Controladores en una API REST con Node y Express

¿Qué son los controladores?

En una API REST desarrollada con Node.js y Express, los *controladores* (controllers) son una parte clave del patrón de arquitectura MVC (Modelo - Vista - Controlador). Su función principal es actuar como intermediarios entre la lógica de negocio y las peticiones HTTP que llegan desde el cliente.

Para entenderlo mejor, pensemos en una analogía:

Analogía didáctica

Analogía: Restaurante y los Meseros

Imagina una API REST como un restaurante:

- El cliente (navegador/app) es quien hace un pedido.
- El mesero (controlador) recibe el pedido, lo interpreta y lo comunica a la cocina.
- La cocina (servicio o modelo) prepara el plato (la lógica y datos).
- El mesero luego lleva el plato de regreso al cliente.

En esta analogía, **el controlador es el mesero**. No cocina, no diseña los platos, solo se encarga de gestionar pedidos, validar que todo esté correcto y entregar la respuesta adecuada.

Funciones principales de un controlador

¿Qué hace un controlador?

1. Recibe las peticiones HTTP (GET, POST, PUT, DELETE).
2. Extrae datos relevantes de la petición (params, query, body).
3. Llama a servicios o modelos para ejecutar la lógica de negocio.
4. Devuelve una respuesta adecuada al cliente (res.json, res.status, etc.).
5. Maneja errores y los comunica correctamente.

Controladores en una API REST con Node y Express

Ejemplo básico:

Supongamos que tenemos un recurso "usuarios":

```js
// controllers/userController.js

const User = require('../models/User');

exports.getAllUsers = async (req, res) => {
    try {
        const users = await User.find();
        res.json(users);
    } catch (error) {
        res.status(500).json({ message: 'Error al obtener usuarios' });
    }
};
```

Este controlador:

- No sabe cómo se conectan a la base de datos los modelos.
- Solo se enfoca en qué hacer con la petición y la respuesta.

Estructura de proyecto con controladores

Estructura común en un proyecto con controladores:

```
/controllers
    userController.js
```

Controladores en una API REST con Node y Express

/models
    userModel.js
/routes
    userRoutes.js
/app.js
```

En `/routes/userRoutes.js` puedes tener algo así:

```js
const express = require('express');
const router = express.Router();
const userController = require('../controllers/userController');

router.get('/users', userController.getAllUsers);
```

Y en `app.js` importas las rutas:

```js
const userRoutes = require('./routes/userRoutes');
app.use('/api', userRoutes);
```

Buenas prácticas:

- Mantén los controladores delgados: solo deben contener lógica relacionada con la petición y la respuesta.
- Usa servicios o helpers para lógica compleja.
- Separa responsabilidades claramente (modelo, controlador, servicio).

Resumen
```

Controladores en una API REST con Node y Express

Resumen final:

Los controladores permiten organizar y mantener el código limpio y escalable. Así como en un restaurante el mesero no se mete a cocinar, el controlador tampoco debería encargarse de la lógica de negocio profunda. Su rol es recibir pedidos, derivarlos correctamente y entregar respuestas.

Este enfoque facilita el mantenimiento, la escalabilidad y la reutilización del código.