***Decisiones de estado local/global:***

**Estado Global:**

El estado global se maneja con Redux Toolkit en [**features/globalSlice.ts**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) y se accede mediante hooks personalizados en [**store/hooks.ts**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)  
Se utiliza para datos que deben ser accesibles desde múltiples componentes o que representan el estado general de la aplicación.

**¿Qué se guarda en el estado global?**

* **contadorNotificaciones:** contador de notificaciones nuevas.
* **notificaciones**: lista de notificaciones en la bandeja de entrada.
* **archivadas**: lista de notificaciones archivadas.

Estos datos se comparten entre componentes como [**BellBtnWidget**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)**,**[**MainModal**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), y los subcomponentes del modal.

**Ventajas del estado global:**

* Permite que cualquier componente acceda y modifique el estado de notificaciones.
* Facilita la sincronización de la UI cuando llegan nuevas notificaciones (por ejemplo, vía WebSocket en [**useNotifications**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)).

**Estado Local:**

El estado local se maneja con [**useState**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) dentro de los componentes para controlar aspectos de UI que solo afectan a ese componente o a su jerarquía inmediata.

**Ejemplos de estado local:**

* **isModalOpen** en [**App.tsx**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): controla si el modal principal está abierto o cerrado.
* **isFullOpen**en [**MainModal.tsx**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): controla si el modal está maximizado.
* **actualSection** en [**MainModal.tsx**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): controla si se muestra la bandeja de entrada o archivadas.
* Estado y pagina en [**ModalBody.tsx**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): filtran y paginan las notificaciones mostradas.
* **isSelect** en [**ModalHeader.tsx**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): controla si el selector de idioma está abierto.

**Ventajas del estado local:**

* Mantiene la lógica de UI encapsulada y simple.
* Evita recargar el estado global con detalles que solo importan a un componente específico.

***Features/Updates:***

* Papelera de reciclaje para recuperar elementos eliminados
* Selector múltiple para marcar como leído/archivado varios elementos
* Añadir límites al **BellBtnWidget** de forma tal que el arrastre no lo mueva de los límites del componente padre
* Servicio de Traducción del lado del BackEnd.
* Añadir Notificaciones en IOs
* Añadir variables de entorno
* Empaquetar APP como un micro frontend usando herramientas como [Module Federation](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) (Webpack 5) o [Single-SPA](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).

***Riesgos de rendimiento:***

1. **Re-renderizados innecesarios:**

Uso de estado global para grandes listas:  
Al almacenar listas completas de notificaciones y archivadas en el estado global, cualquier cambio (agregar, archivar, marcar como leído) puede provocar re-renderizados en todos los componentes que usan estos datos, incluso si solo una notificación cambió.

Solución:  
Usar selectores memorizados (reselect) para evitar renders innecesarios y dividir el estado si es posible.

1. **Selectores no memorizados:**

Si los selectores ([**useAppSelector**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)) no están memorizados, cada vez que el estado global cambia, los componentes que los usan pueden volver a renderizarse, aunque el dato que usan no haya cambiado.

Solución:  
Implementar selectores memorizados.

1. **WebSocket y actualizaciones frecuentes**

Si llegan muchas notificaciones en poco tiempo, cada una dispara una actualización del estado global y, por lo tanto, múltiples renders.

Solución:  
Considerar técnicas de "batching" (agrupar actualizaciones) o throttling/debouncing para reducir la frecuencia de renders.

**4. Componentes que dependen de todo el estado**  
Si un componente usa todo el array de notificaciones o archivadas, se re-renderizará ante cualquier cambio, aunque solo una notificación haya cambiado.

Solución:  
Dividir los componentes para que cada uno reciba solo la notificación que le corresponde, o usar identificadores para renders más eficientes.

**5. Carga inicial de datos**  
Al cargar todas las notificaciones y archivadas al inicio, si el volumen es grande, puede haber un retardo perceptible.

Solución:  
Implementar paginación o carga bajo demanda.

**6. Persistencia y sincronización**  
Si el estado global crece mucho y se sincroniza frecuentemente con el backend, puede haber cuellos de botella en la red o el renderizado.

Solución:  
Optimizar las llamadas al backend y considerar almacenar solo los datos necesarios en el estado global.

**7. Uso de useSelector sin filtros**  
Si se usa [**useAppSelector**](vscode-file://vscode-app/c:/Users/Manuel%20Rodr%C3%ADguez/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) para obtener grandes porciones del estado sin filtrar, cualquier cambio menor puede causar renders innecesarios.

Solución:  
Seleccionar solo los datos estrictamente necesarios para cada componente.