Memoria Actividad Entregable Unidad 1

Alumno: César O. Osorio A.

https://github.com/cesarozorioa/unidad1_cesar_osorio.git

1.- Enunciado:

Desarrollar un servidor utilizando el módulo http.
El servidor tendrá que devolver los datos que encontramos en el siguiente fichero.
<https://github.com/fanzeyi/pokemon.json/blob/master/pokedex.json>
El funcionamiento deseado será el siguiente:
Cuando llamemos al servidor, seguido del nombre (en cualquier idioma) o del id, se nos mostrara toda la información referente a ese Pokémon.
Por ejemplo:

1. localhost:3000/1.
2. localhost:3000/Bulbasaur .
3. localhost:3000/フシギダネ .
4. localhost:3000/妙蛙**种子** .
5. localhost:3000/Bulbizarre.

Respuesta:

6. Tipo: [“Grass”, “Poison”] .
7. HP: 45,
8. Attack: 49,
9. Defense: 49,
10. Sp. Attack: 65,
11. Sp. Defense: 65,
12. Speed: 45

Si el ID o nombre no existe en el JSON, se devolverá un string explicando esto.
Importante que para un mismo objeto/pokemon, podemos acceder a sus datos tanto por id como por todos los nombres.

2.- Explicación:

Para la realización del ejercicio se importa el módulo http de Node js para crear un servidor HTTP y también se importa el módulo fs para leer el archivo pokedex.json, el mismo que es descargado del sitio web:

https://github.com/fanzeyi/pokemon.json/blob/master/ y guardado en la carpeta D:\Backend\nodejs\UEM\actividad-unidad1 para su lectura respectiva.

Se define una variable PORT donde se asigna el nro de puerto en el que se ejecutará el servidor.

En el programa el tratamiento que se le da al archivo JSON es en primer lugar convertirlo a un array de objetos de JavaScript usando el método JSON.parse(<archivo leído>).

Este paso es fundamental para poder acceder a cada uno de los atributos de cada uno de los objetos en particular para el atributo name que tiene distintos valores haciendo referencia al mismo objeto.

La creación del servidor se lo hace a través de la línea
const server = http.createServer((req, res) =>{} y para obtener el id o nombre del pokemon de la URL usamos **Split('/')** que divide la URL en partes usando este carácter como separador, de tal forma que la instrucción const param = urlParams[1]; Obtiene el segundo elemento de las partes de la URL, que debería ser el nombre o ID del Pokémon.

La búsqueda del pokemon por el id o nombre en el array pokedex se realiza en la siguiente línea:

```
// Buscar el Pokemon por nombre o id
let pokemon = pokedex.find(p => p.id === parseInt(param) || Object.values(p.name).includes
(param));
```

Donde:

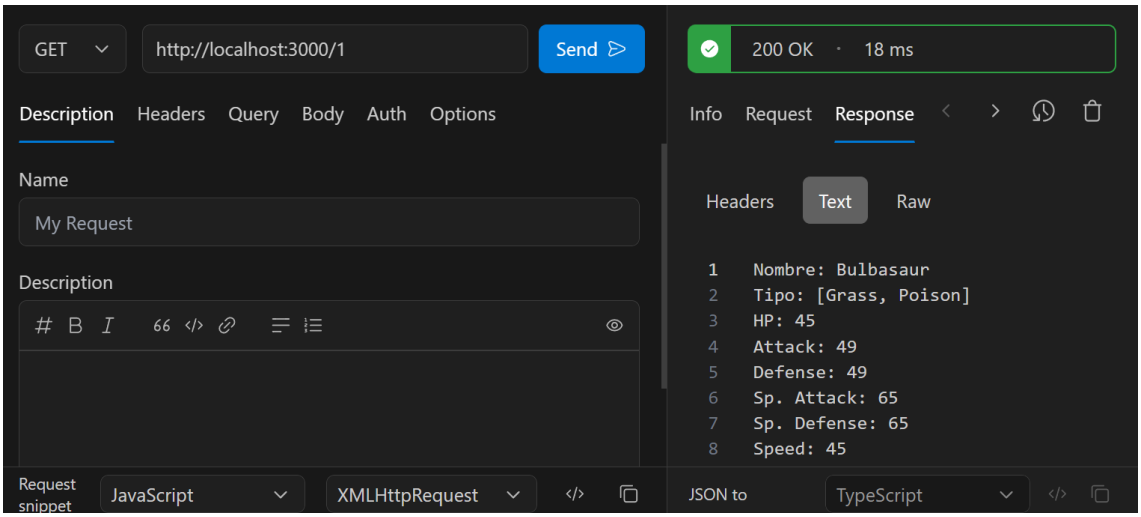
1. pokedex.find(...): pokedex es un array que contiene objetos de Pokémon. El método find() se utiliza para buscar en este array un objeto que cumpla con cierta condición.
2. (p => ...): Esto es una función de flecha (arrow function) que se utiliza como argumento para find(). Toma un parámetro p, que representa cada elemento del array pokedex que se está iterando.
3. p.id === parseInt(param): Aquí estamos verificando si el ID del Pokémon (p.id) es igual al parámetro proporcionado en la URL después de convertirlo a un número entero utilizando parseInt(param). El parseInt() se utiliza para convertir una cadena en un número entero. Esto es necesario porque el ID del Pokémon en el archivo JSON es un número, pero el parámetro de la URL es una cadena.
4. Object.values(p.name).includes(param): Esto busca en el objeto name del Pokémon todas sus propiedades y verifica si alguna de ellas incluye el parámetro proporcionado en la URL. Para hacer esto, primero usamos Object.values(p.name) para obtener un array de los valores de todas las propiedades del objeto name del Pokémon. Luego, utilizamos includes(param) para verificar si alguna de estas propiedades incluye el parámetro de la URL. Esto nos permite buscar el Pokémon por su nombre en cualquier idioma.

Entonces, en resumen, esta línea de código busca en el array pokedex un Pokémon cuyo ID coincida con el parámetro proporcionado en la URL o cuyo nombre (en cualquier idioma) coincida con el parámetro. Si encuentra un Pokémon que cumpla alguna de estas condiciones, lo asigna a la variable pokemon.

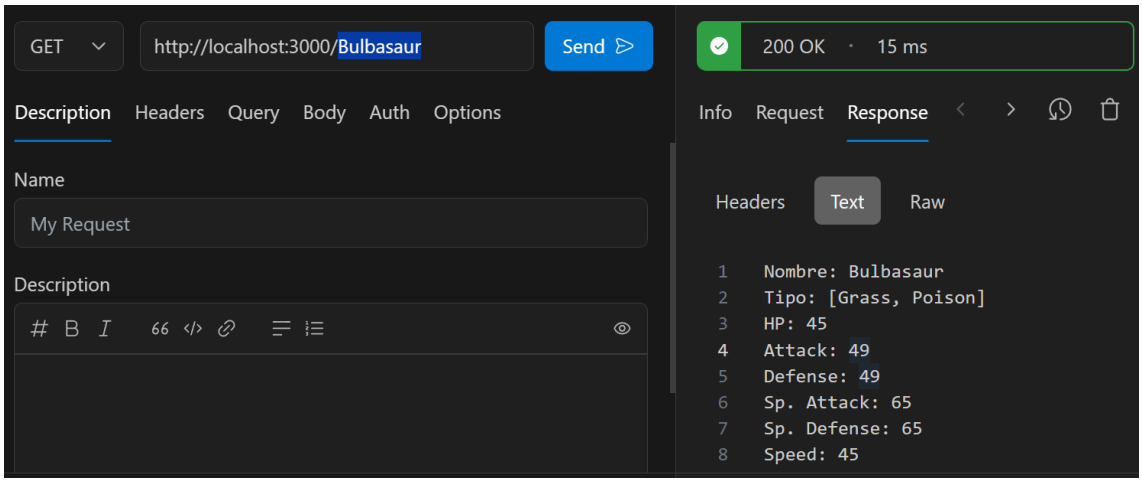
Finalmente a través de un estructura selectiva if verificamos si la variable pokemon guarda el valor de verdadero, lo que determina que se encontró lo buscado, y a continuación el servidor nos envía la información requerida en la actividad.

Captura de Salidas del Servidor

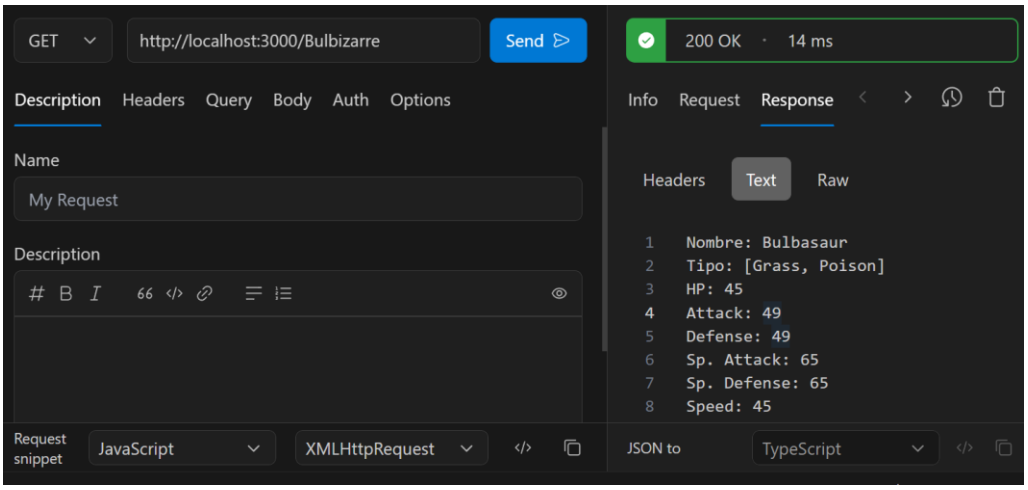
Prueba No 1 por Nro de id del pokemon



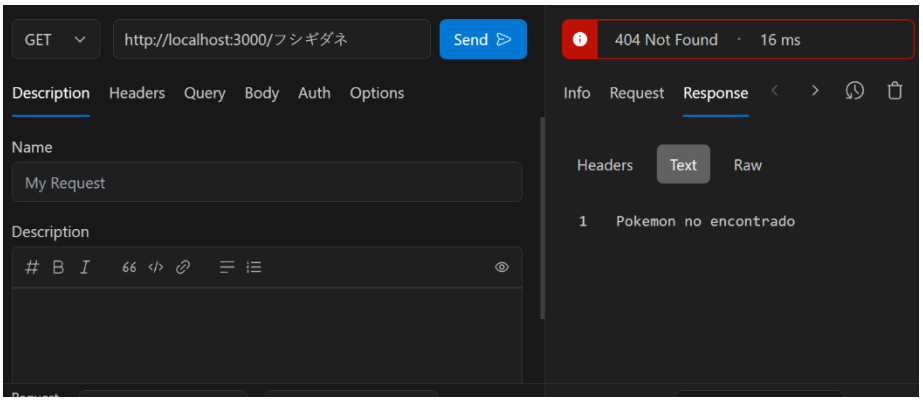
Prueba No 2 por nombre en ingles del pokemon



Prueba No 3 por nombre en frances del pokemon



Prueba No 4 por nombre en chino y japones del pokemon



Para estos dos idiomas que son símbolos especiales, no se logra resolver el problema, pienso que los caracteres leídos tienen un diferente formato irreconocible con la propiedad utf-8

Se comprueba de esta manera que también al no encontrar el pokemon ya sea por id o name, el servidor despliega un error No 404 not found.

Conclusiones

1. **Manejo de archivos JSON:** Se aprende con este ejercicio cómo cargar y manipular datos de un archivo JSON en Node.js.
2. **Uso del módulo HTTP:** El código utiliza el módulo HTTP de Node.js para crear un servidor web básico. Esto permite recibir solicitudes HTTP y enviar respuestas, lo que es fundamental para la creación de aplicaciones web y APIs.
3. **Rutas y parámetros en la URL:** Se muestra cómo extraer parámetros de una URL para realizar acciones específicas en función de ellos. En este caso, se busca un Pokémon por su ID o nombre en cualquier idioma.
4. **Búsqueda y filtrado de datos:** Se emplea el método find() para buscar un Pokémon en función de su ID o nombre en el array pokedex. Esto demuestra cómo realizar operaciones de búsqueda y filtrado en datos estructurados en JavaScript.

5. **Respuestas HTTP:** El servidor responde con códigos de estado y contenido adecuado según la solicitud. Si se encuentra el Pokémon, devuelve su información; de lo contrario, devuelve un mensaje de error 404.
6. **Uso de funciones de flecha y métodos de array:** El código hace uso de funciones de flecha y métodos de array como find() y includes(), lo que muestra la eficiencia y claridad que proporcionan estas características de JavaScript moderno.
7. En conclusión, el ejercicio ilustra cómo crear un servidor web básico en Node.js, manejar solicitudes HTTP, procesar datos en formato JSON y proporcionar respuestas adecuadas según la solicitud del cliente. Es un buen ejemplo para entender los fundamentos de la creación de servidores web y el manejo de datos en Node.js.