Documentación - Quisur Challenge

Estructura General de la Aplicación

La aplicación está desarrollada en React utilizando Vite, con soporte para WebSockets, consumo de datos desde un backend con json-server y estálizacion basada en HeroUI y TailwindCSS. La arquitectura está dividida en páginas, componentes reutilizables y hooks personalizados.

Estructura de Componentes

App

- Punto de entrada principal.
- Encapsula la aplicación con el HeroUIProvider, el GlobalContextProvider y el BrowserRouter.

DefaultLayout

- Layout general de todas las vistas.
- Incluye la barra de navegación, título y estructura del contenido principal.

Navbar

- Barra superior que contiene el conmutador de tema (ThemeSwitch) y el sistema de notificaciones.

ThemeSwitch

- Permite alternar entre modo claro y oscuro.
- El estado del tema se gestiona desde un GlobalContext personalizado.

Notifications

- Muestra las notificaciones en tiempo real.

- Se alimenta desde WebSocket y muestra una lista con posibilidad de ver detalles en un modal.

NotificationModal

- Modal que muestra el contenido de una notificación.

OptionsComponent

- Componente que renderiza las opciones de acción como exportar CSV, crear nuevo item y selector de fechas.

- Recibe props como canCreate, canExportAsCSV, canChangeDate, para hacerlo reutilizable.

TableComponent

- Renderiza una tabla con los datos de productos usando columnas configuradas dinámicamente.

- Permite abrir modales para editar registros.

ModalComponent

- Modal genérico que contiene un formulario para registrar o actualizar datos.

ModalRegister

- Formulario que se renderiza dentro del ModalComponent.

- Utiliza un hook useForm para manejar los valores del formulario.

DonutChart

- Componente de gráfico tipo dona usando ApexCharts.

- Agrupa los productos por categoría y suma la cantidad total de cada una, utilizando su color según la categoría.

LineChart

- Componente de gráfico de línea usando ApexCharts.
- Muestra la evolución del totalPrice acumulado por día y por categoría dentro de un rango de fechas.

Hooks Personalizados

usePost

- Se encarga de manejar el alta y modificación de productos.
- Envía datos por WebSocket tras creación o modificación.

useForm

- Maneja los cambios del formulario y devuelve los valores y el handler handleInputChange.

useGetProductsData

- Encapsula la lógica para obtener los datos de productos y generar las columnas para la tabla.

useSetData

- Hook genérico para consumir un endpoint y retornar los datos.

useWebSocket

- Establece una conexión WebSocket y escucha mensajes entrantes.
- Utilizado tanto en Notifications como en usePost.

useSetDonutContent

- Devuelve los datos agrupados por categoría para alimentar el DonutChart.

- Utiliza useMemo para evitar recálculos innecesarios.
- Filtra por fechas.

useSetLineContent

- Devuelve datos para el LineChart, agrupando totalPrice por fecha y categoría.
- Utiliza date-fns para manipular intervalos de fechas.

Decisiones de Diseño

WebSocket

- Se utiliza para enviar y recibir notificaciones en tiempo real.
- Cada vez que se crea o modifica un producto, se envía una notificación a través del WebSocket.

Contextos Globales

- Se usa GlobalContext para manejar el tema actual y el estado global del spinner.
- ContextRegister se utiliza para manejar la lógica de refresh de datos y control de modales en formularios.

Optimizaciones

- useMemo y useCallback se utilizan para evitar renderizados innecesarios, especialmente en la generación de datos para gráficos y tablas.

Tecnologías Utilizadas

- React + Vite
- TypeScript
- ApexCharts
- HeroUI + TailwindCSS
- json-server
- WebSocket nativo
- date-fns