MÓDULO 7. DESARROLLO AVANZADO DE BACKEND Y APIS

Tarea 1: Fundamentos del Desarrollo Backend con Node.js

Diego Matilla De La Cierva

Descripción del Desarrollo

El archivo app.js es el punto de entrada de toda nuestra API y su responsabilidad principal es orquestar el flujo de las peticiones HTTP, delegando en los controladores la lógica de negocio. En el arranque, se importa el módulo http de Node.js junto con el enrutador básico construido con url.parse, lo que nos permite trabajar sin frameworks externos y comprender al detalle cómo el servidor atiende cada solicitud. A continuación, se establecen las rutas principales (/rooms, /reservations) y sus métodos correspondientes, para leer, crear o eliminar datos; cualquier petición que no coincida con estas rutas recibe una respuesta 404 con un mensaje JSON uniforme.

listRooms(res)

La función listRooms se encarga de proporcionar al cliente la lista completa de salas disponibles para reserva. Para ello, utiliza el helper readJsonFile sobre el archivo rooms.json, que contiene un array de objetos con la información de cada sala (nombre, capacidad e identificador). Una vez obtenidos los datos, la función escribe la cabecera HTTP con código 200 y tipo de contenido JSON, y envuelve el array en JSON.stringify para enviarlo al cuerpo de la respuesta. Esta aproximación simplificada abstrae la persistencia en disco tras una interfaz limpia: quien llame a /rooms recibe directamente la colección actualizada de salas sin preocuparse por los detalles de la lectura de archivos.

listReservations(res)

De manera muy similar a listRooms, listReservations está diseñada para devolver todas las reservas activas que se encuentran almacenadas en reservations.json. Gracias al helper de lectura de archivos, no se realizan comprobaciones adicionales; se asume que el archivo siempre responde con un array (vacío o no). Tras cargar las reservas, la función monta un encabezado con estado 200 y envía la serialización JSON de la lista. Esto permite a cualquier cliente consultar el estado global de las reservas sin necesidad de manejar parámetros ni lógica de filtrado en el servidor.

createReservation(req, res)

La función createReservation es la encargada de procesar las peticiones POST /reservations que intentan generar una nueva reserva. Inicialmente, recopila el cuerpo de la petición en un buffer de texto, y una vez recibidos todos los fragmentos realiza el parseo a un objeto JavaScript. A continuación, extrae los campos obligatorios (roomId, timeSlot y userName) y valida su existencia, devolviendo un 400 si falta alguno. Para garantizar la coherencia, carga primero la definición de salas y luego las reservas ya existentes; si la sala indicada no existe, devuelve un 404. A continuación, comprueba que no haya solapamientos de franja horaria para diferentes usuarios y que la capacidad máxima de la sala no se exceda. Solo si se cumplen todas las condiciones, crea un objeto de reserva con un identificador único (uuid), lo añade al

array de reservas y persiste los cambios con writeJsonFile. Finalmente, responde con un estado 201 y el objeto de reserva recién creado.

deleteReservation(req, res, reservationId)

La función deleteReservation permite cancelar una reserva mediante su identificador único. Tras leer todas las reservas con el helper correspondiente, busca el índice del elemento cuyo id coincide con reservationId. Si no lo encuentra, responde con un 404 y un mensaje claro. En caso contrario, elimina el elemento del array, vuelve a escribir el JSON actualizado en disco y envía un 204 No Content, indicando que la operación fue exitosa sin necesidad de devolver un cuerpo. Esta lógica mantiene limpio el array de reservas y asegura que las cancelaciones queden reflejadas inmediatamente en el archivo de persistencia.

Conclusiones y Observaciones

Al centralizar en app.js la gestión de rutas sobre un servidor HTTP puro de Node.js, hemos logrado comprender de forma profunda el ciclo de vida de una petición sin abstraernos tras frameworks. La separación clara entre enrutamiento y lógica de negocio (controladores) permite extender el proyecto con nuevas rutas o validaciones sin tocar el núcleo del servidor.

Por otro lado, al crear funcionalidades de uso de archivos JSON como fuente de persistencia simplifica la implementación y es adecuado para prototipos o entornos de baja concurrencia, aunque presenta limitaciones de escalabilidad y bloqueo de acceso en escenarios de alta demanda. Finalmente, las validaciones incorporadas en createReservation garantizan la consistencia de los datos y previenen condiciones de carrera o sobrecarga de capacidad, lo que, junto con respuestas HTTP coherentes.