Plan de mejoramiento

Juan José Buriticá Ruiz

Instructor: Luis Fernando

Ficha: 3223873

Documentación, diseño conceptual

Alcance y Definición de Recursos

Objetivo: Identificar y definir los recursos principales para la gestión de tareas.

Descripción del Recurso: El recurso principal es Tasks (Tareas), orientado a rastrear actividades en tres estados: todo, in-progress y done.

Atributos del Recurso:

id: Identificador único (UUID o numérico).

description: Detalle de la tarea.

status: Estado actual del recurso.

createdAt: Fecha de creación del registro.

updatedAt: Fecha de la última modificación.

1.2 Diseño de Endpoints y URIs

Siguiendo las buenas prácticas RESTful, se han definido las siguientes rutas utilizando sustantivos en plural y jerarquías claras:

Listar todas las tareas: GET /v1/tasks

Consultar una tarea: GET /v1/tasks/:id

Crear tarea: POST /v1/tasks

Actualización total: PUT /v1/tasks/:id

Actualización parcial: PATCH /v1/tasks/:id

Eliminación: DELETE /v1/tasks/:id

1.3 Estrategia de Versionado y Buenas Prácticas

Versionado: Se implementó el prefijo /v1/ en las URIs para permitir la evolución de la API sin afectar a los clientes actuales.

Idempotencia: Se garantiza que los métodos GET, PUT y DELETE sean idempotentes para asegurar la consistencia de los datos.

2. Implementación y Configuración (50%)

2.1 Configuración del Entorno

Tecnologías: Node.js y Express.

Dependencias: express, dotenv, nodemon.

Persistencia: Uso del módulo nativo fs para interactuar con un archivo JSON local.

2.2 Gestión de Peticiones y Respuestas

Manejo de datos: Uso de express.json() y express.urlencoded() para procesar el cuerpo de las peticiones (req.body).

Identificación dinámica: Uso de req.params para la captura de IDs y req.query para la aplicación de filtros de búsqueda.

Códigos de Estado HTTP:

200 OK: Petición exitosa.

201 Created: Creación exitosa de recurso.

204 No Content: Eliminación exitosa.

400 Bad Request: Error en la validación de datos enviados.

404 Not Found: Recurso no identificado.

500 Internal Server Error: Fallos generales del servidor.

2.3 Estructura de Versionamiento en GitHub

Se sigue una estrategia de ramas para el desarrollo organizado:

Ramas base: main (protegida) y develop (integración).

Commits realizados:

Commit 1: Scaffold inicial del proyecto.

Commit 2: Implementación de rutas GET.

Commit 3: Lógica de creación con POST.

Commit 4: Actualización con PUT y PATCH.

Commit 5: Eliminación con DELETE.

Commit 6: Validación de parámetros y estados HTTP.

3. Plan de Sustentación en Video (25%)

Guion estructurado para la presentación de resultados:

1. Introducción (30 seg): Presentación del aprendiz y recursos del caso.
2. Explicación de Código (2 min): Demostración de estructura (carpetas, controladores), rutas implementadas y manejo de validaciones.
3. Uso Operativo en Postman (2 min): Demostración del CRUD completo (GET, POST,

PUT/PATCH, DELETE) y visualización de cambios en el archivo JSON.

Razón por la cual no entregue los trabajos a tiempo:

Por medio de este documento, entrego mi plan de mejoramiento sobre la API Task Tracker. Quiero ser sincero y reconocer que no entregué los trabajos en las fechas que tocaba porque me faltó compromiso y no organicé bien mi tiempo en ese momento. Me dejé llevar por la desmotivación del momento y no le di la importancia que merecía a mi formación.

Sin embargo, he aprovechado esta oportunidad para reflexionar y corregir esa actitud. He trabajado a conciencia en este proyecto siguiendo todos los pasos de la guía para demostrar que sí tengo la capacidad de ser un desarrollador responsable y que realmente me interesa aprender y cumplir con mis metas académicas y en algún momento profesionales.