# Investigación sobre AJAX y Fetch

**AJAX (Asynchronous JavaScript and XML)** es una técnica que permite a las páginas web comunicarse con un servidor y actualizarse dinámicamente sin necesidad de recargar. Aunque originalmente usaba XMLHttpRequest, hoy en día es más común emplear el método fetch, introducido en ES6.

**Fetch API** es una interfaz moderna que simplifica las peticiones HTTP. Ventajas de fetch:

1. Sintaxis más limpia y legible.
2. Devuelve una *Promise*, lo que permite usar .then() y catch().
3. Compatible con async/await, haciendo el manejo de tareas asíncronas más claro.
4. Permite manejar métodos HTTP como GET, POST, PUT, y DELETE.

# **Implementación del Proyecto**

A continuación, desarrollo una página web con un formulario de búsqueda y filtrado de datos que interactúa con la API de [JSONPlaceholder](https://jsonplaceholder.typicode.com/).

## Tabla Descripción generada automáticamenteDiseño del formulario y la tabla:

1. El formulario incluye un campo de entrada (input) para búsquedas y un botón para enviar la consulta.
2. La tabla tiene encabezados (thead) y un cuerpo (tbody) vacío donde se mostrarán los datos.

## Configuración de Fetch:

1. Implementé fetch para obtener datos de la API de usuarios de JSONPlaceholder (/users).
2. Texto

   Descripción generada automáticamenteUsé async/await para simplificar el manejo de la Promesa y garantizar un flujo de trabajo limpio.

## Filtrado de datos:

1. Texto

   Descripción generada automáticamenteLos datos recibidos se filtran usando .filter() para coincidir con el nombre o nombre de usuario introducido.
2. Se hace comparación en minúsculas (toLowerCase()) para evitar problemas con mayúsculas/minúsculas.

## Actualización de la tabla:

1. Texto

   Descripción generada automáticamenteLos resultados filtrados se renderizan dinámicamente en el cuerpo de la tabla utilizando insertAdjacentHTML.

## Manejo de errores:

1. Si fetch falla o no se encuentran resultados, se muestra un mensaje al usuario.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# Conclusión

Este ejercicio me permitió explorar el uso práctico de fetch y aplicar conceptos de programación asíncrona con async/await. El desafío principal fue manejar el filtrado de datos dinámicamente y garantizar que la interfaz respondiera sin recargar la página. Al final, logré una implementación funcional y fluida, lo que refuerza la importancia de las buenas prácticas al trabajar con APIs en proyectos web.