¿Qué es una API y como interactuamos con ella en esta práctica?

En esta práctica interactuamos con la API haciendo peticiones a ciertas "Rutas" o "Endpoints" usando HTTP como GET, POST, PUT y DELETE. Cada ruta corresponde a una acción especifica (como por ejemplo obtener un listado de datos, agregar o actualizar información). Estas peticiones se hacen desde el código en Node.js con express, donde configuramos las rutas de la API para que, al recibir una solicitud, el servidor sepa que datos devolver o cómo manejar la información que enviamos

¿Qué rol juega Express en el servidor que creamos?
actúa como el framework que estructura nuestro servidor en Node.js. Proporciona herramientas para crear rutas que gestionan las solicitudes HTTP y nos permite responder con los datos necesarios. También permite servir archivos estáticos, como HTML, CSS y JavaScript del frontend, facilitando la interacción del cliente con el servidor.

¿Cómo se implementó la simulación de una base de datos en la práctica?

Utilizamos Express para configurar las rutas principales de la API RESTful. Al recibir una solicitud HTTP (GET, POST, DELETE), Express llama a funciones específicas que manejan cada caso, como obtener o eliminar productos, o agregar nuevos. Estas funciones permiten gestionar solicitudes y responder con JSON o con archivos del frontend.

¿cómo se gestionó la información de los productos?

La gestión de la información de los productos se realizó utilizando una estructura de datos simple en el servidor, generalmente un arreglo de objetos que representa cada producto.

¿Qué limitaciones tiene este enfoque?

El enfoque de utilizar un arreglo en memoria para simular una base de datos presenta varias limitaciones Si el servidor se reinicia, todos los datos se pierden, y control de acceso que ofrece una base de datos real, lo que podría ser un problema si la aplicación crece y necesita gestionar información sensible.

¿Cómo utilizaste el método Fetch para interactuar con la API?

Utilicé el método Fetch para realizar solicitudes HTTP desde el frontend a la API que creamos Por ejemplo, para obtener la lista de productos, envié una solicitud GET a la API. Cuando el servidor respondía con los datos, los procesaba y los mostraba en la página. Para agregar un nuevo producto, usé una solicitud POST

¿Qué ventajas tiene Fetch frente a otras formas de realizar solicitudes HTTP?

Fetch es más moderno y ofrece una API más simple y promesas integradas, lo que facilita el manejo de respuestas asíncronas y errores. Comparado con XMLHttpRequest, es más claro y tiene menos código.

¿Por qué es importante el uso de promesas en Fetch?

Las promesas facilitan el manejo de operaciones asíncronas al permitir que el código siga ejecutándose sin esperar el resultado. Así, se evita bloquear la interfaz

mientras se reciben los datos y se manejan errores con más claridad, mejorando la experiencia del usuario.

¿Cómo se vinculó el frontend con el backend en esta aplicación?

HTML y CSS crean la interfaz, mientras JavaScript interactúa con el backend usando Fetch. Las solicitudes van al servidor en Node.js, que procesa y responde. Esto permite que la página muestre o actualice la lista de productos, interactuando sin recargar.

¿Qué hace cuando un producto es eliminado y cómo se refleja en la interfaz?

Al hacer una solicitud DELETE con Fetch, el servidor elimina el producto especificado de la "base de datos" simulada. La respuesta confirma la eliminación y JavaScript actualiza el DOM para reflejar el cambio en la interfaz sin recargar.

¿Cómo cambia la página de manera dinámica al agregar o eliminar productos?

a respuesta del servidor y, usando el DOM, agrega o elimina elementos en la lista visible de productos. Esto hace que la interfaz sea más fluida, ya que solo actualiza los elementos específicos.

¿Qué aprendiste sobre la estructura y organización de archivos en una aplicación web?

La estructura de archivos organizada ayuda a mantener claro el flujo de trabajo y facilita el desarrollo. Tener carpetas separadas para el frontend y el backend, y nombrar bien los archivos, mejora la legibilidad y el mantenimiento del proyecto

¿Cómo facilita esto el desarrollo y mantenimiento de la aplicación?

Tener una buena organización de archivos y una estructura clara en el proyecto facilita el desarrollo y mantenimiento de la aplicación (Claridad y navegación, Separación de preocupaciones, Facilidad para colaborar, Mantenimiento más sencillo,Escalabilidad,Testing y calidad del código

¿Qué mejoras implementarías en esta aplicación si fuera un proyecto real de e-commerce?

Incluiría una base de datos real para persistencia y seguridad, autenticación de usuarios, carrito de compras, y un panel de administración. También refinaría la organización de archivos y haría el diseño escalable para futuras mejoras.

¿Qué nuevas características agregarías y cómo mejorarías la estructura del proyecto?

Si este proyecto fuera parte de un e-commerce más grande, agregaríamos varias características para mejorar la funcionalidad y la experiencia del usuario (Base de datos real, Autenticación de usuarios, Carrito de compras, Panel de administración, Métodos de pago