KOIBANX

EJERCICIO DE CÓDIGO

Descripción del problema

a. Se debe construir una barra de búsqueda para una tabla que muestra datos de comercios.
b. La tabla es la siguiente:

ID	Comercio	CUIT	Concepto 1	Concepto 2	Concepto 3	Concepto 4	Concepto 5	Concepto 6	Balance actual	Activo	Ultima venta

c. Tipos de datos:
   - ID: string
   - Comercio: string
   - CUIT: string
   - Concepto 1 al 6: number
   - Balance actual: number
   - Activo: boolean (1 o 0)
   - Última venta: Date (DD/MM/YYYY)
d. La primera fila indica cómo se llaman los campos.
e. La url base de la api que muestra los datos es: https://api.koibanx.com/stores
f. Esta api no existe realmente, es anecdótica. El ejercicio simula que su compañero backend está trabajando en ese endpoint.
g. La API trabaja con la siguiente especificación en cuanto a los parámetros: https://restdb.io/docs/querying-with-the-api#restdb. Tomar esto como guía.
h. La base de datos cuenta con información de **10 millones** de comercios.
i. No se requiere autenticación para el trabajo con la api.
j. La tabla recibe de la api siempre la siguiente respuesta (los valores son de ejemplo):

```
{
 data: [...], // data es un array de comercios
 page: 1,
 pages: 100,
 rowsPerPage: 10,
 total: 1000
}
```

Requerimientos

a. Se necesita una barra de búsqueda que hable con la api para poder filtrar la información de manera útil para el usuario y se la pase a la tabla.
b. El componente barra de búsqueda debe ser separado del componente tabla.
c. La tabla debe permitir ordenar por las columnas Comercios y Cuit.
d. Se necesita poder filtrar la información por ID y/o CUIT y/o Nombre de comercio. Ya sea porque es exacto como parecido a Ejemplo: "34" puede buscar al ID 34 o 340, al cuit 30-3454… o al comercio "pancho 34". (Tip: lo importante es cómo se forman los query params, la búsqueda la realiza la api, no el front).
e. Se necesita poder filtrar por activos o no activos
f. Se necesita poder combinar todos los filtros anteriores.
g. No hace falta construir ni traer la data, el código deberá simular que recibe datos como una especie de mock.
h. No se requiere que se recargue visualmente el ejercicio, con un diseño estándar alcanza. No forma parte de la evaluación lo lindo que se ve, sino más bien la funcionalidad pero deberá ser lo más limpio posible. (si se desea puede utilizar una librería de UI)
i. Para poder ver las url generadas, las mismas deberán imprimirse por consola.
j. Tener en cuenta que la tabla indica rows per page y current page.
k. Tener en cuenta que en el día de mañana la tabla puede ser utilizada para mostrar todas las transacciones de los comercios o datos adicionales en comercios.

Consideraciones

a. El ejercicio debe ser resuelto en React (se puede utilizar Create React App o similar)
b. El diseño de la solución es libre.
c. La tabla no puede ser un componente de una librería de UI.
d. Se puede emular el servicio de api, pero no es necesario en este ejercicio.
e. Podrá usar hooks si lo desea pero no es obligatorio.
f. Podrá usar Redux o similar pero no es necesario.

Forma de entrega

a. El código se deberá subir a github o gitlab en un repositorio público.
b. Se deberá correr con npm i y luego npm start (Agregar las instrucciones en el Readme.md)
c. Si requiere algún proceso adicional para instalarlo, deberá explicarlo en el readme.
d. Si tomó alguna consideración o hipótesis para la confección deberá aclararlo en el readme.
e. Enviar un mail a tech@koibanx.com con asunto ‘Challenge’ y con el link del repositorio.

Por dudas o consultas, escribir a tech@koibanx.com