UNIVERSIDAD NUEVA ESPARTA

FACULTAD DE CIENCIAS DE LA INFORMÁTICA

ESCUELA DE COMPUTACIÓN

Complementaria II

PERIODO LECTIVO: PL-165

Asignación semana dos PROMPTS

Estudiante:

Muchacho Figueroa, Andrea Paola Del Valle

C.I: 31.131.098

Caracas, Enero, 2026

LINK DEL REPOSITORIO EN GITHUB:
https://github.com/andr31a/miune-docs-api

1. CONTEXTO DEL USO DE IA

- **Qué partes del proyecto usaron IA:** Se utilizó IA para la configuración inicial del proyecto, generación de la estructura de carpetas MVC y la creación de los archivos de configuración base para una API REST con Node.js.
- **Qué esperaban lograr:** Obtener una base sólida y estandarizada que cumpla con las mejores prácticas requeridas por la rúbrica, asegurando que las dependencias y scripts de desarrollo estén correctamente configurados antes de escribir la lógica de negocio.

Prompt 1: Configuración inicial

> "Actúa como un desarrollador senior de Node.js. Necesito configurar un proyecto de API REST desde cero para el recurso 'Documentos' de mi Módulo de Gestión Documental. Requisitos basados en mi rúbrica: 1. Node.js con Express v4.18. 2. Estructura de carpetas profesional MVC (models, controllers, routes, middleware, utils). 3. package.json con dependencias (express, dotenv, cors, joi) y devDependencies (nodemon), además de los scripts 'start' y 'dev'. 4. Un archivo .env.example. 5. Un archivo app.js básico configurado. Proporcióname el código y la estructura para iniciar mi proyecto."

Prompt 2: Implementación del Modelo

> "Necesito implementar un modelo de datos en memoria para el recurso 'Documento' de mi sistema de gestión. Especificaciones:
>
> 1. El recurso debe tener los campos: id (UUID), titulo (string), tipo (pdf/docx), peso (string), estado (borrador/publicado), createdAt y updatedAt.
> 2. Debe incluir un array con 3 registros de ejemplo (seed data).
> 3. Debe exportar funciones asíncronas (simulando promesa): getAll, getById, create, update, delete.
> 4. Usa el módulo nativo crypto de Node.js para generar los UUIDs.
> 5. Maneja errores básicos (ej: si no encuentra el ID al actualizar). Genera el código para el archivo src/models/documentoModel.js."

Prompt 3: Controllers y Rutas

> "Actúa como un desarrollador senior de Node.js. Ya tengo mi modelo documentoModel.js con las funciones CRUD en memoria. Necesito que implementes: 1. El archivo src/controllers/documentoController.js con las funciones: getAll, getById, create, update y deleteResource. Usa

async/await, try-catch, y retorna respuestas JSON consistentes con códigos de estado HTTP correctos (200, 201, 404). Pasa los errores al next(). 2. El archivo src/routes/documentoRoutes.js mapeando los endpoints RESTful estándar (GET, POST, PUT, DELETE) a las funciones del controlador. 3. Indícame cómo debo actualizar mi app.js para registrar estas rutas bajo el prefijo /api/documentos. El código debe ser modular y estar comentado."

Prompt 4: Validación con Joi

"Necesito implementar validación robusta para mi API de Documentos usando Joi. Datos a validar:

- titulo: string, requerido, mínimo 3 caracteres, máximo 150.
- tipo: string, requerido, solo puede ser pdf, docx o xlsx.
- peso: string, requerido.
- estado: string, requerido, solo puede ser 'borrador' o 'publicado'. Implementa el archivo src/middleware/validation.js con esquemas para creación (todo obligatorio) y actualización (opcionales), integrando mensajes de error claros."

Prompt 5: Manejo de Errores Global

"Necesito implementar un sistema profesional de manejo de errores para mi API Express.

1. Crea la clase AppError en src/utils/AppError.js extendiendo de Error con propiedades statusCode e isOperational.
2. Crea el middleware global en src/middleware/errorHandler.js que envíe respuestas con formato consistente y muestre el stack solo en desarrollo.
3. Indica cómo actualizar los controladores y rutas para usar este sistema."

5. REFLEXIÓN

¿Qué aprendieron sobre Node.js/Express? Aprendí que Express requiere una configuración meticulosa del orden de los middlewares. Descubrí por experiencia propia que el middleware de manejo de errores (errorHandler) debe ir obligatoriamente al final de todas las definiciones (app.use), o de lo contrario no capturará las excepciones. También comprendí la importancia de configurar dotenv en la primera línea de ejecución para evitar errores de variables undefined en el resto de la aplicación.

¿Qué aprendieron sobre trabajar con IA? La IA es excelente generando código base, pero puede pasar por alto detalles del entorno operativo. Por ejemplo, al

generar el archivo AppError.js, la IA no consideró que Windows no distingue entre mayúsculas y minúsculas, lo que generó conflictos de importación (AppError vs appError) que tuve que depurar manualmente. Aprendí que debo ser muy específica con la nomenclatura de archivos al pedir código.

¿Qué dudas técnicas siguen teniendo? Me gustaría profundizar en cómo conectar esta estructura de API con una base de datos real en la nube sin exponer las credenciales, y cómo manejar la subida de archivos físicos (imágenes/PDFs) ya que por ahora solo estamos validando sus metadatos en JSON.