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

Una API (Application Programming Interface) es un conjunto de definiciones y protocolos que permiten a diferentes sistemas o aplicaciones comunicarse entre sí. En esta práctica, interactuamos con una API a través de peticiones HTTP en diferentes rutas o "endpoints" usando métodos como GET, POST, PUT y DELETE. Cada ruta está diseñada para realizar una acción específica (como obtener datos, agregar o actualizar información). Estas peticiones se envían desde el código en Node.js usando Express, que facilita la configuración de las rutas de la API para que el servidor sepa cómo manejar cada solicitud y qué datos devolver.

Rol de Express en el servidor:

Express actúa como un 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.

Simulación de una base de datos:

En esta práctica, configuramos las rutas principales de la API RESTful con Express para manejar las solicitudes (GET, POST, DELETE). Las rutas llaman a funciones específicas que se encargan de las operaciones solicitadas, como obtener o eliminar productos, o agregar nuevos. Esto permite responder con datos en formato JSON o con archivos del frontend, simulando una base de datos en memoria.

Gestión de la información de productos:

La información de los productos se almacenó en un arreglo de objetos en el servidor, donde cada objeto representaba un producto.

Limitaciones de este enfoque:

Utilizar un arreglo en memoria tiene limitaciones, ya que los datos se pierden si el servidor se reinicia. Además, carece de las características avanzadas de una base de datos real, como persistencia y control de acceso, lo cual se vuelve crítico si la aplicación crece o maneja información sensible.

Uso de Fetch para interactuar con la API:

Desde el frontend, se utilizó el método Fetch para realizar solicitudes HTTP a la API. Por ejemplo, para obtener la lista de productos, se envió una solicitud GET, y al recibir la respuesta, los datos se mostraron en la página. Para agregar un nuevo producto, se envió una solicitud POST.

Ventajas de Fetch frente a otros métodos:

Fetch es una API más moderna y simple, con promesas integradas que facilitan el manejo de respuestas asíncronas y errores. En comparación con XMLHttpRequest, Fetch requiere menos código y es más claro.

Importancia de las promesas en Fetch:

Las promesas permiten gestionar operaciones asíncronas, lo que evita bloquear la interfaz. Esto permite que el código siga ejecutándose mientras se espera el resultado de la operación, 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 recarga.

¿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 se facilita la página de manera dinámica al agregar elementos producto?

A respuesta del servidor, y usando el DOM, agrego o elimino 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 el desarrollo web?

La estructura de archivos organizada ayuda a mantener claro el flujo de trabajo y facilita el desarrollo. Nos permite separar el backend y frontend, y permite un fácil acceso a cada componente y archivo, sin comprometer el proyecto con archivos desordenados.

¿Cómo facilita esto el desarrollo y mantenimiento del código en el proyecto?

La estructura modular permite a cualquier colaborador adaptarse al proyecto en menos tiempo y modifica solo lo necesario de forma eficiente (Claridad y navegación).

¿Cuál fue el objetivo del proyecto, cómo se identifican los roles en el proyecto?

Simular una base de datos para representar y gestionar el inventario de la tienda para practicar en la creación de una API y el diseño de su estructura modular. Administración de productos, roles de los desarrolladores fueron back y front.

¿Qué resultado fue de la API de e-commerce más grande implementada que más te sorprendió?

El proyecto fue un conjunto de componentes para manejar una tienda en línea.