Ejercicio Práctico: Manejo Avanzado del DOM y Elementos ReactJS para el Proyecto del Hospital

Contexto:

En este ejercicio práctico, los estudiantes profundizarán en el manejo del **DOM virtual**, el uso de referencias para manipular elementos del DOM, y la integración de **componentes avanzados** en ReactJS. Implementarán funcionalidades avanzadas en el sistema del hospital, como la gestión del DOM en el cliente y servidor, la optimización de rendimiento, y el uso de **fragmentos** y **componentes de orden superior** para mejorar la modularidad y eficiencia del sistema.

Duración: **2 horas**

Requisitos:

1. **Manejo del DOM Virtual en ReactJS (1 punto)**

- Implementa una sección del sistema del hospital donde se gestione eficientemente la renderización de datos utilizando el **DOM Virtual**.
    - Explica cómo ReactJS utiliza el DOM virtual para mejorar el rendimiento de la interfaz del hospital (secciones como listado de doctores o servicios).
    - Implementa **componentDidMount** o el uso de `useEffect` para gestionar la actualización del DOM al cargar los datos.

2. **Creación y Uso de Referencias en React (1.5 puntos)**

- Utiliza **referencias** en uno de los componentes para manipular elementos del DOM directamente. Por ejemplo:
    - Crea una referencia para controlar un campo de formulario en el que los usuarios agendan citas o consultas médicas.
    - Usa referencias mediante **callback** para interactuar con elementos del DOM cuando se realiza alguna acción del usuario (como enfocar un campo en el formulario de contacto del hospital).

3. **Uso de Fragmentos y Contexto en ReactJS (1.5 puntos)**

- Usa **Fragmentos** (`<React.Fragment>`) para evitar elementos innecesarios en el DOM y mejorar la estructura del código en el sistema del hospital.
    - Utiliza **Context** para gestionar el estado global de la aplicación, como el acceso a los datos de un usuario o doctor a lo largo de la aplicación sin necesidad de pasar **props** entre múltiples componentes.

4. **Verificación de Tipos con PropTypes (1 punto)**

- Implementa **PropTypes** para verificar el tipo de datos que se pasan a los componentes `DoctorCard`, `ServiceList`, y `AppointmentForm`.
    - Asegúrate de que los tipos de datos como strings, arrays y objetos se validen correctamente y muestra un mensaje de error en caso de que el tipo de dato no sea el correcto.

5. **Uso de Componentes de Orden Superior y Portales (1.5 puntos)**

- Implementa un **Componente de Orden Superior (HOC)** para reutilizar la lógica de un componente en otras secciones del sistema.
    - Crea un portal para renderizar un modal que muestre información detallada de un doctor o servicio en una capa superior sin interferir con la estructura del DOM principal.

6. **Optimización de Rendimiento y Profiler en ReactJS (0.5 puntos)**

- Usa herramientas de optimización de ReactJS como **Profiler** para identificar cuellos de botella en la renderización y mejorar el rendimiento del sistema del hospital, especialmente en secciones que cargan muchos datos, como el listado de doctores o citas.

Herramientas a Utilizar:

- **ReactJS** para el manejo del DOM virtual, referencias, y herramientas de optimización.
- **React Developer Tools** y **Profiler** para identificar y mejorar el rendimiento de la aplicación.
- **PropTypes** para la verificación de tipos de datos en los componentes.

Entrega:

- **Formato de entrega**:
    - Opción 1: Enviar un **enlace al repositorio de GitHub** con el proyecto ReactJS actualizado, incluyendo los componentes y funcionalidades avanzadas implementadas.
    - Opción 2: Entregar un archivo **ZIP comprimido** con el proyecto React, incluyendo el código de los componentes, el uso de referencias y la optimización del DOM.