

Ejercicio Práctico

Node.JS

Clase N° 12 - Capa lógica

Ejercicio Práctico

Obligatorio

Organización de Rutas y Capas en tu API

Sabrina y Matías se acercan a revisar tu progreso. Matías sonríe con aprobación: "Has avanzado muchísimo, pero ahora es momento de subir un nivel. Necesitamos que configures la capa de rutas de manera profesional."



Matías, con una expresión confiada, te dice: "Es hora de que trabajes como un verdadero desarrollador profesional. Sabemos que manejar rutas en el archivo principal está bien para comenzar, pero si tu aplicación crece, esa organización será insostenible."

Sabrina da un paso adelante y añade: "Por eso, queremos que trabajes con **Express Router** para separar tus rutas en módulos, que crees controladores para manejar la lógica y que comiences a trabajar con una capa de servicios para preparar datos simulados que luego enviarás a través de tus rutas."



Ejercicio Práctico

Obligatorio

Matías asiente y explica: "Este enfoque no solo hará que tu código sea más limpio, sino que también será más fácil de mantener y escalar. ¡Manos a la obra!"

Misión:

1. Crear rutas organizadas con Express Router
 - Migra las rutas de tu archivo principal a archivos separados en una carpeta llamada **routes** por ejemplo: **products.routes.js**
 - Usa **Express Router** para configurar el o los archivos de rutas y asegúrate de exportarlos correctamente para que pueda ser utilizado en el archivo principal.
2. Implementar controladores para manejar la lógica
 - Crea un archivo llamado `product.controller.js` dentro de la carpeta `controllers`.
 - Crea los controladores necesarios para responder a las rutas definidas en el ejercicio anterior.
 - Mueve la lógica de las rutas al controlador correspondiente y asegúrate de que las funciones sean claras y reutilizables.

Ejercicio Práctico

Obligatorio

3. Añadir una capa de servicios con datos simulados

- Crea un archivo llamado `product.service.js` dentro de la carpeta `services`.
- Simula datos en formato JSON, como una lista de productos o usuarios, y utiliza estas funciones en los controladores para devolver respuestas dinámicas.

Ejemplo práctico:

Supongamos que tienes una ruta que devuelve una lista de productos. Este es el flujo que debes implementar:

- **Archivo `productRoutes.js` (en la carpeta `routes`)**
Define las rutas principales y vincúlalas a las funciones del controlador.
- **Archivo `productController.js` (en la carpeta `controllers`)**
Implementa la lógica de cada ruta, como obtener productos o filtrar por categoría.
- **Archivo `productService.js` (en la carpeta `services`)**
Crea funciones que devuelvan datos simulados para que puedan ser utilizados en el controlador.

Ejercicio Práctico

Obligatorio



Cuando termines, no olvides probar todo con **POSTMAN**. Es la herramienta ideal para asegurarte de que las rutas, controladores y servicios estén funcionando perfectamente.

Este ejercicio es crucial para consolidar lo que has aprendido. Cuando termines, tendrás una base sólida para cualquier proyecto que emprendas. ¡Buena suerte!

Tu desafío está claro. Ahora es momento de estructurar, modularizar y organizar como un profesional. ¡El código te espera!





¡Nuevo cuestionario en Campus!



No olvides que los cuestionarios son de carácter obligatorio para poder avanzar con la cursada.