JSON-SERVER

Qué es una REST API?

- REST = Representational State Transfer
- API = Application Programming Interface
- Debe responder a los request de HTTP:GET,POST,PUT,PATCH, DELETE
- Tiene una forma ordenada y estructurada de poner los recursos

Verbos HTTP:

- GET --> Obtener
- POST --> Enviar datos al Servidor/Creación
- PUT/PATCH --> Actualizar
- DELETE --> Eliminar
- Cuenta con endpoints (urls) para hacer CRUDS
- Para listar todos los clientes
    - GET /clientes
- Para obtener un solo cliente
    - GET /clientes/10 (el id)
- Crear un nuevo cliente
    - POST /clientes
- Editar un cliente
    - PUT /clientes/12 (el id)
- Borrar un cliente
    - DELETE /clientes/32 (el id)

> A diferencia de una API, una REST API tiene esta estructura en sus rutas

---

Para la instalación hay que ejecutarlo como administrador

```
> npm i - g json-server
```

Creando la base de datos de mi primer REST API

Creo un archivo en la raíz llamado db.json Entre llaves y usando comillas dobles, creo el arreglo vacío clientes

```
{
    "clientes": []
}
```

- Para correr el servidor con el watch en el archivo por el puerto 4000

  > json-server --watch db.json --port 4000

- Para enviar los datos a la API, en Formulario.jsx hago async a handleSubmit
- Uso un try y un catch
- Cuando cre un registro, debo enviarlo a /clientes
- Cómo POST genera un nuevo registro.
- Por defecto, cuando hago un fetch, GET es por defecto
- Pero yo puedo especificarle el modo que quiero con esta sintaxis
  - No sé cuanto tiempo va a tomar la creación de ese nuevo registro, uso el await
  - Le estoy diciendo que a esta URL de tipo POST debo pasarle los datos del body
  - El server solo acepta strings. por eso el método stringify
  - Meto la respuesta formateada a json en una variable llamada resultado
  - Lo imprimo en consola

```
const handleSubmit=async(valores)=>{
    try {
      const url="http://localhost:4000/clientes"

     const respuesta = await fetch(url, {
        method: 'POST',
        body: JSON.stringify(valores)
      })

      const resultado= await respuesta.json()
      console.log(resultado)
      console.log(respuesta)

    } catch (error) {
      console.log(error)
    }
  }
```

- JSON server tiene sus reglas, y dice que cuando se usa una petición que no sea GET hay que agregarle estas líneas en el header

```
try {
     const url="http://localhost:4000/clientes"

     const respuesta = await fetch(url, {
       method: 'POST',
       body: JSON.stringify(valores),

       headers:{
         'Content-Type':'application/json'
       }
   }
}
```

- De esta manera ya agrego con un id único que me provee json server los datos al servidor