

# MÓDULO V:ARQUITECTURA RESTful

**Docente:** José Rosalío Serrano Carvajal

## **Integrantes - grupo 1:**

Mario Augusto Lúe Morales

Freddy Alberto Benitez Gomez

Wilber Denilson Lopez Perez

Felix Gerardo Granadino Rauda

Sábado, 31 de agosto de 2024.

---

## **Prueba final Modulo 5**

Presentar una API que cumpla con las características de RESTful de su preferencia ya sea que esté entre sus proyectos anteriores o que se encuentre en el mercado, de eso definir:

- Su funcionalidad
- Documentación ejecución de pruebas y despliegue.
- Además, colocar el enlace del repositorio de GitHub donde se aloja el proyecto.

## **SkyBook API**

La API SkyBook está diseñada para gestionar libros en una biblioteca digital, proporcionando funcionalidades esenciales que permiten a los desarrolladores realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre diversa cantidad de libros. Además de estas operaciones básicas, la API también permite consultar el estado del servidor, garantizando que se pueda verificar su correcto funcionamiento en cualquier momento.

## **Requisitos Previos al uso de la api**

Antes de utilizar la API SkyBook, es esencial que el entorno esté configurado correctamente. Esto incluye tener "Node.js" instalado en el sistema, ya que la API se ejecuta sobre este entorno de JavaScript en el servidor. Además, SkyBook utiliza "MongoDB" como base de datos para almacenar y gestionar los datos de los libros, lo que significa que necesitas tener una instancia de MongoDB en funcionamiento y correctamente configurada. Esto asegura que la API pueda interactuar con la base de datos para realizar operaciones como la creación de nuevos libros o la actualización de los existentes.

## Rutas de la API

La API SkyBook ofrece varias rutas para interactuar con los libros en la biblioteca:

- 1. Obtener Todos los Libros (`GET /books`):** Esta ruta permite recuperar una lista de todos los libros “disponibles” en la base de datos. Puedes filtrar los resultados por género utilizando el parámetro de consulta `genre`.
- 2. Obtener un Libro por ID (`GET /books/:id`):** Permite recuperar un libro específico utilizando su ID único.
- 3. Crear un Nuevo Libro (`POST /books`):** Esta ruta permite añadir un nuevo libro a la base de datos. Es necesario proporcionar detalles del libro en el cuerpo de la solicitud en formato JSON.
- 4. Actualizar un Libro por ID (`PUT /books/:id`):** Utilizada para modificar la información de un libro existente mediante su ID. Solo es necesario enviar los campos que deseas actualizar.
- 5. Eliminar un Libro por ID (`DELETE /books/:id`):** Permite eliminar un libro de la base de datos utilizando su ID.
- 6. Obtener Estado de la API (`GET /status`):** Esta ruta devuelve el estado actual del servidor, asegurando que la API esté funcionando correctamente.

## Middleware

SkyBook utiliza varios middlewares para mejorar la seguridad de cierta manera y el rendimiento de la API:

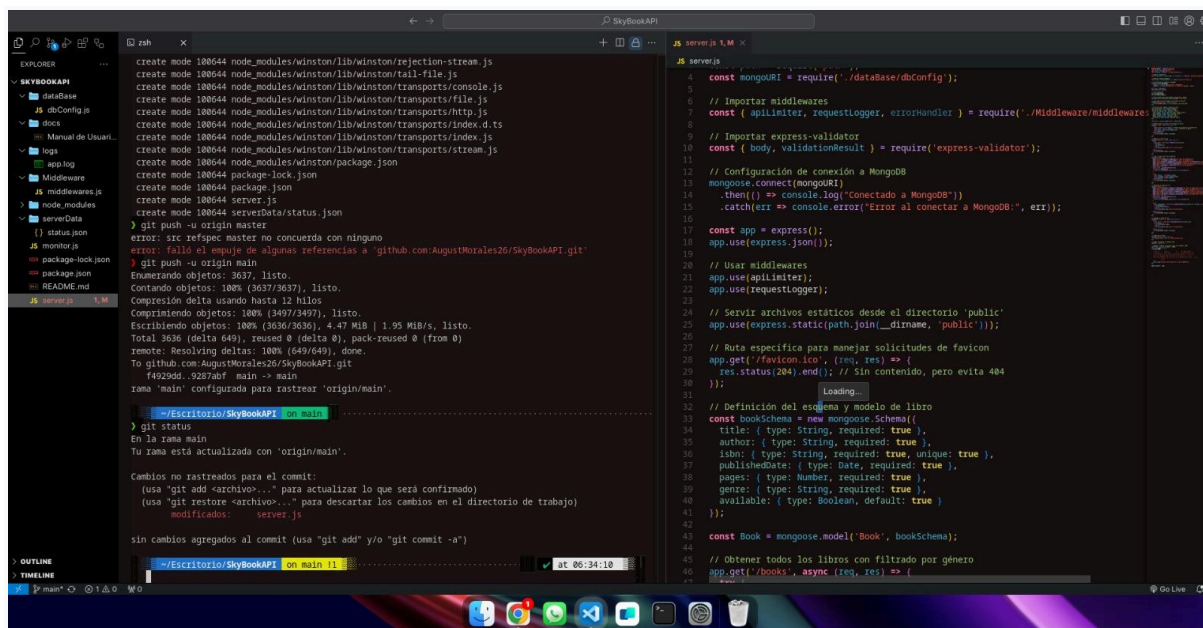
- **apiLimiter:** Limita el número de solicitudes que una sola IP puede hacer en un tiempo determinado, ayudando a prevenir abusos como ataques DDoS.
- **requestLogger:** Registra todas las solicitudes que llegan al servidor, lo que es útil para auditorías y depuración.
- **errorHandler:** Captura y maneja errores de manera centralizada, proporcionando respuestas consistentes y evitando caídas inesperadas del servidor.

## Monitor de Estado

Además, SkyBook incluye un monitor en segundo plano que registra el estado del servidor cada 10 minutos, de esta manera puede estar guardando esta información en un archivo JSON. Este monitor es para mantener un seguimiento continuo del rendimiento del servidor.

En resumen, la API SkyBook conlleva un conjunto completo de herramientas para gestionar eficientemente esta biblioteca digital, con un enfoque en lo que viene siendo la facilidad de uso, la seguridad y la escalabilidad.

## Capturas efectuadas de funcionamiento de la api SkyBook



```
create mode 100644 node_modules/winston/lib/winston/rejection-stream.js
create mode 100644 node_modules/winston/lib/winston/tail-file.js
create mode 100644 node_modules/winston/lib/winston/transports/console.js
create mode 100644 node_modules/winston/lib/winston/transports/file.js
create mode 100644 node_modules/winston/lib/winston/transports/http.js
create mode 100644 node_modules/winston/lib/winston/transports/index.d.ts
create mode 100644 node_modules/winston/lib/winston/transports/index.js
create mode 100644 node_modules/winston/lib/winston/transports/stream.js
create mode 100644 node_modules/winston/package.json
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 server.js
create mode 100644 serverData/status.json
> git push -u origin master
error: src refspec master no concuerda con ninguno
error: falló el empuje de algunas referencias a 'github.com:AugustMorales26/SkyBookAPI.git'
> git push -u origin main
Enumerando objetos: 3637, listo.
Contando objetos: 100% (3637/3637), listo.
Comprimiendo delta usando hasta 12 hilos.
Comprimiendo objetos: 100% (3497/3497), listo.
Escribiendo objetos: 100% (3636/3636), 4.47 MiB | 1.95 MiB/s, listo.
Total 3638 (delta 649), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (649/649), done.
To github.com:AugustMorales26/SkyBookAPI.git
f4929dd..9287abf main -> main
rama 'main' configurada para rastrear 'origin/main'.

~/Escritorio/SkyBookAPI on main
> git status
En la rama main
Tu rama está actualizada con 'origin/main'.

Cambios no rastreados para el commit:
(usa 'git add <archivo>...' para actualizar lo que será confirmado)
(usa 'git restore <archivo>...' para descartar los cambios en el directorio de trabajo)
modificados:   server.js

sin cambios agregados al commit (usa 'git add' y/o 'git commit -a')

~/Escritorio/SkyBookAPI on main | 11:06:24 AM

server.js
1  const mongoose = require('mongoose');
2
3  // Importar middlewares
4  const { apiLimiter, requestLogger, errorHandler } = require('./Middleware/middlewares');
5
6  // Importar express-validator
7  const { body, validationResult } = require('express-validator');
8
9  // Configuración de conexión a MongoDB
10 mongoose.connect(mongooseURI)
11   .then(() => console.log('Conectado a MongoDB'))
12   .catch(err => console.error('Error al conectar a MongoDB:', err));
13
14 const app = express();
15 app.use(express.json());
16
17 // Usar middlewares
18 app.use(apiLimiter);
19 app.use(requestLogger);
20
21 // Servir archivos estáticos desde el directorio 'public'
22 app.use(express.static(path.join(__dirname, 'public')));
23
24 // Ruta específica para manejar solicitudes de favicon
25 app.get('/favicon.ico', (req, res) => {
26   res.status(204).end(); // Sin contenido, pero evita 404
27 });
28
29 // Definición del esquema y modelo de libro
30 const bookSchema = new mongoose.Schema({
31   title: { type: String, required: true },
32   author: { type: String, required: true },
33   isbn: { type: String, required: true, unique: true },
34   publishedDate: { type: Date, required: true },
35   pages: { type: Number, required: true },
36   genre: { type: String, required: true },
37   available: { type: Boolean, default: true }
38 });
39
40 const Book = mongoose.model('Book', bookSchema);
41
42 // Obtener todos los libros con filtrado por género
43 app.get('/books', async (req, res) => {
```



The screenshot displays the Visual Studio Code editor interface for a project named 'SkyBookAPI'. The left sidebar shows the Explorer view with the following files and folders:

- SKYBOOKAPI
  - database
  - dbConfig.js
  - docs
  - Manual de Usuario de...
  - logs
  - applog
  - Middleware
  - middlewares.js
  - model\_modules
  - serverData.js
  - status.json
  - monitor.js
  - package-lock.json
  - package.json
  - README.md
  - server.js

The main editor area shows the 'server.js' file, which contains the following code:

```
const express = require('express');
const mongoose = require('mongoose');
const path = require('path');
const mongoURI = require('./database/dbConfig');

// Importar middlewares
const { apiLimitier, requestLogger, errorHandler } = require('...');

// Importar express-validator
const { body, validationResult } = require('express-validator');

// Configuración de conexión a MongoDB
mongoose.connect(mongoURI)
  .then(() => console.log('Conectado a MongoDB'))
  .catch(err => console.error('Error al conectar a MongoDB'));
```

The bottom terminal window shows the command 'node server.js' being executed, and the console output displays a series of log messages indicating the server is running and handling requests:

```

Servidor ejecutándose en el puerto 4000
{"level":"info","message":"[2024-09-01T00:42:50.521Z] GET /status - Status: 200,\"method\":\"GET\",\"status\":\"200\",\"timestamp\":\"2024-09-01T00:42:50.541Z\",\"url\":\"/status\"}
Estado guardado en /home/augustodev/Escritorio/SkyBookAPI/serverData/status.json
Conectado a MongoDB
{"level":"info","message":"[2024-09-01T00:43:03.172Z] GET /Books - Status: 304,\"method\":\"GET\",\"status\":\"304\",\"timestamp\":\"2024-09-01T00:43:03.332Z\",\"url\":\"/Books\"}
{"level":"info","message":"[2024-09-01T00:43:03.402Z] GET /favicon.ico - Status: 204,\"method\":\"GET\",\"status\":\"204\",\"timestamp\":\"2024-09-01T00:43:03.405Z\",\"url\":\"/favicon.ico\"}
{"level":"info","message":"[2024-09-01T00:43:22.602Z] GET /Books/66d246b4378c1d91540928f - Status: 304,\"method\":\"GET\",\"status\":\"304\",\"timestamp\":\"2024-09-01T00:43:22.721Z\",\"url\":\"/Books/66d246b4378c1d91540928f\"}
{"level":"info","message":"[2024-09-01T00:43:22.773Z] GET /favicon.ico - Status: 204,\"method\":\"GET\",\"status\":\"204\",\"timestamp\":\"2024-09-01T00:43:22.775Z\",\"url\":\"/favicon.ico\"}

```

The screenshot shows a code editor interface. On the left, the Explorer sidebar displays a project structure for 'SKYBOOKAPI'. The 'serverData' folder is expanded, and 'status.json' is selected. The main editor area shows the content of 'status.json', which is a JSON object with the following structure:

```
1 {
2   "timestamp": "2024-09-01T00:42:50.570Z",
3   "serverStatus": {
4     "status": "API is running smoothly"
5   },
6   "systemInfo": {
7     "hostname": "fedora",
8     "platform": "linux",
9     "arch": "x64",
10    "totalMemory": 16536137728,
11    "freeMemory": 12412825600,
12    "uptime": 3972.8
13  }
14 }
```

Nota: en este status como se observa se ejerce una actualización cada 10 min

```
app.log M X status.json M
logs > app.log
1 ["level":"info","message":"[2024-09-01T23:47:07.576Z] GET /Books - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T23:47:08.147Z","url":"/Books"}
2 ["level":"info","message":"[2024-09-01T00:04:22.187Z] GET /status - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T00:04:22.202Z","url":"/status"}
3 ["level":"info","message":"[2024-09-01T00:05:52.135Z] GET /status - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T00:05:52.151Z","url":"/status"}
4 ["level":"info","message":"[2024-09-01T00:06:05.000Z] GET /Books - Status: 204","method":"GET","status":204,"timestamp":"2024-09-01T00:06:05.149Z","url":"/Books"}
5 ["level":"info","message":"[2024-09-01T00:07:39.268Z] GET /status - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T00:07:39.282Z","url":"/status"}
6 ["level":"info","message":"[2024-09-01T00:11:43.290Z] GET /status - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T00:11:43.306Z","url":"/status"}
7 ["level":"info","message":"[2024-09-01T00:13:08.616Z] GET /status - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T00:13:08.630Z","url":"/status"}
8 ["level":"info","message":"[2024-09-01T00:15:33.283Z] GET /status - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T00:15:33.298Z","url":"/status"}
9 ["level":"info","message":"[2024-09-01T00:19:41.702Z] GET /status - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T00:19:41.721Z","url":"/status"}
10 ["level":"info","message":"[2024-09-01T00:20:26.100Z] GET /Books/66d2648d4370c1d91540928f - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T00:20:26.255Z","url":"/Books/66d2648d4370c1d91540928f"}
11 ["level":"info","message":"[2024-09-01T00:20:26.308Z] GET /favicon.ico - Status: 204","method":"GET","status":204,"timestamp":"2024-09-01T00:20:26.312Z","url":"/favicon.ico"}
12 ["level":"info","message":"[2024-09-01T00:42:50.511Z] GET /status - Status: 200","method":"GET","status":200,"timestamp":"2024-09-01T00:42:50.541Z","url":"/status"}
13 ["level":"info","message":"[2024-09-01T00:43:03.172Z] GET /Books - Status: 304","method":"GET","status":304,"timestamp":"2024-09-01T00:43:03.332Z","url":"/Books"}
14 ["level":"info","message":"[2024-09-01T00:43:03.402Z] GET /favicon.ico - Status: 204","method":"GET","status":204,"timestamp":"2024-09-01T00:43:03.405Z","url":"/favicon.ico"}
15 ["level":"info","message":"[2024-09-01T00:43:22.600Z] GET /Books/66d2648d4370c1d91540928f - Status: 304","method":"GET","status":304,"timestamp":"2024-09-01T00:43:22.721Z","url":"/Books/66d2648d4370c1d91540928f"}
16 ["level":"info","message":"[2024-09-01T00:43:22.773Z] GET /favicon.ico - Status: 204","method":"GET","status":204,"timestamp":"2024-09-01T00:43:22.775Z","url":"/favicon.ico"}
17
```