Bienvenido a React Query

CÓDIGO

+

MÚSICA

+

SIMBA

GERARDO FERNÁNDEZ MORENO

Powerful asynchronous state management for React

Estado en el servidor

Está almacenado en un sitio externo de nuestra aplicación.

Necesitamos llamadas asíncronas para acceder a él.

Se puede acceder a él simultáneamente.

Las aplicaciones que lo consumen pueden tener una versión antigua.


PROBLEMAS,
PROBLEMAS EVERYWHERE
LIGHTYEAR

Estado asíncrono de una aplicación

Rendimiento

Peticiones dependientes

Caché

Estado de la llamada

Actualizaciones

Estado desactualizado

Gestión de errores

Ejemplo básico

Recuperar una lista de usuarios:

- Mostrar cargando mientras la llamada se completa.

- Gestionar el error.

- Acceder a los datos

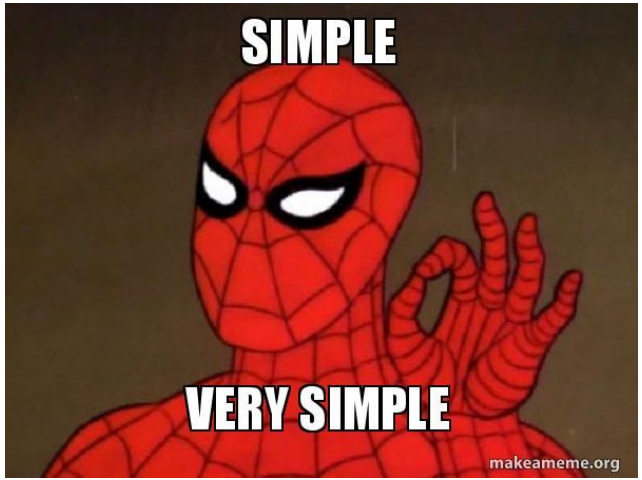
Estado de *useQuery*

Nos da información acerca de los **resultados**:

isLoading: la petición todavía no tiene resultados.

isError: la petición se resolvió con un error.

isSuccess: los resultados ya están disponibles.



FetchStatus

Nos da información acerca de la **llamada**:

fetching: la llamada está en proceso.

paused: la petición está en pausa.

idle: la petición no está haciendo nada.

Características

- Llamadas dependientes.
  - isEnabled
- Automatic “Window Focus Refetching”.
- Paginación mejorada.
  - *previousData*
  - Paginación infinita
- Reintentos.
- useIsFetching (indicador global).
- Llamadas en paralelo.
  - useQueries

Query keys

Query keys

- Las query keys permiten *cachear* el resultado de las llamadas.
- En React Query, son arrays con strings u objetos dentro.
  - Se serializan de forma determinista.
- Podemos emplearlas para pasar información a la llamada.

Caché

Caché

staleTime vs cacheTime

cacheTime: conserva los datos del “garbage collector” para permitir su reutilización durante el tiempo establecido.

staleTime: determina el tiempo durante el cual los datos se consideran “frescos”, es decir, no es necesario lanzar una llamada en segundo plano para refrescarlos.

Un ejemplo para cachearlo todo

Mutaciones

Mutaciones

- Son usadas para crear, editar o eliminar información en el servidor.
- Se gestionan mediante el hook *useMutation*.
- Admiten los callbacks *onSuccess*, *onError* y *onSettled*.
  - Combinados con las funciones *invalidateQueries* y *setQueryData* son una herramienta muy poderosa.

Y sí... React Query admite la API Suspense de React 18

¡Gracias!

https://youtube.com/c/latteandcode

https://www.linkedin.com/in/gerardofernandezmoreno/

https://latteandcode.substack.com/