API REST en React

Módulo 2 - Unidad 2

Listado de usuarios desde una API pública

Objetivos

Practicar:

- Consumo de una API REST con fetch.
- Manejo de estados loading, error y data.
- Uso de useState y useEffect.
- Renderizado condicional.
- Buenas prácticas en el manejo de errores.

Consigna

Consigna

1. **Preparar el proyecto**

- Crear un proyecto React (puede ser en Vite).
- Limpiar archivos innecesarios.

2. **Componente Usuarios.jsx**
   - Crear tres estados:
     - usuarios (array vacío inicialmente).
     - loading (*booleano*, inicia en true).
     - error (string vacío inicialmente).
   - En useEffect:
     - Hacer una solicitud a https://jsonplaceholder.typicode.com/users.
     - Usar try/catch para manejar errores.
     - Validar res.ok antes de procesar la respuesta.
     - En caso de éxito, guardar la data en usuarios y cambiar loading a false.
     - En caso de error, guardar un mensaje en error y cambiar loading a false.
3. **Renderizado condicional**

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

- Si loading es true, mostrar: **"Cargando usuarios..."**.
- Si error tiene valor, mostrar el mensaje de error.
- Si hay datos en usuarios, renderizar una lista <ul> con sus nombres y correos.

4. **Extras opcionales**
   - Mostrar un botón "Recargar" que vuelva a ejecutar la solicitud.
   - Implementar una barra de búsqueda que filtre usuarios por nombre.
   - Dividir la lista en un componente hijo UsuarioCard.

Formato de presentación

La entrega debe realizarse a través de un **repositorio en GitHub**, el cual debe incluir:

- Proyecto creado en **React (Vite recomendado)**, con archivos innecesarios eliminados. Declarador en .gitignore.
- **Componente Usuarios.jsx** con la lógica de:
  - Estados usuarios, loading, error.
  - useEffect con la solicitud a la API y manejo de errores mediante try/catch.
  - Renderizado condicional según el estado.

- **Componente hijo UsuarioCard.jsx** (si se implementa el extra de dividir la lista).
- **Archivo de estilos CSS** (global o por componente) para la presentación de la lista.
- **Archivo README.md** con:
    - Descripción breve del proyecto.
    - Instrucciones para clonar, instalar dependencias y ejecutar (npm install / npm run dev).
    - Capturas de pantalla mostrando los tres estados (cargando, error, datos).
    - Créditos del autor (nombre del estudiante, curso, unidad).
    - Citación de fuentes (bibliografía y créditos de imágenes o recursos, si corresponde).

Criterios de evaluación

- Correcta implementación de **fetch con async/await** dentro de `useEffect`.
- Manejo adecuado de **estados en React** (`useState`, `useEffect`).
- Uso correcto de **renderizado condicional** para mostrar *loading*, error o datos.
- Validación con `res.ok` antes de procesar la respuesta.
- Manejo explícito y claro de **errores**.

- Organización del código en componentes (`Usuarios`, `UsuarioCard` si aplica).
- Presentación visual clara y prolija de la lista de usuarios.
- Organización correcta del repositorio (estructura limpia, README completo).
- Claridad, indentación y buenas prácticas de sintaxis en React y JavaScript.

Bibliografía utilizada y sugerida

Libros y otros manuscritos

Banks, A. y Porcello, E. *Learning React: Modern Patterns for Developing React Apps*. 2ª ed. O'Reilly Media; 2020.

Flanagan, D. *JavaScript: The Definitive Guide. Master the World's Most-Used Programming Language*. 7ª ed. Estados Unidos: O'Reilly Media; 2020. https://share.google/NWtI0mj2jOuHpwbuu

Artículos y documentación en línea

MDN Web Docs. (s.f.). *Window: fetch() method*. Mozilla Corporation. https://developer.mozilla.org/en-US/docs/Web/API/fetch

React. (s.f.-a). *useState Hook*. https://react.dev/reference/react/useState

React. (s.f.-b). *useEffect Hook*. https://react.dev/reference/react/useEffect