**REACT**

1. Es una librería
2. Es declarativo
3. Eficiente -> Si se actualiza un componente o una parte de él, el DOM solo actualizará esa parte que cambió no va a montar o renderizar todo el componente.
4. Es predecible
5. App móvil con React Native
6. Server-side con Node

Graphical user interface, text, application

Description automatically generated

**JSX:** JavaScript + XML

**BABEL:** traduce el código actual o moderno al código que si entiende el navegador.

Graphical user interface

Description automatically generated with medium confidence

**COMPONENTES**

Es una pequeña pieza de código encapsulada que realiza un trabajo en específico y puede tener un estado o no. Uno se diferencia del otro por los argumentos que recibe.

**REACT DOM**

ReactDOM= es importante que usemos react DOM porque nos permite crear nuestro árbol de componentes y nos provee una comunicación entre componentes de manera sencilla.

**REDUCER**

**Graphical user interface, text, application

Description automatically generated**

Una función pura se refiere a que todo se debe resolver dentro de la función sin necesidad de llamar a otra función.

Text

Description automatically generated

**CONTEXT**

Es un espacio/contenedor creado a nivel superior de los componentes que lo van a usar para guardar la información, métodos que dichos componentes podrían usar o ejecutar sin necesidad de pasarlo por las props.

Diagram

Description automatically generated

**REDUX**

1. Es un contenedor predecible del estado de la aplicación. Es una forma de controlar donde se encuentra la información de mi aplicación en todo momento y también ayuda a que la modificación de la información sea en una sola vía de manera predecible con el objetivo de prevenir cambios accidentales en la misma.
2. STORE: fuente única de verdad. Donde se encuentra la información que va a consumir la aplicación.
3. Reducer es una función pura que maneja un estado. El estado sirve para que la vista lo lea, la vista puede ejecutar un action que llega al dispatcher del reducer para cambiar algo en el estado, esta action llega a un reducer específico para generar un nuevo estado, y este nuevo estado se mostrará en la vista.

<https://res.cloudinary.com/practicaldev/image/fetch/s--m5BdPzhS--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_66%2Cw_880/https://i.imgur.com/riadAin.gif>

Diagram

Description automatically generated

DIFERENCIA ENTRE PUSH Y REPLACE DE HISTORY

History.push(‘/path’) -> agrega a la ruta un path más

History.replace(‘/path’) -> reemplaza la historia con el nuevo path, cuando le de hacia atrás no navegarà a ninguna ruta porque no existira historia anterior

History.goBack() -> vuelve al path inmediatamente anterior, pero si no hay es mejor usar una validación, como por ejemplo cuando está en incognito.

If (history.length <= 2) {

History.push(“/”)

} else {

History.goBack()

}

**USEMEMO**

Se usa cuando quiero memorizar el valor de una variable cada vez que cambia su dependencia. Retorna un valor. Optimiza mucho las funciones y procesos pesados.

Const hero = useMemo(() => funcionAEjecutar, [heroId]);

FUNCION DE BÚSQUEDA

Es importante que si se necesita implementar una búsqueda, en caso de que exista en el front, un input que permita hacer búsquedas. Hacerlo mediante query string.

handleSearch = () => {

history.push(‘?q=${ valor ingresado input}’)

}

Revisar useLocation y paquete queryString

<https://github.com/Klerith/react-hero-app/releases/tag/v0.14.0>

REPOSITORIOS PROFE CURSO UDEMY: <https://github.com/Klerith?page=2&tab=repositories>

NOTAS:

Context provider, también posee el router context, ósea que cualquier componente que está debajo del context puede hacer uso del context del history, ejemplo, useHistory()

<https://github.com/Klerith/react-hero-app/releases/tag/v0.15.0>

PRUEBAS REACT ROUTER V5: <https://github.com/Klerith/react-hero-app/releases/tag/v0.16.0>

NO SE DEBE USAR EL INDEX COMO KEY

Porque el key se usa para que React sepa que elemento se eliminó del árbol, y cuando se usa el index, por ejemplo, se elimina el index 0, este elemento siempre va a existir en un arreglo con longitud entonces React se puede confundir y eliminar un elemento que no debería

GIFCARD APP: <https://github.com/Klerith/react-vite-gif-expert/tree/fin-seccion-6>

Es Buena práctica separar los useEffect según las responsabilidades y no hacer todo en uno solo.

ALL HOOKS: <https://github.com/Klerith/react-hooks/tree/fin-seccion-9>

APP TODO – HOOK: <https://github.com/Klerith/react-hooks/tree/fin-seccion-10>

CUSTOM HOOKS: <https://github.com/Klerith/custom-hooks>

HEROES APP – ROUTER V6: <https://github.com/Klerith/react-heroes/tree/fin-seccion-14>

PRUEBAS UNITARIAS ROUTER V6: <https://github.com/Klerith/react-heroes/tree/fin-seccion-16>