

L U M B R E



Diseñar tus propios endpoints: una API para tu aplicación

Materia: Desarrollo de Software con IA

07/10/2026

Antes de empezar

Antes de pedir que la IA construya tu backend, diseña su API: la lista de operaciones que ofrecerá, cada una con verbo, ruta, qué recibe y qué devuelve. Ese catálogo es el plano del que ya hablaste en contratos, pero a escala de sistema. Aquí aprendes a dibujar tu API en una tabla que la IA implementará sin adivinar.

QUÉ APRENDERÁS

- Definir una tabla de endpoints (verbo, ruta, entrada, salida) y pedirla a la IA como contrato.

Tu app también es una API para su propio frontend

Consumir APIs ajenas (módulo anterior) y **diseñar la tuya** son la misma cara de dos monedas. Tu backend expone endpoints que tu propio frontend llama. Si diseñas esa lista con orden —verbos claros, rutas coherentes, entradas y salidas definidas—, la IA construye un sistema que se entiende a sí mismo.

El verbo importa

GET lee, POST crea, PUT cambia, DELETE borra: la convención que hace tu API predecible.

Verbo	Para qué se usa	Ejemplo
GET	Leer / consultar datos (no cambia nada)	GET /contactos
POST	Crear un recurso nuevo	POST /contactos
PUT	Actualizar / reemplazar uno existente	PUT /contactos/{id}
DELETE	Eliminar un recurso	DELETE /contactos/{id}

Diseña la API en una tabla

Esta tabla ES tu API. La pegas en el prompt y la IA no tiene que inventar ni una ruta.

Operación	Verbo y ruta	Entra	Sale
Listar	GET /contactos	filtro opcional por nombre	lista de contactos + total
Crear	POST /contactos	nombre, teléfono, email	contacto creado con su id
Ver uno	GET /contactos/{id}	—	el contacto o 404
Editar	PUT /contactos/{id}	campos a cambiar	contacto actualizado

Operación	Verbo y ruta	Entra	Sale
Borrar	DELETE /contactos/{id}	—	confirmación de borrado

Para reforzar verbos y JSON

Prompt de diseño

Del plano al código en dos pasos

Diseña e implementa la API de Contactos segun esta tabla (respeta verbos y rutas, no inventes otras):\n- GET /contactos?nombre= -> lista + total; solo los del usuario autenticado\n- POST /contactos {nombre*, telefono, email} -> 201 con id; 400 si falta nombre\n- GET /contactos/{id} -> contacto o 404\n- PUT /contactos/{id} {campos} -> actualizado o 404\n- DELETE /contactos/{id} -> 204 al borrar\n\nDevuelve primero la documentacion de la API (una tabla igual a esta con sus codigos de estado) y luego el codigo de cada endpoint.

Prueba

1. ¿Qué verbo y código de estado tienen más sentido para crear un contacto con éxito?

- ☐ GET /contactos que devuelva 404.
- ☐ POST /contactos que devuelva 201 (creado).
- ☐ DELETE /contactos que devuelva 200.
- ☐ PUT /contactos que devuelva 500.

2. Rutas como /obtenerContactos y /borrarContacto son tan válidas como GET /contactos y DELETE /contactos/{id}.

- ☐ Verdadero
- ☐ Falso

Detecta el diseño inconsistente

ACTIVIDAD
INTERACTIVA
DISPONIBLE EN
LUMBRE

**3. Revisas el diseño de API de una IA.
Señala la incoherencia que la volverá
difícil de usar:**



4. Escribe la tabla de endpoints de una entidad de tu proyecto (5 filas: listar, crear, ver, editar, borrar) con verbo, ruta, entra y sale. Pídele a la IA primero la documentación y después el código.

Si tu API es predecible, tu frontend se escribe casi solo.

Referencia curricular

Materia:	Desarrollo de Software con IA
----------	-------------------------------

Versión interactiva completa (con actividades, videos y autoevaluación):



<https://empresa.lumbre.lat/object/fsi-m09-0a02>

Material educativo generado desde Lumbre · lumbre.lat