Aprendizaje Basado en Proyectos: Consumo de API en el Sistema del Hospital

Contexto:

En este proyecto, los estudiantes deberán implementar el **consumo de una API** para obtener y mostrar datos del sistema del hospital, como la información de doctores o servicios médicos. Utilizando **useEffect** y **useState**, se espera que realicen peticiones asíncronas, gestionen el estado y manejen los errores de manera eficiente. Podrán utilizar **Fetch API** o **Axios** según su preferencia.

Duración: **2 horas**

Requisitos:

1. **Implementación de Peticiones con useEffect y useState (2 puntos)**

- Usa los Hooks **useEffect** y **useState** para realizar peticiones a una API externa (puedes simular una API REST) que devuelva datos relacionados con los servicios médicos o el equipo de doctores del hospital.
    - Asegúrate de que los datos se carguen cuando el componente se monte en el DOM.
    - Los datos deben mostrarse correctamente en una lista o tabla en la vista correspondiente.

2. **Uso de Fetch API o Axios para el Consumo de la API (1.5 puntos)**

- Implementa las peticiones a la API utilizando **Fetch API** o **Axios** para obtener los datos de manera asíncrona.
    - Explica en el **README** por qué has elegido una u otra opción.
    - Asegúrate de manejar correctamente los errores de la petición (por ejemplo, mostrar un mensaje de error si la API no responde o devuelve un error).

3. **Peticiones Basadas en Eventos del Usuario (1 punto)**

- Permite que el usuario realice una petición a la API mediante una interacción, como un botón para recargar la lista de doctores o servicios médicos.

- Asegúrate de que el botón realice la petición y actualice los datos en la interfaz.

4. **Manejo de Errores en Peticiones Asíncronas (1 punto)**

- Implementa una estrategia de manejo de errores cuando la API falle o no responda.
    - Muestra un mensaje en la interfaz indicando que ocurrió un error, y permite al usuario intentar realizar la petición nuevamente.

5. **Optimización del Rendimiento al Omitir Efectos en useEffect (0.5 puntos)**

- Implementa una optimización en **useEffect** para evitar que las peticiones se realicen múltiples veces innecesariamente. Asegúrate de que la petición se realice solo cuando el componente se monte o cuando haya un cambio relevante (por ejemplo, al hacer clic en el botón para recargar los datos).

Herramientas a Utilizar:

- **ReactJS** para el consumo de API y gestión de Hooks.
- **Fetch API** o **Axios** para realizar las peticiones asíncronas.
- **React Developer Tools** para verificar el comportamiento de los Hooks y el manejo del estado.

Entrega:

- **Formato de entrega**:
    - Opción 1: Enviar un **enlace al repositorio de GitHub** con el proyecto ReactJS, que incluya la integración de la API y el manejo de los errores.
    - Opción 2: Entregar un archivo **ZIP comprimido** con el proyecto, asegurándose de incluir las peticiones a la API, el manejo de errores, y un archivo README explicando la elección entre **Fetch API** y **Axios**.