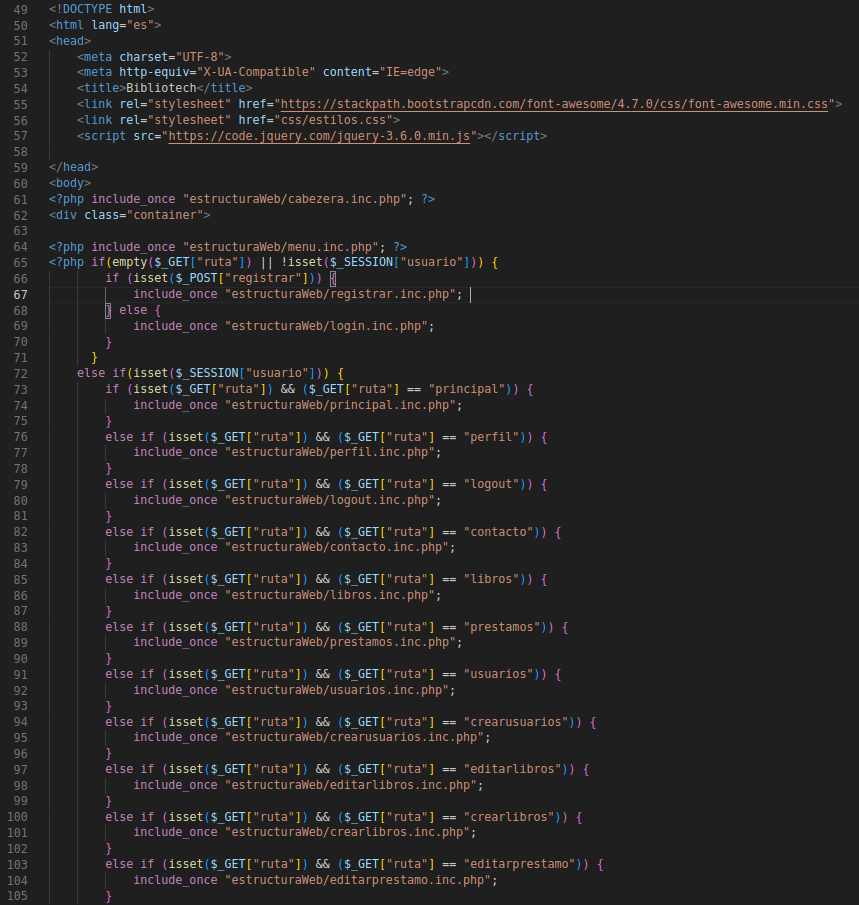
  
RegistrarUsuario:  


Index.php:  
Este código PHP se encarga de manejar la lógica del inicio de sesión, registro de usuarios y enrutamiento de páginas web:

→Manejo de sesiones: Inicia la sesión y verifica si el formulario de inicio de sesión ha sido enviado (`isset($\_POST["enviarLogin"])`) para autenticar al usuario utilizando la función `verificarInicioSesion`.

→Registro de usuarios: Si se ha enviado el formulario de registro (`isset($\_POST["registrarse"])`), recopila los datos del formulario y utiliza la función `registrarUsuario` para insertar un nuevo usuario en la base de datos.

→Redirección después de inicio de sesión: Si la autenticación es exitosa, se establecen variables de sesión y se redirige al usuario a la página principal.  
→Manejo de roles: Verifica el rol del usuario y redirige si es necesario (por ejemplo, si un alumno intenta acceder a la ruta de administrador).

→Páginas HTML: La parte final del código representa la estructura HTML de la página web, incluyendo la inclusión de diferentes fragmentos de código dependiendo de la ruta y la sesión del usuario.  
Parte de Login y Registro de index.php:  
  
Parte de Redirecciones de la aplicación:  


Registrar.inc.php:  
Este código PHP maneja el registro de usuarios a través de un formulario HTML. Aquí tienes una descripción resumida:

→ Registro de usuarios: Verifica si el formulario de registro ha sido enviado (`isset($\_POST["registrarse"])`). Si es así, recopila los datos del formulario y utiliza la función `registrarUsuario` para insertar un nuevo usuario en la base de datos.

→ Alerta de éxito: Si el registro es exitoso, muestra una alerta JavaScript indicando que los datos se han insertado correctamente.

→ Formulario HTML de registro: Muestra un formulario HTML que recopila información como nombre, apellidos, correo electrónico, usuario, contraseña y rol del usuario. El formulario incluye opciones para seleccionar el rol (en este caso, "alumno") y un botón de registro.

→ Formulario de inicio de sesión: También se presenta un segundo formulario que redirige a "index.php" cuando se envía, pero este fragmento de código no maneja la lógica de inicio de sesión aquí.

Parte Registar.php:  


Login.inc.php:  
Este código PHP maneja el inicio de sesión a través de un formulario HTML. Aquí tienes una descripción resumida:

→ Inicio de sesión: Verifica si el formulario de inicio de sesión ha sido enviado (`isset($\_POST["enviarLogin"])`). Si es así, recopila los datos del formulario (`$usuario` y `$password`) y utiliza la función `verificarInicioSesion` para autenticar al usuario.

→ Manejo de errores: Si la autenticación falla, establece la variable `$errValido` a `true` para mostrar un mensaje de error en el formulario HTML.

→ Formulario HTML de inicio de sesión: Muestra un formulario HTML que recopila el nombre de usuario y la contraseña. Si se ha establecido un error (`$errValido`), se muestra un mensaje de error en el formulario.

→ Botón de registro: Además del botón para enviar el formulario de inicio de sesión (`enviarLogin`), hay un segundo botón para registrarse (`registrar`). Sin embargo, en el código proporcionado, no hay una lógica específica para manejar el registro. Puede ser necesario implementar una funcionalidad adicional para el registro.

→ Redirección después del inicio de sesión: Si la autenticación es exitosa, se establecen variables de sesión (`$\_SESSION["usuario"]` y `$\_SESSION["rol"]`) y se redirige al usuario a la página principal.

Parte Login.inc.php:  

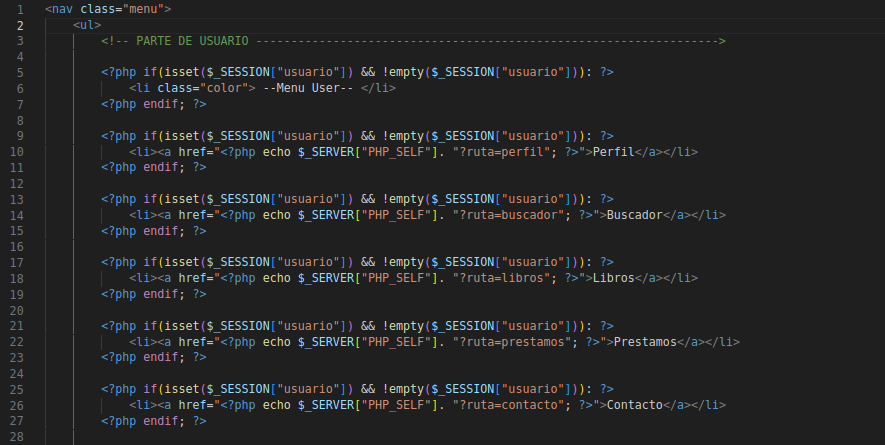

Menu.inc.php:  
Este código PHP maneja la generación dinámica de un menú de navegación basado en la sesión del usuario y su rol. Aquí tienes una descripción resumida:

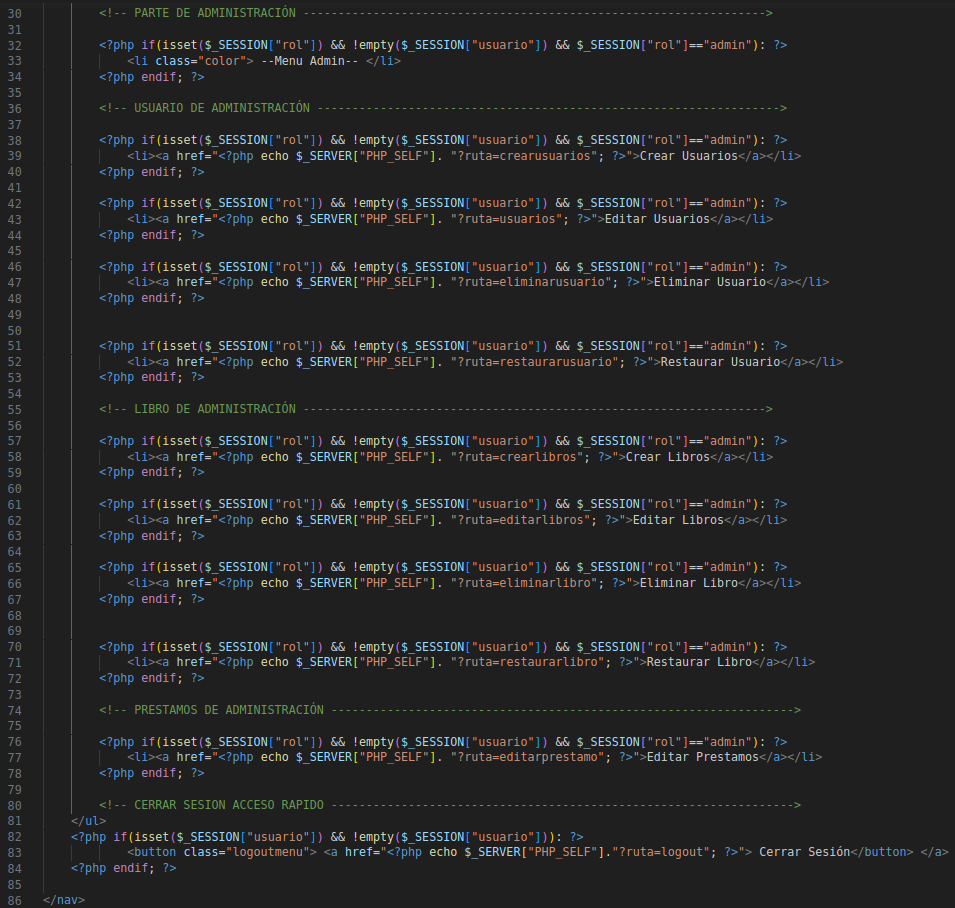
→ Menú del Usuario: Muestra diferentes elementos del menú si hay un usuario autenticado (`isset($\_SESSION["usuario"]) && !empty($\_SESSION["usuario"])`). Estos elementos incluyen enlaces a las páginas de "Perfil", "Buscador", "Libros", "Préstamos" y "Contacto".

→ Menú de Administrador: Si el usuario tiene el rol de administrador (`$\_SESSION["rol"]=="admin"`), se muestran elementos adicionales en el menú, incluyendo enlaces a las páginas de administración como "Crear Usuarios", "Editar Usuarios", "Eliminar Usuario", "Restaurar Usuario", "Crear Libros", "Editar Libros", "Eliminar Libro", "Restaurar Libro" y "Editar Préstamos".

→ Botón de Cerrar Sesión: Se muestra un botón para cerrar sesión si hay un usuario autenticado. El botón redirige al usuario a la página de cerrar sesión (`?ruta=logout`).

Parte Usuario Menu.inc.php:



Parte Admin Menu.inc.php:  


**3- Resto de ficheros de la aplicación**Buscador.inc.php:  
Este código PHP maneja un formulario de búsqueda de libros y muestra los resultados en una tabla HTML. Aquí tienes una descripción resumida:

→ Procesamiento del formulario: Si el método de solicitud es POST (`$\_SERVER["REQUEST\_METHOD"] == "POST"`), se recopilan y limpian los datos del formulario utilizando la función `limpiarEntrada` y se realiza una búsqueda de libros utilizando la función `buscarLibros`. El resultado se almacena en la variable `$librosFiltrados`.

→ Formulario de Búsqueda: Muestra un formulario HTML con campos para el ISBN, título, autor y editorial, y un botón de búsqueda.

→ Mostrar Resultados: Dependiendo de si hay resultados en `$librosFiltrados`, se muestra una tabla HTML con la información de los libros filtrados. Si no hay resultados, se muestra una tabla vacía.

→ Mensajes predeterminados: Si no se ha enviado el formulario o no hay resultados, se muestra una tabla vacía como mensaje predeterminado.

Parte Buscador.inc.php:  


Crearlibros.inc.php:  
Este código PHP maneja la creación de un nuevo libro a través de un formulario, pero solo permite el acceso a usuarios con el rol de administrador. Aquí tienes una descripción resumida:

→ Verificación de Permisos: Verifica si el usuario tiene el rol de administrador (`isset($\_SESSION['rol']) && $\_SESSION['rol'] === 'admin'`). Si el usuario no tiene el rol de administrador, muestra un mensaje indicando que no tiene permisos.

→ Procesamiento del Formulario: Si el método de solicitud es POST (`$\_SERVER["REQUEST\_METHOD"] == "POST"`) y se ha enviado el formulario (`isset($\_POST["crearLibro"])`), recopila los datos del formulario (ISBN, título, autor, editorial) y utiliza las funciones `isbnExistente` y `crearLibro` para verificar la existencia del ISBN y crear el libro, respectivamente.

→Mensajes de Éxito o Error: Después de procesar el formulario, muestra mensajes de éxito o error mediante alertas de JavaScript. Si el libro se crea con éxito, se muestra una alerta informando al usuario y se recarga la página.

→ Formulario de Creación de Libro: Muestra un formulario HTML con campos para el ISBN, título, autor y editorial, y un botón para crear el libro.

→ Mensaje de Permiso Denegado: Si el usuario no tiene permisos de administrador, muestra un mensaje indicando que no tiene acceso a la página.

Parte Crearlibros.inc.php:  


Editarlibro.inc.php:  
Este código PHP maneja la visualización de todos los libros en una tabla, la modificación de un libro seleccionado y la actualización de la información del libro modificado. Aquí tienes una descripción resumida:

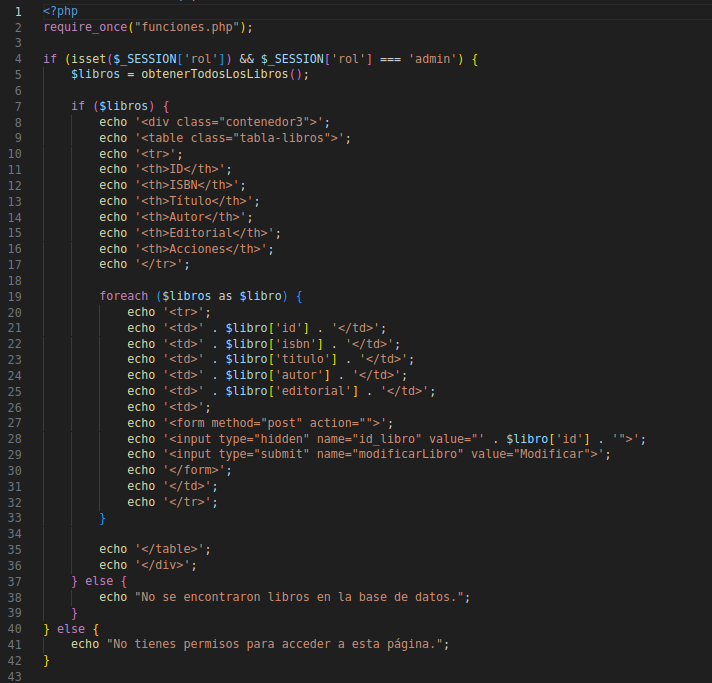
→ Verificación de Permisos: Verifica si el usuario tiene el rol de administrador (`isset($\_SESSION['rol']) && $\_SESSION['rol'] === 'admin'`). Si el usuario no tiene el rol de administrador, muestra un mensaje indicando que no tiene permisos.

→ Obtención de Libros: Obtiene todos los libros de la base de datos utilizando la función `obtenerTodosLosLibros()` y los muestra en una tabla HTML. Cada fila de la tabla incluye un botón "Modificar" que permite acceder a la opción de modificar el libro.

→ Modificación de Libro: Si se hace clic en el botón "Modificar" para un libro específico, se muestra un formulario prellenado con la información del libro seleccionado, utilizando la función `obtenerLibroPorId`.

→ Guardar Cambios: Si se envía el formulario de modificación (`isset($\_POST['guardarCambiosLibro'])`), se recopilan los datos modificados y se utiliza la función `actualizarLibro` para guardar los cambios en la base de datos.

→ Mensajes de Éxito o Error: Después de procesar la modificación, se muestra un mensaje de éxito o error mediante alertas de JavaScript. Si los cambios se guardan con éxito, se muestra una alerta informando al usuario y se recarga la página.  
Parte I Editarlibro.inc.php:  


Parte II Editarlibro.inc.php:  


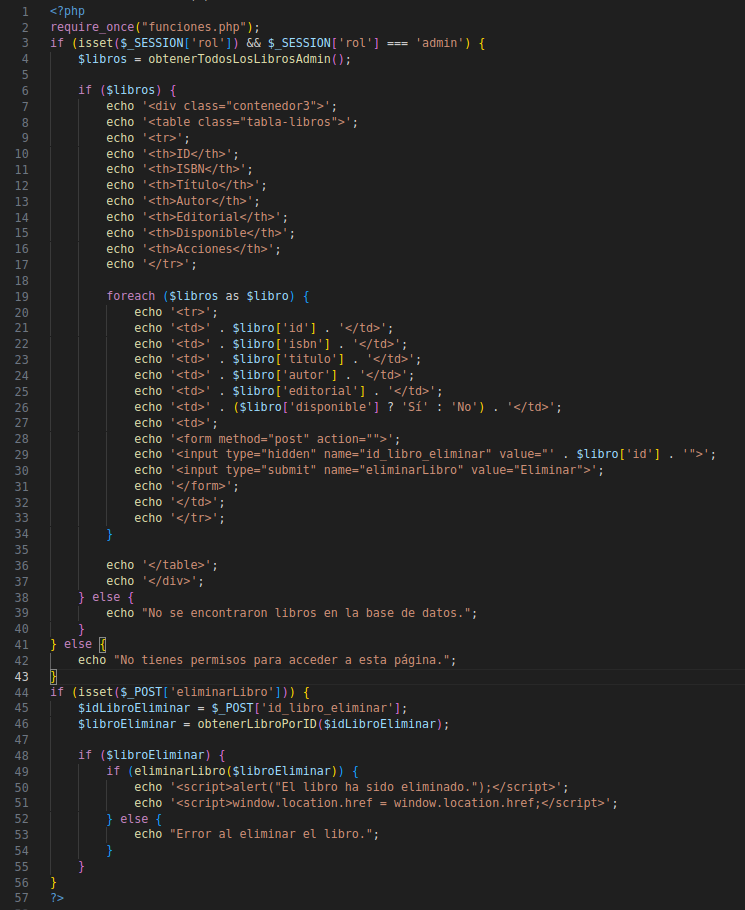
Eliminarlibro.inc.php:  
Este código PHP maneja la visualización de todos los libros en una tabla, incluida la información sobre si un libro está disponible o no, y proporciona la opción de eliminar libros. Aquí tienes una descripción resumida:

→ Verificación de Permisos: Verifica si el usuario tiene el rol de administrador (`isset($\_SESSION['rol']) && $\_SESSION['rol'] === 'admin'`). Si el usuario no tiene el rol de administrador, muestra un mensaje indicando que no tiene permisos.

→ Obtención de Libros: Obtiene todos los libros de la base de datos utilizando la función `obtenerTodosLosLibrosAdmin()` y los muestra en una tabla HTML. Cada fila de la tabla incluye un botón "Eliminar" que permite eliminar el libro correspondiente.

→ Eliminación de Libro: Si se hace clic en el botón "Eliminar" para un libro específico (`isset($\_POST['eliminarLibro'])`), se recopila la información del libro seleccionado utilizando la función `obtenerLibroPorID` y se utiliza la función `eliminarLibro` para eliminar el libro de la base de datos.

→ Mensajes de Éxito o Error: Después de procesar la eliminación, se muestra un mensaje de éxito o error mediante alertas de JavaScript. Si el libro se elimina con éxito, se muestra una alerta informando al usuario y se recarga la página.

Parte Eliminarlibro.inc.php:  


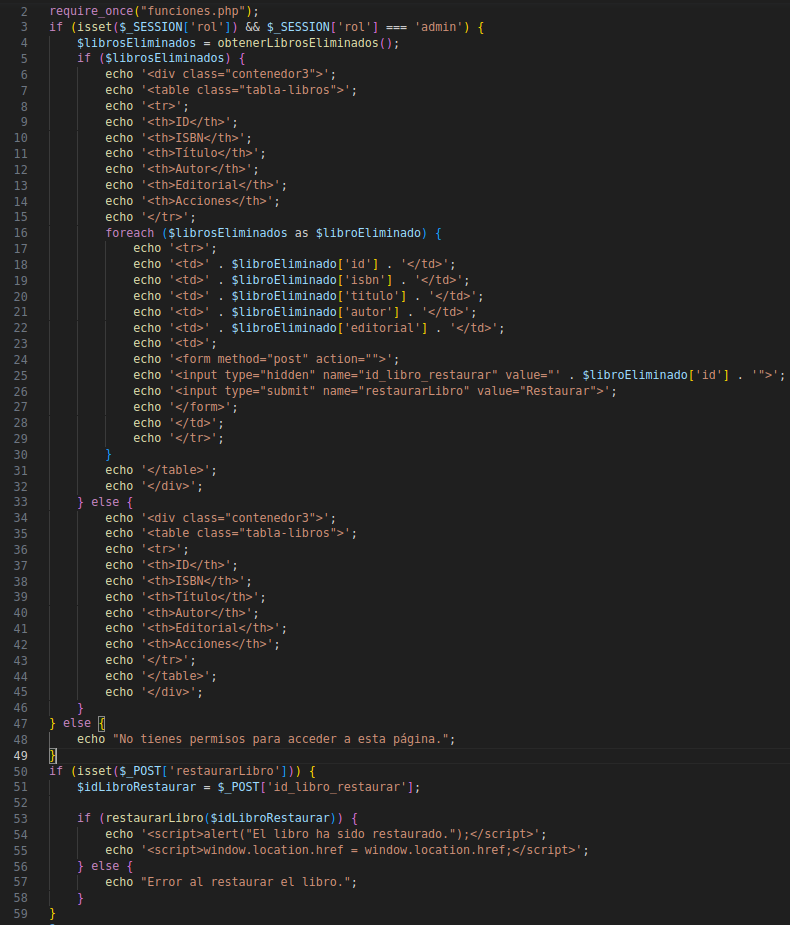
Restaurarlibro.inc.php:  
Este código PHP gestiona la visualización y restauración de libros eliminados para usuarios con el rol de administrador. Aquí tienes una descripción resumida:

→ Verificación de Permisos: Verifica si el usuario tiene el rol de administrador (`isset($\_SESSION['rol']) && $\_SESSION['rol'] === 'admin'`). Si el usuario no tiene el rol de administrador, muestra un mensaje indicando que no tiene permisos.

→ Obtención de Libros Eliminados: Obtiene todos los libros eliminados de la base de datos utilizando la función `obtenerLibrosEliminados()` y los muestra en una tabla HTML. Cada fila de la tabla incluye un botón "Restaurar" que permite restaurar el libro correspondiente.

→ Restauración de Libro: Si se hace clic en el botón "Restaurar" para un libro específico (`isset($\_POST['restaurarLibro'])`), se recopila la información del libro seleccionado utilizando la función `restaurarLibro` para restaurar el libro en la base de datos.

→ Mensajes de Éxito o Error: Después de procesar la restauración, se muestra un mensaje de éxito o error mediante alertas de JavaScript. Si el libro se restaura con éxito, se muestra una alerta informando al usuario y se recarga la página.

Parte Restauralibro.inc.php:  


Prestamos.inc.php:  
Este código PHP gestiona la visualización y cancelación de préstamos de libros para usuarios que han iniciado sesión. Aquí tienes una descripción resumida:

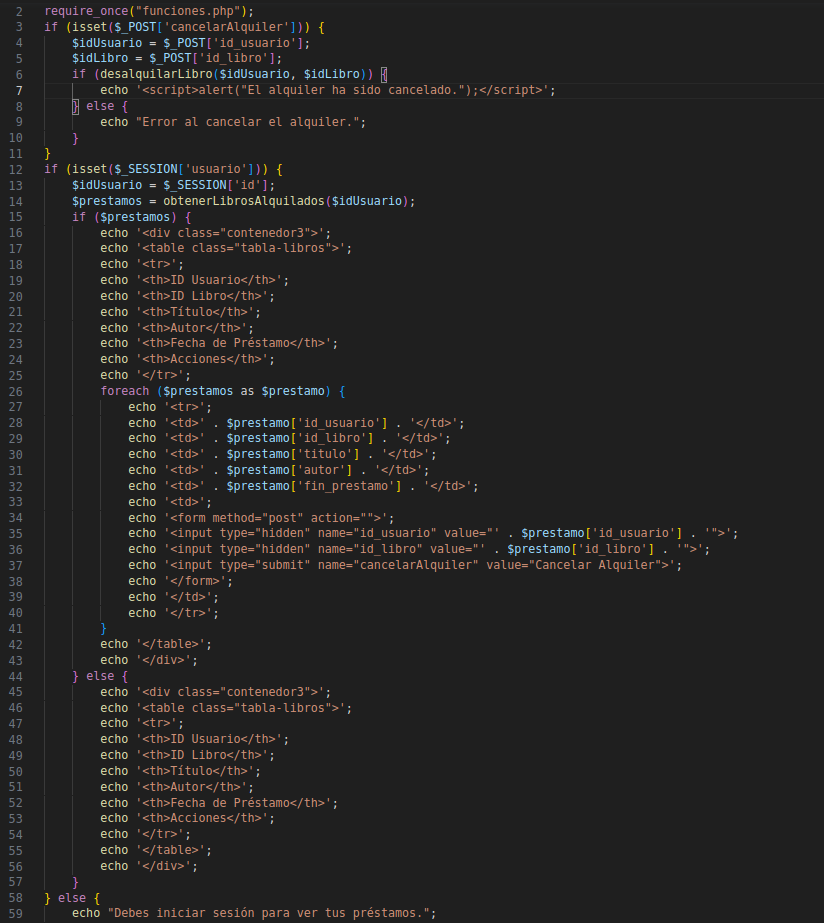
→ Cancelación de Alquiler: Si se recibe un formulario para cancelar un alquiler (`isset($\_POST['cancelarAlquiler'])`), se procesa utilizando la función `desalquilarLibro`. Se muestra una alerta de JavaScript informando sobre el resultado.

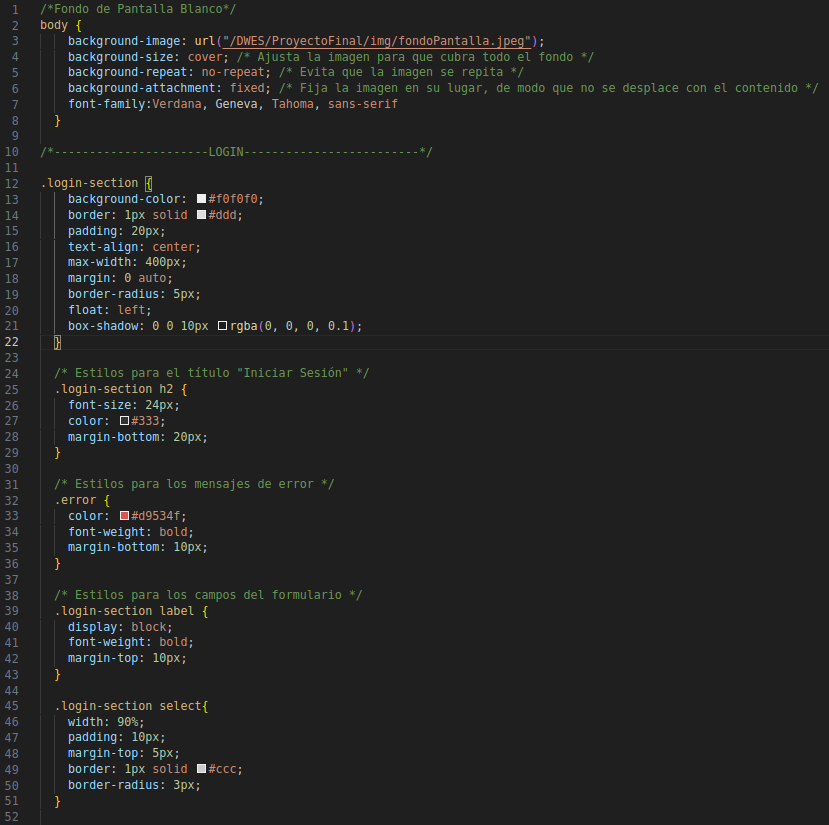
→ Visualización de Préstamos del Usuario: Si el usuario ha iniciado sesión (`isset($\_SESSION['usuario'])`), se obtienen los préstamos del usuario con la función `obtenerLibrosAlquilados`. Luego, se muestran los préstamos en una tabla HTML, incluyendo información como ID de usuario, ID de libro, título, autor y fecha de préstamo. Además, se incluye un botón "Cancelar Alquiler" para cada préstamo.

→ Mensajes de Éxito o Error: Después de procesar la cancelación de alquiler, se muestra un mensaje de éxito o error mediante alertas de JavaScript. Si el alquiler se cancela con éxito, se muestra una alerta informando al usuario.

→ Mensaje de Inicio de Sesión Requerido: Si el usuario no ha iniciado sesión, se muestra un mensaje indicando que deben iniciar sesión para ver sus préstamos.

Parte Prestamos.inc.php:



**4- Fichero CSS**Estilos.css:  
A continuación, una pequeña vista de los estilos que hay en toda la aplicación (+470 Líneas de estilos):  


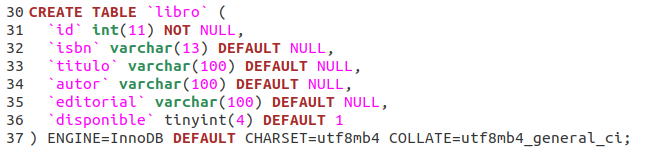
**5- Fichero BBDD**A continuación, una pequeña vista de las tablas SQL que hay en toda la aplicación:  
Tabla Libro:  


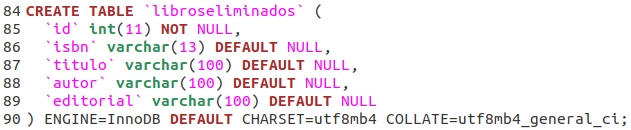
Tabla Libroeliminados:  


Tabla Prestamo:

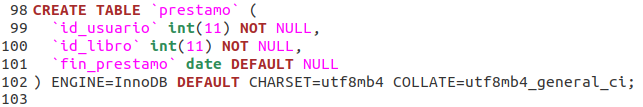


Tabla Usuarios:

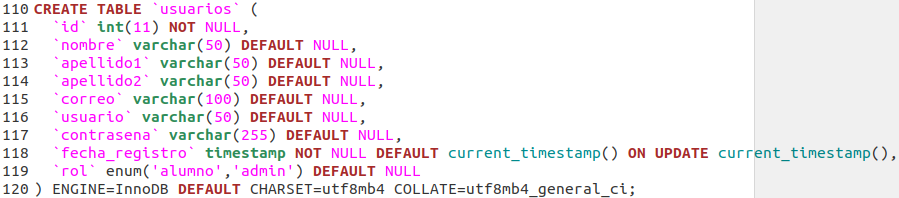
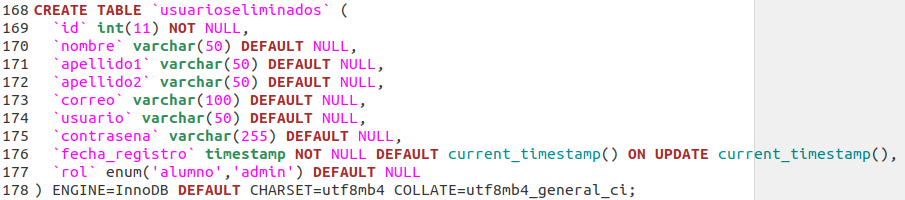


Tabla Usuarioeliminados:



**6- Conclusión y Justificación**

En la aplicación he usado ciertos recursos que no hemos llegado a dar, pero gracias a ellos, la aplicación es más eficiente:  
  
-En la parte de funciones.php, como has podido observar, he estado usando PDO ya que es más seguro que el MySqli y más sencillo de realizar las consultas a la base de datos.  
  
Es cierto que se debería de usar MySqli con lo explicado, pero estuve mirando videos y foros que es mucho mejor y esta es la razón por la que la he usado.

-En la parte de eliminar libros o eliminar usuarios, tuve una idea la cual cuando se borrase un usuario (por ejemplo), se hiciera un insert a una tabla llamada “usuario eliminado” y un delete en la parte de usuarios. De esta manera nos aseguramos que no estaba en la tabla usuarios. Y cuando quisiera restablecer ese usuario, tendría que hacer lo mismo de manera inversa.   
  
Es cierto que se podría poner visible y no visible y nos ahorramos una tabla, pero en su momento lo vi de esta manera y funciona sin problema.  
  
-En la parte de préstamos, cuando alguien alquila el libro y luego lo desalquila, no se queda ningún tipo de registro de que esa persona ha alquilado ese libro. Para ello se podría crear una tabla de registro de préstamos la cual coja la información de ese usuario, el libro alquilado, y las veces que lo ha estado alquilando ese usuario ese libro. Por falta de tiempo no lo he implementado, pero se cómo se podría hacer.  
  
En resumen, hay cosas que se podrían mejorar, cosas que se podrían depurar y quede la aplicación mucho mejor de lo que ya está de por sí.

**7- Bibliografía**He estado usando todas estas webs la cual gracias a ellas me ha sido de apoyo para realizar este proyecto:  
  
-<https://www.youtube.com/> (Funcionalidad de PDO para poder realizar las Funciones.php)  
-<https://chat.openai.com/> (Errores que me salian y consultas SQL de base de datos, estilos CSS de la página)  
-<https://www.php.net/> (Propiedades, funciones…)