2025

Memoria

Api públicas

Aquí podremos ver mi proyecto realizado para M6 UF4, veremos que es un trabajo mejorando el último que hicimos utilizando Fetch, y aprendiendo a utilizar una Api pública, Google Charts y Open Street Map.

Oriol Chiva Hidalgo
INS La Ferreria
27/01/2025

Contenido

Introducción

Este proyecto consiste en una aplicación en React que utiliza diferentes tecnologías para consumir y representar datos externos: json-server como base de datos local, una API externa (https://randomuser.me), Google Charts para visualización de datos y Leaflet con Open Street Map para mostrar la ubicación de los usuarios.

Descripción del Proyecto

Parte 1: Fetch

Se mejoró el sistema para editar, borrar, añadir u obtener datos gracias a Fetch:

- Obtener datos (GET)
- Añadir nuevos personajes (POST)
- Editar personajes (PUT)
- Eliminar personajes (DELETE)

```
export const TemaProvider = ({ children }) => {
    const [personajes, setPersonajes] = useState(personajesData);

    const get = () => {
        fetch("http://localhost:4000/data")
            .then((response) => response.json())
            .then((jsonData) => {setPersonajes(jsonData)})
            .catch((error) => console.error("Error:", error))

    }

    const post = (dada) => {
        fetch("http://localhost:4000/data", {
            method: "POST",
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(dada),
        })
            .then((response) => response.json())
            .then((data) => console.log("Succes: ", data))
            .catch((error) => console.error("Error: ", error))

    };
```

```
const put = (dada, id) => {
    fetch("http://localhost:4000/data/" + id,{
        method: "PUT",
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(dada),
    })
        .then((response) => response.json())
        .then((data) => console.log("Succes: ", data))
        .catch((error) => console.error("Error: ", error));
}

    const remove = (dada, id) => {
    fetch("http://localhost:4000/data/" + id,{
        method: "DELETE",
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(dada),
    })
        .then((response) => response.json())
        .then((data) => console.log("Succes: ", data))
        .catch((error) => console.error("Error: ", error));
}
```

Se implementó en el proyecto, sobretodo en ItemForm y EditForm todo lo demás sigue siendo igual al proyecto anterior.

Parte 2: API pública

Utilice una API pública: randomuser.me, donde se pueden encontrar usuarios aleatorios de diferentes partes del mundo. Así que implemente una interfaz dividida en dos columnas:

- A la izquierda: botones de búsqueda aleatoria o por nombre, y opciones para mostrar nombre, país o foto.
- A la derecha: el resultado actual que cambia cada 5 segundos.

Tuve que añadir un contador para buscar por nombre, debido que la API te busca personas aleatorias no podía buscar por la persona en concreto que se quisiera.

Parte 3: Google Charts

Añadí un gráfico donde se muestran los porcentajes de los países de la nacionalidad de cada usuario, cada vez que buscamos un usuario nuevo se actualiza.

Parte 4: Open Street Map

Instale la biblioteca Leaflet para poder añadir un mapa de la ubicación de los usuarios de nuestra API pública. El mapa podemos ampliarlo o disminuirlo para ayudarnos en nuestras búsquedas.

Dificultades Encontradas y Soluciones

Una de las cosas que dificulto mi proyecto es que al añadir nuevos usuarios, estos no podía editarlos o resérvalos. También al añadir un usuario después no me carga a la página de “Catalogo” igual que a la hora de editarlo, pero sí hacia la edición correctamente. Al ser también un servidor json-server tuve problemas con las id creadas, porque al hacer pruebas se van quitando o modificando que en un futuro me daría problemas.

Con la API publica tengo que mencionar que funciona todo correctamente aunque no lo parezca, me explico; esta API publica que utilice funciona correctamente, pero todo lo hace con usuarios aleatorios, tu no puedes buscar uno en concreto, por eso al darle a buscar seguramente no encuentre el usuario que queremos, por eso mismo añadí un contador y un mensaje avisando que después de x búsquedas no encontramos el usuario que queríamos, pero esto no es culpa mía ya que la API funciona así.

Igual también paso, que las coordenadas de los países de la API están inventadas, esto hace que nos diga que una mujer puede ser polaca, pero en Open Street Map nos esté señalando el océano indico por ejemplo, esto vuelve a ser por la API y yo ahí ya no podía hacer nada más, el código está bien implementado, si en un futuro modificaran la API y pusieran coordenadas reales funcionaría sin ningún tipo de problema.

Apartados No Realizados

Los únicos apartados opcionales que no se han realizado fueron por falta de tiempo, que son los dos siguientes: mostrar más de un mapa y añadir algún grafico más, que me hubiera gustado añadir uno más teniendo en cuenta las franjas de edades de los usuarios que iban saliendo, pero pensaba que este trabajo se podía enviar más tarde y no para el 12/5/2025 así que no me dio tiempo a realizar dichos puntos.

Mejoras realizadas y ampliaciones

Se añadió la opción de buscar por nombre (aunque al ser random, no sirva de mucho) pero por ello mismo se añadió un contador para avisar al cliente.

El grafico se va actualizando a tiempo real.

Conclusiones

Este trabajo es una total mejora comparado con el anterior por utilizar Fetch, cosa que ayudar muchísimo para los json-server. Súper importante saber utilizar las dos formas.

También me pareció un trabajo curioso y divertido poder añadir gráficos a nuestro gusto o poder poner el mapa con las ubicaciones.

Se notó que este proyecto también era más sencillo comparado con los anteriores, y que ya es final de curso y nuestro nivel también ha ido mejorando.