

I

| INFORMACIÓN BÁSICA | | | | | |
|--|--------------------|----------------------|-------|----------------|-----|
| ASIGNATURA: | PROGRAMACIÓN WEB 2 | | | | |
| TÍTULO DE LA PRÁCTICA: | PRÁCTICA SQLITE | | | | |
| NÚMERO DE PRÁCTICA: | 04 | AÑO LECTIVO: | 2025 | NRO. SEMESTRE: | III |
| FECHA DE PRESENTACIÓN | 18/05/2025 | HORA DE PRESENTACIÓN | 22:00 | | |
| INTEGRANTE (s): Auccacusi Conde Brayan Carlos | | | | NOTA: | |
| DOCENTE(s): Corrales Delgado Carlo Jose Luis | | | | | |

| SOLUCIÓN Y RESULTADOS |
|---|
| <p>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</p> <p>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.</p> <p>RESPUESTA</p> <p>Mi ejercicio se trata de un visualizador de tablas de la base de datos. Se trata de hacer una lista a la cual se le agrega un botón, cuando se presiona este me muestra los datos, en formato tabla, de la base de datos.</p> <p>Use SQLite, html, css, js, ajax, node y express</p> <p>ESTRUCTURA DEL PROYECTO</p> <pre> 03_Pratica SQLite --node_modules --imdb.db --index.html --index.js --package-lock.json --package.json --server.js --styles.css </pre> |

SCRIPTS

index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>IMDB Explorer</title>
7   <link rel="stylesheet" href="styles.css">
8 </head>
9 <body>
10  <h1>IMDB Database Explorer</h1>
11  <form id="formulario">
12    <label for="opciones">Selecciona qué quieres ver:</label>
13    <select id="opciones" name="tabla">
14      <option value="Actor">Actores</option>
15      <option value="Casting">Casting</option>
16      <option value="Movie">Películas</option>
17    </select>
18    <button type="button" id="btnConsultar">Consultar</button>
19  </form>
20
21  <div id="resultados">
22    <!-- Aquí se mostrarán los resultados -->
23  </div>
24  <script src="index.js"></script>
25 </body>
26 </html>
```

Se crea un formulario que será enviado al servidor, dentro de una lista para que el usuario seleccione qué tabla de la base de datos quiere ver con un botón, finalmente se conecta al script [index.js](#) que manejará el ajax

server.js

```
1  const express = require('express');
2  const sqlite3 = require('sqlite3').verbose();
3  const app = express();
4  const port = 3000;
5
6  app.use(express.json());
7  app.use(express.static('.'));
8
9  // Conexión a la base de datos
10 const db = new sqlite3.Database('./imdb.db', sqlite3.OPEN_READWRITE, (err) => {
11   if (err) {
12     console.error(err.message);
13   }
14   console.log('Conectado a la base de datos imdb.db');
15 });
16
17 // Obtener datos de las tablas
18 app.post('/api/data', (req, res) => {
19   const { table } = req.body;
20
21   if (!['Actor', 'Casting', 'Movie'].includes(table)) {
22     return res.status(400).json({ error: 'Tabla no válida' });
23   }
24
25   db.get(`SELECT name FROM sqlite_master WHERE type='table' AND name=?`, [table], (err, row) => {
26     if (err) {
27       return res.status(500).json({ error: err.message });
28     }
29
30     if (!row) {
31       return res.status(404).json({ error: `La tabla ${table} no existe en la base de datos` });
32     }
33
34     db.all(`SELECT * FROM ${table}`, [], (err, rows) => {
35       if (err) {
36         return res.status(500).json({ error: err.message });
37       }
38       res.json(rows || []);
39     });
40   });
41 });
42
43 app.listen(port, () => {
44   console.log(`Servidor corriendo en http://localhost:${port}`);
45 });
```

Aquí se configura un servidor que escucha en el puerto 3000, sirve archivos estáticos desde el directorio actual y permite recibir datos en formato JSON. Se conecta a una base de datos llamada imdb.db, y expone una ruta POST (/api/data) que, al recibir el nombre de una tabla (Actor, Casting o Movie) en el cuerpo de la solicitud, primero verifica que el nombre sea válido y que la tabla exista en la base de datos. Si todo es correcto, devuelve todos los registros de esa tabla en formato JSON; si ocurre un error o la tabla no existe, responde con el mensaje de error correspondiente.

index.js

```
1 document.getElementById('btnConsultar').addEventListener('click', function() {
2   const tabla = document.getElementById('opciones').value;
3
4   fetch('/api/data', {
5     method: 'POST',
6     headers: {
7       'Content-Type': 'application/json',
8     },
9     body: JSON.stringify({ table: tabla })
10  })
11  .then(response => response.json())
12  .then(data => {
13    mostrarResultados(data, tabla);
14  })
15  .catch(error => {
16    console.error('Error:', error);
17    document.getElementById('resultados').innerHTML =
18      `<p style="color: red;">Error al cargar los datos: ${error.message}</p>`;
19  });
20 });
21
22 function mostrarResultados(data, tabla) {
23   // Verificar si data es null/undefined o no es un array
24   if (!data || !Array.isArray(data)) {
25     document.getElementById('resultados').innerHTML =
26       `<p style="color: red;">Error: No se recibieron datos válidos para ${tabla}</p>`;
27     return;
28   }
29
30   if (data.length === 0) {
31     document.getElementById('resultados').innerHTML =
32       `<p>No se encontraron registros en la tabla ${tabla}</p>`;
33     return;
34   }
35
36   // Crear tabla HTML
37   let html = `<h2>${tabla} (${data.length} registros)</h2>`;
38   html += `<table><thead><tr>`;
39
40   // Encabezados de la tabla
41   const columnas = Object.keys(data[0]);
42   columnas.forEach(col => {
43     html += `<th>${col}</th>`;
44   });
45   html += `</tr></thead><tbody>`;
46
47   // Filas de datos
48   data.forEach(fila => {
49     html += `<tr>`;
50     columnas.forEach(col => {
51       html += `<td>${fila[col] !== null ? fila[col] : 'NULL'}</td>`;
52     });
53     html += `</tr>`;
54   });
55
56   html += `</tbody></table>`;
57   document.getElementById('resultados').innerHTML = html;
58 }
```

Este script JavaScript se encarga de manejar la interacción del usuario con una página web para consultar datos de una tabla de base de datos. Cuando el usuario hace clic en el botón con ID btnConsultar, se obtiene el valor seleccionado en el elemento con ID opciones (que representa el nombre de una tabla como Actor, Casting o Movie). Luego, se hace una solicitud POST a la ruta /api/data del servidor, enviando el nombre de la tabla en formato JSON. Cuando se recibe la respuesta, se llama a la función mostrarResultados, la cual verifica si los datos recibidos son válidos, y si es así, construye una tabla HTML dinámica mostrando los resultados (con encabezados y filas correspondientes a los campos y valores de los registros). Si ocurre un error durante la solicitud o los datos no son válidos, se muestra un mensaje de error en el elemento con ID resultados.

MÓDULOS

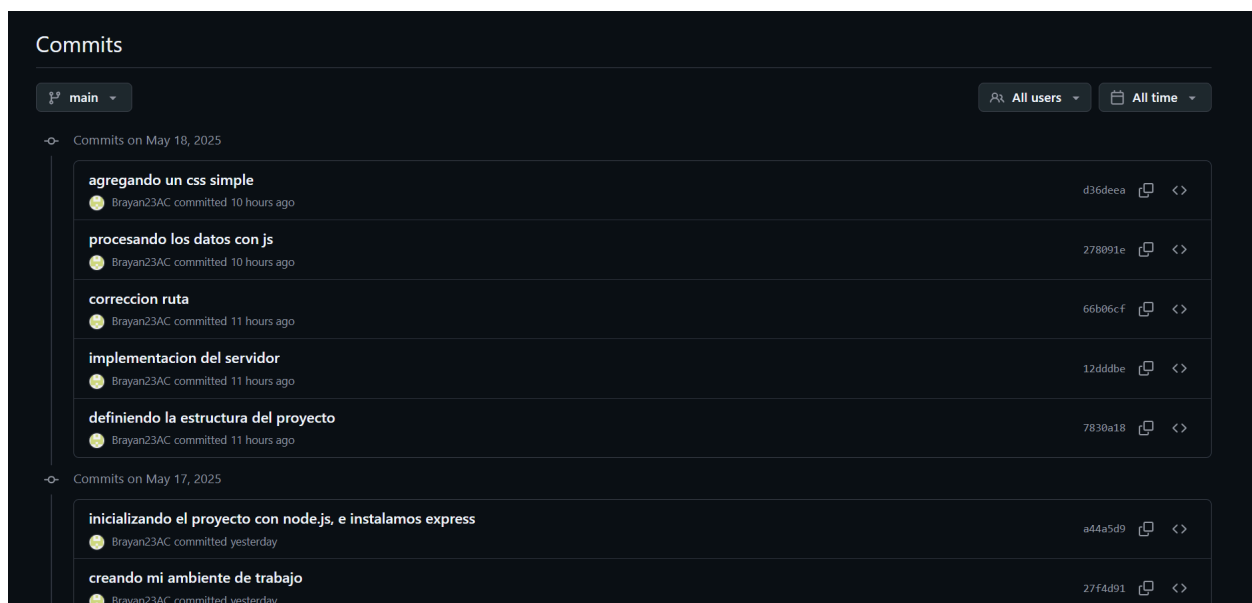
Para usar node.js usamos:

- npm init -y
- npm install express sqlite3

LINK GITHUB

<https://github.com/BCarlosAC/Programacion-Web-2-Teoria.git>

COMMITTS



Commits

main

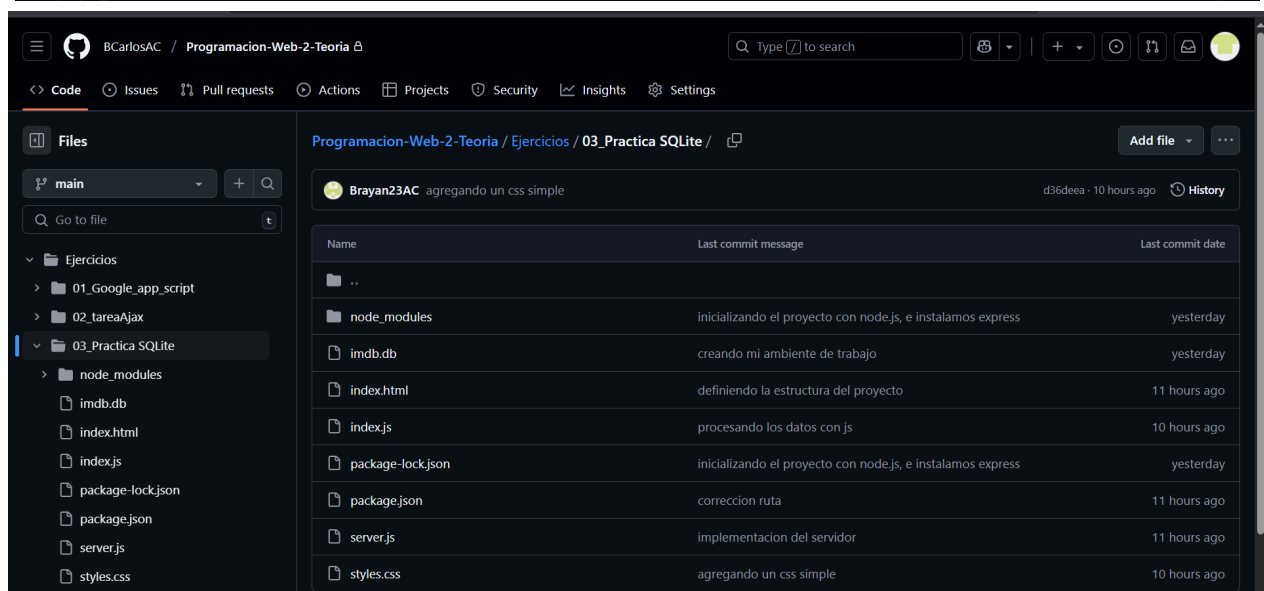
All users All time

Commits on May 18, 2025

- agregando un css simple d36deea 10 hours ago
- procesando los datos con js 278091e 10 hours ago
- correccion ruta 66b06cf 11 hours ago
- implementacion del servidor 12dddbe 11 hours ago
- definiendo la estructura del proyecto 7830a18 11 hours ago

Commits on May 17, 2025

- inicializando el proyecto con node.js, e instalamos express a44a5d9 yesterday
- creando mi ambiente de trabajo 27f4d91 yesterday



BCarlosAC / Programacion-Web-2-Teoria

Type to search

Code Issues Pull requests Actions Projects Security Insights Settings

Files

main

Go to file

- Ejercicios
 - 01_Google_app_script
 - 02_tareaAjax
 - 03_Practica SQLite
 - node_modules
 - imdb.db
 - index.html
 - index.js
 - package-lock.json
 - package.json
 - server.js
 - styles.css

Programacion-Web-2-Teoria / Ejercicios / 03_Practica SQLite

Add file

Brayan23AC agregando un css simple d36deea · 10 hours ago History

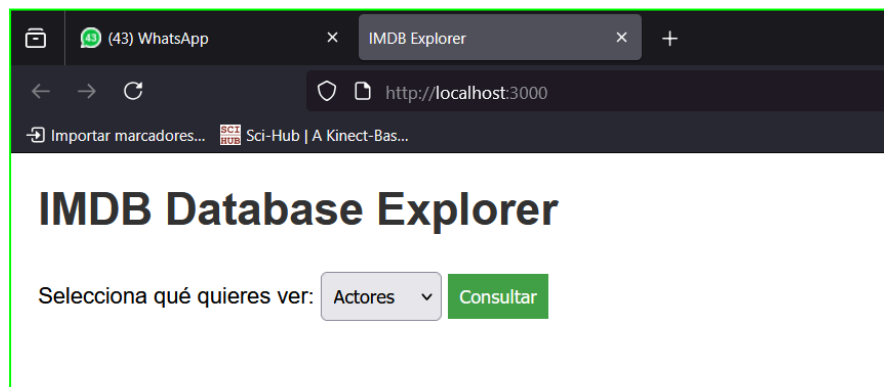
| Name | Last commit message | Last commit date |
|-------------------|---|------------------|
| .. | | |
| node_modules | inicializando el proyecto con node.js, e instalamos express | yesterday |
| imdb.db | creando mi ambiente de trabajo | yesterday |
| index.html | definiendo la estructura del proyecto | 11 hours ago |
| index.js | procesando los datos con js | 10 hours ago |
| package-lock.json | inicializando el proyecto con node.js, e instalamos express | yesterday |
| package.json | correccion ruta | 11 hours ago |
| server.js | implementacion del servidor | 11 hours ago |
| styles.css | agregando un css simple | 10 hours ago |

II. SOLUCIÓN DEL CUESTIONARIO

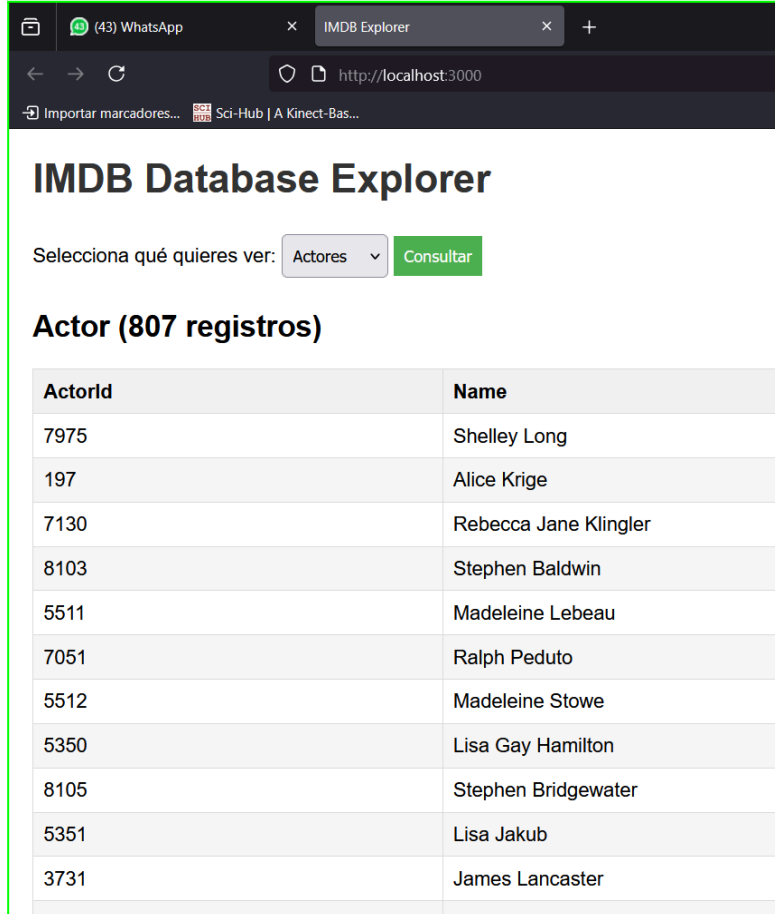
```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS D:\02_BRAYAN_UNIVERSIDAD\01_Programacion Web 2> cd ".\Programacion-Web-2-Teoria\Ejercicios\03_Practica SQLite\"
PS D:\02_BRAYAN_UNIVERSIDAD\01_Programacion Web 2\Programacion-Web-2-Teoria\Ejercicios\03_Practica SQLite> node server.js
Servidor corriendo en http://localhost:3000
Conectado a la base de datos imdb.db
PS D:\02_BRAYAN_UNIVERSIDAD\01_Programacion Web 2\Programacion-Web-2-Teoria\Ejercicios\03_Practica SQLite> node server.js
Servidor corriendo en http://localhost:3000
Conectado a la base de datos imdb.db
  
```

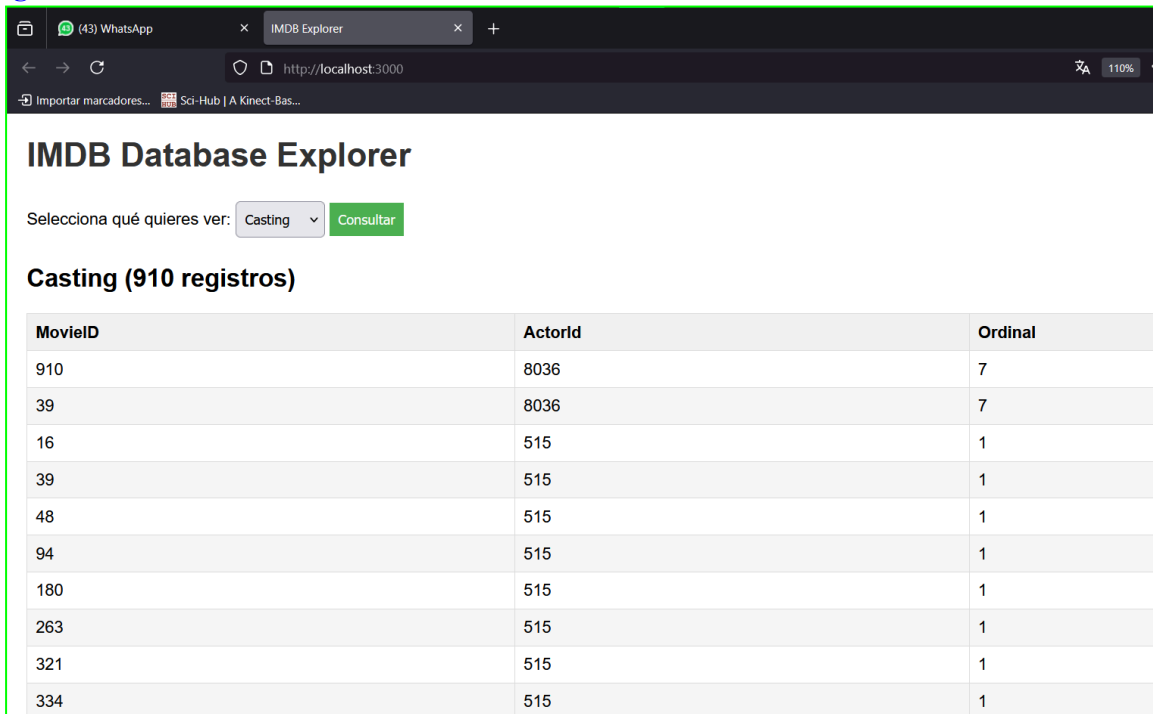


Actores



| ActorId | Name |
|---------|-----------------------|
| 7975 | Shelley Long |
| 197 | Alice Krige |
| 7130 | Rebecca Jane Klingler |
| 8103 | Stephen Baldwin |
| 5511 | Madeleine Lebeau |
| 7051 | Ralph Peduto |
| 5512 | Madeleine Stowe |
| 5350 | Lisa Gay Hamilton |
| 8105 | Stephen Bridgewater |
| 5351 | Lisa Jakub |
| 3731 | James Lancaster |
| 5514 | Madeleine Stowe |

Casting



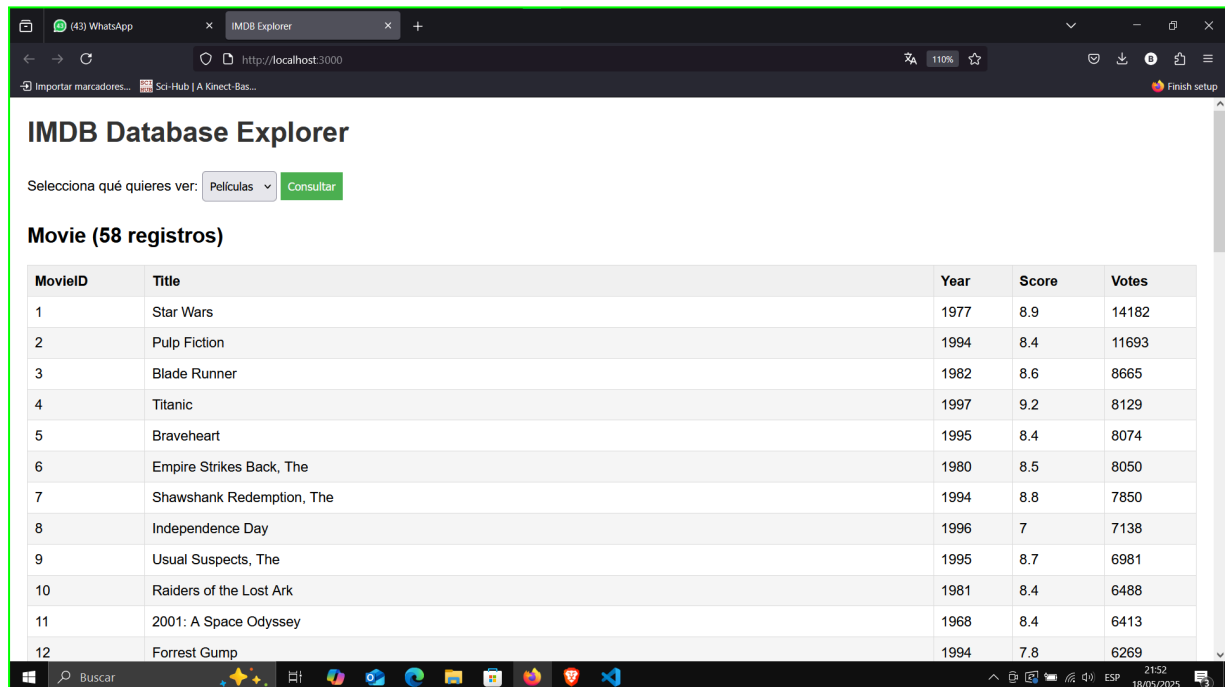
IMDB Database Explorer

Selecciona qué quieres ver: Casting Consultar

Casting (910 registros)

| MovieID | ActorId | Ordinal |
|---------|---------|---------|
| 910 | 8036 | 7 |
| 39 | 8036 | 7 |
| 16 | 515 | 1 |
| 39 | 515 | 1 |
| 48 | 515 | 1 |
| 94 | 515 | 1 |
| 180 | 515 | 1 |
| 263 | 515 | 1 |
| 321 | 515 | 1 |
| 334 | 515 | 1 |

Películas



IMDB Database Explorer

Selecciona qué quieres ver: Películas Consultar

Movie (58 registros)

| MovieID | Title | Year | Score | Votes |
|---------|---------------------------|------|-------|-------|
| 1 | Star Wars | 1977 | 8.9 | 14182 |
| 2 | Pulp Fiction | 1994 | 8.4 | 11693 |
| 3 | Blade Runner | 1982 | 8.6 | 8665 |
| 4 | Titanic | 1997 | 9.2 | 8129 |
| 5 | Braveheart | 1995 | 8.4 | 8074 |
| 6 | Empire Strikes Back, The | 1980 | 8.5 | 8050 |
| 7 | Shawshank Redemption, The | 1994 | 8.8 | 7850 |
| 8 | Independence Day | 1996 | 7 | 7138 |
| 9 | Usual Suspects, The | 1995 | 8.7 | 6981 |
| 10 | Raiders of the Lost Ark | 1981 | 8.4 | 6488 |
| 11 | 2001: A Space Odyssey | 1968 | 8.4 | 6413 |
| 12 | Forrest Gump | 1994 | 7.8 | 6269 |

III. CONCLUSIONES

Usar esta tecnología permitió crear una aplicación web sencilla y funcional que se conecta a una base de datos SQLite, consulta datos dinámicamente desde el frontend con JavaScript y Express en el backend, y muestra los resultados de forma clara al usuario, todo sin necesidad de recargar la página.

REFERENCIAS Y BIBLIOGRAFÍA

- <https://www.sqlite.org/>
- <https://developer.mozilla.org/es/docs/Web/JavaScript>
- <https://www.w3schools.com/js/DEFAULT.asp>