**DAM 2 10/10/2024**

**Professor:** Ignasi López

**Assignatura:** MP07

# Guia d'ús d'APIs en React

Les APIs (Application Programming Interfaces) permeten que les aplicacions es comuniquin entre si. En React, podem utilitzar APIs per obtenir dades d'un servidor i mostrar-les en la interfície d'usuari. En aquesta guia, veurem com fer-ho utilitzant fetch i com gestionar l'estat de l'aplicació amb useState i useEffect.

## Pas 1: Configuració del projecte React

Si encara no tens un projecte creat, pots crear-ne un de nou utilitzant **Create React App**.

npx create-react-app api-example

cd api-example

npm start

Aquestes comandes crearan i iniciaran un projecte React bàsic.

## Pas 2: Entendre useState i useEffect

useState és un hook que permet a React gestionar l'estat d'un component. Aquest estat pot canviar amb el temps i React s'encarregarà d'actualitzar la UI automàticament.

**useEffect**

useEffect s'utilitza per gestionar efectes secundaris com ara peticions a APIs. Aquest hook s'executa després que el component s'hagi renderitzat. És especialment útil per fer fetch de dades quan el component es munta per primera vegada.

## Pas 3: Fer una petició a una API

Suposem que volem obtenir dades d'una API pública, com l'API de **JSONPlaceholder**, que ens permet obtenir una llista de publicacions (posts). Aquesta API ens retorna dades en format JSON.

Primer, crearem un component funcional en React per fer la petició a l'API.

import React, { useState, useEffect } from 'react';

function PostList() {

const [posts, setPosts] = useState([]); // Estat per desar les dades de l'API

const [loading, setLoading] = useState(true); // Estat per mostrar que s'estan carregant les dades

const [error, setError] = useState(null); // Estat per gestionar errors

// useEffect per fer la petició quan el component es munta

useEffect(() => {

fetch('https://jsonplaceholder.typicode.com/posts')

.then((response) => {

if (!response.ok) {

throw new Error('Error en carregar les dades'); // Si hi ha un error en la resposta

}

return response.json();

})

.then((data) => {

setPosts(data); // Actualitzem l'estat amb les dades de l'API

setLoading(false); // Deixem de mostrar el missatge de càrrega

})

.catch((error) => {

setError(error.message); // Actualitzem l'estat amb l'error

setLoading(false);

});

}, []); // La dependència buida [] assegura que només s'executi una vegada al muntatge

// Renderització condicional per mostrar l'estat de càrrega, error o les dades

if (loading) {

return <p>Carregant dades...</p>;

}

if (error) {

return <p>Error: {error}</p>;

}

return (

<div>

<h1>Llista de Publicacions</h1>

<ul>

{posts.map((post) => (

<li key={post.id}>

<h2>{post.title}</h2>

<p>{post.body}</p>

</li>

))}

</ul>

</div>

);

}

export default PostList;

**Explicació del codi**

1. **useState([])**: Inicialitzem l'estat dels posts com un array buit. Aquest serà el lloc on desarem les dades de l'API.
2. **useEffect(() => {...}, [])**: Utilitzem useEffect per fer la petició a l'API quan el component es munta. L'array buit [] assegura que l'efecte només s'executi una vegada.
3. **fetch()**: fetch és la funció que fem servir per obtenir dades de l'API. Si la resposta és correcta, convertim les dades en JSON i les desem a l'estat amb setPosts.
4. **Renderització condicional**: Primer comprovem si les dades s'estan carregant o si hi ha hagut un error. Quan les dades estan disponibles, recorrem la llista de posts i els mostrem en pantalla.

## Pas 4: Executar el projecte

Després d'implementar el component, assegura't que el projectes estigui funcionant correctament. Afegeix el component PostList dins de l'arxiu App.js:

import React from 'react';

import './App.css';

import PostList from './PostList'; // Importem el component

function App() {

return (

<div className="App">

<PostList /> {/\* Mostrem la llista de publicacions \*/}

</div>

);

}

export default App;

Ara, quan vagis al navegador (probablement a http://localhost:3000), hauries de veure una llista de publicacions carregades des de l'API.