2-3-2023

M06: Desarrollo web en entorno cliente

UF4 – Proyecto: Marvel API

Samuel Rebollo Rodriguez | Xavier Aranda Salinas
15585039.CLOT@FJE.EDU 15585072.CLOT@FJE.EDU

ÍNDEX

1. Introducción

Descripción del proyecto:

Se pide realizar una aplicación que permite obtener información de la API de Marvel, https://developer.marvel.com/.

Esta información debe permitir obtener primero todos los cómics de un superhéroe que introducimos en un campo de búsqueda. Nos muestra en forma de columnas las portadas de todos los cómics y su título.

Al seleccionar uno concreto nos muestra todos los detalles del mismo en un espacio lateral. Estos llamamientos a la API deben realizarse en 3 tecnologías. XHR, Fetch API y WebSockets. Para realizar las llamadas con WebSockets habrá que hacer en NodeJS un Proxy que haga las llamadas a Marvel.

Realización del proyecto:

Hemos realizado este trabajo con las 3 diferentes tecnologías mencionadas anteriormente. En las 3 hemos hecho el mismo proceso, desarrollar la página web en la que el usuario es capaz de buscar los cómics de su personaje favorito y junto a esta, tiene que tener el soporte de una de las tecnologías que utilizamos ya que si la página no puede realizar peticiones a la API no tendrá contenido.

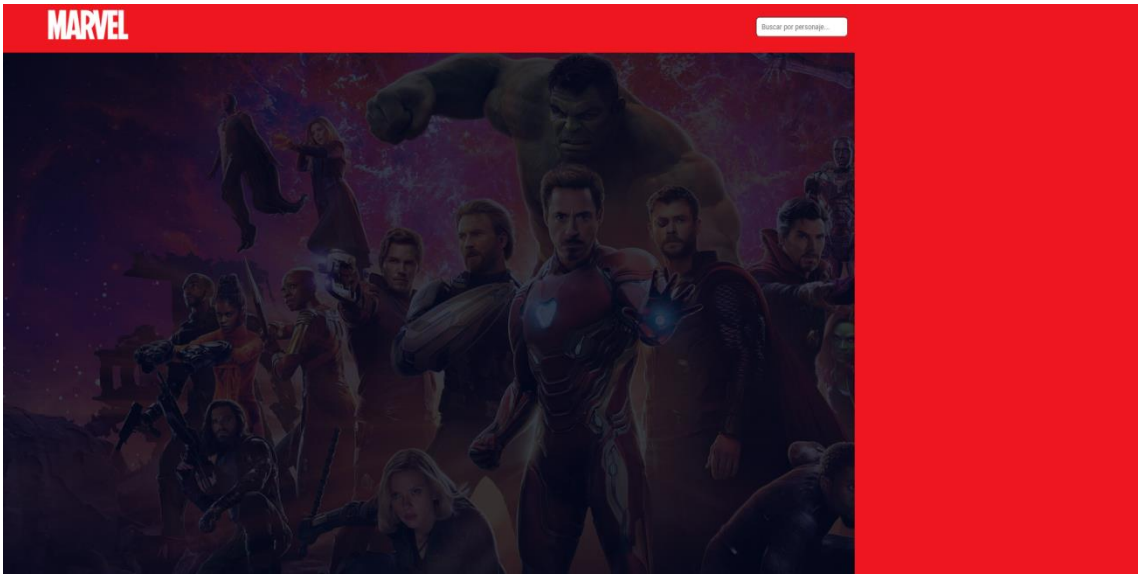
Primero hemos realizado XHR y Fetch API ya que trabajan de la misma forma y una vez las teníamos, hemos hecho el proyecto en WebSockets ya que necesita la ayuda de express (en nuestro caso) para poder utilizarlo.

2. Conceptos generales

En este apartado veremos todo lo que comparten las diferentes tecnologías y luego tendremos un apartado específico para cada uno con las cosas que las hacen diferente en este proyecto.

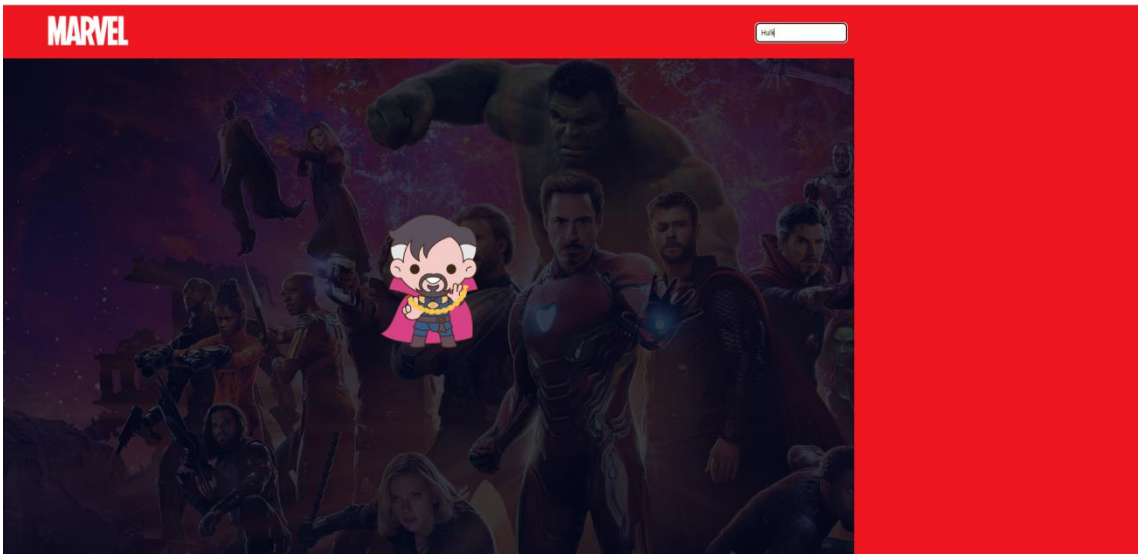
Esto a nivel visual ya que no entraremos en el código.

Página inicial:



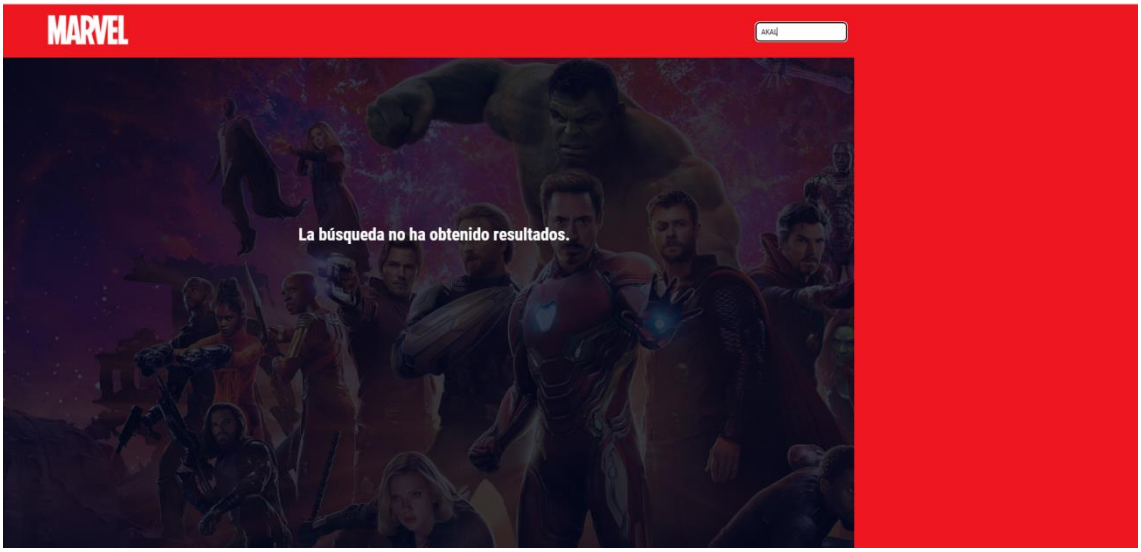
Esto es lo que veremos nada más entrar, una foto de fondo de los Avengers de Marvel con una interfaz muy simple y un buscador en la parte superior derecha.

Cargando petición:



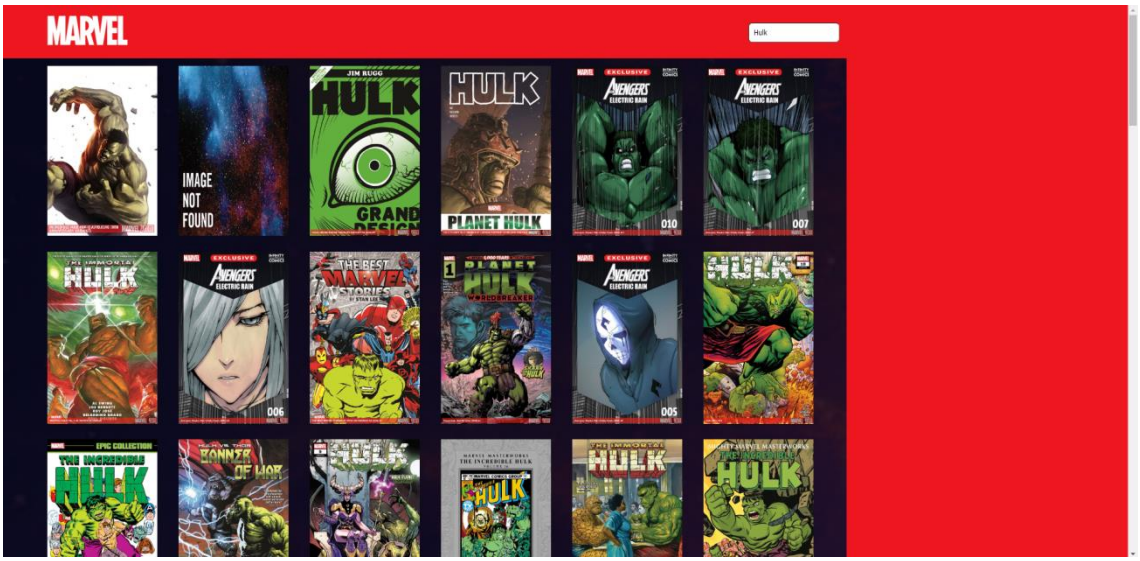
Hemos puesto un gif de Dr. Strange para que el usuario sepa cuando está cargando la petición que ha realizado.

Petición no encontrada:



Cuando la petición no obtiene unos resultados de la respuesta o bien no ha podido realizar la petición nos mostrará que no hemos obtenido ningún resultado.

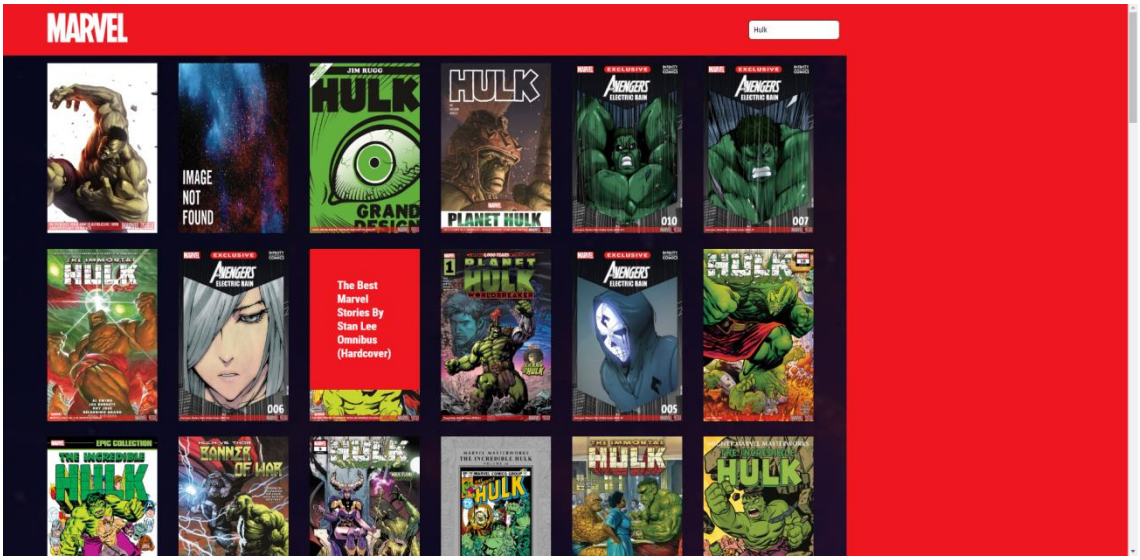
Petición realizada correctamente:



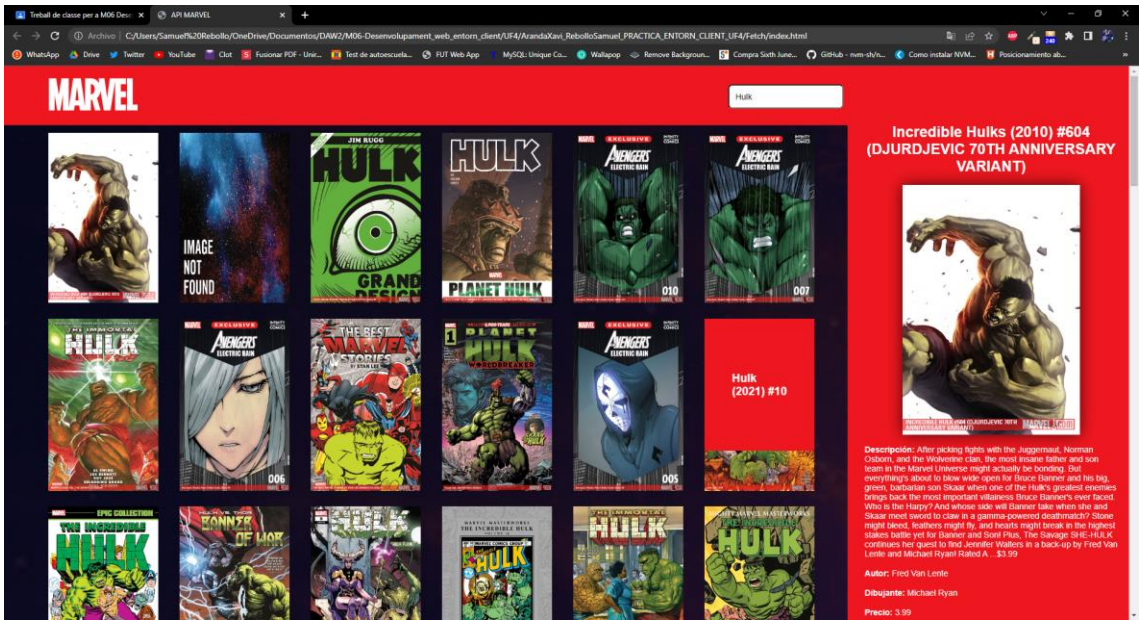
Una vez realizada, podemos ver todos los cómics del personaje buscado.

Títulos de los cómics:

Para saber el nombre del cómic tenemos dos opciones, o bien con la portada del mismo, o con un efecto “hover” cuando pasas por encima del cómic que quieres.



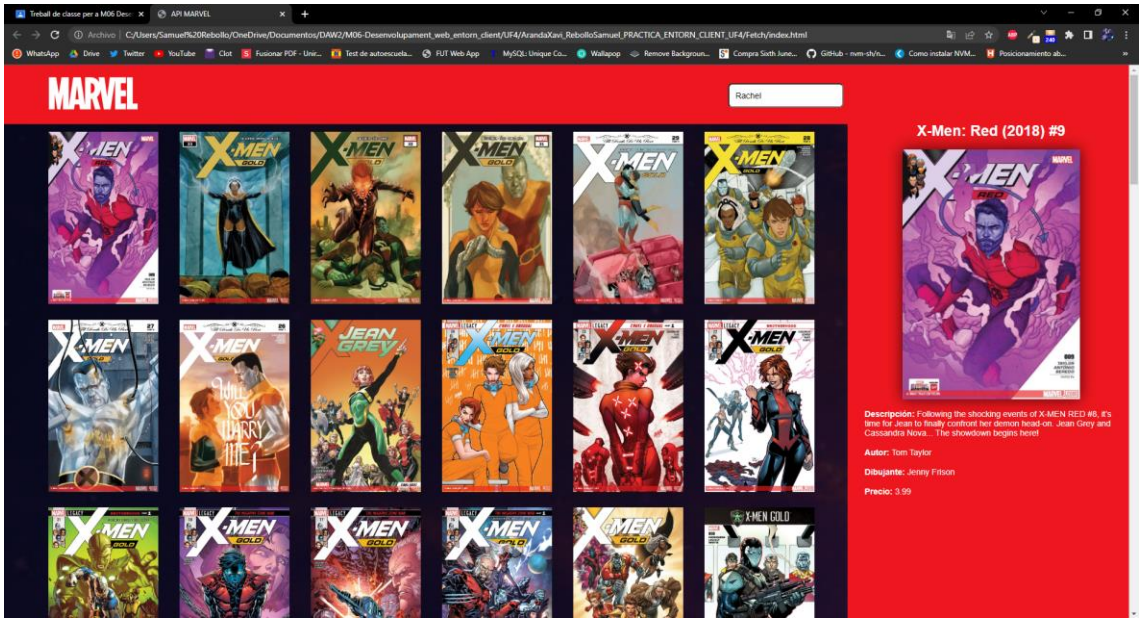
Información sobre el cómic:



Para obtener información sobre un cómic es tan fácil como hacer clic sobre él para que nos aparezca su información en la parte derecha de nuestra pantalla.

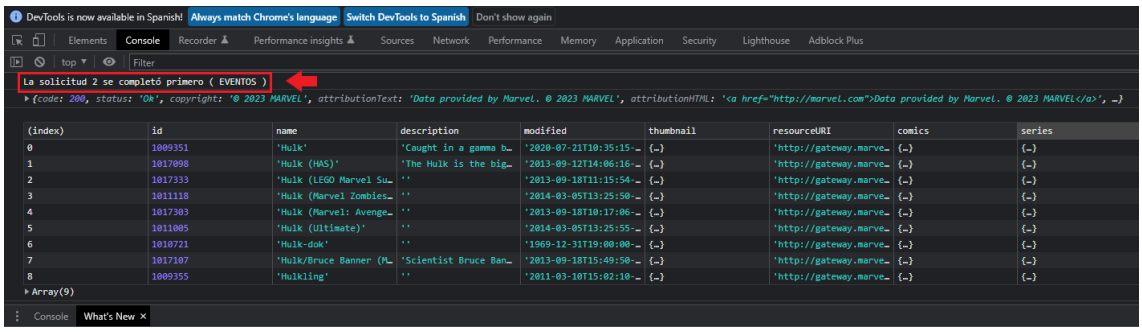
3. FETCH API

Página inicial:



Petición RACE:

```
let peticion1 = fetch(`https://gateway.marvel.com:443/v1/public/comics?ts=1&apikey=3c043a9e457ce749d34745ac17502e1f&hash=197925ff0
let peticion2 = fetch(`https://gateway.marvel.com:443/v1/public/events?ts=1&apikey=3c043a9e457ce749d34745ac17502e1f&hash=197925ff0

Promise.race([peticion1, peticion2])
    .then(response => {
        if (response.url === peticion1.url) {
            console.log('La solicitud 1 se completó primero ( COMICS )');
        } else {
            console.log('La solicitud 2 se completó primero ( EVENTOS )');
        }
        return response.json();
    })
    .then(data => console.log(data))
    .catch(error => console.error(error));
```



Hemos hecho una petición RACE a los eventos y a los comics para saber cual se realizaba antes y lo mostramos por la consola de la página.

4. XHR

Página inicial:

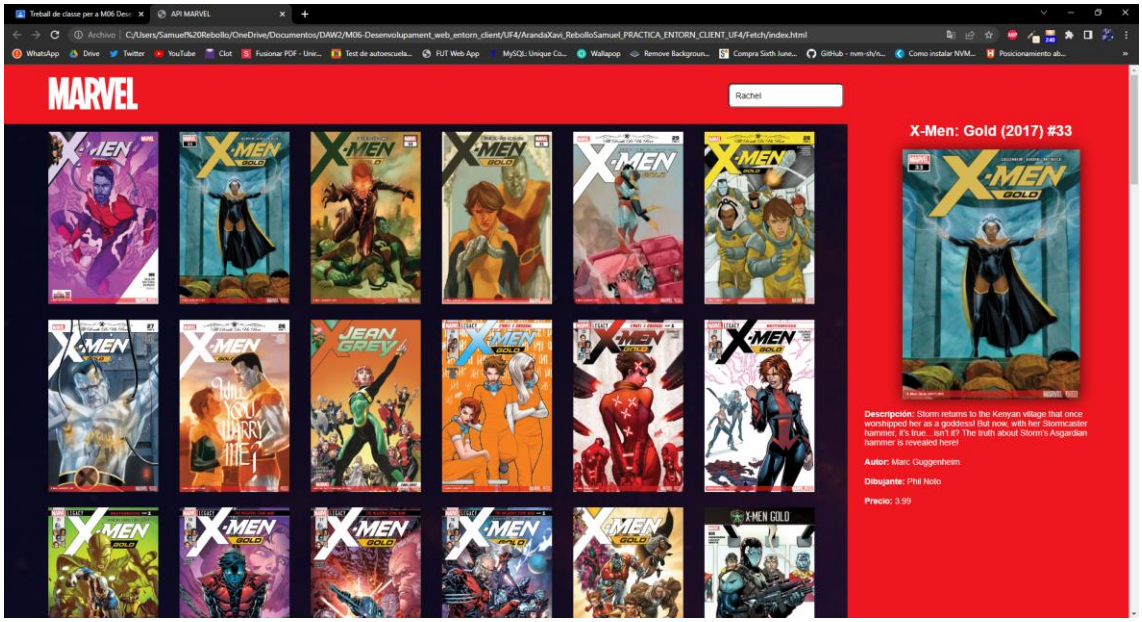
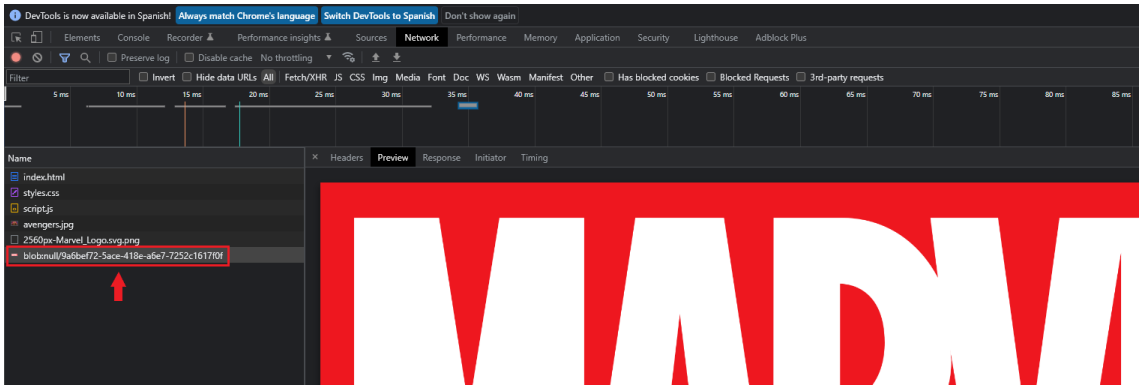


Imagen blob:

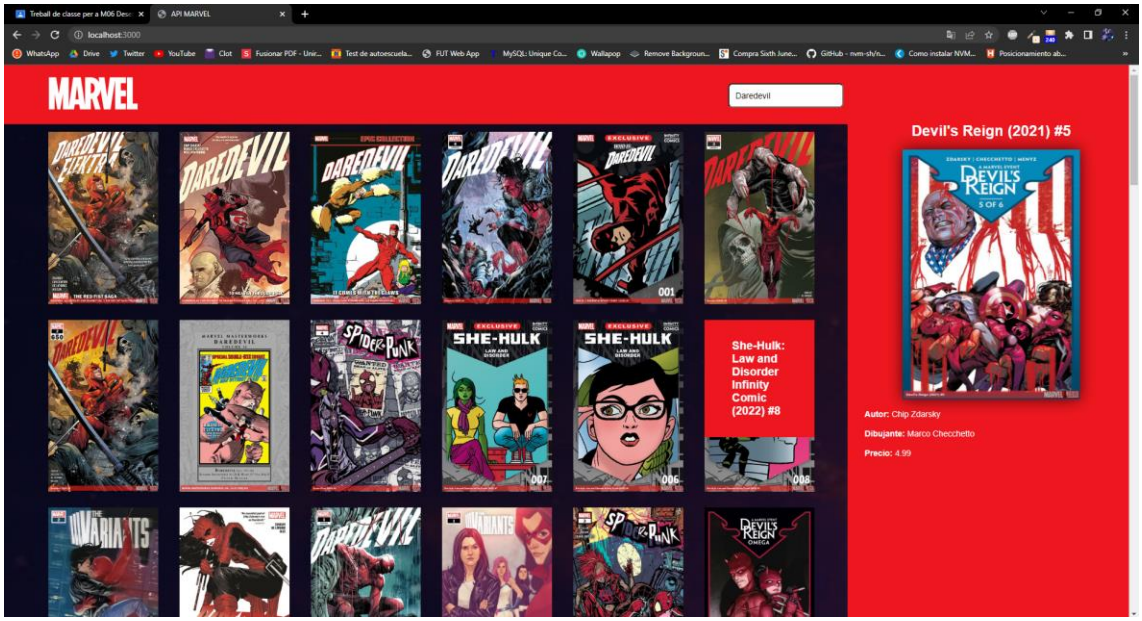
```
function carregarLogo(){
    let xhr = new XMLHttpRequest();
    // Hacemos la petición para bajarnos la imagen del logo de Marvel y situarla como logo (trabajamos con datos binarios)
    xhr.open('GET', 'https://upload.wikimedia.org/wikipedia/commons/thumb/b/b9/Marvel_Logo.svg/2560px-Marvel_Logo.svg.png', true);
    xhr.responseType = 'blob';

    xhr.onload = function(e) {
        if (this.status == 200) {
            let blob = this.response;
            let img = document.createElement('img');
            let URL = window.URL || window.webkitURL;
            img.src = URL.createObjectURL(blob);
            contenedorLogo.appendChild(img);
        };
    };
    xhr.send(null);
}
```



Para esta tecnología necesitábamos hacer uso de datos binarios, por tanto, hemos puesto el logo de la página a través de una petición “xhr” cargando la imagen del logo como un dato “blob”.

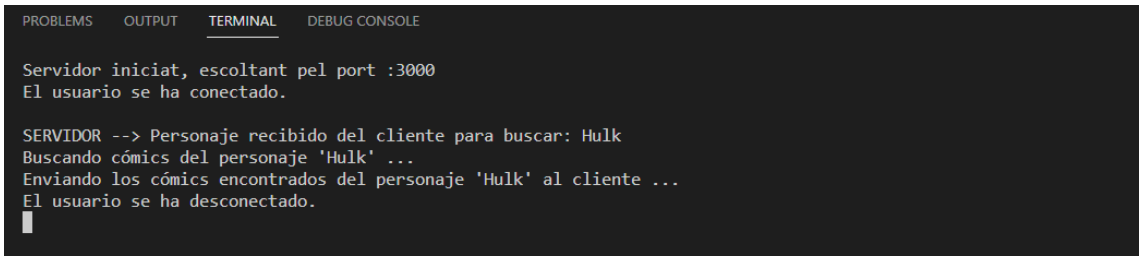
5. Web Sockets

Página inicial:



Servidor Express recibiendo mensajes del cliente:

Demostración del servidor express soportando los Web Sockets i mostrando cuando se conecta el usuario, la búsqueda que ha realizado y la desconexión del propio usuario.



Cliente recibiendo mensajes del servidor con los resultados de los cómics: