Decisiones de Diseño

1. **Estructura del Proyecto:**
    - Organización de Archivos: La estructura del proyecto se organiza en carpetas para componentes (src/components), páginas (src/pages), y activos (src/assets). Esto mejora la mantenibilidad y escalabilidad del código, permitiendo un acceso y gestión más fáciles de los diferentes elementos del proyecto.
2. **Separación de Estilos y Lógica**:
    - CSS en Archivos Separados: Los estilos CSS se han movido a la carpeta assets, manteniendo la lógica y el estilo desacoplados. Esto facilita la lectura y modificación de los estilos sin interferir con la lógica de la aplicación.
3. **Componentización:**
    - Componentes Reutilizables: Se han creado componentes reutilizables (TaskItem, TaskEditForm) para tareas individuales y el formulario de edición. Esto promueve la reutilización del código y facilita la prueba y el mantenimiento de componentes individuales.
4. **Uso de useState y useEffect:**
    - Estado Local: useState se utiliza para gestionar el estado local del componente, como la lista de tareas (tasks), la nueva tarea (newTask) y el índice de edición (editingIndex).
    - Efectos Secundarios: useEffect se usa para cargar tareas desde localStorage al inicializar el componente y para guardar tareas en localStorage cada vez que cambian. Esto asegura que los datos persisten entre recargas de página.
5. **Interactividad y Manejo de Eventos:**
    - Eventos de Formularios: Se manejan eventos como la adición de tareas (handleAddTask), edición de tareas (handleEditTask), eliminación de tareas (handleDeleteTask), y actualización de estado de las tareas (handleToggleState). Esto asegura una interacción fluida y una buena experiencia de usuario.
    -

Implementación de Buena Práctica Adicional

Principio de Responsabilidad Única (SRP):

- Descripción: SRP es uno de los principios SOLID que establece que un módulo o clase debería tener una sola responsabilidad, es decir, solo una razón para cambiar.
- Aplicación:
    - Componentes Dedicados: Cada componente tiene una responsabilidad clara y única.
        - TaskItem se encarga de representar una tarea individual, gestionar su estado y manejar su eliminación.
        - TaskEditForm se encarga de la edición de una tarea existente.

        - TaskList gestiona la lista completa de tareas, incluida la adición, edición y eliminación de tareas.
    - Desacoplamiento: La lógica de cada acción (añadir, editar, eliminar, cambiar estado) está desacoplada y encapsulada en funciones dedicadas dentro de los componentes correspondientes. Esto facilita la prueba y el mantenimiento del código.
- **Beneficios:**
    - Mantenibilidad: Al tener responsabilidades claras y únicas, los componentes son más fáciles de entender, mantener y modificar.
    - Reutilización: Componentes bien definidos y desacoplados pueden ser reutilizados en diferentes partes de la aplicación o en otros proyectos.
    - Pruebas: Es más fácil escribir pruebas unitarias para componentes que tienen una única responsabilidad, mejorando la fiabilidad del código.