

Contenido

1	INTRODUCCION.....	2
2	Configuración de entorno	2
2.1	Instalar PHP	2
2.2	Instalar un servidor web (Apache)	2
3	Base de Datos MySQL – XAMPP	3
3.1	Crear Base de Datos y Tabla	3
3.2	Conexión a Base de Datos	5
4	Código fuente PHP.....	6
4.1	Endpointst de la API.....	6
4.1.1	Agregar Libro	6
4.1.2	Editar Libro.....	7
4.1.3	Eliminar Libro	7
4.1.4	Obtener Libro	8
4.1.5	Obtener Libro por Id.....	8
4.2	Documentación y Ejecución de API con Postman	9
5	Código Fuente Interfaz Web	12
5.1	Interfaz Web.....	12
5.1.1	Index.html.....	12
5.1.2	Scripts.js.....	14
5.1.3	styles.css.....	17
6	Test de Automatización (PHPUnit)	19
6.1	Configuración PHPUnit.....	20
6.2	Ejecución de Pruebas	29

1 INTRODUCCION

En este archivo se deja de manera clara y concisa la documentación de cada paso a realizar para replicar el proyecto, el cual refiere a una API Restful para un catalogo de libros, usando código backend PHP. Contando también con una interfaz web simple para interactuar con la API (Js, Html, Css). También se documenta la manera de hacer las pruebas automatizadas utilizando PHPunit. Todo realizado desde un IDE, Visual Studio Code (VSC).

Para crear una API REST en PHP que gestione un catálogo de libros, vamos a seguir los siguientes pasos:

2 Configuración de entorno

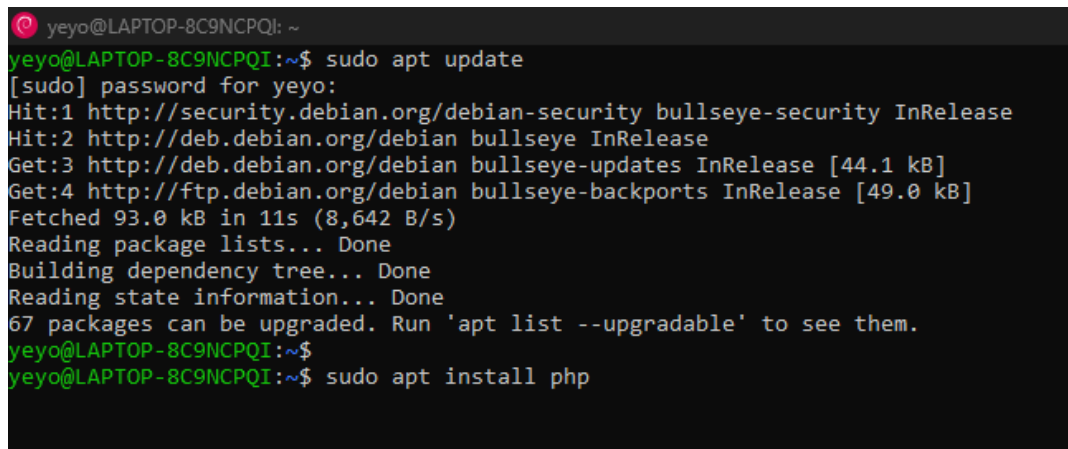
Instalar PHP y un servidor web como Apache o Nginx. Por medio de una terminal de comando WSL (Windows subsystem for Linux), en mi caso estoy utilizando **Debian**.

2.1 Instalar PHP

Se deberá ejecutar lo siguientes comandos.

- `sudo apt update` (Si cuentas con contraseña de Usuario debes ingresarla)
- `sudo apt install php`

De manera que quedaría así:



```
yeyo@LAPTOP-8C9NCPQJ: ~  
yeyo@LAPTOP-8C9NCPQJ:~$ sudo apt update  
[sudo] password for yeyo:  
Hit:1 http://security.debian.org/debian-security bullseye-security InRelease  
Hit:2 http://deb.debian.org/debian bullseye InRelease  
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]  
Get:4 http://ftp.debian.org/debian bullseye-backports InRelease [49.0 kB]  
Fetched 93.0 kB in 11s (8,642 B/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
67 packages can be upgraded. Run 'apt list --upgradable' to see them.  
yeyo@LAPTOP-8C9NCPQJ:~$  
yeyo@LAPTOP-8C9NCPQJ:~$ sudo apt install php
```

Como ya cuento con PHP instalado no era necesario ejecutar el comando. En caso contrario debes de ejecutarlo y lo demás se hará por sí solo.

2.2 Instalar un servidor web (Apache)

Se deberá ejecutar lo siguientes comandos.

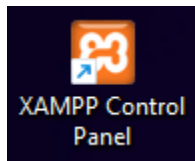
- `sudo apt update` (Si cuentas con contraseña de Usuario debes ingresarla)
- `sudo apt install apache2`

De manera que quedaría así:

```
yeyo@LAPTOP-8C9NCPQI: ~  
yeyo@LAPTOP-8C9NCPQI:~$ sudo apt update  
[sudo] password for yeyo:  
Hit:1 http://security.debian.org/debian-security bullseye-security InRelease  
Hit:2 http://deb.debian.org/debian bullseye InRelease  
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]  
Get:4 http://ftp.debian.org/debian bullseye-backports InRelease [49.0 kB]  
Fetched 93.0 kB in 11s (8,642 B/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
67 packages can be upgraded. Run 'apt list --upgradable' to see them.  
yeyo@LAPTOP-8C9NCPQI:~$  
yeyo@LAPTOP-8C9NCPQI:~$ sudo apt install apache2
```

3 Base de Datos MySQL – XAMPP

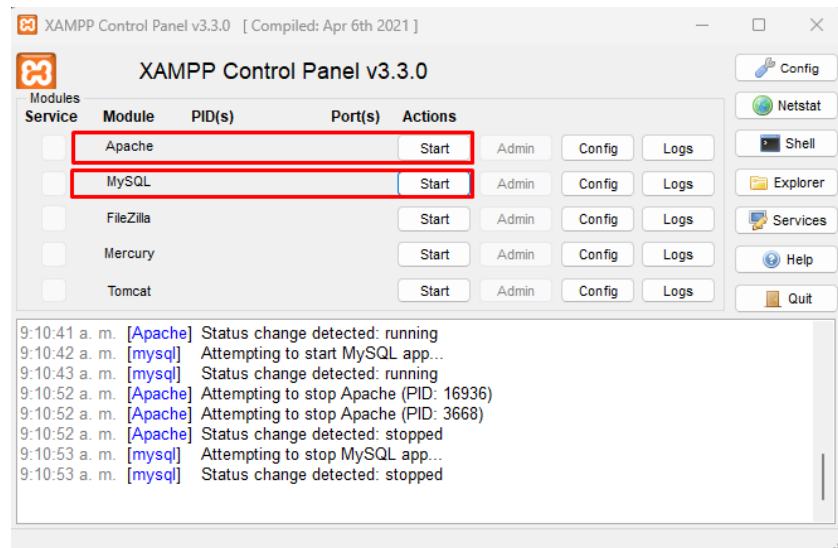
Para el proyecto vamos a usar la herramienta de XAMPP para trabajar con el servidor Apache y la Base de Datos MySQL. En este sentido deberíamos tener instalado la herramienta: Si está instalada correctamente, en el escritorio del equipo tendremos un acceso directo a Xampp.



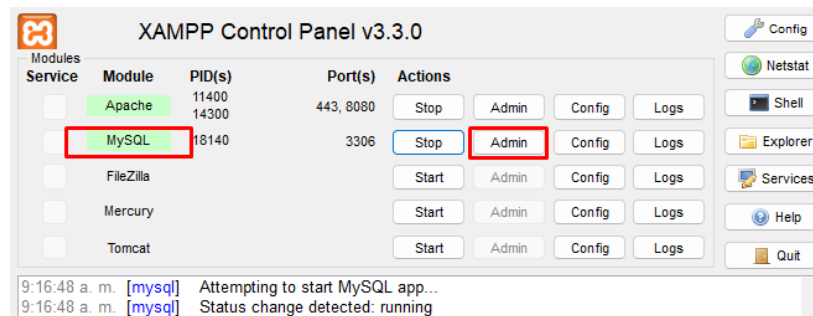
3.1 Crear Base de Datos y Tabla

Procedemos a abrir la aplicación y visualizaremos una ventana con un panel de control, donde solo nos interesará los servicios de **Apache** y **MySQL**. Entendiendo esto procedemos con lo siguiente paso:

- Se debe de iniciar primeramente el servicio de Apache
- Seguido se inicia el servicio de MySQL



-Para el abrir y usar servicio de Base de Datos en la web vamos a dar click en “Admin”



Esto nos llevara al servidor de base de datos por medio de la web donde construiremos la consulta SQL para la creación de base de datos y tablas necesarias para el proyecto. Para ello seguiremos los siguientes pasos:

-En la pagina inicial buscaremos en el menú de barra, específicamente que nos indica “SQL”.

-Allí se dejará en siguiente código SQL para la creación de base de datos y tabla:

CREATE DATABASE libreria;

USE libreria;

CREATE TABLE libros (

id INT AUTO_INCREMENT PRIMARY KEY,

titulo VARCHAR(255) NOT NULL,

autor VARCHAR(255) NOT NULL,

```
    ano_publicacion INT NOT NULL,  
    genero VARCHAR(100) NOT NULL  
);
```

-Procediendo con la ejecución le daremos clic en “**Continuar**” en la parte inferior derecha del recuadro del código.

Una vez ejecutado correctamente el código ya estaríamos listos para crear una conexión desde PHP a MySQL.

3.2 Conexión a Base de Datos

En nuestro entorno de desarrollo, es decir, VSC. Tenemos a disposición una carpeta llamada “**bookstore**” el cual contendrá nuestro proyecto.

[NOTA: Para tener en cuenta, ya que se esta utilizando XAMPP como medio para un servidor web en php y una integración a base de datos. El proyecto deberá estar ubicado en la siguiente ruta “**C:\xampp\htdocs\vsc_yeyowork\bookstore**” en mi caso cree un espacio personal (vsc_yeyowork) para alojar mi proyecto. (Recomendación, crear el espacio igualmente para no tener problemas de enrutamiento con la api)]

- Iniciamos con la creación de una carpeta “api”.
- Dentro de esta misma crearemos el archivo “**db.php**”
- Allí se encontrará el código para conexión a base de datos:

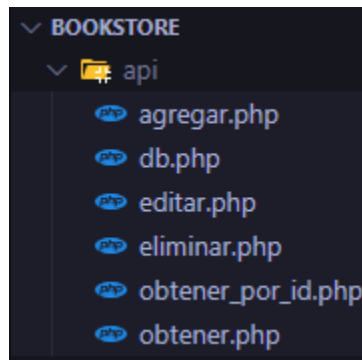
```
<?php  
$host = 'localhost';  
$db = 'libreria';  
$user = 'root';  
$pass = '';  
  
try {  
    $pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);  
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
} catch (PDOException $e) {  
    die("Connection failed: " . $e->getMessage());  
}  
?>
```

4 Código fuente PHP

4.1 Endpoints de la API

Para comenzar a programar en PHP nuestra Api para el catalogo de libros tendremos que realizar la creación de unos archivos los cuales serán para la funcionalidad de agregar, editar, eliminar, obtener y obtener por id, todo esto dentro de la carpeta “api”.

Consiguiendo la siguiente estructura:



4.1.1 Agregar Libro

En el archivo encontraremos el siguiente código:

```
<?php
require 'db.php';

$data = json_decode(file_get_contents('php://input'), true);

if (!isset($data['titulo']) || !isset($data['autor']) ||
    !isset($data['ano_publicacion']) || !isset($data['genero'])) {
    echo json_encode(['error' => 'Todos los campos son obligatorios']);
    exit;
}

$titulo = $data['titulo'];
$autor = $data['autor'];
$ano_publicacion = $data['ano_publicacion'];
$genero = $data['genero'];

$stmt = $pdo->prepare("INSERT INTO libros (titulo, autor, ano_publicacion,
genero) VALUES (?, ?, ?, ?)");
$stmt->execute([$titulo, $autor, $ano_publicacion, $genero]);

echo json_encode(['message' => 'Libro agregado exitosamente']);
?>
```

4.1.2 Editar Libro

En el archivo encontraremos el siguiente código:

```
<?php
require 'db.php';

$data = json_decode(file_get_contents('php://input'), true);

if (!isset($data['id']) || !isset($data['titulo']) || !isset($data['autor'])
|| !isset($data['ano_publicacion']) || !isset($data['genero'])) {
    echo json_encode(['error' => 'Todos los campos son obligatorios']);
    exit;
}

$id = $data['id'];
$titulo = $data['titulo'];
$autor = $data['autor'];
$ano_publicacion = $data['ano_publicacion'];
$genero = $data['genero'];

$stmt = $pdo->prepare("UPDATE libros SET titulo = ?, autor = ?,
ano_publicacion = ?, genero = ? WHERE id = ?");
$stmt->execute([$titulo, $autor, $ano_publicacion, $genero, $id]);

echo json_encode(['message' => 'Libro editado exitosamente']);
?>
```

4.1.3 Eliminar Libro

En el archivo encontraremos el siguiente código:

```
<?php
require 'db.php';

$data = json_decode(file_get_contents('php://input'), true);

if (!isset($data['id'])) {
    echo json_encode(['error' => 'El ID es obligatorio']);
    exit;
}

$id = $data['id'];

$stmt = $pdo->prepare("DELETE FROM libros WHERE id = ?");
$stmt->execute([$id]);
```

```
echo json_encode(['message' => 'Libro eliminado exitosamente']);
?>
```

4.1.4 Obtener Libro

En el archivo encontraremos el siguiente código:

```
<?php
require 'db.php';

$stmt = $pdo->query("SELECT * FROM libros");
$libros = $stmt->fetchAll(PDO::FETCH_ASSOC);

echo json_encode($libros);
?>
```

4.1.5 Obtener Libro por Id

En el archivo encontraremos el siguiente código:

```
<?php
header('Content-Type: application/json');

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "libreria";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die(json_encode(["message" => "Conexión fallida: " . $conn->connect_error]));
}

$id = isset($_GET['id']) ? intval($_GET['id']) : 0;

if ($id > 0) {
    $stmt = $conn->prepare("SELECT * FROM libros WHERE id = ?");
    $stmt->bind_param("i", $id);
    $stmt->execute();
    $result = $stmt->get_result();
}
```



```

    if ($result->num_rows > 0) {
        $libro = $result->fetch_assoc();
        echo json_encode($libro);
    } else {
        echo json_encode(["message" => "No se encontró ningún libro con el
ID $id."]);
    }

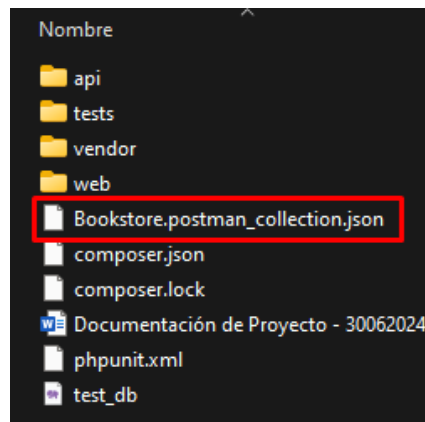
    $stmt->close();
} else {
    echo json_encode(["message" => "ID no válido."]);
}

$conn->close();
?>

```

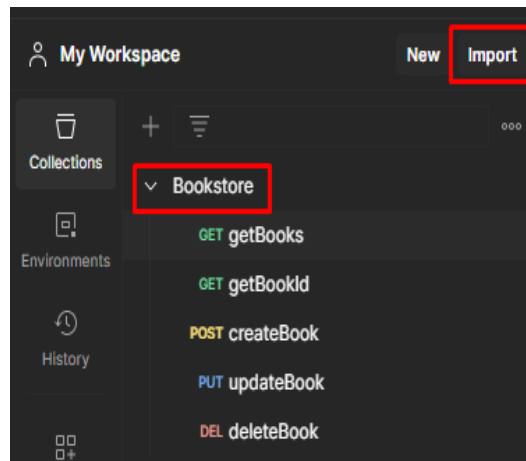
4.2 Documentación y Ejecución de API con Postman

Resumiendo, las configuraciones de postman las podremos encontrar dentro del proyecto “bookstore”, encontramos la colección .json para realizar la pruebas correspondientes:



Utilizando postman podremos importar la colección:

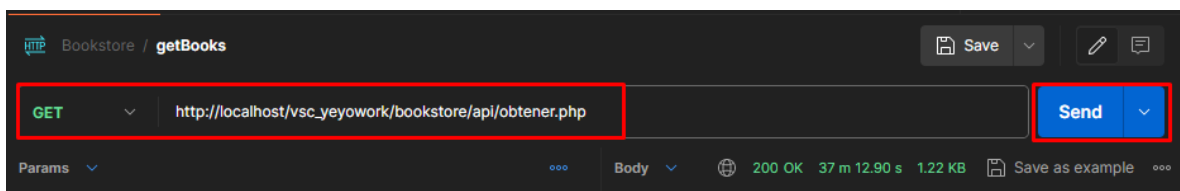
- Desde el espacio de trabajo visualizamos el botón de “importar”
- Al hacer clic nos dejará seleccionar el archivo .json de la colección
- Al terminar la importación nos habilitara las peticiones con la siguiente estructura:



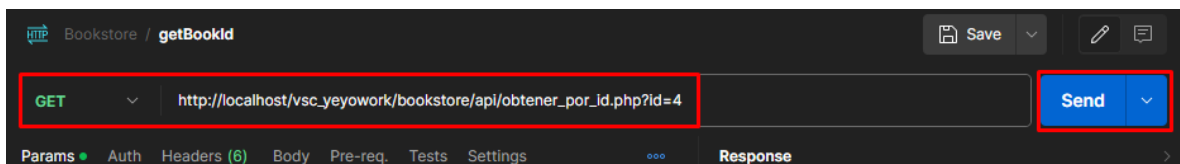
Allí podremos realizar las peticiones de cada método de la siguiente manera:

URL: http://localhost/vsc_yeyowork/bookstore/api/{Nombre del archivo .php}

-Obtener Libro: Comprobamos la URL que se encuentre correctamente, el método es 'GET', después daremos clic en "Send" y nos arrojará los resultados.



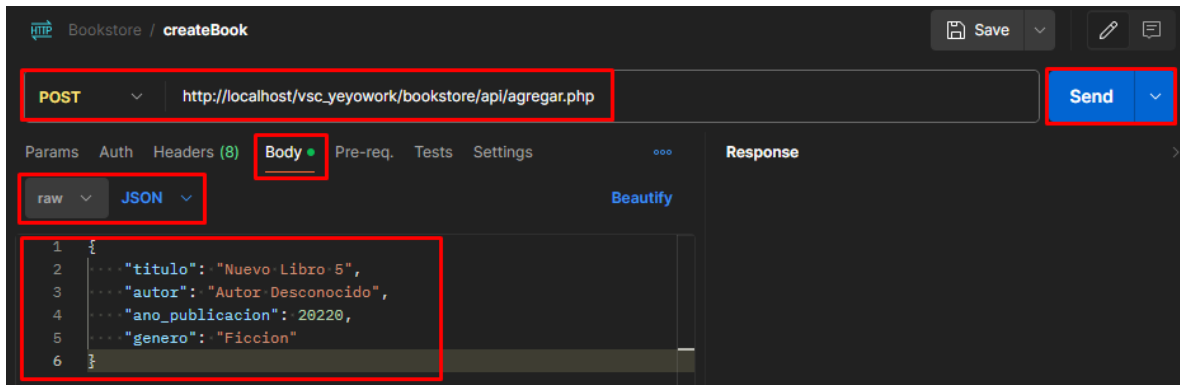
-Obtener Libro por Id: Comprobamos la URL que se encuentre correctamente, el método es 'GET', después daremos clic en "Send" y nos arrojará los resultados.



-Crear Libro: Comprobamos la URL que se encuentre correctamente, el método es 'POST'. Configura el cuerpo de la petición en Postman seleccionando "Body" -> "raw" y eligiendo "JSON" en el menú desplegable. Colocando la siguiente información (Lo creas de acuerdo con preferencias).

```
{
  "titulo": "Nuevo Libro",
  "autor": "Autor Desconocido",
  "ano_publicacion": 2020,
  "genero": "Ficcion"
}
```

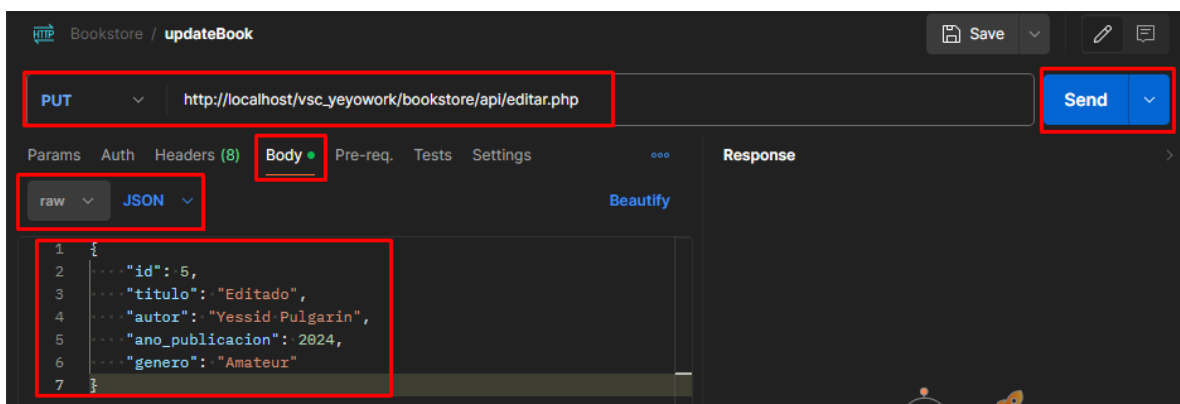
después daremos clic en “Send” y nos arrojará los resultados.



-Editar Libro: Comprobamos la URL que se encuentre correctamente, el método es 'PUT'. Configura el cuerpo de la petición en Postman seleccionando "Body" -> "raw" y eligiendo "JSON" en el menú desplegable. Colocando la siguiente información (Lo editas de acuerdo con preferencias).

```
{
  "id": 5,
  "titulo": "Editado",
  "autor": "Yessid Pulgarin",
  "ano_publicacion": 2024,
  "genero": "Amateur"
}
```

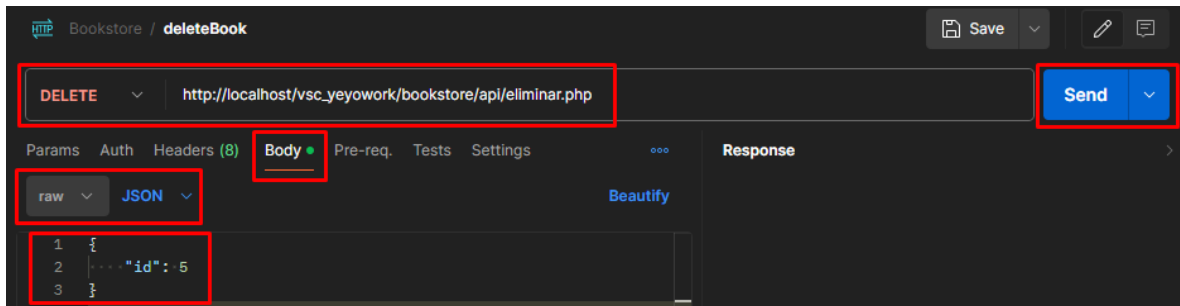
después daremos clic en “Send” y nos arrojará los resultados.



-Eliminar Libro: Comprobamos la URL que se encuentre correctamente, el método es 'DELETE'. Configura el cuerpo de la petición en Postman seleccionando "Body" -> "raw" y eligiendo "JSON" en el menú desplegable. Colocando la siguiente información (Lo eliminas de acuerdo con preferencias).

```
{
  "id": 5
}
```

después daremos clic en “Send” y nos arrojará los resultados.

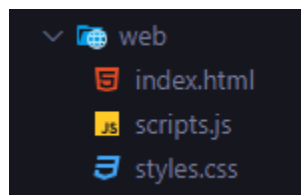


5 Código Fuente Interfaz Web

5.1 Interfaz Web

Creemos una carpeta dentro del proyecto llamada “**web**”, donde se contendrá cada archivo con los que podremos visualizar una interfaz y así poder interactuar entre la web y la api. Creamos la carpeta y allí incluimos la creación de los siguientes archivos de index.html, scripts.js, styles.css:

Quedaremos con la siguiente estructura:



5.1.1 Index.html

- En el archivo **index.html** encontraremos el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Catálogo de Libros</title>
    <link rel="stylesheet" href="styles.css" />
    <!-- Enlazar el archivo de estilos -->
  </head>
  <body>
```

```
<div class="container">
  <h1>Catálogo de Libros</h1>

  <div class="form-group">
    <input type="hidden" id="id" />
    <label for="titulo">Título</label>
    <input type="text" id="titulo" placeholder="Título" />
  </div>

  <div class="form-group">
    <label for="autor">Autor</label>
    <input type="text" id="autor" placeholder="Autor" />
  </div>

  <div class="form-group">
    <label for="ano_publicacion">Año de Publicación</label>
    <input
      type="number"
      id="ano_publicacion"
      placeholder="Año de Publicación"
    />
  </div>

  <div class="form-group">
    <label for="genero">Género</label>
    <input type="text" id="genero" placeholder="Género" />
  </div>

  <div class="form-group">
    <button onclick="agregarLibro()">Agregar</button>
    <button onclick="editarLibro()">Editar</button>
  </div>

  <div class="form-group">
    <label for="buscarId">Buscar por ID:</label>
    <input type="number" id="buscarId" placeholder="ID del libro" />
    <button onclick="buscarLibro()">Buscar por ID</button>
  </div>

  <h2>Lista de Libros</h2>
  <ul id="lista"></ul>

  <script src="scripts.js"></script>
  <!-- Enlazar el archivo de JavaScript -->
</div>
```

```
</body>
</html>
```

5.1.2 Scripts.js

- En el archivo **scripts.js** encontraremos el siguiente código:

```
const baseUrl = "http://localhost/vsc_yeyowork/bookstore/api";

async function agregarLibro() {
  const titulo = document.getElementById("titulo").value;
  const autor = document.getElementById("autor").value;
  const ano_publicacion = document.getElementById("ano_publicacion").value;
  const genero = document.getElementById("genero").value;

  const response = await fetch(`${baseUrl}/agregar.php`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ titulo, autor, ano_publicacion, genero }),
  });

  const data = await response.json();
  alert(data.message);
  obtenerLibros();
}

async function editarLibro() {
  const id = document.getElementById("id").value;
  const titulo = document.getElementById("titulo").value;
  const autor = document.getElementById("autor").value;
  const ano_publicacion = document.getElementById("ano_publicacion").value;
  const genero = document.getElementById("genero").value;

  const response = await fetch(`${baseUrl}/editar.php`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ id, titulo, autor, ano_publicacion, genero }),
  });

  const data = await response.json();
  alert(data.message);
  obtenerLibros();
}
```

```

async function eliminarLibro(id) {
  const response = await fetch(`${baseUrl}/eliminar.php`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ id }),
  });

  const data = await response.json();
  alert(data.message);
  obtenerLibros();
}

async function obtenerLibros() {
  const response = await fetch(`${baseUrl}/obtener.php`);
  const libros = await response.json();

  const lista = document.getElementById("lista");
  lista.innerHTML = "";

  libros.forEach((libro) => {
    const item = document.createElement("li");
    item.innerHTML = `
      <span>${libro.titulo} - ${libro.autor} -
      ${libro.ano_publicacion} - ${libro.genero}</span>
      <div class="button-container">
        <button onclick="editarFormulario(${libro.id},
        '${libro.titulo}', '${libro.autor}', ${libro.ano_publicacion},
        '${libro.genero}')">Editar</button>
        <button
        onclick="eliminarLibro(${libro.id})">Eliminar</button>
      </div>
    `;
    lista.appendChild(item);
  });
}

async function buscarLibro() {
  const id = document.getElementById("buscarId").value;

  if (!id) {
    alert("Ingrese un ID válido para buscar.");
    return;
  }
}

```

```

}

try {
  const response = await fetch(`${baseUrl}/obtener_por_id.php?id=${id}`);
  const libro = await response.json();

  if (libro && !libro.message) {
    const lista = document.getElementById("lista");
    lista.innerHTML = "";

    const item = document.createElement("li");
    item.innerHTML = `
      <span>${libro.titulo} - ${libro.autor} -
    ${libro.ano_publicacion} - ${libro.genero}</span>
      <div class="button-container">
        <button onclick="editarFormulario(${libro.id},
    '${libro.titulo}', '${libro.autor}', ${libro.ano_publicacion},
    '${libro.genero}')">Editar</button>
        <button
    onclick="eliminarLibro(${libro.id})">Eliminar</button>
      </div>
    `;
    lista.appendChild(item);
  } else {
    alert(
      libro.message || "No se encontró ningún libro con el ID
proporcionado."
    );
  }
} catch (error) {
  console.error("Error al buscar el libro:", error);
  alert(
    "Ocurrió un error al buscar el libro. Verifique la consola para más
detalles."
  );
}
}

function editarFormulario(id, titulo, autor, ano_publicacion, genero) {
  document.getElementById("id").value = id;
  document.getElementById("titulo").value = titulo;
  document.getElementById("autor").value = autor;
  document.getElementById("ano_publicacion").value = ano_publicacion;
  document.getElementById("genero").value = genero;
}

```



```
document.addEventListener("DOMContentLoaded", obtenerLibros);
```

5.1.3 styles.css

- En el archivo **styles.css** encontraremos el siguiente código:

```
body {
  font-family: Arial, sans-serif;
  margin: 20px;
}

.container {
  max-width: 600px;
  margin: 0 auto;
}

h1, h2 {
  text-align: center;
}

.form-group {
  margin-bottom: 15px;
}

label {
  display: block;
  margin-bottom: 5px;
}

input[type="text"], input[type="number"] {
  width: 100%;
  padding: 8px;
  margin-bottom: 10px;
  box-sizing: border-box;
}

button {
  padding: 10px 15px;
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
  margin-right: 5px;
}

button:hover {
```

```

        background-color: #45a049;
    }

    ul {
        list-style-type: none;
        padding: 0;
    }

    li {
        background-color: #f9f9f9;
        padding: 10px;
        margin-bottom: 10px;
        display: flex;
        justify-content: space-between;
        align-items: center;
        border: 1px solid #ddd;
        border-radius: 4px;
    }

    li span {
        flex-grow: 1;
    }

    .button-container {
        display: flex;
        justify-content: space-between;
    }

    .button-container button {
        margin-left: 10px;
        background-color: #008CBA;
    }

    .button-container button:nth-child(2) {
        background-color: #f44336;
    }

```

Una vez realizada la parte de interfaz web podremos ejecutar el proyecto en el navegador con la siguiente URL: http://localhost/vsc_yeyowork/bookstore/web/.

Aquí podremos visualizar la siguiente página:

Catálogo de Libros

Título

Autor

Año de Publicación

Género

Buscar por ID:

Lista de Libros

Como visualizamos, existe un formulario simple para agregar y/o editar un libro, con opciones de buscar un libro por id específico, al final podremos ver cada libro agregado y su respectiva información, donde también se tendrá opciones a un costado de “Editar” y “Eliminar” el libro.

6 Test de Automatización (PHPUnit)

Para escribir pruebas automatizadas para la API de la librería, podemos usar una herramienta como PHPUnit para PHP. PHPUnit es un framework de pruebas para PHP que facilita la creación y ejecución de pruebas unitarias y de integración.

6.1 Configuración PHPUnit

Primeramente, instalamos "Composer" si aun no lo tenemos, ejecutando las siguientes líneas de comando en la terminal WSL (en mi caso es debian):

- `sudo apt update` (Actualizamos dependencias)
- `sudo apt install curl php-cli php-mbstring git unzip` (instalamos Composer)
- `curl -sS https://getcomposer.org/installer | php` (Obtenemos las dependencias de composer)
- `sudo mv composer.phar /usr/local/bin/composer` (movemos composer de manera local)

```
yeyo@LAPTOP-8C9NCPQI: ~  
yeyo@LAPTOP-8C9NCPQI:~$ - sudo apt update  
yeyo@LAPTOP-8C9NCPQI:~$ - sudo apt install curl php-cli php-mbstring git unzip  
yeyo@LAPTOP-8C9NCPQI:~$ - curl -sS https://getcomposer.org/installer | php  
yeyo@LAPTOP-8C9NCPQI:~$ - sudo mv composer.phar /usr/local/bin/composer  
yeyo@LAPTOP-8C9NCPQI:~$
```

Si aún no lo tienes instalado, puedes instalarlo globalmente mediante Composer, ejecutando el siguiente comando en la terminal WSL (en mi caso es debian):

- `composer global require phpunit/phpunit`

```
Selecciónar yeyo@LAPTOP-8C9NCPQI: ~  
yeyo@LAPTOP-8C9NCPQI:~$ composer global require phpunit/phpunit
```

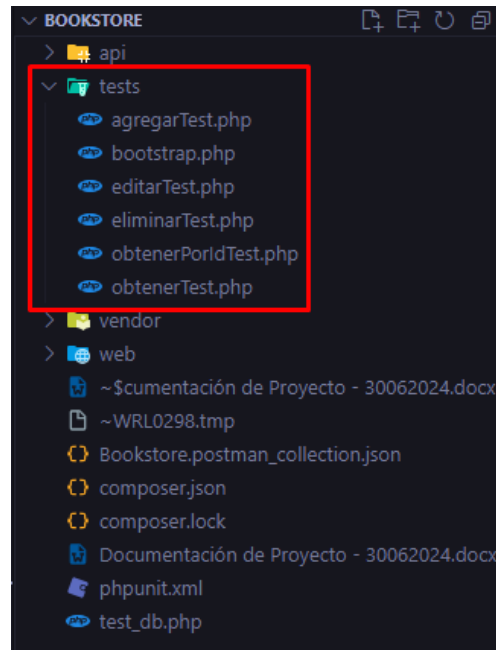
Recordemos instalar las dependencias de Composer en el directorio del proyecto para poder realizar la ejecución de las pruebas PHPUnit:

- `composer install`

Ingresamos al directorio del proyecto mediante la terminal WSL (`cd "/mnt/c/xampp/htdocs/vsc_yeyowork/bookstore"`) y ejecutamos el comando anterior

```
yeyo@LAPTOP-8C9NCPQI: /mnt/c/xampp/htdocs/vsc_yeyowork/bookstore  
yeyo@LAPTOP-8C9NCPQI:/mnt/c/xampp/htdocs/vsc_yeyowork/bookstore$ composer install
```

Crear la estructura de directorios para pruebas: Asegúrate de que el proyecto tenga una estructura como la siguiente:



Crear el archivo de configuración de PHPUnit: En la raíz del proyecto, crea un archivo phpunit.xml para configurar PHPUnit con el siguiente código.

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="tests/bootstrap.php">
  <testsuites>
    <testsuite name="API Tests">
      <directory>tests</directory>
    </testsuite>
  </testsuites>
  <php>
    <server name="DB_HOST" value="127.0.0.1"/>
    <server name="DB_PORT" value="3306"/>
    <server name="DB_USER" value="root"/>
    <server name="DB_PASS" value=""/>
    <server name="DB_NAME" value="libreria"/>
    <server name="DB_SOCKET" value="/var/run/mysqld/mysqld.sock"/>
  </php>
</phpunit>
```

Crear un archivo de arranque: En el directorio “tests”, crea un archivo **bootstrap.php** para configurar el entorno de prueba:

```

<?php
// tests/bootstrap.php
require __DIR__ . '/../vendor/autoload.php';

$host = getenv('DB_HOST');
$port = getenv('DB_PORT');
$db    = getenv('DB_NAME');
$user  = getenv('DB_USER');
$pass  = getenv('DB_PASS');
$charset = 'utf8mb4';
$socket = getenv('DB_SOCKET');

$dsn = "mysql:host=$host;dbname=$db;charset=$charset;unix_socket=$socket";
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES  => false,
];

try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e->getCode());
}

$GLOBALS['pdo'] = $pdo;

// Incluir archivos de la API
require_once __DIR__ . '/../api/agregar.php';
require_once __DIR__ . '/../api/editar.php';
require_once __DIR__ . '/../api/eliminar.php';
require_once __DIR__ . '/../api/obtener.php';
require_once __DIR__ . '/../api/obtener_por_id.php';

```

A continuación, podremos empezar a escribir el código para las pruebas, teniendo en cuenta que se debe de validar que no se puedan agregar o editar libros sin los campos requeridos:

- En el archivo **agregarTest.php** encontraremos el siguiente código:

```

<?php
// tests/agregarTest.php

use PHPUnit\Framework\TestCase;

```

```

class AgregarTest extends TestCase
{
    protected $pdo;

    protected function setUp(): void
    {
        $this->pdo = $GLOBALS['pdo'];
        $this->pdo->exec("DELETE FROM libros WHERE titulo = 'Libro de
Prueba'");
    }

    public function testAgregarLibroExitoso()
    {
        $data = [
            'titulo' => 'Libro de Prueba',
            'autor' => 'Autor de Prueba',
            'ano_publicacion' => 2023,
            'genero' => 'Ficción'
        ];

        $_POST = $data;
        ob_start();
        require '../api/agregar.php';
        $output = ob_get_clean();

        $this->assertStringContainsString('Libro agregado exitosamente',
$output);

        // Verificar que el libro fue agregado en la base de datos
        $stmt = $this->pdo->prepare("SELECT * FROM libros WHERE titulo =
?");
        $stmt->execute([$data['titulo']]);
        $book = $stmt->fetch();

        $this->assertNotFalse($book);
    }

    public function testAgregarLibroSinCamposRequeridos()
    {
        $data = [
            'titulo' => '',
            'autor' => '',
            'ano_publicacion' => '',
            'genero' => ''
        ];
    }
}

```

```

        $_POST = $data;
        ob_start();
        require '../api/agregar.php';
        $output = ob_get_clean();

        $this->assertStringContainsString('Todos los campos son
obligatorios', $output);
    }
}

```

- En el archivo **editarTest.php** encontraremos el siguiente código:

```

<?php

// tests/editarTest.php

use PHPUnit\Framework\TestCase;

class EditarTest extends TestCase
{
    protected $pdo;

    protected function setUp(): void
    {
        $this->pdo = $GLOBALS['pdo'];
        $this->pdo->exec("DELETE FROM libros WHERE titulo = 'Libro de Prueba
Editado'");
        $this->pdo->exec("INSERT INTO libros (titulo, autor,
ano_publicacion, genero) VALUES ('Libro de Prueba', 'Autor de Prueba', 2023,
'Ficción')");
    }

    public function testEditarLibroExitoso()
    {
        $data = [
            'id' => 1,
            'titulo' => 'Libro Editado',
            'autor' => 'Autor Editado',
            'ano_publicacion' => 2023,
            'genero' => 'No Ficción'
        ];

        $_POST = $data;
    }
}

```



```

        ob_start();
        require '../api/editar.php';
        $output = ob_get_clean();

        $this->assertStringContainsString('Libro editado exitosamente',
$output);

        // Verificar que el libro fue editado en la base de datos
        $stmt = $this->pdo->prepare("SELECT * FROM libros WHERE id = ?");
        $stmt->execute([$data['id']]);
        $book = $stmt->fetch();

        $this->assertEquals('Libro de Prueba Editado', $book['titulo']);
        $this->assertEquals('Autor de Prueba Editado', $book['autor']);
        $this->assertEquals(2024, $book['ano_publicacion']);
        $this->assertEquals('No Ficción', $book['genero']);
    }

    public function testEditarLibroSinCamposRequeridos()
    {
        $data = [
            'id' => 1,
            'titulo' => '',
            'autor' => '',
            'ano_publicacion' => '',
            'genero' => ''
        ];

        $_POST = $data;
        ob_start();
        require '../api/editar.php';
        $output = ob_get_clean();

        $this->assertStringContainsString('Todos los campos son
obligatorios', $output);
    }
}

```

- En el archivo **eliminarTest.php** encontraremos el siguiente código:

```

<?php

// tests/eliminarTest.php

```

```

use PHPUnit\Framework\TestCase;

class EliminarTest extends TestCase
{
    protected $pdo;

    protected function setUp(): void
    {
        $this->pdo = $GLOBALS['pdo'];
        $this->pdo->exec("DELETE FROM libros WHERE titulo = 'Libro de Prueba
a Eliminar'");
        $this->pdo->exec("INSERT INTO libros (titulo, autor,
ano_publicacion, genero) VALUES ('Libro de Prueba a Eliminar', 'Autor de
Prueba', 2023, 'Ficción')");
    }

    public function testEliminarLibroExitoso()
    {
        $stmt = $this->pdo->prepare("SELECT id FROM libros WHERE titulo =
'Libro de Prueba a Eliminar'");
        $stmt->execute();
        $book = $stmt->fetch();

        $data = [
            'id' => $book['id']
        ];

        $_POST = $data;
        ob_start();
        require '../api/eliminar.php';
        $output = ob_get_clean();

        $this->assertStringContainsString('Libro eliminado exitosamente',
$output);

        // Verificar que el libro fue eliminado de la base de datos
        $stmt = $this->pdo->prepare("SELECT * FROM libros WHERE id = ?");
        $stmt->execute([$data['id']]);
        $book = $stmt->fetch();

        $this->assertFalse($book);
    }

    public function testEliminarLibroSinId()
    {

```

```

        $data = ['id' => ''];

        $_POST = $data;
        ob_start();
        require '../api/eliminar.php';
        $output = ob_get_clean();

        $this->assertStringContainsString('ID del libro es obligatorio',
$output);
    }
}

```

- En el archivo **obtenerPorIdTest.php** encontraremos el siguiente código:

```

<?php

// tests/obtenerPorIdTest.php

use PHPUnit\Framework\TestCase;

class ObtenerPorIdTest extends TestCase
{
    protected $pdo;

    protected function setUp(): void
    {
        $this->pdo = $GLOBALS['pdo'];
        $this->pdo->exec("DELETE FROM libros WHERE titulo = 'Libro de Prueba
por ID'");
        $this->pdo->exec("INSERT INTO libros (titulo, autor,
ano_publicacion, genero) VALUES ('Libro de Prueba por ID', 'Autor de
Prueba', 2023, 'Ficción')");
    }

    public function testObtenerLibroPorIdExitoso()
    {
        $stmt = $this->pdo->prepare("SELECT id FROM libros WHERE titulo =
'Libro de Prueba por ID'");
        $stmt->execute();
        $book = $stmt->fetch();

        $data = [
            'id' => $book['id']
        ];
    }
}

```

```

        $_GET = $data;
        ob_start();
        require '../api/obtener_por_id.php';
        $output = ob_get_clean();

        $this->assertStringContainsString('success', $output);
        $this->assertStringContainsString('data', $output);
        // $libro = json_decode($output, true);
        // $this->assertIsArray($libro);
        // $this->assertArrayHasKey('id', $libro);
    }

    public function testObtenerLibroPorIdInvalido()
    {
        $_GET['id'] = 'invalid';
        ob_start();
        require '../api/obtener_por_id.php';
        $output = ob_get_clean();

        $this->assertStringContainsString('ID del libro es obligatorio',
$output);
    }
}

```

- En el archivo **obtenerTest.php** encontraremos el siguiente código:

```

<?php

// tests/obtenerTest.php

use PHPUnit\Framework\TestCase;

class ObtenerTest extends TestCase
{
    protected $pdo;

    protected function setUp(): void
    {
        $this->pdo = $GLOBALS['pdo'];
    }

    public function testObtenerLibros()
    {

```

```

        ob_start();
        require '../api/obtener.php';
        $output = ob_get_clean();

        $this->assertStringContainsString('success', $output);
        $this->assertStringContainsString('data', $output);

        // $libros = json_decode($output, true);
        // $this->assertIsArray($libros);
    }
}

```

6.2 Ejecución de Pruebas

Navegar al Directorio del Proyecto mediante la terminal WSL (`cd "/mnt/c/xampp/htdocs/vsc_yeyowork/bookstore"`):

```

yeyo@LAPTOP-8C9NCPQI: /mnt/c/xampp/htdocs/vsc_yeyowork/bookstore
yeyo@LAPTOP-8C9NCPQI:~$ cd "/mnt/c/xampp/htdocs/vsc_yeyowork/bookstore"
yeyo@LAPTOP-8C9NCPQI:/mnt/c/xampp/htdocs/vsc_yeyowork/bookstore$

```

Ejecutar las Pruebas: ejecutando el siguiente comando en el directorio del proyecto.

- `phpunit`

```

yeyo@LAPTOP-8C9NCPQI: /mnt/c/xampp/htdocs/vsc_yeyowork/bookstore
yeyo@LAPTOP-8C9NCPQI:~$ cd "/mnt/c/xampp/htdocs/vsc_yeyowork/bookstore"
yeyo@LAPTOP-8C9NCPQI:/mnt/c/xampp/htdocs/vsc_yeyowork/bookstore$ phpunit

```

En caso de que la ejecución de las pruebas sea fallida, también podrás intentarlo con el siguiente comando:

- `vendor/bin/phpunit`

```

yeyo@LAPTOP-8C9NCPQI: /mnt/c/xampp/htdocs/vsc_yeyowork/bookstore
yeyo@LAPTOP-8C9NCPQI:~$ cd "/mnt/c/xampp/htdocs/vsc_yeyowork/bookstore"
yeyo@LAPTOP-8C9NCPQI:/mnt/c/xampp/htdocs/vsc_yeyowork/bookstore$ vendor/bin/phpunit

```

Con estos pasos, estarás listo para instalar PHPUnit y ejecutar tus pruebas de forma efectiva en el entorno de desarrollo. Esto ejecutará todas las pruebas en el directorio “tests” y te mostrará un resumen de los resultados. Con estas pruebas, estarás validando cada endpoint de la API.