Laboratorio Práctico: Sistema de Gestión de Biblioteca con Node.js y Express.js

1 Objetivo

Crear un sitema Backend básico para gestionar libros en una biblioteca, implementando todos los métodos HTTP principales (GET, POST, PUT, PATCH, DELETE).

2 Requisitos Previos

- Node.js instalado en tu sistema
- Conocimientos básicos de JavaScript
- Un editor de código (como Visual Studio Code)
- Postman o una herramienta similar

3 Paso 1: Configuración del Proyecto

1. Crea un nuevo directorio para tu proyecto:

```
mkdir biblioteca-backend
cd biblioteca-backend
```

2. Inicializa un nuevo proyecto Node.js:

```
npm init -y
```

3. Instala las dependencias necesarias:

```
npm install express body-parser uuid
```

4. Crea un archivo `app.js` en el directorio raíz del proyecto.

4 Paso 2: Implementación Básica

En tu archivo `app.js`, implementa la estructura básica de la aplicación:

```
const express = require('express');
const bodyParser = require('body-parser');
const uuid = require('uuid');

const app = express();
app.use(bodyParser.json());

// Simular una base de datos con un array
let libros = [];

// Implementar rutas aquí

const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Servidor corriendo en http://localhost:${PORT}`);
});
```

5 Paso 3: Implementar Rutas para Cada Método HTTP

5.1 GET - Obtener todos los libros

```
app.get('/libros', (req, res) => {
    res.json(libros);
});
```

5.2 GET - Obtener un libro específico

```
app.get('/libros/:id', (req, res) => {
    const libro = libros.find(l => l.id === req.params.id);
    if (libro) {
        res.json(libro);
    } else {
        res.status(404).send('Libro no encontrado');
    }
});
```

5.3 POST - Agregar un nuevo libro

```
app.post('/libros', (req, res) => {
    const nuevoLibro = {
        id: uuid.v4(),
        titulo: req.body.titulo,
        autor: req.body.autor,
        anioPublicacion: req.body.anioPublicacion
    };
```

```
    libros.push(nuevoLibro);
    res.status(201).json(nuevoLibro);
});
```

5.4 PUT - Actualizar un libro existente

```
app.put('/libros/:id', (req, res) => {
    const index = libros.findIndex(l => l.id === req.params.id);
    if (index >= 0) {
        libros[index] = { ...libros[index], ...req.body, id: req.
    params.id };
        res.json(libros[index]);
    } else {
        res.status(404).send('Libro no encontrado');
    }
});
```

5.5 PATCH - Actualizar parcialmente un libro

```
app.patch('/libros/:id', (req, res) => {
    const libro = libros.find(l => l.id === req.params.id);
    if (libro) {
        Object.assign(libro, req.body);
        res.json(libro);
    } else {
        res.status(404).send('Libro no encontrado');
    }
});
```

5.6 DELETE - Eliminar un libro

```
app.delete('/libros/:id', (req, res) => {
    const index = libros.findIndex(l => l.id === req.params.id);
    if (index >= 0) {
        libros.splice(index, 1);
        res.status(204).send();
    } else {
        res.status(404).send('Libro no encontrado');
    }
});
```

6 Paso 4: Pruebas

1. Inicia tu servidor:

   ```
   node app.js
   ```

2. Usa Postman o una herramienta similar para probar cada endpoint:

- GET `http://localhost:3000/libros`
- GET `http://localhost:3000/libros/{id}`
- POST `http://localhost:3000/libros`
  Body: { "titulo": "1984", "autor": "George Orwell", "anioPublicacion": 1949 }
- PUT `http://localhost:3000/libros/{id}`
  Body: { "titulo": "1984", "autor": "George Orwell", "anioPublicacion": 1948 }
- PATCH `http://localhost:3000/libros/{id}`
  Body: { "anioPublicacion": 1949 }
- DELETE `http://localhost:3000/libros/{id}`

7 Paso 5: Mejoras y Desafíos Adicionales

1. Implementa validación de datos en las rutas POST y PUT.
2. Agrega manejo de errores más robusto.
3. Implementa paginación en la ruta GET que obtiene todos los libros.
4. Agrega funcionalidad de búsqueda, permitiendo filtrar libros por título o autor.
5. Implementa autenticación básica para proteger las rutas de modificación (POST, PUT, PATCH, DELETE).

8 Conclusión

Este laboratorio práctico te ha guiado a través de la creación de un Backend básico para un sistema de gestión de biblioteca utilizando Node.js y Express.js. Has implementado los cinco métodos HTTP principales (GET, POST, PUT, PATCH, DELETE) y has aprendido cómo manejar diferentes tipos de solicitudes y respuestas.

Recuerda que este es solo el comienzo. Un Backend robusto en producción requeriría consideraciones adicionales como:

- Implementación de una base de datos real (por ejemplo, MongoDB o PostgreSQL)
- Manejo de errores más sofisticado
- Validación y sanitización de entrada de datos
- Autenticación y autorización
- Logging y monitoreo

- Implementación y documentación de una API RestFul robusta
- Pruebas unitarias e integración