Informe de SQLite

Tema: Práctica SQLite

Nota

Estudiante	Escuela	Asignatura
Victor Gonzalo Maldonado Vilca vmaldonadov@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 1702122

Tarea	Tema	Duración
04	Práctica SQLite	2 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 10 de mayo de 2024	Al 26 de mayo de 2024

1. Introducción

1.1. Node.js

Es una plataforma de tiempo de ejecución de JavaScript que permite ejecutar código JavaScript en el servidor. Con Node.js, podemos crear servidores web eficientes y escalables. Es especialmente útil en el desarrollo de aplicaciones web en tiempo real y aplicaciones de una sola página que requieren una comunicación constante entre el cliente y el servidor.

1.2. Ajax

Es una tecnología que permite actualizar partes específicas de una página web sin necesidad de recargarla por completo. Con AJAX, las solicitudes al servidor se realizan de forma asíncrona, lo que significa que el usuario puede interactuar con la página sin interrupciones. Esto mejora significativamente la experiencia del usuario al proporcionar respuestas rápidas y dinámicas.

1.3. Json

Es un formato ligero y legible para el intercambio de datos entre el cliente y el servidor. Es fácil de entender tanto para los humanos como para las máquinas, lo que lo hace ideal para la comunicación en aplicaciones web. JSON se utiliza comúnmente para enviar datos entre el cliente y el servidor, ya sea en respuesta a solicitudes AJAX o para almacenar datos en bases de datos.

1.4. SQL

Es un lenguaje de programación utilizado para administrar bases de datos relacionales. Con SQL, podemos crear, leer, actualizar y eliminar datos en una base de datos de manera eficiente. Es fundamental en el desarrollo web para almacenar y manipular datos de manera estructurada.

1.5. mySQL

Es un sistema de gestión de bases de datos relacionales (RDBMS) ampliamente utilizado y de código abierto. Destaca por su rendimiento, confiabilidad y facilidad de uso, siendo una opción popular para una variedad de aplicaciones, desde pequeños sitios web hasta grandes sistemas empresariales.

2. Objetivos

- Crear una aplicación web que utilice SQLite para poder manipular información de películas.
- Implementar la comunicación entre el cliente y el servidor utilizando AJAX y JSON.

3. Tarea

- Realice un ejemplo de sqlite libre, con la base de datos imdb.db, tendrá que usar ajax y json para la comunicación entre cliente y servidor; así como git para sus versiones. No es necesario que entregue un programa funcionando, pero si es importante que muestre los errores encontrados.

4. Entregables

- Informe del proyecto
- Repositorio en gitHub
- Link del vídeo Explicativo

5. Equipos, materiales y temas utilizados

- Node.js
- JavaScript
- Ajax
- Json
- Xampp
- SQL
- mySQL

6. URL de Repositorio Github

- URL: Repositorio SQLite en GitHub
- `https://github.com/Victor-Gonzalo-Maldonado-Vilca/SQLite.git`

7. Link de Video

- Link del vídeo explicativo
- `https://www.youtube.com/watch?v=GqJOymFuurA`

8. Metodología

8.1. Creación del entorno de trabajo

(Tener instalado Node.js)

8.1.1. Creación de carpetas

- Directorio Actual *(Nos encontraremos en la siguiente ruta)*

Listing 1: Directorio Actual

```
D:\UNSA\PW2\Teoria\SQLite\src
```

- Creamos carpeta de Trabajo

Listing 2: Carpeta de Trabajo

```
mkdir Programa && cd Programa
```

8.1.2. Instalar modulos necesarios

(Recordar: Agregar la ruta de instalación de Node.js y npm a las variables de entorno)

- **Instalar Modulo Express:**

Listing 3: Install Express

```
npm install express
```

- **Instalar Modulo mySQL:**

Listing 4: Install mySQL

```
npm install mysql
```

8.2. Uso de Xampp

(Tener instalado xampp)

8.2.1. Descripción

Es un paquete de software que incluye Apache, MySQL, PHP y Perl. Permite crear fácilmente un entorno de desarrollo web local en sistemas Windows, Linux y macOS. Es ampliamente utilizado por desarrolladores web para comenzar a desarrollar y probar aplicaciones sin tener que configurar cada componente por separado.

8.2.2. Activar puertos mysql y apache de xampp

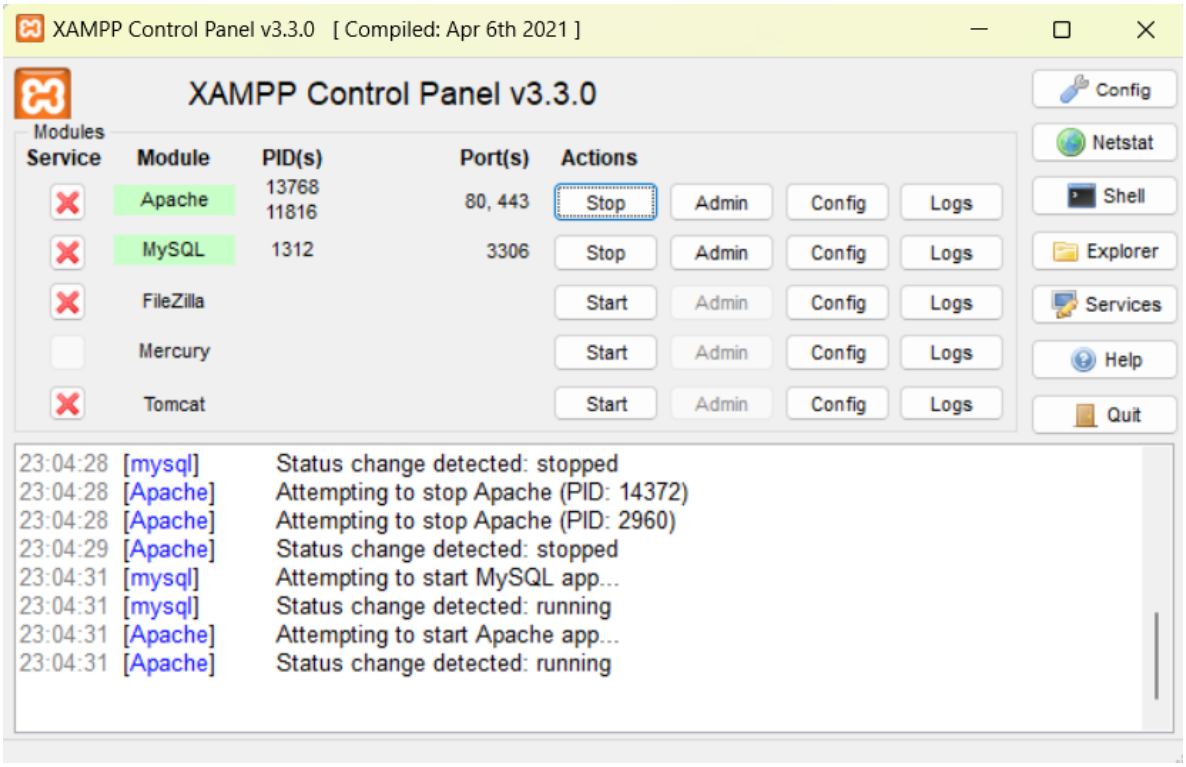

Figura 1: Activación de puertos

8.2.3. Uso del archivo imdb.db

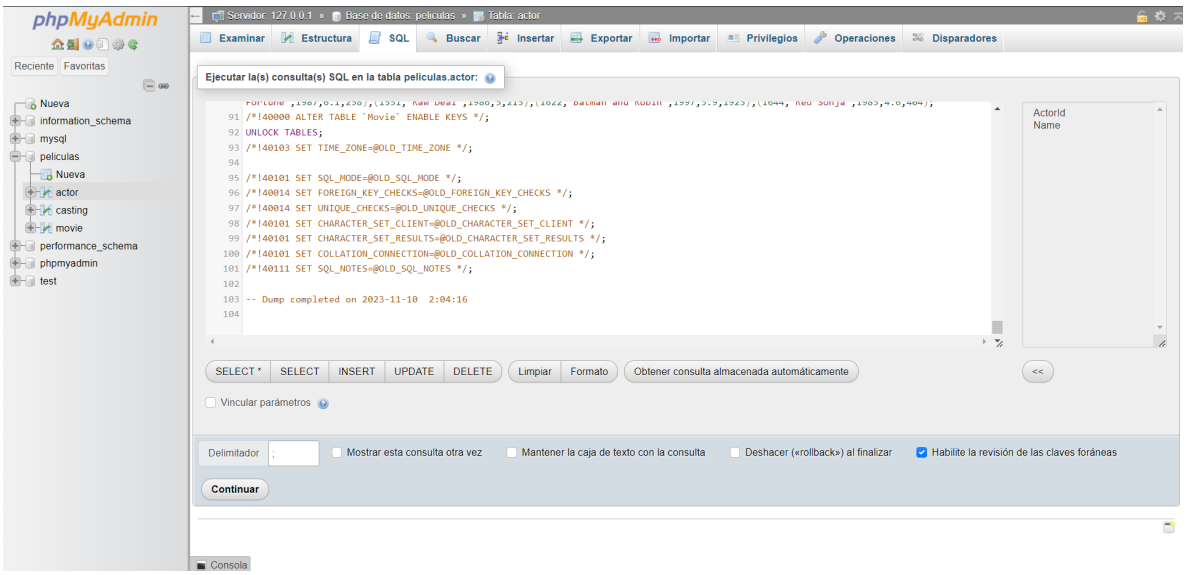

Figura 2: Uso de imdb.db

9. Desarrollo del trabajo

9.1. Archivo index.js *(Parte - Servidor)*

9.1.1. Fragmentos del Código

- En esta parte se importa los módulos path, express, body-parser y mysql, y luego crea una instancia de la aplicación Express. Estos módulos son utilizados para manejar rutas de archivos, configurar el servidor web, analizar datos enviados desde formularios HTML, y conectar y comunicarse con una base de datos MySQL.

Listing 5: Importación

```
1   const path = require('path');
2   const express = require('express');
3   const bp = require('body-parser');
4   const mysql = require('mysql')
5   const app = express();
```

- Esta sección del código configura el servidor Express para servir archivos estáticos desde el directorio 'pub' , analizar datos JSON y analizar datos de formularios HTML.

Listing 6: Configuración

```
7   app.use(express.static('pub'));
8   app.use(bp.json());
9   app.use(bp.urlencoded({ extended: true }));
```

- Este fragmento de código inicia el servidor Express en el puerto 3000 y muestra un mensaje en la consola indicando que el servidor está escuchando en la dirección http://localhost:3000.

Listing 7: Servidor Inicial

```
12  app.listen(3000, () => {
13    console.log("Escuchando en: http://localhost:3000")
14  });
```

- Este bloque de código establece una conexión a una base de datos MySQL en el servidor local. Se especifican los detalles de la conexión, incluyendo el host, el puerto, el nombre de usuario, la contraseña y el nombre de la base de datos a la que se desea conectar. En este caso, se está intentando conectar a una base de datos llamada 'peliculas' con el usuario 'Victor' y la contraseña proporcionada.

Listing 8: conexion mySQL

```
20  const conexion = mysql.createConnection({
21    host: 'localhost',
22    port: 3306,
23    user: 'Victor',
24    password: 'maldonado100204.',
25    database: 'peliculas'
26  });
```

- Este bloque de código establece una conexión a una base de datos MySQL y define una ruta en la aplicación Express para manejar solicitudes POST en '/basedata'.
  Cuando se recibe una solicitud POST en esa ruta, se extrae un dato del cuerpo de la solicitud y se realiza una consulta SQL a la base de datos utilizando ese dato como parámetro. Si la consulta es exitosa, se devuelve el resultado al cliente en formato JSON. Si hay un error durante la conexión o la consulta, se devuelve un mensaje de error al cliente.

Listing 9: consulta mySQL

```
28      conexion.connect((err) => {
29        if (err) {
30          console.error('Error de conexion mysql ', err);
31          return;
32        } else {
33          console.log('Conexion Establecida mysql')
34        }
35
36        app.post('/basedata',(request, response) => {
37          const fecha = request.body.year;
38          console.log(fecha);
39          //consulta sql
40          conexion.query('SELECT * FROM movie WHERE Year =
                ?',[fecha],(err,filas) => {
41            if (err) {
42              console.error('Error en la consulta')
43              response.status(500).json({error: 'Error en la consulta'});
44            } else {
45              //Respuesta del servidor
46              console.log('Consulta realizada')
47              response.json(filas);
48            }
49          });
50        });
51      });
```

9.2. Archivo index.html

9.2.1. Fragmentos del Código

- Este fragmento de código HTML define la estructura básica de una página web con el título "SQLitez metadatos como el autor y la descripción. También enlaza un archivo CSS externo llamado .estilos.css"para aplicar estilos a la página.

Listing 10: Head

```
4       <title>SQLite</title>
5       <meta charset="utf-8"/>
6       <meta name="author" content="Victor Gonzalo Maldonado Vilca"/>
7       <meta name="description" content="Ejemplo Libre SQLite"/>
8       <link rel="stylesheet" href="estilos.css"/>
```

- Este fragmento de código HTML crea una página web que muestra un título Tabla de Películas seguido de un formulario para ingresar un año. El formulario incluye un campo de entrada de tipo número con restricciones de rango para el año (de 1900 a 2024) y un botón de envío etiquetado como Generar. Cuando el formulario se envía, se espera que se genere una tabla de películas relacionadas con el año ingresado. El resultado de la tabla se mostrará en un contenedor con el id table.

Listing 11: Body

```
12      <h1>Tabla de Peliculas</h1>
13      <form id="formulario">
14        <label for="year">Ingrese Year:</label>
15        <input type="number" placeholder="Year superior a 1899 e inferior 2025"
              id="year" name="year" min="1900" max="2024"/>
16        <input type="submit" value="Generar" id="submit"/>
17      </form>
18      <div id="table"></div>
```

- Esta etiqueta script enlaza el archivo JavaScript programa.js con la página HTML actual. Esto permite que el navegador cargue y ejecute el código JavaScript contenido en el archivo programa.js cuando se renderiza la página web.

Listing 12: Script

```
20      <script src="programa.js"></script>
```

9.3. Archivo programa.js *(Se creo directorio 'pub' => Parte - Cliente)*

9.3.1. Fragmentos del Código

- Esta función recibe un parámetro fecha que representa el año de la película a buscar. Luego, realiza una solicitud POST al servidor para obtener datos de películas relacionadas con ese año. Una vez que recibe la respuesta del servidor, procesa los datos y los muestra en una tabla en la página HTML.

Listing 13: function movies()

```
1       function movies(fecha){
2         const url = 'http://localhost:3000/basedata';
3         const data = {
4           year: parseInt(fecha)
5         }
6         console.log(data);
7         //Enviando datos al servidor, uso de json
8         const request = {
9           method: 'POST',
10          headers: {
11            'Content-Type': 'application/json',
12          },
13          body: JSON.stringify(data),
14        }
15
16        fetch(url, request)
```

```
        .then(response => response.json())
        .then(datos => {
          console.log(datos);
          const contenedorTabla = document.querySelector('#table');
          contenedorTabla.innerHTML = '';
          //Verificando que los datos extraidos no esten vacios
          if (datos.length > 0) {

            const tabla = document.createElement('table');
            tabla.classList = 'tablePeliculas';
            const encabezado = tabla.createTHead();
            const filaE = tabla.insertRow();

            //Agregar el encabezado, extrayendolo de la base de datos
            for (const key in datos[0]){
              if(key != 'id' && key != 'Year'){
                const th = document.createElement('th');
                th.textContent = key.toUpperCase();
                filaE.appendChild(th);
              }
            }

            //Insertar filas donde se encontraran los datos correspondientes
            const cuerpoTabla = tabla.createTBody();
            datos.forEach(dato => {
              const fila = cuerpoTabla.insertRow();
              for (const key in dato){
                if(key != 'id' && key != 'Year'){
                  const td = fila.insertCell();
                  td.textContent = dato[key];
                }
              }
            })
            contenedorTabla.appendChild(tabla);
          } else {
            //Caso contrario se genera un parrafo con un texto correspondiente
            const texto = document.createElement('p');
            texto.id = 'text';
            texto.textContent = 'No hay Peliculas que mostrar';
            contenedorTabla.appendChild(texto);
          }
        })
    }
```

- Este bloque de código se ejecuta cuando se carga completamente la página. Busca el elemento HTML correspondiente al campo de entrada de year y configura un evento para el formulario. Cuando se envía el formulario, se llama a la función movies(fecha.value) para obtener y mostrar las películas relacionadas con el año ingresado.

Listing 14: Evento

```
    document.addEventListener('DOMContentLoaded', function(){
```

```
      //Extraer parametro del formulario
      const fecha = document.querySelector(#year);
      //Realizar ciertas funciones al presionar el boton del formulario
      document.querySelector('#formulario').onsubmit = (event) => {
        //Evita que se cargue nuevamente la pagina
        event.preventDefault();
        //llamada al metodo movies()
        movies(fecha.value);
        return false;
      }
    });
```

9.4. Estilos *(En el directorio 'pub')*

- Estos estilos CSS definen el formato básico para una página web que incluye una tabla de películas y un formulario para ingresar el año de las películas a visualizar. Incluyen ajustes de alineación, bordes y colores para garantizar una apariencia consistente y agradable en la interfaz de usuario.

Listing 15: Estilos

```
.tablePeliculas{
  text-align: center;
  border-collapse: collapse;
  margin-top: 10px;
}

.tablePeliculas td, .tablePeliculas th{
  border: 1px solid black;
}

h1 {
  color: #700000;
}

label {
  color: #700000;
}

#year {
  border-radius: 10px;
  border: 1px solid black;
  padding-left: 10px;
  width: 225px;
}

#submit {
  color: white;
  border: 1px solid #640000;
  background: #7e1919;
  border-radius: 10px;
}

```

```
33      #submit:hover {
34        border: 1px solid #7e1919;
35        background: #9a4c4c;
36      }
37
38      #text{
39        margin-top: 20px;
40        color: #7e1919;
41        font-size: 25px;
42        font-style: italic;
43      }
```

9.5. Ejecución

- En caso de que haya películas en la base de datos:

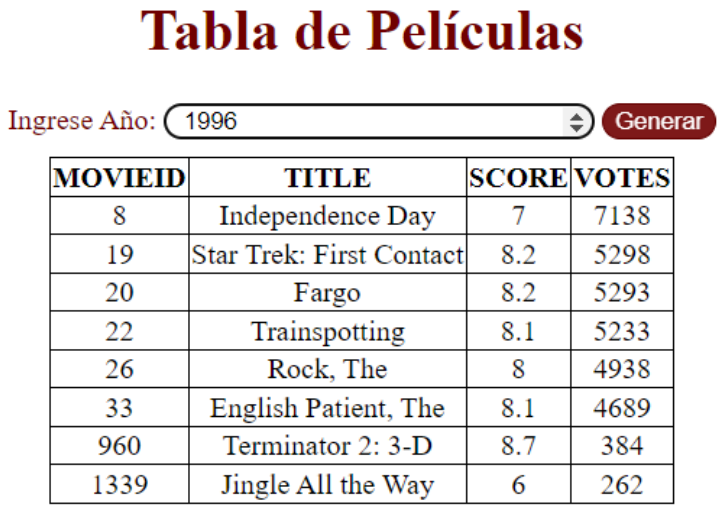

Figura 3: Ejecución 1

- En caso de que no haya películas en la base de datos:

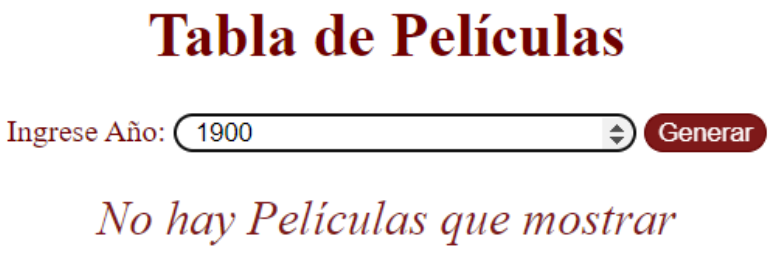

Figura 4: Ejecución 2

9.6. Uso de GitHub

9.6.1. Usuario de GitHub

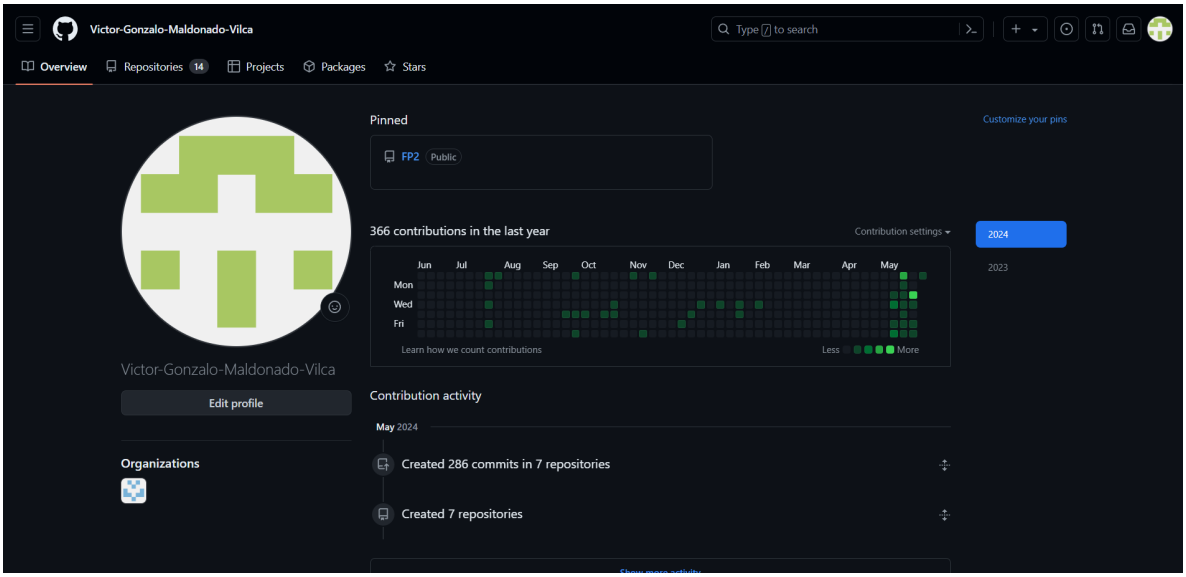

Figura 5: Usuario GitHub

9.6.2. Creación de un Nuevo Repositorio

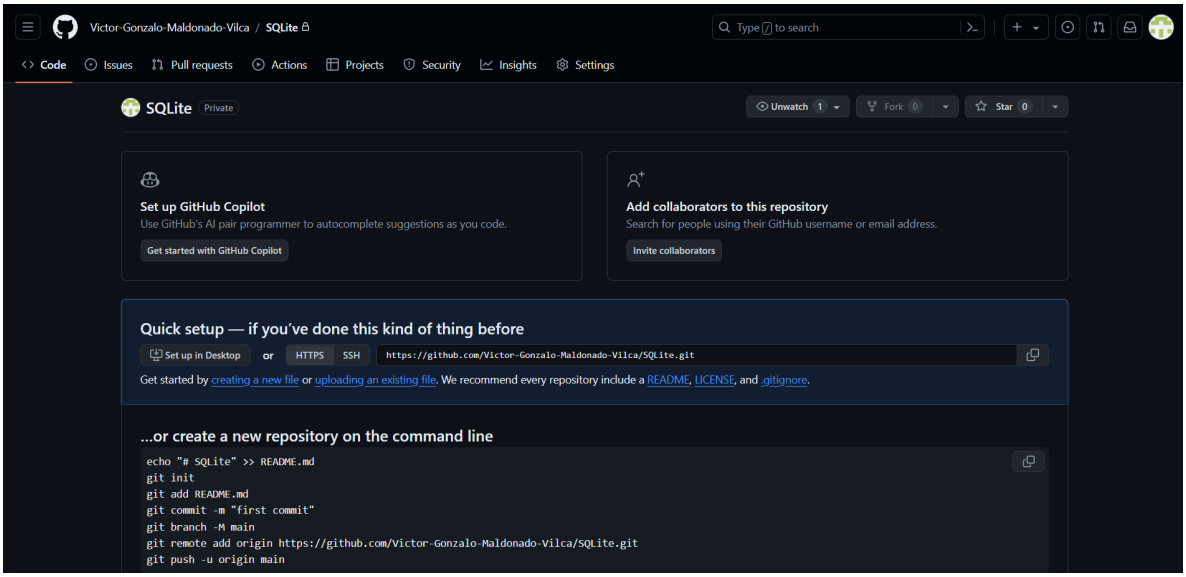

Figura 6: Nuevo Repositorio SQLite

9.6.3. Comandos de Configuración

Listing 16: Configuración Inicial

```
echo "# SQLite" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin
    https://github.com/Victor-Gonzalo-Maldonado-Vilca/SQLite.git
git push -u origin master
```

9.6.4. Implementación de Readme.md

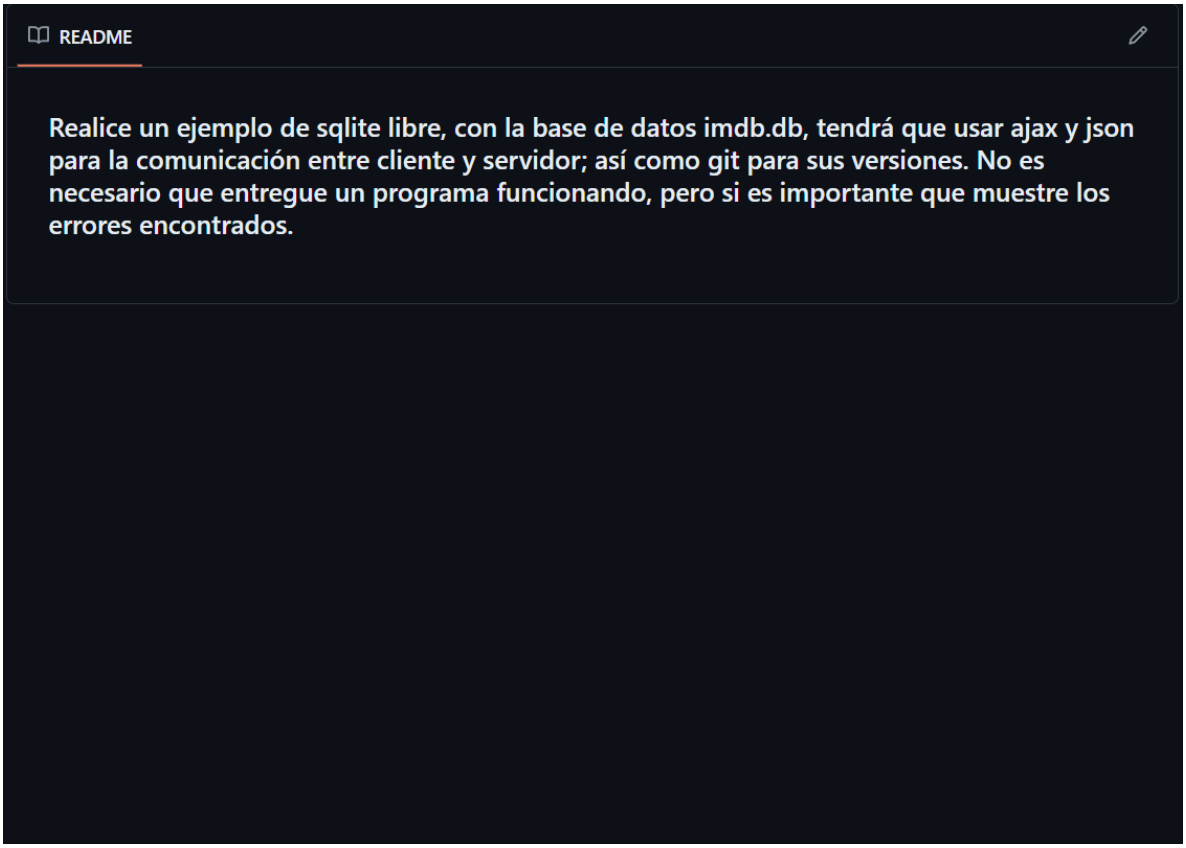

Figura 7: Readme.md

9.6.5. Registro de cambios en mi código

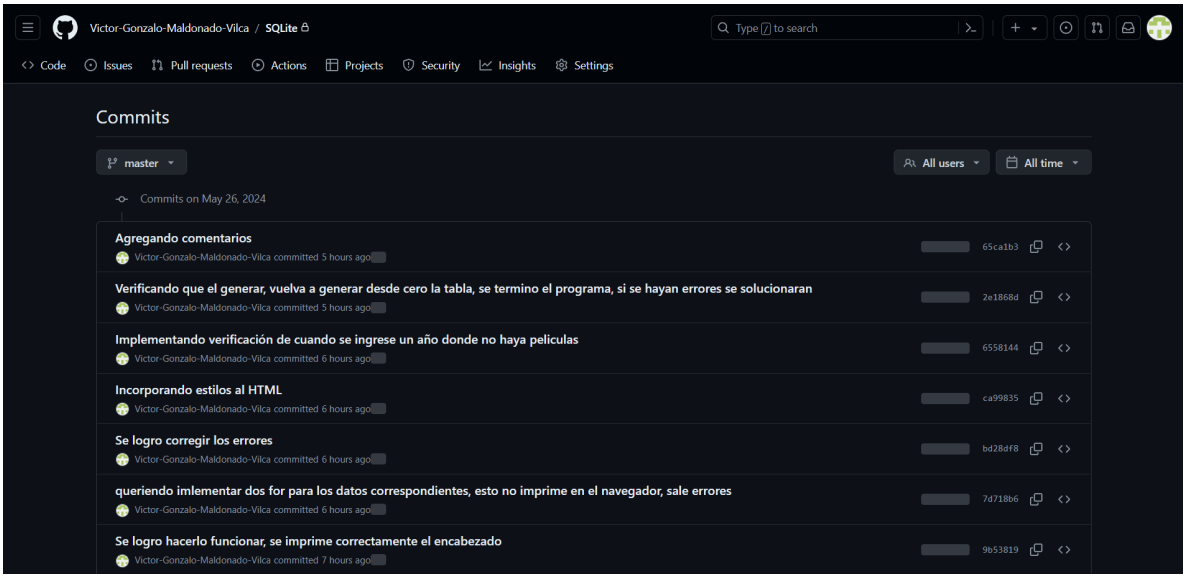

Figura 8: Commits

9.6.6. Repositorio

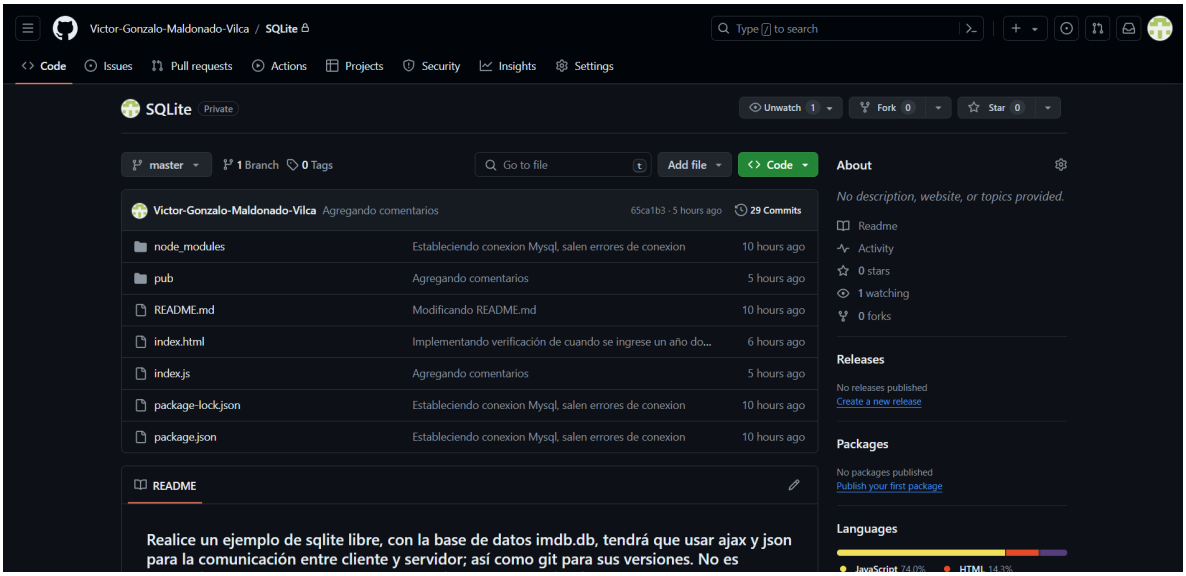

Figura 9: Repositorio SQLite

9.6.7. Proyecto compartido con el profesor de github

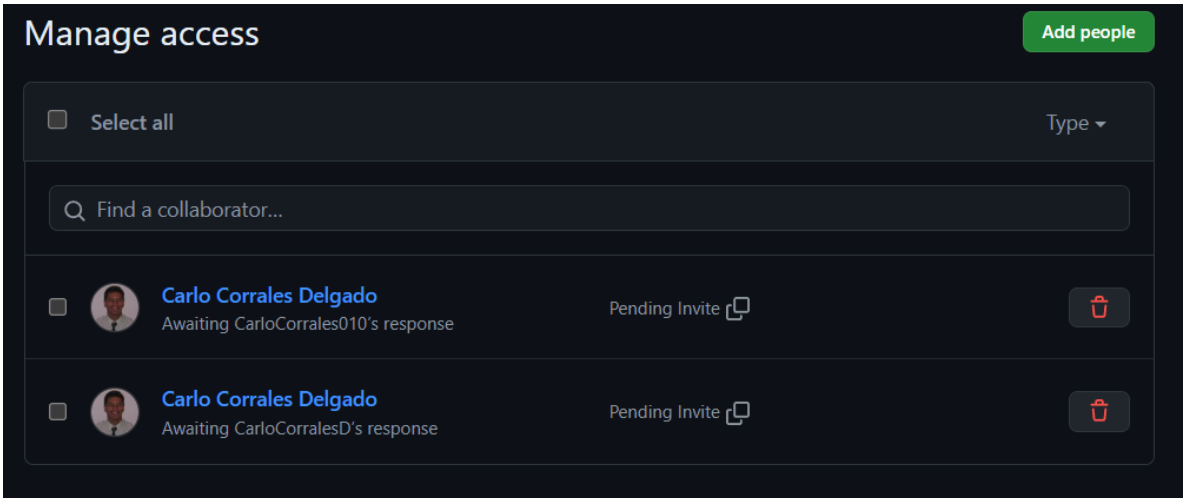

Figura 10: Compartir con el Docente

10. Recomensaciones

- Implementar un manejo adecuado de errores al procesar datos JSON en tu servidor. Esto incluye la detección y manejo de errores de análisis JSON incorrecto, así como la gestión de excepciones en tu lógica relacionada con los datos JSON.
- Implementar un manejo adecuado de errores en tus solicitudes AJAX para informar al usuario sobre posibles problemas de conexión o errores en el servidor.
- Fomentar las mejores prácticas de colaboración, como hacer commits atómicos y descriptivos para nuevas características o correcciones de errores.

11. Conclusiones

- Node.js es una excelente opción para desarrollar servidores debido a su naturaleza asíncrona y su capacidad para manejar grandes cantidades de conexiones concurrentes de manera eficiente.
- AJAX facilita la comunicación entre el cliente y el servidor, lo que permite a las aplicaciones web enviar y recibir datos de forma rápida y eficiente sin necesidad de recargar la página.
- JSON es un formato de datos simple y legible que facilita el intercambio de datos entre servidores y clientes.
- JSON y AJAX son tecnologías fundamentales en el desarrollo de aplicaciones web modernas, mientras que Git es una herramienta esencial para la gestión del código fuente y la colaboración en proyectos de software

11.1. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 2: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		20	

12. Referencias

- https://nodejs.org/en
- https://www.apachefriends.org/es/index.html
- https://www.w3schools.com/