REACT – QUERY

Vamos a comenzar con una aplicación que recupera artículos de una base de datos json. También permite crear nuevos recursos.

Para ello montaremos un servidor local que entregará acceso a la base de datos. Tomaremos los mismos de una conocida API pública: <https://jsonplaceholder.typicode.com/>

Montaremos uno o varios endpoints en un archivo .json y lo serviremos con la dependencia JSON Server: <https://www.npmjs.com/package/json-server>

Archivo de DB debería verse así:

{

  "posts": [

    {

      "userId": 1,

      "id": 1,

      "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",

      "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"

    },

{

      "userId": 10,

      "id": 100,

      "title": "at nam consequatur ea labore ea harum",

      "body": "cupiditate quo est a modi nesciunt soluta\nipsa voluptas error itaque dicta in\nautem qui minus magnam et distinctio eum\naccusamus ratione error aut"

    }

  ]

}

Instalamos json-server: **npm i json-server**

Creamos un script para lanzarlo en el archivo package.json:

"scripts": {

    "db": "json-server --watch db.json --delay 1500 --port 3001",

Ahora que tenemos los datos servidos en un puerto, creemos los métodos para acceder a ellos. Instalaremos la dependencia Axios (podríamos trabajar también con Fetch)

**npm i axios**

Y creamos una carpeta **api** donde vamos a incluir dos archivos:

1. API.js (la instancia de Axios con la url base)
2. posts.js (los métodos para acceder a los datos)

**API.js**

import axios from "axios";

export const API = axios.create({

    baseURL: "http://localhost:3001"

});

**posts.js**

import { API } from "./API";

export const getPosts = async () => {

    const { data } = await API.get(`/posts`);

    return data;

};

export const getPostById = async (postId) => {

    const { data } = await API.get(`/posts/${postId}`);

    return data;

};

export const createNewPost = async (post) => {

    const { data } = await API.post('/posts', post);

    return data;

};

Creamos una carpeta **components** con tres componentes:

Posts.jsx; Post.jsx; NewPost.jsx

**Posts.jsx**

import { useEffect, useState } from "react";

import { getPosts } from "../api/posts";

import { Spinner, Table } from "react-bootstrap"

export default function Posts({ setPostId }) {

    const [isLoading, setIsLoading] = useState(true);

    const [error, setError] = useState(false);

    const [posts, setPosts] = useState(null);

    const fetchData = async () => {

        try {

            const data = await getPosts();

            setPosts(data);

        } catch (error) {

            setError(true);

        }

        setIsLoading(false);

    };

    useEffect(() => {

        fetchData();

    }, []);

    if (isLoading) {

        return (

            <div className="d-flex">

                Loading <Spinner animation="border" variant="success" role="status" /> Posts...

            </div>

        );

    }

    if (error) {

        return (

            <section className="alert alert-danger">

                Error fetching posts: {error.message}

            </section>

        );

    }

    return (

        <section>

            <h2>Posts:</h2>

            <Table bgcolor="mistyrose" hover>

                <thead className="bg-dark text-white">

                    <tr>

                        <th>Title</th>

                    </tr>

                </thead>

                <tbody style={{ display: "block", maxHeight: '70vh', overflow: "auto" }}>

                    {posts.map((post) => (

                        <tr key={post.id} onClick={() => setPostId(post.id)} >

                            <td >{post.title.toUpperCase()}

                            </td>

                        </tr>

                    ))}

                </tbody>

            </Table>

        </section>

    );

}

**Post.jsx**

import { useEffect, useState } from "react";

import { getPostById } from "../api/posts";

import { Spinner } from "react-bootstrap"

export default function Post({ postId }) {

    const [isLoading, setIsLoading] = useState(true);

    const [error, setError] = useState(false);

    const [post, setPost] = useState(null);

    useEffect(() => {

        const fetchData = async () => {

            try {

                const data = await getPostById(postId);

                setPost(data);

            } catch (error) {

                setError(true);

            }

            setIsLoading(false);

        };

        fetchData();

    }, [postId]);

    if (isLoading) {

        return (

            <div className="d-flex">

                Loading <Spinner animation="grow" variant="success" /> Post...

            </div>

        );

    }

    if (error) {

        return (

            <div className="alert alert-danger">

                Error fetching post: {error.message}

            </div>

        );

    }

    return (

        <article>

            <h1>{post.title}</h1>

            <p>{post.body}</p>

        </article>

    );

}

**NewPost.jsx**

import React, { useState } from "react";

import { Form } from "react-bootstrap";

import { createNewPost } from "../api/posts";

function NewPost() {

    const [title, setTitle] = useState("");

    const [body, setBody] = useState("");

    const [isLoading, setIsLoading] = useState(false);

    const [error, setError] = useState(false);

    const handleSubmit = async (e) => {

        e.preventDefault();

        setIsLoading(true);

        try {

            await createNewPost({ title, body });

            setTitle("");

            setBody("");

        } catch (error) {

            setError(true);

        }

        setIsLoading(false);

    };

    return (

        <section>

            <h2>Create Post:</h2>

            <Form onSubmit={handleSubmit} className="">

                <div className="mb-2">

                    <Form.Label htmlFor="title">

                        <b>Title:</b>

                    </Form.Label>

                    <Form.Control

                        value={title}

                        onChange={(e) => setTitle(e.target.value)}

                        id="title"

                    />

                </div>

                <div className="mb-2">

                    <Form.Label htmlFor="content">

                        <b>Content:</b>

                    </Form.Label>

                    <Form.Control as={'textarea'}

                        value={body}

                        onChange={(e) => setBody(e.target.value)}

                        style={{ resize: "vertical" }}

                        id="content"

                        rows={5}

                    ></Form.Control>

                </div>

                <button disabled={isLoading || !title || !body} className="btn btn-success mb-2">

                    {isLoading ? (

                        <>

                            Submitting...<span className="spinner-border spinner-border-sm"></span>

                        </>

                    ) : (

                        "Submit"

                    )}

                </button>

                {error && (

                    <p className="alert alert-danger">Error creating the post</p>

                )}

            </Form>

        </section>

    );

}

export default NewPost;

Con los componentes listos, hay que montar el mecanismo inicial en el archivo **App.jsx**

import { useState } from "react";

import NewPost from "./components/NewPost";

import Post from "./components/Post";

import Posts from "./components/Posts";

export default function App() {

  const [postId, setPostId] = useState(null);

  return (

    <main className="container">

      <h1 className="mb-4">React-Query Demo</h1>

      {postId && (

        <div>

          <a onClick={() => setPostId(null)} href="#">

            Back

          </a>

        </div>

      )}

      {postId ? (

        <Post postId={postId} setPostId={setPostId} />

      ) : (

        <div className="row gap-4">

          <div className="col-md">

            <NewPost />

          </div>

          <div className="col-md">

            <Posts setPostId={setPostId} />

          </div>

        </div>

      )}

    </main>

  );

}