Stechs challenge

El objetivo de este documento a diferencia del readme es centrarse en cuestiones más funcionales y técnicas del proyecto, no solo en cómo levantar las aplicaciones.

Implemente docker para facilitar la ejecución de los servicios, de igual forma el readme del repo especifica el paso a paso en caso de querer ejecutar los servicios sin docker.

La lógica utilizada para resolver la problemática del challenge.

1. Genero una búsqueda a la base de datos, si obtengo registros significa que el **vendor** existe, si la no obtengo registros directamente significa que la búsqueda es sobre un **vendor** inexistente.
2. En caso de tener registros con respecto a la busqueda mencionada anteriormente, paso a leer el archivo JSON.
3. Luego valido si existe el vendor en los ítems del JSON, en caso de que si existan me quedo con estos.
4. Luego aplico álgebra relacional en la función ***leftOuterJoin*** donde pasó como registros left a los que obtuve como resultado en el punto 1, y como registros right a los registros resultantes del punto 3. Al no tener ids por donde comparar paso campos de comparación tanto para los registros left como para los registros right.
5. El resultado del punto 4 son los registros del punto 1 que no existen en el los registros del punto 3.

Esto se puede ver en el controller de ambos endpoins (backend/src/controllers/modems.ts).

Funcionalidades de Frontend web.

- Un buscador con los siguientes resultados de búsqueda posible:
    - Si la búsqueda no existe en el json ni en la bd, tira el siguiente mensaje:

  "No se encontraron fabricantes en la base de datos con respecto a la búsqueda realizada."
    - Si la búsqueda tiene registros encontrados en la bd que no se encontraron en el JSON los muestra en grilla.
    - Si la búsqueda encuentra en BD todos los registros que tambien existen en el JSON la búsqueda arroja el siguiente mensaje:
      "El fabricante existe, pero no hay modelos desconocidos."

- La grilla tiene las siguientes funcionalidades:
    - Ordenamiento por cada una de las columnas que muestra
    - Selección múltiple
    - Selección total
    - Paginado
    - Si hay uno o más registros seleccionados se muestra un botón de AGREGAR registros.

- El botón de “agregar” agrega al JSON manteniendo y respetando su formato original, nuevos objetos con los siguientes atributos que figuran en la grilla: Modelo, fabricante y versión que corresponden a la siguiente estructura del json:

```
{
    "vendor": "",
    "name": "",
    "soft": ""
}
```

Funcionalidades de Backend (servicio api).

Cuenta con dos endpoint

- **api/modems/:*vendor***: utiliza el verbo GET donde recibe el parámetro vendor, que se utiliza como filtro.
  El endPoint puede devolver lo siguiente:
  - Status 200: Si encontró registros en la BD que no están en el JSON. Devuelve un array de registros.
  - Status 304: Si se hace una petición idéntica a la previa. Devuelve un array de registros.
  - Status 204: Si no existe en la BD el *vendor*(fabricante) que se ingresó como parámetro.
  - Status 500: Si ocurre algún error en el servicio mientras ocurre la solicitud.
- **api/modems:** utiliza el verbo POST donde recibe como body un array de registros que deben ser insertados en el JSON.
  El endPoint puede devolver lo siguiente:
  - Status 201: Si la inserción de los registros al JSON fue exitosa. También devuelve un array de registros actualizado, para evitar una nueva búsqueda del usuario, y poder proveer información actualizada a la grilla sin tener que hacer una nueva búsqueda.
  - Status 500: Si ocurre algún error en el servicio mientras ocurre la solicitud.

Especificación técnica frontend.

- Utilice Reactjs como framework de desarrollo, y typeScript por encima del lenguaje javaScript.
- Se usaron Interfaces para definir objetos.
- Utilice el patron flux para la interacción de datos.
- Use reduxjs/toolkit para integrar el patrón flux y hacer las llamadas asíncronas a la api.
- Use componentes no clases.
- Agregue la librería materialUI para poder usar la DataGrid que provee, también Container y box ya que tenía la librería instalada. La idea es dejar el codigo propio lo más limpio y transparente posible.
- Como el JSON en su estructura no tenía un id, tuve que agregar lógica adicional en la selección del registro, para poder hacer su identificación inequívoca y luego poder

transformarlo en un ítem en formato json para cargarlo en el body del request de tipo post.
- Le agregue .docker/dockerfile, para poder dockerizar el repositorio.

Especificación técnica backend.

- Utilice la librería express para levantar el servicio de tipo api.
- Utilice la librería mysql como conectar de la api y el moton de BD.
- Agregue typeScript por encima del lenguaje javaScript.
- Todo tipo de lógica quedó en utils de forma desacoplada.
- Le agregue .docker/dockerfile, para poder dockerizar el repositorio.

Lo que quedó fuera:

- UnitTest.
- Refinamiento de nombre de variables y constantes.
- Desacople más abstracto del conector de base de datos en la api.
- Tener todos los objetos pertenecientes a Interfaces.

Como todo desarrollo de software siempre está sujeto a un evolutivo constante, este es un MVP de tiempo acotado.