Mini Proyecto Api-First

Cristian Harol Romero Arias

Mini proyecto de Integración Continua

Profesora

Diana Marcela Toquica Rodríguez

Ingeniera de Sistemas

Materia

Ingeniería WEB 1

Universidad Manuela Beltrán

Ingeniería de Software Virtual

2025

Contenido

1. Resumen

Este documento describe el desarrollo de una Api Rest Full utilizando Node.js y Express.js siguiendo el enfoque API firts, se documenta el proceso desde la instalación hasta la implementación

1.1 introducción

El desarrollo de la API es fundamental en la arquitectura moderna de software, este proyecto implementa una API **monolítica** con express.js, estructurada de forma sencilla y documentada con OpenAPI.

2. Objetivos

Primero : Desarrollar una Api funcional sencilla que gestione de usuarios y productos

Segundo: Aplicar buenas prácticas de documentación

Tercero: Implementar pruebas automatizadas con Jest

3. Metodología

Primero se procede a la instalación de Dependencias como Nodejs,npm,Express.js posterior a la configuración del proyecto en estructura de carpetas utilizamos el IDE Visual estudio Code donde estructuramos carpetas y archivos principales.

3.1 Estructura Carpetas

Esta es una imagen de la estructura:

```
APIFIRST/
├── src/
│   ├── app.js          # Lógica principal de la API
│   ├── index.js        # Punto de entrada del servidor
│   ├── server.js       # Inicio del servidor
├── test.js             # Pruebas automatizadas
├── openapi.yaml        # Especificación OpenAPI
├── package.json
└── README.md
└── Arquitectura.md
```

Figura 1.Estructura de Carpetas

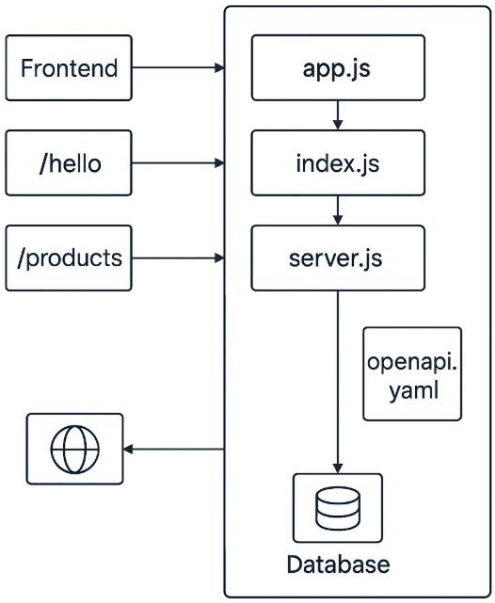


Figura 2. Diagrama de Arquitectura

3.2 Lógica Principal de la APP.JS

Se estructura así:

1.Importacion de Dependencias

2.inicializacion de la aplicación

3.Seguridad HTTP

4.Limitar el numero de peticiones por IP

5.Configuracion de OpenAPI

6. Implementación de la documentación den Swagger UI desde openapi.yaml

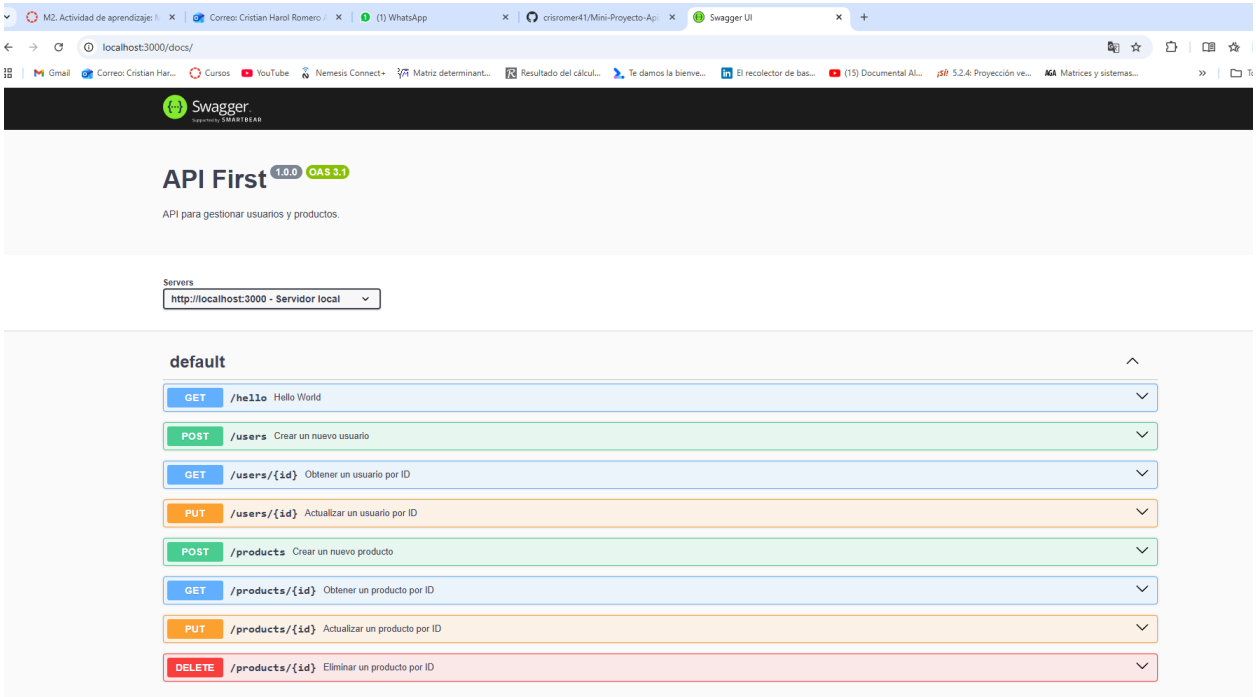


Figura 3. Documentación con Swagger

7. Endpoints principales

Usuarios

POST /users→ Crear usuario

GET /users/{id}→ Obtener usuario por ID

PUT /users/{id}→ Actualizar usuario por ID

Productos

POST /products→ Crear producto

GET /products/{id}→ Obtener producto por ID

PUT /products/{id}→ Actualizar producto por ID

DELETE /products/{id}→ Eliminar producto por ID

Otros

GET /hello→ Endpoint de prueba

8.Documentacion OpenApi

La documentación completa de la API está disponible en http://localhost:3000/docs en un repositorio de GitHub https://github.com/crisromer41/Mini-Proyecto-Api.git, generado con Openapi.yaml algunos ejemplos de uso como son el Post/users

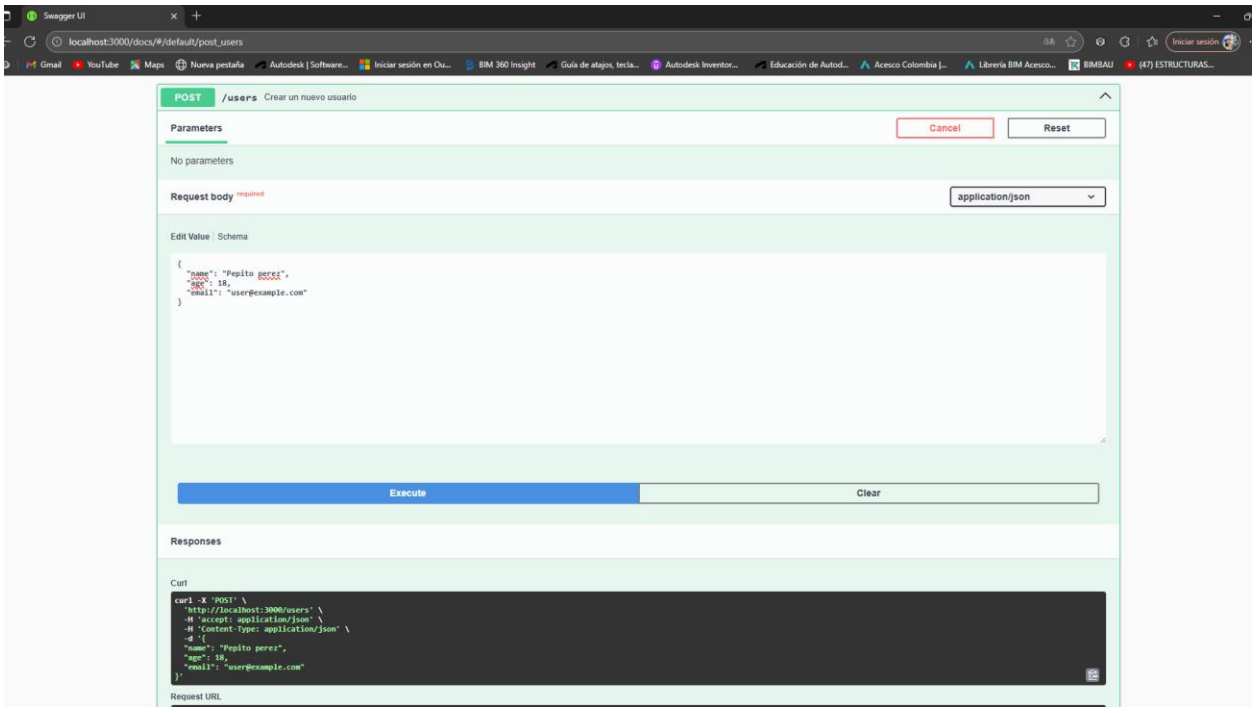


Figura 4. Validación POST

Se procede a actualizar usuario utilizando un PUT, gracias a la documentación interactiva con swagger puedo realizar un ejemplo de cómo corre a la aplicación al modificar un usuario

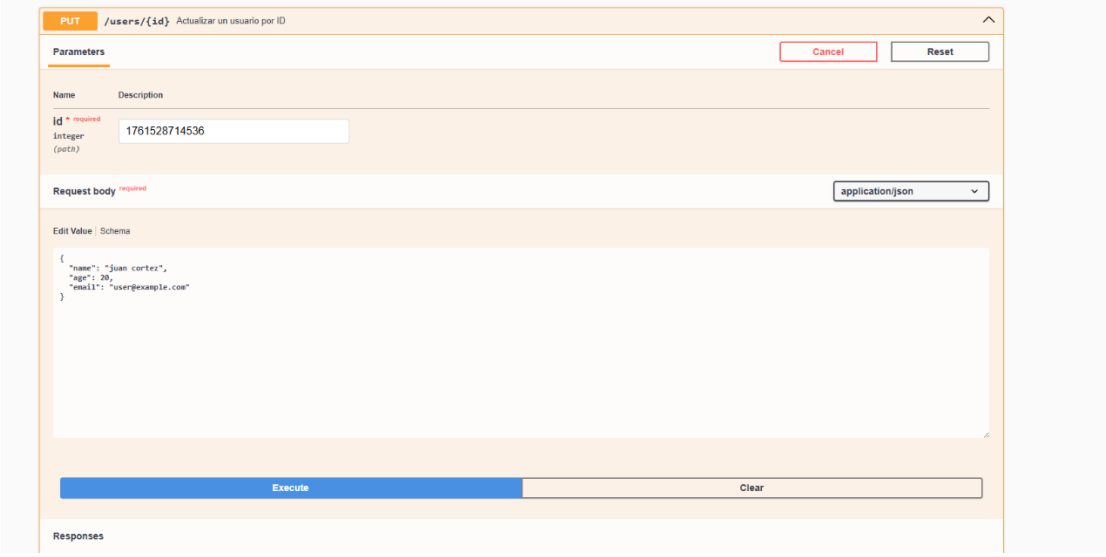
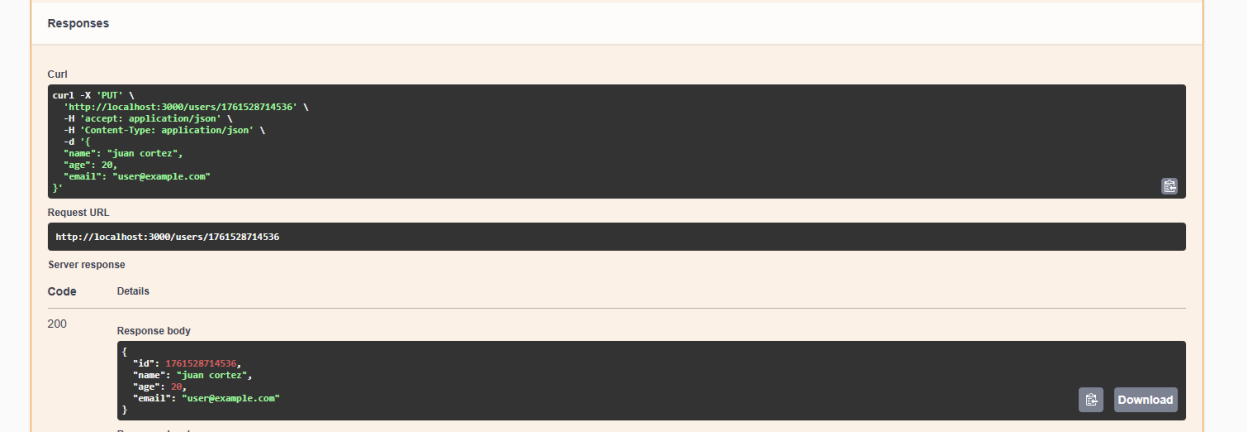


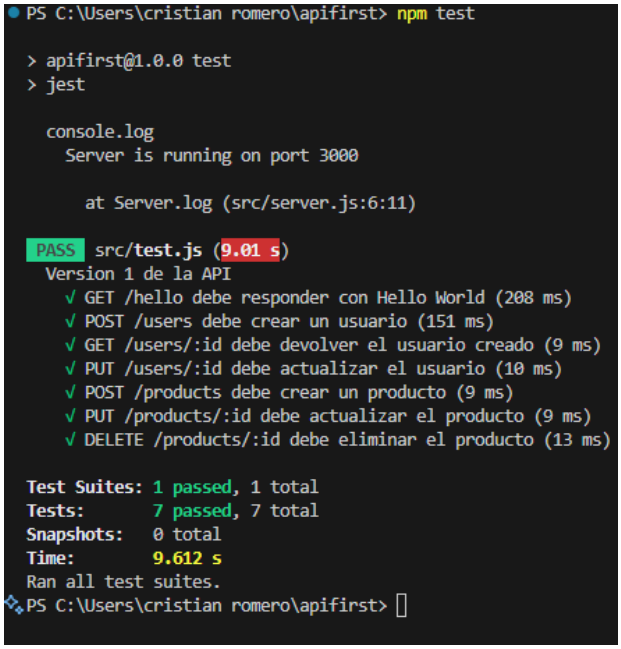
Figura 5. Validación PUT

Obtenemos una respuesta con código 200



9. Al ejecutar el comando npm test corriendo en jest, el framework de pruebas configurado en el proyecto, Se ejecutan pruebas automatizadas para validad la funcionalidad de los endpoints definidos en el api, el resultado fue exitoso, con 7 pruebas de 7 en total, lo que confirma que la

API cumple con los criterios de evaluación, endpoints funcionales, código limpio y documentado con la cobertura adecuada en el pipeline de pruebas.



4. Resultados

4.1 Estructura

La aplicación esta compuesta por los siguientes archivos principales:

index.js**:** Punto de entrada de la aplicación.
app.js: Contiene toda la lógica de la API, incluyendo definición de endpoints y middlewares.
server.js: Inicializa el servidor HTTP.
openapi.yaml: Define la especificación OpenAPI para la documentación.
test.js: Pruebas automatizadas con Jest y Supertest.

4.2 Flujo de ejecución

1. `index.js` importa `app.js` y lo pasa a `server.js`.
2. `server.js` inicia el servidor Express en el puerto definido.
3. `app.js` define los endpoints (`/hello`, `/users`, `/products`) y configura middlewares.
4. La documentación Swagger se genera automáticamente desde `openapi.yaml`.
5. Las pruebas se ejecutan con `npm test` usando Jest y Supertest.

4.2 Conclusiones

Esta versión de APIFIRTS no implementa controladores ni rutas separadas, toda la lógica esta centralizada en app.js, lo que simplifica el desarrollo inicial y facilita la compresión del fujo, esta estructura es adecuada para proyectos pequeños, prototipos rápidos, APIS con bajo nivel de complejidad, esto permite una implementación rápida y funcional de una API RESFUL, a medida que el proyecto crezca se recomienda la lógica en controladores y rutas para mejorar la mantenibilidad, la api permite realizar operaciones CRUD sobre usuarios y productos, con documentación en http://localhost:3000/docs .

5.Referencias

OpenAPI Initiative. (2023). OpenAPI Specification. https://www.openapis.org

Node.js Foundation. (2023). Node.js Documentation. https://nodejs.org

Express.js Team. (2023). Express.js Guide. https://expressjs.com

Jest. (2023). Jest Documentation. https://jestjs.io