1. App (líneas 3 a 78)

El componente **App** es el componente raíz de toda la aplicación, y su responsabilidad principal es gestionar el estado global de los usuarios, las tareas, el usuario seleccionado y el tema (claro/oscuro). aquí se utilizan dos contextos de React:

- **ThemeContext**: permite cambiar entre tema claro y oscuro a nivel global, afectando al className del <body>.
- **UserContext**: contiene toda la lógica relacionada con los usuarios y sus tareas, incluyendo funciones como addUser, selectUser, addTask, deleteTask, toggleTask, editTask, etc.

El contenido de la interfaz se divide en dos zonas principales:

- <Sidebar /> a la izquierda, que contiene la gestión de usuarios y la información lateral.
- <Main /> a la derecha, que muestra las tareas del usuario seleccionado.

La jerarquía de componentes está bien organizada para que todos los datos fluyan a través de los contextos de forma clara, sin necesidad de prop-drilling excesivo. además, se utilizan estados controlados para los inputs de usuario y tarea, tal como pide el enunciado.

2. Sidebar (líneas 80 a 93)

El componente **Sidebar** representa el menú lateral izquierdo. aquí se muestra:

- el título "Usuarios"
- la lista de usuarios (<UserList />)
- el formulario para añadir un nuevo usuario (<UserForm />)
- un bloque adicional (<UserInfo />) que solo aparece si hay un usuario seleccionado, mostrando datos sobre sus tareas
- un botón para cambiar el tema, que cambia entre light y dark a través del ThemeContext

Este componente se mantiene fiel a la estructura visual original del HTML imperativo (**<aside class="sidebar card">**) y organiza su contenido de forma vertical con flexbox.

3. Main (líneas 95 a 110)

- El componente **Main** representa el contenido principal del área derecha de la aplicación.
- Si no hay usuario seleccionado, muestra el mensaje "Selecciona un usuario".
- Si hay usuario seleccionado, muestra:
    - La lista de tareas (<TaskList />)
    - El formulario para crear nuevas tareas (<TaskForm />)

Esto se encuentra envuelto en un , para mantener el mismo diseño visual que la versión original. también se utilizan condicionales para mostrar u ocultar secciones dinámicamente.

4. UserList (líneas 112 a 126)

Este componente renderiza la lista completa de usuarios disponibles. se accede a la lista de usuarios desde el **UserContext** y se genera un <li> por cada uno.

- Si el usuario está seleccionado, se le aplica un estilo especial (fontWeight: 'bold') para resaltarlo.
- Al hacer clic sobre un usuario, se ejecuta selectUser(index) para seleccionarlo y mostrar su información y tareas.

Esto replica el comportamiento imperativo original en renderUsers(), pero con un enfoque completamente declarativo.

5. UserInfo (líneas 128 a 138)

Este componente solo se muestra cuando hay un usuario seleccionado (selectedUserIndex !== null) en ese caso, se presenta:

- El nombre del usuario (user.name)
- Un resumen de sus tareas (completadas / totales)
- Un botón para deseleccionar al usuario, que ejecuta la función deselectUser()

Esto se corresponde con el bloque #userInfo que estaba oculto en la versión original hasta que se seleccionaba un usuario. ahora, en React, esto se gestiona con JSX condicional.

6. UserForm (líneas 140 a 149)

UserForm contiene el campo de texto para escribir el nombre de un nuevo usuario, junto con un botón para añadirlo. internamente:

- usa un estado local (useState) para almacenar el valor actual del input
- cuando se pulsa el botón, se ejecuta addUser(name), que está definida en el UserContext

Tras añadir el usuario, el input se resetea. esto reemplaza completamente el bloque addUser() del JS original con una solución declarativa y controlada.

7. TaskList (líneas 151 a 165)

El componente TaskList se encarga de mostrar las tareas del usuario actualmente seleccionado.

- Cada tarea se muestra como un <li> que incluye:
    - El texto de la tarea, con un evento onClick para marcarla como completada (esto cambia su clase a "completed")
    - Un grupo de botones para editarla (usando prompt) o eliminarla

El componente usa className="completed" condicionalmente para aplicar el estilo tachado si la tarea está finalizada. este comportamiento era parte del renderTasks() original.

8. TaskForm (líneas 167 a 175)

TaskForm es el formulario que permite crear nuevas tareas. igual que UserForm, usa un estado local para controlar el valor del input.

- Si el usuario ha escrito algo y hay un usuario seleccionado, se llama a addTask(text) para añadir la tarea a la lista
- Después, se limpia el input

Esto reemplaza el bloque imperativo addTask() por un componente bien encapsulado y reutilizable.