Módulo 6 – Global state

Explicar la nueva implementación de React-Hook-Form  
Existen muchas opciones como Mantiene (Explicar que es UI y Utilidades) pero Hook-Form funciona tanto en web com native

Explicar estos puntos de React-Hook-Form

* Integración con Zod y el uso de constrains en Zod.
* Definición con type, uso y valores retornado del Hooks.
* Como inicializar el form con valores o reiniciarlo con otros.
* El uso de “handleSubmit” y cuando es que se ejecuta.
* Como usar la función “register” y porque se definió un nuevo prop inputProps.

# ¿Qué es Global State?

El global state es un concepto ampliamente aplicado en diversas aplicaciones, ya sea web, nativas, incluso en ciertos servidores. En ReactJS, el estado global nos permite almacenar cualquier tipo dato, ya sea primitivo, JSON, Clase etc. Además en ReactJS se acostumbra a definir además de los valores, también las funciones que realizan las modificaciones, del estado.

# ¿Por qué usar Global State?

Los componentes pueden compartir valores entre sí con la ayuda de props. Sin embargo, el problema está cuando deseamos compartir valores o funciones, entre componentes hermanos, o que el flujo va de un componente hijo a un padre. Incluso si fuera de un componente padre a un “nieto” eso involucra que se necesita enviar valores a un componente intermedio. Que probablemente no necesite esos valores, pero los recibe solo a manera de “Componente tránsito”, lo que provoca que el componente tenga mas código del que debería.

Es por esto que el uso de un global state es necesario. Muchas veces los global state son usados para almacenar información de las secciones que está visitan el usuario. Por ejemplo, al entrar en la sección de “Mi cuenta”, se realiza una consulta de los datos del usuario y se guarda en un “Slide”, por ejemplo “UserProfile”, del Global State la información. Así desde los diferentes componentes ligados a la sección “Mi cuenta” pueden acceder a los valores que ellos necesiten y no todos los disponibles.

Esto último es muy importante, previamente se mencionó el término “Slide”, que como su nombre lo indica, es una rebanada o porción del Global State. Este concepto nos ayuda a dividir el Global State en porciones enfocadas a ciertas secciones, funcionalidades o lógica de nuestra aplicación web. Lo que ayuda a modularizar la creación de Global State en la estructura del proyecto y que sea mucho mas fácil leer y mantener.

# ¿Qué opciones hay para crearlo?

Como se ha mencionado previamente, el entorno del desarrollo web es muy frenético. Suele salir nuevas librerías, incluso, nuevas versiones de las que ya conocemos y pueden cambiar mucho, u ofrecer otro enfoque para realizar “lo mismo”. Es imposible quizás poder implementarlo y probarlo al 100% todo, sin embargo, lo que sí es importante es estar consiente de estas opciones emergentes y leer su documentación como mínimo. Entender su funcionamiento, detectar sus fortalezas y debilidades para saber cuándo optar por una o no. Generalmente todo depende de la combinación de la complejidad y tamaño del equipo de trabajo.

Actualmente existe 3 opciones bastante conocidas para el manejo del Global State:

* React Context, curva de aprendizaje muy rápida pero su estructura no es la ideal para el manejo de petición asíncronas. Se pueden realizar, pero no es lo ideal.
* Redux (Sagas o Thunks), curva de aprendizaje un poco más compleja. Sin embargo, es una estructura más robusta y pensaba para peticiones asíncronas.
* Zustand, curva de aprendizaje muy sencilla, si bien no está pensada para peticiones asíncronas se puede usar.

En este curso revisaremos implementaciones de las 2 primeras y la compararemos al final con la última opción. Es importante conocer las librerías que llevan más tiempo como Context o Redux, si buen pueden parecer más viejas o “anticuadas” en comparación a otras, recordemos que muchas veces llegamos a proyecto ya consolidades que muy probablemente usen alguna de esas dos. Por lo que tratar de migrar todo es simplemente inviable. El uso de nuevas librerías o enfoques es bueno practicarlo para proyectos personales, prácticas o en el levantamiento de nuevos.

# Definición de implementación

A continuación, usaremos el global State para mocker los datos de un usuario logeado que puede tener el rol de “admin” o “saas-user” y “user”.

* Admin, puede entrar a todas las secciones y crear nuevos usuarios.
* Saas-user, puede entra a todas las secciones y crear usuarios, pero en el campo ciudad el valor permitido solo será el mismo valor del campo “city” del usuario logeado.
* User, no puede crear nuevos usuarios solo listar usuarios o alguno con su URL directo.

# React Context

1. Crear archivo “src/type/context.ts” para definir el type/schema de UserLogin. Con los campos name, role y city.
2. Crear el archivo “src/context/index.ts”. Donde se creará el context. Usar el tipo UserLogin.
3. Añadir al archivo “main.tsx” el componente de Context.
4. En el componente Layout añadir la lógica en un useEffect para validar que el usuario tiene permiso de estar en el path ingresado.
5. Modificar el componente Input, para que si recibe el prop value (para inicializar) y actualizar el componente Users (donde se busca) para actualizarlo.
6. En el componente del form añadir validaciones de permisos según “saas-user”.
7. En el componente Home poner el nombre del usuario logeado y ocultar botones.

# Redux-Thunks

1. Instalar las dependencias necesarias
   1. npm install @reduxjs/toolkit Redux react-redux
2. Crear la estructura de carpetas necesaria
   1. Crear la carpeta “store”.
   2. Crear el archivo index.ts
   3. Crear el archivo “store/slice/user.ts” y “store/slice/theme.ts”.
3. Definir el type necesario para Theme.
4. Definir las variables necesarias dentro de cada slice:
   1. InitialState, con sus valores iniciales de cada slice.
   2. CreateSlice, para definir name, initialState y los respectivos reducers.
   3. Exportar los Actions.
   4. Exportar los select que deseemos crear.
   5. Exportar todo el reducer.
5. En el archivo “store/index.ts” importar y crear el store con los reducers definidos y exportar el store creado.
6. En el archivo “main.tsx” importar y wrapear al App con el Provider de Redux.
7. Crear los Hooks para el dispatch y el select necesarios en el archivo “hooks/store.ts”.
8. Actualizar la definición para usar el rol en el componente Layout.tsx.
9. Actualizar la definción para usar el rol en el componente home/index.tsx. Usar name y rol para mostrar el nombre y ocultar el botón según las reglas.

# Extra