Problemas de Laboratorio - Express.js

1. API de Registro de Estudiantes

Crea una API REST en Express.js para registrar estudiantes en una universidad. El sistema debe permitir:
- Obtener la lista completa de estudiantes (GET /students)
- Registrar un nuevo estudiante (POST /students)
- Validar que el nombre y la carrera no estén vacíos
- Cada estudiante debe tener: id, nombre, edad, carrera

Agrega manejo de errores si faltan campos obligatorios. Usa una estructura modular con rutas, controladores y un archivo de datos.

2. API de Libros Favoritos

Desarrolla una API que permita almacenar libros favoritos. Cada libro debe tener: id, titulo, autor, anio.

La aplicación debe permitir:
- Obtener todos los libros (GET /books)
- Agregar un nuevo libro (POST /books)
- Mostrar un solo libro por ID (GET /books/:id)
- Manejar errores cuando un libro no se encuentra
- Validar que el año sea un número entero

3. API de Tareas Pendientes

Crea una API para gestionar tareas de un usuario. Cada tarea contiene: id, descripcion, completada (true/false).

Implementa lo siguiente:
- Listar todas las tareas (GET /tasks)
- Agregar una nueva tarea (POST /tasks)
- Filtrar solo las tareas completadas (GET /tasks/completed)
- Validar que la descripción no esté vacía
- Manejar errores si se intenta agregar una tarea sin descripción

4. API de Reservas de Restaurante

Implementa una API en Express que gestione reservas. Cada reserva tiene: id, nombre_cliente, personas, fecha.

Debes permitir:
- Ver todas las reservas (GET /reservas)
- Crear una nueva reserva (POST /reservas)
- Validar que el número de personas sea mayor que 0
- Devolver error si falta algún dato obligatorio

5. API de Productos de Tienda

Crea una API para administrar un catálogo de productos. Cada producto tendrá: id, nombre, precio, disponible.

Implementa:
- Listar todos los productos (GET /productos)
- Agregar un nuevo producto (POST /productos)
- Mostrar solo los productos disponibles (GET /productos/disponibles)
- Validar que el precio sea mayor que cero
- Manejar errores si se envía un producto sin nombre o con precio inválido

Utiliza una arquitectura con app.js, routes, controllers, y data.

6. API de Autenticación de Usuarios

Construye una API en Express.js para autenticar usuarios. El sistema debe:
- Registrar usuarios con email y password en memoria (POST /auth/register)
- Iniciar sesión validando credenciales (POST /auth/login)
- Manejar errores de registro (email duplicado) y login (credenciales inválidas)
- Validar formato de email y longitud mínima de password

7. API de Mensajes de Chat

Desarrolla una API para enviar y recibir mensajes de chat. Cada mensaje contiene: id, usuario, mensaje y timestamp.

Funcionalidades:
- Obtener todos los mensajes (GET /chat)
- Enviar un mensaje (POST /chat)
- Filtrar mensajes de un usuario específico (GET /chat/:usuario)
- Validar que el mensaje no esté vacío

8. API de Comentarios de Blog

Crea una API para gestionar comentarios en un blog. Cada comentario: id, autor, contenido, postId.

Debe permitir:
- Listar comentarios de un post (GET /posts/:postId/comments)
- Agregar comentario a un post (POST /posts/:postId/comments)
- Editar un comentario por id (PUT /comments/:id)
- Eliminar un comentario (DELETE /comments/:id)
- Manejar errores cuando el comentario o post no existe

9. API de Gestión de Inventario

Implementa una API para administrar inventario de productos. Cada producto: id, nombre, stock, precio.

Funciones:
- Listar inventario completo (GET /inventory)
- Agregar producto (POST /inventory)
- Validar que stock y precio sean números positivos
- Manejar errores si el producto no existe