BookHub

Crystal Lorena Sánchez García

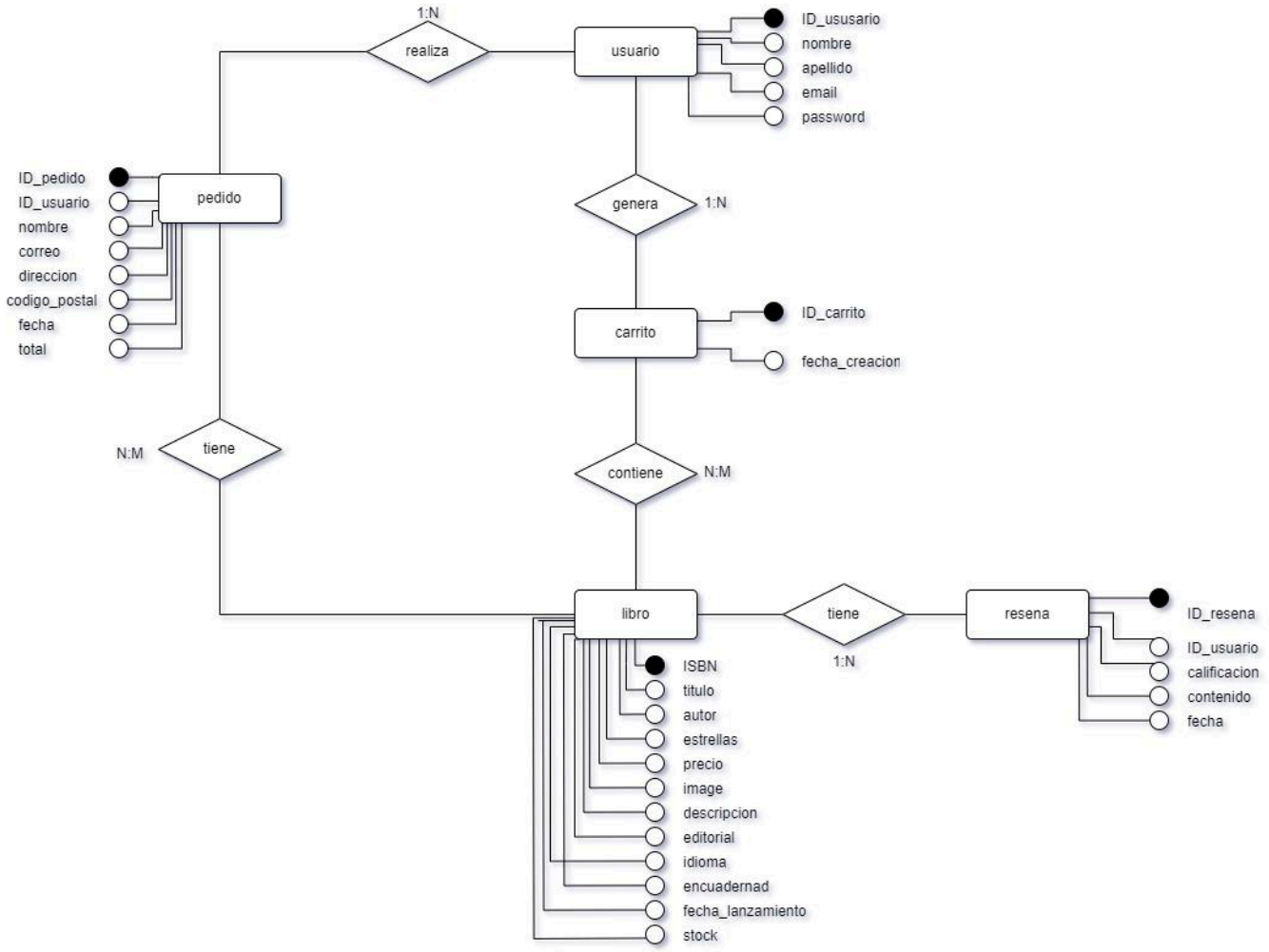
DAW
IES Abdera
Adra - Almería

13/06/2024

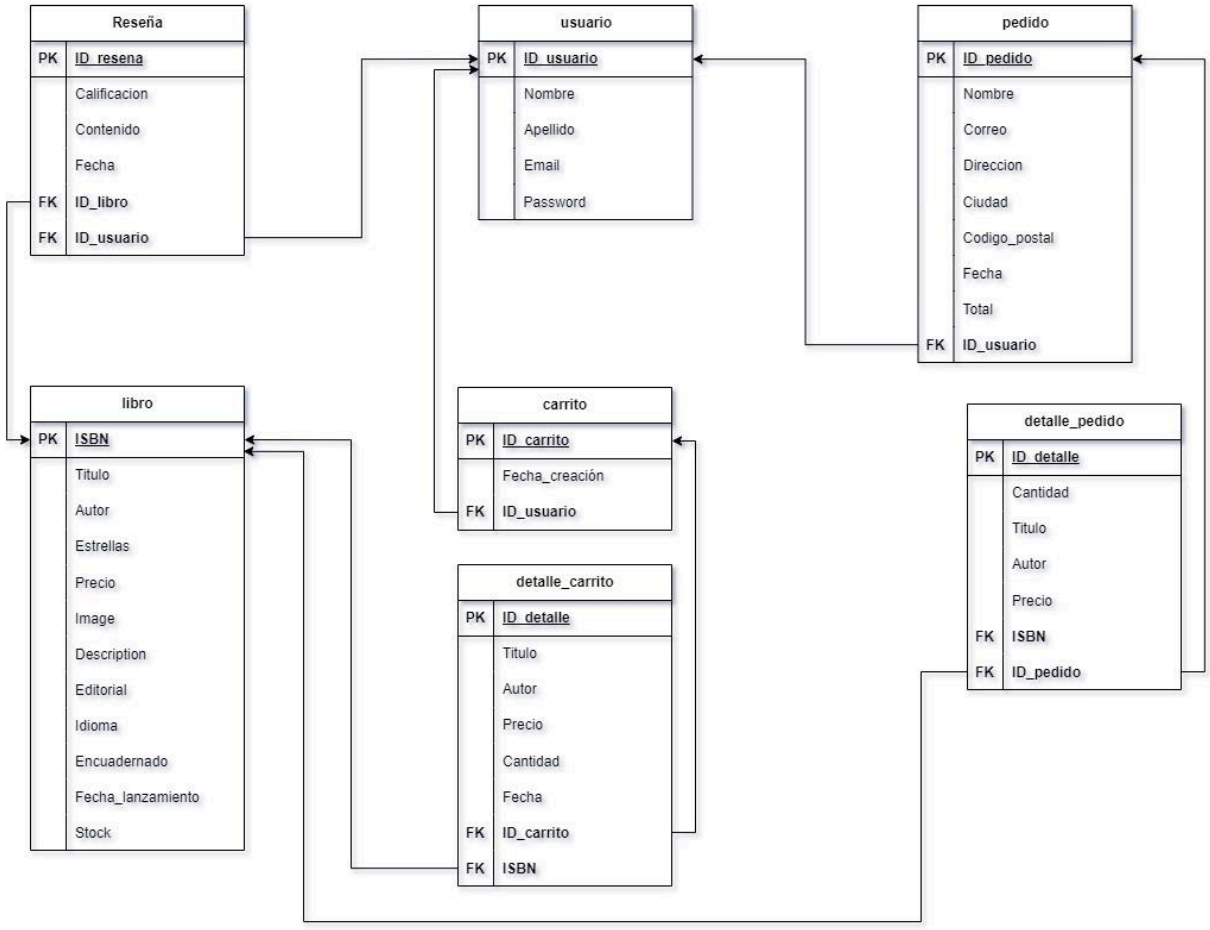
Índice

Manual Técnico

Diagrama Entidad/Relación:



Grafo Relacional:



Aclaraciones:

Tanto la t**abla carrito** y **detalle carrito** se limpian cuando el usuario realiza el pedido, lo cual evita la redundancia de datos.

Ficheros del proyecto:

CSS:

- Styles.css: Este fichero contiene los estilos del proyecto.

IMG:

- Esta carpeta contiene las imágenes que se muestran en el proyecto, tanto de los libros como del carrusel y la página de inicio.

JS:

- scripts.js: Este archivo contiene scripts para el carrusel y el botón de back to top, para ir al inicio de página.

PHP:

- add_to_cart.php: En este archivo se gestiona la agregación de libros al carrito o cesta de compra del usuario. Verifica la autenticación de usuario, maneja la creación y actualización del carrito y valida la disponibilidad de stock del libro.

Verificación de autenticación:

```php
// Verifica si el usuario está logueado
if (!isset($_SESSION['user_id'])) {
    //En caso de no estar logueado, redirige a iniciar
    $_SESSION['message'] = [
        'type' => 'danger',
        'text' => 'Debes iniciar sesión para agregar libros al carrito.'
    ];
    header("Location: ../iniciar.php");
    exit();
}
```

Verificación y creación del carrito:

```php
try {
    // Verificar si el carrito del usuario ya existe
    $stmt = $pdo->prepare("SELECT ID_carrito FROM carrito WHERE ID_usuario = :user_id");
    $stmt->execute(['user_id' => $user_id]);
    $carrito = $stmt->fetch(PDO::FETCH_ASSOC);

    if (!$carrito) {
        // Crear un nuevo carrito para el usuario si no existe
        $stmt = $pdo->prepare("INSERT INTO carrito (ID_usuario) VALUES (:user_id)");
        $stmt->execute(['user_id' => $user_id]);
        $carrito_id = $pdo->lastInsertId(); //Obtienen el ID del carrito generado
    } else {
        $carrito_id = $carrito['ID_carrito']; //Usa el ID del carrito existente.
    }
```

Verificación y actualización de Detalle del carrito:

```php
// Verificar si el libro ya está en el carrito
$stmt = $pdo->prepare("SELECT * FROM detalle_carrito WHERE ID_carrito = :carrito_id AND ISBN
= :isbn");
$stmt->execute(['carrito_id' => $carrito_id, 'isbn' => $isbn]);
$detalle = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$detalle) {
    // Agregar el libro al carrito (sin modificar el stock aquí) agrgandolo con 1 en cantidad.
    $stmt = $pdo->prepare("INSERT INTO detalle_carrito (ID_carrito, ISBN, Titulo, Autor,
    Precio, Cantidad)
                           SELECT :carrito_id, ISBN, Titulo, Autor, Precio, 1 FROM libro
                           WHERE ISBN = :isbn");
    $stmt->execute(['carrito_id' => $carrito_id, 'isbn' => $isbn]);
} else {
    // Si el libro ya esta en el carrito, incrementa la cantidad.
    $new_quantity = $detalle['Cantidad'] + 1;
    $stmt = $pdo->prepare("UPDATE detalle_carrito SET Cantidad = :new_quantity WHERE
    ID_carrito = :carrito_id AND ISBN = :isbn");
    $stmt->execute(['new_quantity' => $new_quantity, 'carrito_id' => $carrito_id, 'isbn' =>
    $isbn]);
}
```

- Confirmacion.php: Este archivo es una nueva página con un mensaje de confirmación de pedido. cuenta con un botón de volver a Inicio

- Procesar pedido.php: En este archivo se gestiona la finalización de compra del pedido. Verifica la autenticación del usuario, se procesa los detalles del pedido y se actualiza en la base de datos.

  Obtiene y valida los datos del carrito:

```php
// Obtener el ID de usuario de la sesión
$id_usuario = $_SESSION['user_id'];

// Recibir los datos del formulario
$nombre = $_POST['nombre'];
$correo = $_POST['correo'];
$direccion = $_POST['direccion'];
$ciudad = $_POST['ciudad'];
$codigo_postal = $_POST['codigo_postal'];

// Verifica si el carrito del usuario ya existe
$stmt = $pdo->prepare("SELECT ID_carrito FROM carrito WHERE ID_usuario = :user_id");
$stmt->execute(['user_id' => $id_usuario]);
$carrito = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$carrito) {
    die("El carrito está vacío.");
}
//Se obtiene el ID del carrito
$carrito_id = $carrito['ID_carrito'];
```

  Se calcula el total del pedido:

```php
$total = 0;

// Calcular el total del pedido
foreach ($detalles_carrito as $detalle) {
    $total += $detalle['Precio'] * $detalle['Cantidad'];
}
```

  Aquí se inserta el pedido en la base de datos:

```php
// Insertar el pedido en la tabla de pedidos
$stmt = $pdo->prepare("INSERT INTO pedido (ID_usuario, Nombre, Correo, Direccion, Ciudad,
Codigo_postal, Total) VALUES (?, ?, ?, ?, ?, ?, ?)");
$stmt->execute([$id_usuario, $nombre, $correo, $direccion, $ciudad, $codigo_postal, $total]);
$id_pedido = $pdo->lastInsertId();
```

Aquí se insertan los detalles del pedido en la base de datos, actualizando el stock.

```php
// Inserta los detalles del pedido en la tabla de detalle_pedido
foreach ($detalles_carrito as $detalle) {
    $stmt = $pdo->prepare("INSERT INTO detalle_pedido (ID_pedido, ISBN, Titulo, Autor, Precio,
    Cantidad) VALUES (?, ?, ?, ?, ?, ?)");
    $stmt->execute([$id_pedido, $detalle['ISBN'], $detalle['Titulo'], $detalle['Autor'], $detalle
    ['Precio'], $detalle['Cantidad']]);

    //Se actualiza el stock del libro
    $stmt = $pdo->prepare("UPDATE libro SET Stock = Stock - ? WHERE ISBN = ?");
    $stmt->execute([$detalle['Cantidad'], $detalle['ISBN']]);
}
```

Por último, se limpia el carrito:

```php
// Limpia el carrito
$stmt = $pdo->prepare("DELETE FROM detalle_carrito WHERE ID_carrito = ?");
$stmt->execute([$carrito_id]);

$stmt = $pdo->prepare("DELETE FROM carrito WHERE ID_carrito = ?");
$stmt->execute([$carrito_id]);
```

- remove_from_cart.php: En este archivo se maneja la eliminación de libros del carrito de compras de un usuario que esté autenticado. Realiza la eliminación del libro del carrie gestiona los mensajes de confirmación o error, aparte de la autenticación de usuario.

  Verifica y recupera el ISBN:

```php
//Verifica si se proporciona un ISBN
if (isset($_POST['isbn'])) {
    $isbn = $_POST['isbn'];// Obtiene el ISBN del libro desde el formulario
```

Recuperación del ID del carrito del Usuario:

```php
// Recuperar el ID del carrito
$stmt = $pdo->prepare("SELECT ID_carrito FROM carrito WHERE ID_usuario = :user_id");
$stmt->execute(['user_id' => $user_id]);
$carrito = $stmt->fetch(PDO::FETCH_ASSOC);
```

Verifica y elimina el libro del carrito:

```php
        // Elimina el libro del carrito
        $stmt = $pdo->prepare("DELETE FROM detalle_carrito WHERE ID_carrito = :carrito_id AND ISBN =
        :isbn");
        $stmt->execute(['carrito_id' => $carrito_id, 'isbn' => $isbn]);

        if ($stmt->rowCount() > 0) {
            //Mensaje de éxito si el libro fue eliminado.
            $_SESSION['message'] = [
                'type' => 'success',
                'text' => 'El libro ha sido eliminado del carrito.'
            ];
        } else {
            //Mensaje de error si el libro no está en el carrito
            $_SESSION['message'] = [
                'type' => 'danger',
                'text' => 'El libro no está en el carrito.'
            ];
        }
    } else {
        // Mensaje de error si no se encuentra el carrito del usuario
        $_SESSION['message'] = [
            'type' => 'danger',
            'text' => 'No se pudo encontrar el carrito del usuario.'
        ];
    }
} else {
    // Mensaje de error si no se proporciona el ISBN
    $_SESSION['message'] = [
        'type' => 'danger',
        'text' => 'ISBN no proporcionado.'
```

- submit review.php: En este archivo se maneja la recepción y almacenamiento de la reseña de libros a la base de datos. Se verifica que la solicitud se realice mediante el método POST y que el usuario esté autenticado. Luego, se inserta la reseña en la base de datos y se manejan posibles errores.

Se inserta la reseña en la base de datos:

```php
try {
    // Prepara la consulta SQL para insertar en la tabla de reseña
    $stmt = $pdo->prepare("INSERT INTO resena (ID_libro, ID_usuario, Calificacion, Contenido)
    VALUES (?, ?, ?, ?)");
    $stmt->execute([$isbn, $userId, $rating, $review]);// Ejecutar la consulta con los datos del
    formulario

    echo "Reseña enviada correctamente"; // Mensaje de confirmación en caso de éxito
} catch (PDOException $e) {
    echo "Error al enviar la reseña: " . $e->getMessage(); // Mensaje de error en caso de
    excepción
}
```

- view_cart.php: En el script de este archivo se le permite al usuario logueado ver los detalles de su carrito, recupera la información del carrito desde la base de datos, calcula el total de la compra y muestra los detalles de los libros del carrito junto con la información del cliente.

  Obtienen el ID del carrito:

```php
// Recuperar el ID del carrito
$stmt = $pdo->prepare("SELECT ID_carrito FROM carrito WHERE ID_usuario = :user_id");
$stmt->execute(['user_id' => $user_id]);
$carrito = $stmt->fetch(PDO::FETCH_ASSOC);

$libros = [];
$total = 0;
```

  Recupera los detalles del carrito y calcula el total:

```php
if ($carrito) {
    $carrito_id = $carrito['ID_carrito'];

    // Recuperar los detalles del carrito
    $stmt = $pdo->prepare("SELECT dc.ISBN, dc.Titulo, dc.Autor, dc.Precio, dc.Cantidad, l.Image
                           FROM detalle_carrito dc
                           JOIN libro l ON dc.ISBN = l.ISBN WHERE dc.ID_carrito = :carrito_id");
    $stmt->execute(['carrito_id' => $carrito_id]);
    $libros = $stmt->fetchAll(PDO::FETCH_ASSOC);

    foreach ($libros as $libro) {
        $total += $libro['Precio'] * $libro['Cantidad'];
    }
}
// Obtiene la cantidad de artículos en el carrito
```

Muestra el contenido del libro agregado a la cesta, de forma dinámica:

```php
<?php
if (!empty($libros)) {
    foreach ($libros as $libro) {
        echo "<div class='book-container mb-3 p-3 border'>
                <div class='book d-flex align-items-center'>
                    <div class='book-image mr-3'>

                        <img src='../{$libro['Image']}' alt='{$libro
                        ['Titulo']}' class='img-fluid' style='max-width:
                        120px;'>
                    </div>
                    <div class='book-details'>
                        <h3>{$libro['Titulo']}</h3>
                        <p>{$libro['Autor']}</p>
                        <p class='price'>{$libro['Precio']} €</p>
                        <p>Cantidad: {$libro['Cantidad']}</p>
                        <form action='remove_from_cart.php' method='post'>
                            <input type='hidden' name='isbn' value='{$libro
                            ['ISBN']}'>
                            <button type='submit' class='btn btn-danger'
                            id='delete'>Eliminar del Carrito</button>
                        </form>
                    </div>
                </div>
            </div>";
    }
} else {
    echo "<p>El carrito está vacío</p>";
}
?>
```

Muestra el formulario para hacer el pedido:

```
<h2>Información del Cliente</h2>
<form action="procesar_pedido.php" method="post">
    <div class="form-group">
        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre" name="nombre" class="form-control"
        required>
    </div>
    <div class="form-group">
        <label for="correo">Correo Electrónico:</label>
        <input type="email" id="correo" name="correo" class="form-control"
        required>
    </div>
    <div class="form-group">
        <label for="direccion">Dirección:</label>
        <input type="text" id="direccion" name="direccion" class="form-control"
        required>
    </div>
    <div class="form-group">
        <label for="ciudad">Ciudad:</label>
        <input type="text" id="ciudad" name="ciudad" class="form-control"
        required>
    </div>
    <div class="form-group">
        <label for="codigo_postal">Código Postal:</label>
        <input type="text" id="codigo_postal" name="codigo_postal"
        class="form-control" required>
    </div>
    <button type="submit" class="btn btn-custom">Enviar Pedido</button>
</form>
```

apache-config.conf:

- En este archivo de configuración de Apache, se define un VirtualHost en el puerto 80 que es el puerto estándar para el tráfico HTTP no cifrado. Principalmente su función es

configurar la raíz del documento, establecer permisos de acceso y definir la ubicación de los archivos de registro.

Definición del VirtualHost:

```
<VirtualHost *:80>
```

Configuración de los Archivos de registro:

```
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

book-detqails.php:

- Este código sirve para mostrar los detalles de los libros obtenidos de la base de datos, también gestiona la funcionalidad de añadir el libro a la cesta y permite a los usuarios dejar una opinión y valoración del libro.

Obtiene el ISBN del libro:

```
$isbn = $_GET['isbn'];
```

Obtiene los detalles del libro basado en el ISBN:

```
//Prepara y ejecuta la consulta SQL para obtener los detalles del libro segun el ISBN.
$query = $pdo->prepare('SELECT * FROM libro WHERE isbn = ?');
$query->execute([$isbn]);
$book = $query->fetch(PDO::FETCH_ASSOC);

if (!$book) {
    echo "Libro no encontrado";
    exit;
}
```

Función para generar las estrellas en HTML:

```php
function getStarsHTML($stars) {
    $fullStars = floor($stars);                    // Número entero de estrellas llenas
    $decimalPart = $stars - $fullStars;            // Parte decimal para determinar fracción de
    estrella
    $halfStar = $decimalPart >= 0.25 && $decimalPart < 0.75 ? 1 : 0; // Determina si hay media
    estrella
    $emptyStars = 5 - $fullStars - $halfStar;      // Número de estrellas vacías necesarias

    $starsHTML = str_repeat('&#9733;', $fullStars);   // Genera las estrellas llenas (&#9733; es ★)
```

Modal de opinión:

```php
<form id="review-form" action="php/submit_review.php" method="POST">
    <div class="modal-body">
        <input type="hidden" name="isbn" id="isbn" value="<?= $book['ISBN'] ?>">
        <?php if (isset($_SESSION['user_id'])): ?>
            <input type="hidden" name="user_id" id="user-id" value="<?php echo
            $_SESSION['user_id']; ?>">
            <div class="form-group">
                <label for="rating">Calificación:</label>
                <select id="rating" name="rating" class="form-control">
                    <option value="5">★★★★★ - 5</option>
                    <option value="4">★★★★☆ - 4</option>
                    <option value="3">★★★☆☆ - 3</option>
                    <option value="2">★★☆☆☆ - 2</option>
                    <option value="1">★☆☆☆☆ - 1</option>
                </select>
            </div>
            <div class="form-group">
                <label for="review">Comentario:</label>
                <textarea id="review" name="review" class="form-control"
                rows="3"></textarea>
            </div>
        <?php else: ?>
            <div class="alert alert-warning">Debe iniciar sesión para comentar.</
            div>
        <?php endif; ?>
    </div>
    <div class="modal-footer">
        <?php if (isset($_SESSION['user_id'])): ?>
            <button type="submit" class="btn btn-primary">Enviar</button>
        <?php endif; ?>
        <button type="button" class="btn btn-secondary"
        data-dismiss="modal">Cerrar</button>
    </div>
</form>
```

database.sql:

- Este documento describe la estructura de las tablas creadas en la base de datos bookhub y los datos insertados en la tabla Libro.
- **Estructura de Tablas:**

  `libro`: Almacena información detallada sobre cada libro, incluyendo título, autor, calificación, precio, imagen, descripción, editorial, idioma, tipo de encuadernado, fecha de lanzamiento y stock.

  `usuario`: Guarda los datos de los usuarios registrados, con un ID único para cada usuario.

  `pedido`: Registra los pedidos realizados por los usuarios, con detalles como nombre, correo, dirección, ciudad, código postal, fecha y total del pedido.

  `resena`: Contiene las reseñas de los libros realizadas por los usuarios, incluyendo la calificación y el contenido de la reseña.

  `detalle_pedido`: Guarda los detalles específicos de cada libro incluido en un pedido, como título, autor, precio y cantidad.

  `carrito`: Almacena el carrito de compras de los usuarios, con la fecha de creación del carrito.

  `detalle_carrito`: Guarda los detalles de los libros incluidos en el carrito de compras, incluyendo título, autor, precio, cantidad y fecha de agregado.

- **Índices Adicionales:**

  Se crean índices adicionales para mejorar el rendimiento en consultas **`JOIN`**, optimizando la velocidad de recuperación de datos relacionados entre las tablas.

db.php:

- En este documento se describe la función principal y los detalles técnicos del código PHP utilizado para conectar a la base de datos booktube.

```php
<?php
$host = 'db'; //Nombre del host de la base de datox (Contenedor Docker)
$dbname = 'bookhub'; //Nomnre de la base de datos
$user = 'user'; //Nombre de usuario de la base de datos
$pass = 'user_password'; //Contraseña del usuario

try {
    // Se crea una nueva instancia PDO para la conexión a ala base de datos
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
    // Se configura PDO para lanzar excepciones en caso de errores.
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    // Captura cualquier excepcion de conexion y muestra un mensaje de error.
    die("Could not connect to the database: " . $e->getMessage());
}
?>
```

docker-compose.yml:

- Este documento describe la función principal y los detalles técnicos del archivo docker-compose.yml, el que se utiliza para configurar y orquestar múltiples contenedores Docker que componen un entorno de desarrollo para la aplicación web. Se utiliza para crear un entorno de desarrollo completo que incluye un servidor PHP, una base de datos MySQL y una interfaz gráfica de usuario PHPMyAdmin.
    - **Versión de Docker Compose:** version: 3.8, especifica la versión de Docker Compose que se utilizará para interpretar al archivo YAML.

- **Servicio php:**

```yaml
php:
  build:
    context: .
    dockerfile: Dockerfile
  container_name: php_container
  volumes:
    - .:/var/www/html
  ports:
    - "8080:80"
  depends_on:
    - db
  networks:
    - mynetwork
```

BUILD: especifica cómo construir la imagen del contenedor PHP usando el contexto actual y el archivo Dockerfile.

Container_name: es el nombre del contenedor.

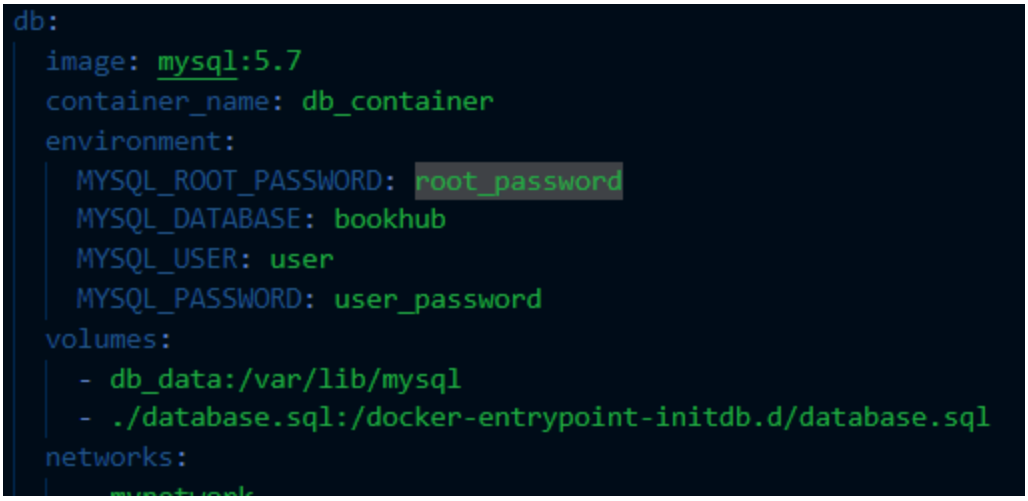
volumes: Monta en el directorio actual en **/var/www/html** dentro del contenedor , permitiendo la edición del archivo a tiempo real.

Ports: Mapea el puerto 8080 del host al puerto 80 del contenedor PHP.

Depends_on: define que este servicio depende del servicio db, asegurando que el contenedor de MySQL esté en funcionamiento antes de iniciar el contenedor PHP.

networks: Conecta el contenedor al nombre de red **mynetwork,** permitiendo la comunicación entre contenedores.

- Servicio **db** (MySQL):



`image`: Utiliza la imagen oficial de MySQL versión `5.7`.

`container_name`: Nombre del contenedor MySQL.

`environment`: Configura las variables de entorno necesarias para la instancia de MySQL, incluyendo la contraseña del usuario root, la base de datos bookhub y las credenciales del usuario `user`.

`volumes`: Permite persistir los datos de la base de datos MySQL y ejecutar scripts de inicialización.

`db_data:/var/lib/mysql`: Almacena los datos persistentes de MySQL.

`./database.sql:/docker-entrypoint-initdb.d/database.sql`: Monta un archivo SQL para inicializar la base de datos al inicio del contenedor.

`networks`: Conecta el contenedor al nombre de red `mynetwork` para la comunicación entre servicios.

- Servicio **phpmyadmin:**

```yaml
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  container_name: phpmyadmin_container
  environment:
    PMA_HOST: db
    MYSQL_ROOT_PASSWORD: root_password
  ports:
    - "8081:80"
  depends_on:
    - db
  networks:
    - mynetwork
```

image: Utiliza la imagen oficial de PHPMyAdmin.

container_name: Nombre del contenedor PHPMyAdmin.

environment: Configura las variables de entorno para PHPMyAdmin, especificando el host de MySQL (`PMA_HOST`) y la contraseña del usuario root de MySQL.

ports: Mapea el puerto `8081` del host al puerto `80` del contenedor PHPMyAdmin.

depends_on: Define que este servicio depende del servicio db, asegurando que el contenedor de MySQL esté en funcionamiento antes de iniciar el contenedor PHPMyAdmin.

networks: Conecta el contenedor al nombre de red `mynetwork` para la comunicación entre servicios.

- **Volúmenes y Redes:**

volumes: Define el volumen `db_data` para persistir los datos de MySQL.

networks: Define la red `mynetwork` para permitir la comunicación entre los contenedores.

Dockerfile:

- Describe la función principal y los detalles técnicos del archivo Dockerfile, el cual se utiliza para construir una imagen Docker que incluye PHP 7.4 con Apache, configurada para ejecutar la aplicación web.
    - **FROM:** utiliza la base oficial de PHP 7.4 con Apache
    - **WORKDIR:** Establece el directorio de trabajo dentro del contenedor donde se copiará el código de la aplicación (**`/var/www/html).`**
    - **COPY:** Copia el contenido del directorio actual al directorio de trabajo del contenedor (**`/var/www/html`**). Incluyendo todos los archivos y subdirectorios necesarios para la aplicación.
    - **RUN (a2enmod):** Ejecuta el comando **a2enmod rewrite** para habilitar el módulo de reescritura de URL de Apache. Esto es necesario para configuraciones que utilicen URLs amigables.
    - **RUN (apt-get):** Actualiza los paquetes disponibles e instala dependencias necesarias para las extensiones de PHP. Configura y habilita la extensión GD para manejar imágenes. Instala las extensiones PDO y PDO MySQL para la conexión con la base de datos MySQL desde PHP.
    - **COPY configuración de apache:** Copia el archivo de configuración personalizado de Apache al archivo de configuración principal de Apache (**`000-default.conf`**).
    - **EXPOSE:** Expone el puerto 80 del contenedor, donde Apache HTTP Server estará escuchando las conexiones entrantes.
    - **CMD:** Define el comando predeterminado que se ejecutará al iniciar el contenedor. En esta ocasión utiliza **apache2-foreground** para iniciar Apache HTTP Server en primer plano.

Index.php:

- Este archivo contiene el código html de la página de inicio. Contiene script que da funcionalidad a la barra de búsqueda.

```php
<?php
    $search = isset($_GET['search']) ? $_GET['search'] : '';
    if ($search) {
        $stmt = $pdo->prepare('SELECT * FROM libro WHERE Titulo LIKE ? OR Autor LIKE ?');
        $stmt->execute(["%$search%", "%$search%"]);
    } else {
        $stmt = $pdo->query('SELECT * FROM libro');
    }
    $books = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>
```

iniciar.php:

- Este archivo contiene el código HTML de la página de registrarse. Contiene script que controla los mensajes que se le mostrarán al usuario.

```php
<?php
session_start();
$logged_in = isset($_SESSION['nombre']);

// Verificar si hay un mensaje de error y mostrarlo
$message = '';
if (isset($_SESSION['message'])) {
    if (is_array($_SESSION['message'])) {
        $message = $_SESSION['message']['text'];
    } else {
        $message = $_SESSION['message'];
    }
    unset($_SESSION['message']); // Limpia el mensaje para que no se muestre nuevamente
}
?>
```

libros.php:

- En este archivo se muestran dinámicamente los libros desde la base de datos, tiene implementada la barra de búsqueda y la función de las estrellas en HTML, además de manejar el mostrar mensajes al usuario.

```php
// Código para imprimir libros
echo '<div class="row">';
foreach ($books as $book) {
    echo '<div class="col-md-3 book">';
    echo '<a href="book-details.php?isbn=' . $book['ISBN'] . '">';
    echo '<img src="' . $book['Image'] . '" alt="' . $book['Titulo'] . '"
    class="img-fluid">';
    echo '</a>';
    echo '<h3>' . $book['Titulo'] . '</h3>';
    echo '<p>' . $book['Autor'] . '</p>';

    // Llama a la función getStarsHTML para imprimir las estrellas
    echo getStarsHTML($book['Estrellas']);

    echo '<div class="price">' . $book['Precio'] . ' €</div>';

    // Verifica el stock para deshabilitar el botón si es necesario
    if ($book['Stock'] <= 0) {
        echo '<button type="button" class="btn add-to-cart" disabled>Agotado</
        button>';
    } else {
        echo '<form action="php/add_to_cart.php" method="post">';
        echo '<input type="hidden" name="isbn" value="' . $book['ISBN'] . '">';
        echo '<button type="submit" class="btn add-to-cart">Comprar</button>';
        echo '</form>';
    }

    echo '</div>';
}
echo '</div>';
?>
```

login.php:

- Este archivo contiene el script de inicio de sesión del usuario, obteniendo los datos de la base de datos y generando mensajes si las credenciales son incorrectas.

```php
// Verificar las credenciales del usuario
try {
    $stmt = $pdo->prepare("SELECT * FROM usuario WHERE Email = :email");
    $stmt->execute(['email' => $email]);
    $user = $stmt->fetch();

    if ($user && password_verify($password, $user['Password'])) {
        // Iniciar sesión
        $_SESSION['user_id'] = $user['ID_usuario']; // Guardar el ID del
        usuario en la sesión
        $_SESSION['email'] = $email;
        $_SESSION['nombre'] = $user['Nombre'];
        header('Location: Index.php');
        exit();
    } else {
        $_SESSION['message'] = 'Credenciales incorrectas.';
        header('Location: iniciar.php');
        exit();
    }
} catch (PDOException $e) {
    $_SESSION['message'] = "Error: " . $e->getMessage();
    header('Location: iniciar.php');
    exit();
}
```

logout.php:

- Este script genera la destrucción de la sesión iniciada.

```php
<?php
session_start();// Inicia una sesion nueva o reanuda una ya creada.
session_destroy(); //Destruye toda la información registrada de la sesion
header('Location: Index.php'); // Redirige al usuario a la pagina de inicio.
exit(); //Termina la ejecucion paara asegurar que la redireccion ocurre.
?>
```

navbar.php:

- Este archivo es una plantilla de la barra de navegación que se implementa en la mayoría de las páginas.

register.php:

- Este script contiene la lógica del registro de usuario a través de la base de datos. Controlando también si el usuario ya tiene el email registrado, y hashea la contraseña para ser más segura.

```php
// Hashear la contraseña
$hashed_password = password_hash($password, PASSWORD_DEFAULT);

// Insertar el usuario en la base de datos
try {
    $stmt = $pdo->prepare("INSERT INTO usuario (Nombre, Apellido, Email,
    Password) VALUES (:name, :lastname, :email, :password)");
    $stmt->execute(['name' => $name, 'lastname' => $lastname, 'email' =>
    $email, 'password' => $hashed_password]);

    // Iniciar sesión
    $_SESSION['email'] = $email;

    $_SESSION['message'] = 'Usuario creado, por favor inicie sesión.';
    $_SESSION['message_type'] = 'alert-success'; // Mensaje de éxito
    header('Location: registro.php');
    exit();
```

```
@var mixed $email
```

registro.php:

- Este documento tiene el HTML de registro de usuario, muestra los mensajes derivados de la lógica del registro (register.php)

Aplicaciones utilizadas

Lenguajes de programación:

- PHP
- JavaScript

Lenguajes de Marcado/Estilo:

- HTML 5
- CSS 3

Framework:

- Bootstrap

RDBMS:

- MySQL

Herramienta gestora de DB MySQL:

- phpMyAdmin

Servidores Web:

- Apache

Herramientas de Desarrollo:

- Visual Studio Code
- XAMPP

Herramientas de despliegue:

- Docker

Herramientas de control de Versiones:

- Git

Bibliografía

La documentación de PHP:

- https://www.php.net/docs.php

YAML:

- https://gettaurus.org/docs/YAMLTutorial/

AJAX:

- https://www.freecodecamp.org/news/ajax-tutorial/

Configurar Apache:

- https://www.dnsstuff.com/apache-web-server-configuration

PHP y DB:

- https://www.hostinger.com/tutorials/how-to-connect-php-to-mysql
- https://www.sqliz.com/posts/php-crud-mysql/