Ejercicios para repasar

Ejercicio 1: Simulación de llamadas a una API con Fetch

1. Crea una función `fetchUserData` que simule la llamada a una API para obtener información de un usuario. Utiliza `fetch` con una URL de prueba (puedes usar `https://jsonplaceholder.typicode.com/users/`).
2. Implementa esta función utilizando Promesas para la obtención de los datos.
3. Implemente con `async/await` para obtener los datos de los usuarios.
4. Almacena los datos de los usuarios en un Array de Objetos y guárdalo en el `LocalStorage` con la clave "usuario".

Ejercicio 2: Manejo de Errores con Async/Await y Promesas

1. Simula una función `fetchPostWithError` que intente obtener un post de una API que no existe (URL inválida).
2. Implementa dos versiones: una utilizando Promesas (`.then()` / `.catch()`) y otra con `async/await` para capturar y manejar el error.
3. Crea un array donde almacenarás los errores ocurridos y utiliza un `Set` para evitar errores duplicados.

Ejercicio 3: Múltiples Llamadas Asíncronas en Paralelo

1. Crea una función `fetchMultipleResources` que haga varias llamadas asíncronas en paralelo para obtener tres recursos diferentes: usuarios, posts y comentarios (por ejemplo, utilizando

`https://jsonplaceholder.typicode.com/users`, `/posts`, y `/comments`).
2. Implementa la función con `Promise.all()` y `async/await`, comparando las diferencias en la implementación.
3. Almacena los resultados de estas llamadas en un `Map` donde la clave sea el nombre del recurso y el valor sea un array de los elementos obtenidos.
4. Implementa la función con `Promise.allSettled()` y `async/await`.
5. Implementa temporizadores para saber cuánto tarda Promise.all y Promise.allSettled en resolver todas las peticiones en paralelo.

Crud básico sobre API json-server

> **Nota:** Un modelo de db.json se encuentra al final para estos ejercicios.

Ejercicio 1: Crear Producto

- **Enunciado**: Crear una función que permita añadir un nuevo producto a la base de datos a través de una solicitud POST, incluyendo una `categoriaId` para relacionarlo con su categoría.
- **Elementos de JavaScript a repasar**:
  - Uso de `fetch` para realizar solicitudes HTTP.
  - Manejo de promesas con `async/await`.
  - Estructuras de datos como objetos y relaciones.

Ejercicio 2: Obtener Productos

- **Enunciado**: Implementar una función que recupere todos los productos de la base de datos mediante una solicitud GET. Mostrar también el nombre de la categoría correspondiente.
- **Elementos de JavaScript a repasar**:
  - Uso de `fetch` para realizar solicitudes HTTP.
  - Manejo de promesas con `async/await`.
  - Manipulación de arrays y objetos.

Ejercicio 3: Actualizar Producto

- **Enunciado**: Desarrollar una función que permita actualizar un producto existente usando una solicitud PATCH, asegurando que se pueda cambiar también la `categoriaId`.
- **Elementos de JavaScript a repasar**:
  - Uso de `fetch` con el método PATCH.
  - Manejo de promesas con `async/await`.
  - Actualización de objetos en JavaScript.

Ejercicio 4: Eliminar Productos

- **Enunciado**: Crear una función que elimine un producto de la base de datos mediante una solicitud DELETE. Implementar el manejo de múltiples eliminaciones usando `Promise.allSettled`.
- **Elementos de JavaScript a repasar**:
  - Uso de `fetch` con el método DELETE.
  - Manejo de promesas con `Promise.allSettled`.
  - Estructuras de control como bucles y condicionales.

Ejercicio 5: Obtener y Crear Categorías

- **Enunciado**: Implementar funciones que permitan obtener todas las categorías y crear una nueva categoría. Utilizar `Promise.all` para ejecutar ambas funciones simultáneamente.
- **Elementos de JavaScript a repasar**:
  - Uso de `fetch` para solicitudes GET y POST.
  - Manejo de promesas con `Promise.all`.
  - Trabajar con objetos y arrays para gestionar datos.

Ejercicio 6: Añadir Comentarios a Productos

- **Enunciado**: Crear una función que permita añadir comentarios a un producto mediante una solicitud POST, incluyendo el nombre del usuario, contenido del comentario y calificación.
- **Elementos de JavaScript a repasar**:
  - Uso de `fetch` para realizar solicitudes HTTP.
  - Manejo de promesas con `async/await`.
  - Estructuras de datos anidadas.

Ejercicio 7: Filtrar Productos

- **Enunciado**: Implementar una función que permita filtrar productos por precio y/o stock, usando parámetros de consulta.
- **Elementos de JavaScript a repasar**:
  - Uso de `fetch` con parámetros de consulta.
  - Manipulación de arrays para filtrar datos.

Ejercicio 8: Consultar Precio Histórico

- **Enunciado**: Crear un endpoint que permita obtener el historial de precios de un producto, implementando una función para mostrar los precios anteriores.
- **Elementos de JavaScript a repasar**:
  - Uso de `fetch` para obtener datos históricos.
  - Manipulación de arrays para mostrar datos.

`Archivo db.json`

```
{
  "productos": [
    {
      "id": 1,
      "nombre": "Producto 1",
      "stock": 50,
      "precio": 29.99,
      "categoriaId": 1
    },
    {
      "id": 2,
      "nombre": "Producto 2",
      "stock": 20,
      "precio": 49.99,
      "categoriaId": 2
    }
  ],
  "categorias": [
    { "id": 1, "nombre": "Electrónica" },
    { "id": 2, "nombre": "Ropa" }
  ],
```

```
  "comentarios": [
    {
      "id": 1,
      "productoId": 1,
      "usuario": "Juan",
      "contenido": "Excelente producto!",
      "calificacion": 5
    },
    {
      "id": 2,
      "productoId": 2,
      "usuario": "Maria",
      "contenido": "Buena calidad.",
      "calificacion": 4
    }
  ],
  "historialPrecios": [
    { "productoId": 1, "precios": [29.99, 24.99, 19.99] },
    { "productoId": 2, "precios": [49.99, 39.99] }
  ]
}
```