08-junio-2025

Módulo 7 – PRÁCTICA 1

# introducción

En este proyecto se ha desarrollado una API utilizando Node.js y el módulo HTTP, con el objetivo de gestionar las reservas de salas de estudio y administrar la disponibilidad de los espacios de una biblioteca. La API permite consultar la lista de salas disponibles, realizar nuevas reservas, consultar las reservas existentes y cancelar las reservas mediante peticiones HTTP. Toda la información se almacena en archivos JSON, facilitando la persistencia de los datos.

# ENLACE A GITHUB

<https://github.com/lolafdezgUE/M7Act1.git>

# endpoints generados

En el proyecto se han generado diferentes endpoints con diferentes funcionalidades.

## GET /rooms

### Este endpoint tiene como objetivo obtener un listado de todas las salas de estudio existentes en la biblioteca. Para ello, no es necesario enviar ningún parámetro, ya que simplemente devuelve una lista con todas las salas. La respuesta consistirá en un JSON que contiene un array de salas, donde cada sala tiene su identificador, su nombre y su capacidad máxima.

### Llamada y respuesta recibida:

### 

## GET /reservations

### Este endpoint tiene como objetivo obtener un listado de todas las reservas de salas realizadas en la biblioteca. Para ello, no es necesario enviar ningún parámetro, ya que simplemente devuelve una lista con todas las reservas. La respuesta consistirá en un JSON que contiene un array de reservas, donde cada reserva tiene su identificador, su nombre y su capacidad máxima.

### Llamada y respuesta recibida:

### 

## POST /reservation

### Este endpoint se utiliza para registrar una nueva reserva en el sistema. Para ello, el cliente debe enviar un objeto JSON con el nombre de usuario (user), id de la sala que quiere reservar (room), fecha de la reserva (date), hora de inicio (hourStart), hora de fin (hourEnd) y el número de personas (persons). El sistema primero validará que qué todos los datos estén informados y posteriormente hará varias validaciones más, siendo la más relevante la validación que evita que la sala pueda estar reservada varias veces simultáneamente. Si todos los datos enviados son válidos, se creará la nueva reserva y se almacenará en el JSON.

### Llamada y respuesta recibida:

### 

### Error forzado intentando una reserva cuando ya existe otra:

### 

### Error forzado metiendo una sala que no existe:

### 

### Error forzado metiendo mal las horas:

### 

## DELETE /reservation/ID

### Este endpoint permite cancelar una reserva existente. Para realizar la cancelación, el cliente debe enviar una solicitud DELETE indicando el identificador único de la reserva en la URL. El sistema buscará la reserva correspondiente a ese identificador y, si la encuentra, eliminará la información asociada a la reserva. De modo que, si la cancelación es exitosa, la respuesta incluirá un mensaje confirmando que la reserva ha sido cancelada. Sin embargo, si el identificador de la reserva no existe, se devolverá un error 404

### Llamada y respuesta recibida:

### 

### Error forzado eliminando una reserva inexistente:

### 

# conclusiones y observaciones

Este proyecto me ha permitido aprender y practicar varios aspectos esenciales en el desarrollo backend, desde el diseño y estructuración de una API REST hasta la gestión de rutas, validación de datos y persistencia de información en archivos. Además, me gustaría destacar las siguientes conclusiones en concreto.

## VALIDACIÓN DE ENTRADAS Y GESTIÓN DE ERRORES

Durante la realización de este proyecto, he comprobado que es fundamental asegurarse de que todos los datos de entrada al sistema sean válidos y completos. Por esta razón, en la configuración de los diferentes endpoints se han implementado verificaciones que garantizan que los datos sean correctos, que existan y que no se solapen reservas en la misma sala y horario. Además, si alguna de estas validaciones detecta un problema, se gestiona el error de manera adecuada, informando al usuario de la causa del fallo.

## USO DE NODE.JS Y EL MÓDULO HHTP

La realización de este proyecto ha sido una excelente oportunidad para familiarizarme con Node.js y el uso del módulo http nativo. Esta aproximación me ha permitido comprender en profundidad cómo funcionan las peticiones y respuestas HTTP, así como la gestión manual de rutas y datos sin depender de frameworks externos. La experiencia adquirida en la construcción de una API, en la implementación de las diferentes operaciones y en el manejo de los datos mediante archivos JSON ha sido altamente enriquecedora.