Repaso para el PI oral

x: Lukas tiene el resumen !..
Alguien/Todos:

0. que es una api ?
API o Application Programming Interface, que en español quiere decir Interfaz de Programación de Aplicaciones, es un conjunto de funciones y procedimientos que permite integrar sistemas, permitiendo que sus funcionalidades puedan ser reutilizadas por otras aplicaciones o software.
API son las siglas en inglés para «application programming interface» o interfaz de programación de aplicaciones. Como su nombre lo indica, las API son interfaces compuestas por reglas y llamados en lenguaje computacional, que sirven para programar el funcionamiento de una aplicación determinada dentro de un software.

1. Pasar de async away a promesas:

2. Agregar la ruta de Delete o put:

3. Crear un nuevo componente de react en Nueva ruta:

4. Crear filtro:

6. como se conecta el back con el front:
A través de las acciones en donde se realizan las peticiones asíncronas al servidor con las direcciones http

18. Dónde se conectan back y front?
A través de las acciones en donde se realizan las peticiones asíncronas al servidor con las direcciones http

7. Paginado con next y previous:

8. Agregar nuevo dato para renderizar en las cartas del home:

9. Crear filtro nuevo con dato diferente:

10. Explicar ciclo/recorrido del post/get:

11. Cuáles son los métodos de sequelize
..cómo se busca un ID con sequelize?

12. Que es primary key?

13. Que es una orm?
Un **ORM** es un modelo **de** programación **que** permite mapear las estructuras **de** una base **de** datos relacional (SQL Server, Oracle, MySQL, etc.), sobre una estructura lógica **de** entidades con el objeto **de** simplificar y acelerar el desarrollo **de** nuestras aplicaciones.

Un ORM, por sus siglas al inglés: Object Relational Mappper, no es más que una pieza de software que nos permite interactuar con nuestra base de datos sin la necesidad de conocer SQL (El lenguaje de consultas). Todo esto utilizando el paradigma de programación orientada a objetos.
Los ORMs se encarga de traducir nuestra instrucción en el lenguaje de programación que estemos utilizando a una sentencia SQL que el gestor de base de datos pueda entender.

un ORM te permite convertir los datos de tus objectos en un formato correcto para poder guardar la información en una base de datos (**mapeo**) creándose una **base de datos virtual** donde los datos que se encuentran en nuestra aplicación, quedan **vinculados** a la base de datos (**persistencia**).
Así, los objetos o entidades de la *base de datos virtual* creada en nuestro *ORM* podrán ser manipulados por medio de algún lenguaje de nuestro interés según el tipo de *ORM* utilizado

Si alguna vez has programado alguna aplicación que se conecta a una base de datos, habrás podido comprobar lo laborioso que es transformar toda la información que recibes de la base datos, principalmente en tablas, en los objetos de tu aplicación y viceversa. A ésto se le denomina **mapeo**. Utilizando un ORM este mapeo será automático, es más, será **independiente** de la base de datos que estés utilizando en ese momento pudiendo cambiar de motor de base de datos según tus necesidades.

14. Que es express?

Express, es un entorno de trabajo para aplicaciones web para el programario Node.js, de código abierto y con licencia MIT. Se utiliza para desarrollar aplicaciones web y APIs **Express** permite definir rutas que corresponden a métodos HTTP **como** son peticiones GET, POST, PUT, DELETE.

Expressjs es un framework rápido, minimalista y flexible de Node.js. Permite crear APIs y aplicaciones web fácilmente, provee un conjunto de características como manejo de rutas (direccionamiento), archivos estáticos, uso de motor de plantillas, integración con bases de datos, manejo de errores, middlewares entre otras.

15. Diferencia entre promesas y async await?

Async Await: En la programación informática, el patrón async/await es una característica sintáctica de muchos lenguajes de programación que permite estructurar una función asíncrona sin bloqueo de forma similar a una función síncrona ordinaria.
La expresión **await** provoca que la ejecución de una función **async** sea pausada hasta que una Promise sea terminada o rechazada, y regresa a la ejecución de la función **async** después del término. Al regreso de la ejecución, el valor de la expresión **await** es la regresada por una promesa terminada.

las promesas y async await resuelven la asincronía de distinta forma. Con las promesas no sabemos cuándo se va a resolver y con async await forzamos una espera en la función.

5. Que es el thunk y como funciona:
Cuando usamos un Redux Store básico, lo único que puedes hacer son actualizaciones síncronas sencillas por medio de una acción. Pero si quieres trabajar con lógica asíncrona para interactuar con el Store, necesitarás algo más. Aquí es donde entra redux-thunk.

Redux-thunk es un middleware que te permite escribir creadores de acciones que retornan una función en vez de un objeto de acción típico. Entonces, el thunk puede ser usado para retrasar el envío de una acción hasta que se cumpla una línea de código asíncrona.

Redux Thunk se usa con mayor frecuencia para comunicarse de manera asíncrona con una API externa y, así, recuperar o guardar datos. Redux Thunk facilita el envío de acciones que siguen el ciclo de vida de una solicitud a una API externa.

Paso a paso de un proceso con Redux-Thunk

-Verificar la acción entrante:
Si es una acción regular, redux-thunk no hace nada y la acción es procesada por el reducer del Store.
-Si la acción es una función:

Redux-thunk la invoca y usa los métodos dispatch y getState y cualquier argumento adicional.
-Después que la función se ejecute:
El thunk envía la acción, la cual actualizará el estado como corresponde.

16. Que son los Middleware?
Un middleware actúa como un puente entre un sistema operativo o base de datos y aplicaciones.

17. Que es axios? Porque usaste axios/fetch?
La fetch API nos permite acceder a recursos de un servidor de manera asíncrona (peticiones Ajax). Este tipo de peticiones nos permiten realizar solicitudes HTTP sin necesidad de recargar toda la página. Para utilizar fetch API no es necesario usar ninguna librería.

Axios es una librería JavaScript que puede ejecutarse en el navegador y que nos permite hacer sencillas las operaciones como cliente HTTP, por lo que podremos configurar y realizar solicitudes a un servidor y recibiremos respuestas fáciles de procesar.

¿Qué es React?

React es una librería Javascript focalizada en el desarrollo de interfaces de usuario

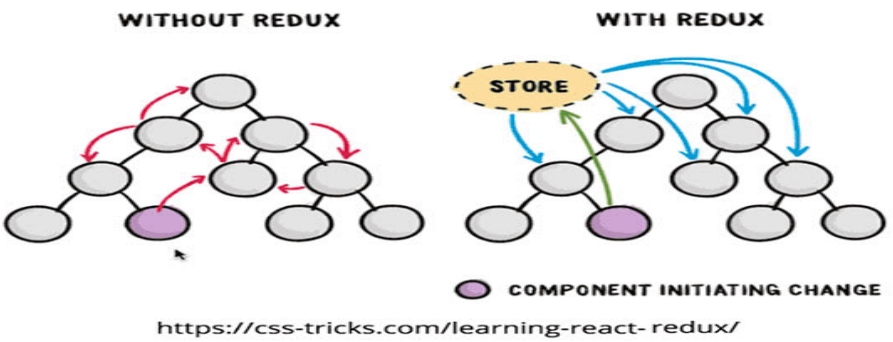
Redux es un contenedor predecible del estado **de** aplicaciones JavaScript. **Redux es** una excelente herramienta **para** manejar el estado **de** una aplicación. Sus principales beneficios son: Estado global e innmutable. Mayor control del estado de la aplicación y el flujo de datos. Arquitectura escalable de datos

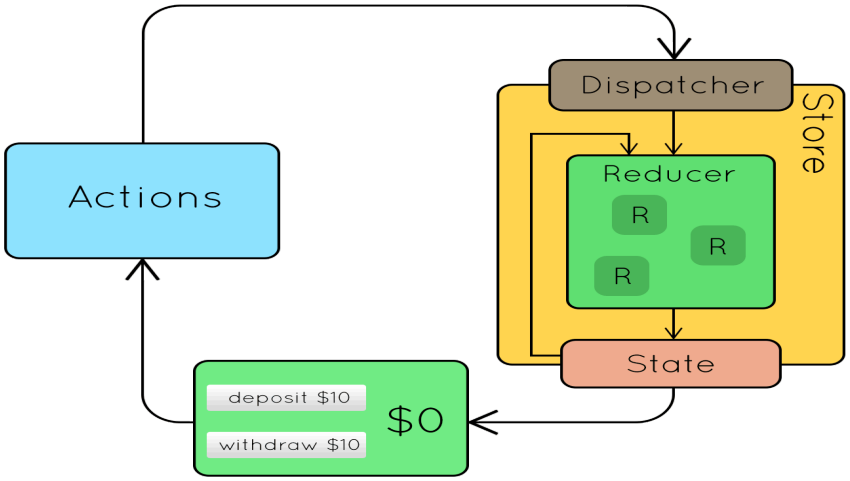
La principal ventaja de Redux es cómo administra los cambios de estado.

1. El Store como la única fuente de la verdad
2. El State es de solo lectura
3. Los cambios al State pueden hacerse únicamente a través de acciones (actions) y funciones puras (reducers)

De esta forma, se logra centralizar el estado de la aplicación y por lo tanto unificar el lugar para realizar cambios. Esto hace el desarrollo muchísimo más simple:

Todo el estado de tu aplicación esta almacenado en un único árbol dentro de un único store. La única forma de cambiar el árbol de estado es emitiendo una acción, un objeto describiendo que ocurrió.





20. Ciclo de vida de un componente?

El ciclo de vida se puede dividir en *3 fases*, el **montado**, **actualización** y **desmontado** del componente. Estas fases a su vez se dividen en varios métodos que puede tener el componente.

Montado: La primera fase ocurre solo una vez por componente cuando este se crea y monta

Actualización: Esa fase puede ocurrir múltiples veces (o incluso ninguna), sucede cuando algún dato del componente (ya sea una *propiedad*, un *estado* o el *contexto*) se modifica y por lo tanto requiere que la UI se vuelva a generar para representar ese cambio de datos.

Desmontado: Esta última fase consiste en un solo método que se ejecuta antes de que un componente se elimine (desmonte) de la UI de nuestra aplicación.

state representan los valores renderizados, es decir, lo que hay actualmente en la pantalla.

¿Cuándo se renderiza un componente en React?

Render. Todo **componente** de **React**, tiene un método Render que es el que **se** encarga de **renderizar** en el navegador el HTML correspondiente al **componente**. Este método **se**

llama automáticamente cuando **se** crea un **componente** y cuando el estado del **componente se** actualiza

27. Para qué sirven los hooks? Un hook de React que se usa para simular las funciones de ciclo de vida. Igual mejor ir agregando en otro espacio las respuestas
Los **Hooks** son funciones que te permiten “enganchar” el estado de **React** y el ciclo de vida desde componentes de función. Los **hooks** no **funcionan** dentro de las clases — te permiten usar **React** sin clases.

21. Que es useSelector?
useSelector es un Hook que nos permite extraer datos del store de Redux utilizando una función selectora
Se llamará al selector con todo el estado de la tienda Redux como su único argumento. El selector se ejecutará cada vez que se represente el componente de la función.

- El selector puede devolver cualquier valor como resultado, no solo un objeto. El valor de retorno del selector se utilizará como valor de retorno del useSelector().
- Cuando se envía una acción, useSelector()se hará una comparación de referencia del valor del resultado del selector anterior y el valor del resultado actual. Si son diferentes, el componente se verá obligado a volver a renderizarse. Si son iguales, el componente no se volverá a renderizar.

22. Que es useEffect?
Al usar este Hook, le estamos indicando a React que el componente tiene que hacer algo después de renderizarse. React recordará la función que le hemos pasado (nos referiremos a ella como nuestro “efecto”), y la llamará más tarde después de actualizar el DOM.

Este hook normalmente es usado para la inicialización de variables, llamadas a APIs o para limpiar un componente antes de desmontarlo del DOM.
La llamada a useEffect acepta una función como argumento. Esta función se ejecuta por defecto cuando el componente se renderiza por primera vez, y después cada vez que el componente se actualice.

También es posible especificar **cuándo** se debe ejecutar esta función con un segundo argumento opcional que le podemos pasar.
Para ello basta con añadir un segundo parámetro a la función, con la lista de los elementos de los que depende. Si el valor de uno de estos elementos que hemos indicado cambia, la función se va a ejecutar con la siguiente actualización.

Otra posibilidad que nos permite este hook es la de especificar que se ejecute sólo una vez. Esto resulta muy útil si sólo queremos hacer una llamada AJAX para rellenar el estado de la aplicación.

23. Que es useState?
Para que un componente funcional tenga estado propio podemos hacer uso del hook useState.
UseState permite trabajar con estados locales
useState() es una función que crea internamente una variable donde podremos almacenar el estado de nuestro componente. Acepta un valor inicial para esa variable y devuelve un array con dos elementos, el valor de la variable y la función para modificarla.
Como el valor devuelto por la función es un array, podemos descomponerlo para acceder a sus elementos de manera individual.
Es importante saber que cuando llamamos a la función set de un useState(), se sobrescribe el contenido de la variable.
const [count, setCount] = useState(0);

24. A quienes reemplazan en los componentes de clase?

25. Explicar el paginado

26.Que hace redux
Redux es una librería JavaScript que emite actualizaciones de estado en respuesta a acciones, con la peculiaridad de realizar dichas modificaciones a través de objetos sencillos, que reciben el nombre de **acciones**, y no a través de cambios directos en el estado.

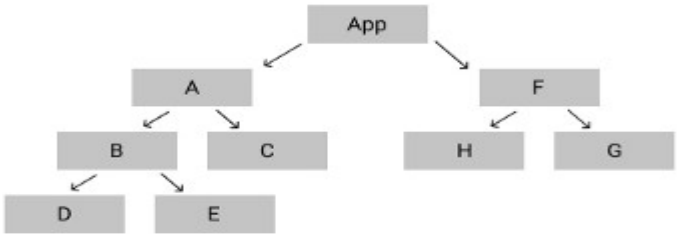
Redux es un patrón de arquitectura de datos que concede **manipular el estado de la aplicación de una forma predecible**. Está creado para disminuir el número de relaciones entre los componentes de la aplicación y conservar un flujo de datos asequible.

Redux reduce la interacción entre componentes, simplificando su complejidad y facilitando su depuración, mantenimiento, y al final, su escalabilidad.

19. Para que sirve react y redux?

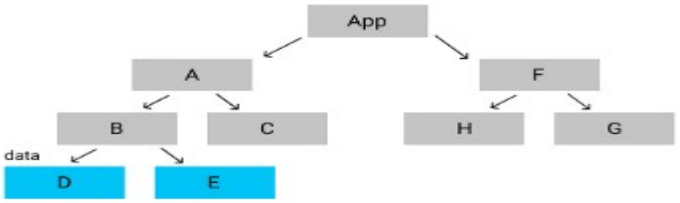
¿Por qué usar Redux?

Cuando creamos aplicaciones en React, éstas se organizan como una serie de componentes anidados, su naturaleza es funcional. En otras palabras reciben información a través de sus argumentos (props) y pasan la información a través de sus valores de retorno y a esto se le llama: *one-way binding*, los datos sólo se transmiten de los componentes a sus hijos:

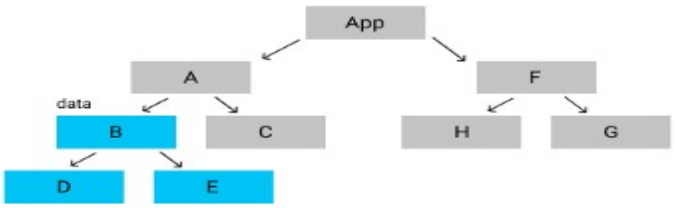


React sin Redux

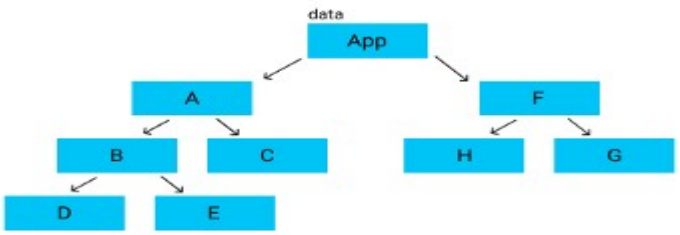
Entonces, imaginemos que en nuestro componente D tenemos un input que almacena cierta información en un estado. Esta información estará disponible para todo el componente D y sus hijos, ¿pero qué pasaria si necesitamos que esa información también esté presente en el componente E?



Una solución sería declarar el estado en el componente B (padre), de esa forma tendremos acceso a la información desde el componente D y E (hijos):

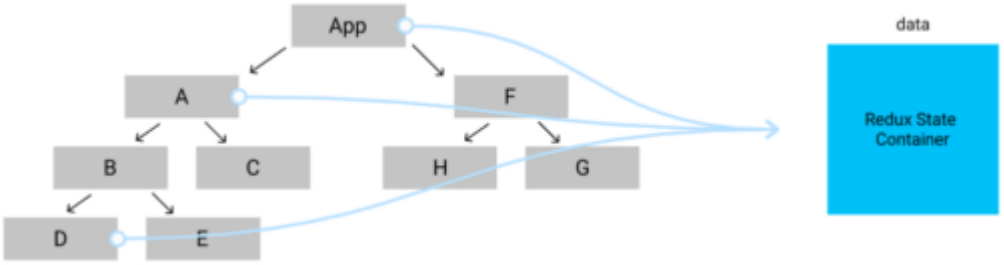


Siguiendo la misma lógica, ¿qué pasaría si necesito tener acceso a esa información en toda mi aplicación? Bueno, entonces declaro el estado en el padre de todos los componentes:



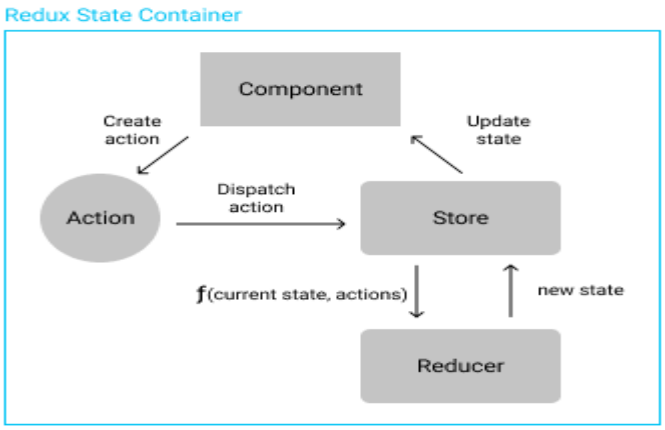
React con Redux

Entonces, necesitamos tener toda esa información en un solo lugar, para que pueda estar disponible en todos nuestros componentes, algo como nuestra única fuente de la verdad:



Patrón de arquitectura Redux

Más arriba mencionamos que Redux es un patrón de arquitectura y vimos a Redux como una caja azul que almacena los estados. Veamos ahora qué tenemos dentro de nuestra caja azul y cómo funciona:



1. Tenemos unComponent que va a emitir un Action, que pueden ser llamadas por algún evento: un click, por ejemplo.
2. El Action pasa al Store que es donde se guardan todos los estados.
3. El Store comunica al Reducer el estado actual y cual fue el Action que se ejecutó.
4. Luego, el Reducer retorna un nuevo estado modificado por la acción que se acaba de ejecutar.
5. El estado se actualiza en el Store.
6. Y el Store devuelve el nuevo estado modificado al componente.

links de referencias:
Hooks y uses:
https://desarrollofront.medium.com/entendiendo-los-hooks-de-react-c%C3%B3mo-usar-usestate-y-useeffect-en-nuestros-componentes-611b9e826dfa

redux thunk:
https://platzi.com/blog/como-funciona-redux-thunk/#:~:text=Redux%2Dthunk%20te%20permite%20escribir,una%20l%C3%ADnea%20de%20c%C3%B3digo%20as%C3%ADncrona.

https://www.digitalocean.com/community/tutorials/redux-redux-thunk-es

ORM y base de datos:
https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html

https://codigofacilito.com/articulos/orm-explicacion