de la aplicación tenga una responsabilidad claramente definida. De este modo, evitamos tener un código disperso y sin propósito, y en su lugar logramos una arquitectura más coherente y fácil de gestionar.

Es importante también resaltar que la división de responsabilidades y la partición del código serían mucho más complejas de implementar sin el uso de **módulos**. Como aprendimos en clases anteriores, los módulos permiten que podamos exportar e importar partes del código de forma ordenada, asegurando que solo las partes necesarias del sistema estén disponibles en los lugares correctos. Esto nos proporciona un control mucho más granular sobre cómo y dónde se utiliza cada fragmento de código, lo que facilita su mantenimiento y evolución a lo largo del tiempo.

Ejercicio Práctico

Estructura y Rutas de la API

Sabrina y Matías te observan con una mirada satisfecha, sabiendo que has hecho grandes avances. "Ahora que ya tienes una buena base trabajando con APIs, es momento de ponerte en los zapatos de un desarrollador que construye una aplicación desde cero", comenta Matías.

"Construir una API no solo se trata de hacer que los datos lleguen a tu aplicación, sino de cómo organizas todo el proyecto de manera que sea fácil de mantener y escalar", explica Sabrina mientras te señala la pantalla.

“Queremos que crees la estructura de directorios para tu API usando arquitectura REST", dice Matías con tono serio, "Esto significa que debes dividir tu código de manera ordenada en capas, tal como lo vimos en la clase. Queremos ver que uses una estructura clara para manejar rutas, controladores, modelos y servicios. Así, cuando tu proyecto crezca, será más fácil mantenerlo organizado y profesional."

Misión:

1. Crea la estructura de directorios para tu aplicación, asegurándote de crear carpetas para **rutas**, **controladores**, **modelos** y **servicios**.
2. En tu archivo principal, crea dos rutas nuevas:
    - Una ruta que devuelva una respuesta en formato **HTML**. Podría ser algo simple, como una página de bienvenida.
    - Otra ruta que devuelva una respuesta en formato **JSON**, por ejemplo, con una lista de usuarios o productos ficticios.
    - Todavía no es necesario colocar las rutas dentro de la carpeta routes, por lo que deberás definirlas en el archivo index.js.

Materiales y Recursos Adicionales:

Guía de Arquitectura REST: Un artículo que detalla los principios básicos de la arquitectura REST, que es la base de la organización de nuestra API.

Tutorial sobre la creación de APIs RESTful con Express y Node.js: Este tutorial paso a paso te llevará desde la creación de un servidor básico hasta la implementación de rutas, controladores, y modelos.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Preguntas para Reflexionar:

- ¿Por qué es importante dividir el código en diferentes capas, como rutas, controladores, modelos y servicios, cuando se está desarrollando una API Rest?
- ¿Cómo impacta la organización adecuada de la estructura de directorios en la escalabilidad y el mantenimiento de una aplicación en el largo plazo?
- ¿Cómo podrías aplicar la arquitectura REST a un proyecto más grande y complejo, como una aplicación con múltiples recursos interrelacionados?
- En tu experiencia, ¿cómo mejora la legibilidad del código cuando se siguen buenas prácticas de organización y se usan herramientas como módulos para separar responsabilidades?

Próximos Pasos:

- **Request y Response**: llegó el momento de aplicar de forma práctica todo nuestro conocimiento sobre la comunicación web.
- **Capa lógica**: controlando la respuesta de nuestra aplicación.
- **Modelo de datos y trabajo con JSON**: consultamos datos de forma interna y los devolvemos al cliente desde nuestra API Rest.

Buenos Aires aprende

Buenos Aires Ciudad