RETO SOLDAI

URL: https://chatbot-soldai.herokuapp.com/

Instrucciones docker:

docker build --no-cache -t ms-soldai .
docker run -p 3000:3000 -d --name ms-soldai ms-soldai

Solución

Primero creé una cuenta de soldai, esto con el fin de poder revisar el panel y entender la forma en la que se usa el api. Una vez creada la cuenta, estuve revisando las diferentes secciones, con la ayuda de su tour fue muy sencillo entender el funcionamiento del bot, además de los ejemplos que tienen en las indicaciones del reto. Luego revise cómo funcionan los ejemplos de chats que tienen, esto es viendo que envían, que devuelven, mediante la herramienta consola de desarrollador de chrome, y realice algunas peticiones con postman.
Una vez que entendí el funcionamiento de soldai, revise la api de pokedex para entender su funcionamiento, revisar la estructura de su respuesta. De igual forma, estuve jugando un poco con la api con la ayuda de postman.

Una vez teniendo claro el funcionamiento de ambas apis, así como de sus respuestas y del reto que se plantea, cree un proyecto en nodejs con la estructura que se indica más abajo. Ademas me di a la tarea de diseñar el endpoint, para saber que necesitaba recibir de parte de soldai cuando se hace match con un elemento, y del cliente (el chat), y al final diseñe la respuesta que se devolvería desde el endpoint.

Debido a que la api de pokedex necesita de un número en su path, para consultar los datos del pokémon, necesitaba recibir ese dato de parte de soldai, cuando un elemento con el nombre del pokémon haga match en la pregunta y cómo debía saber que se está preguntando, si es una habilidad, peso, etc. tuve que añadir en la respuesta la característica y esto separarlo mediante un patrón que sea fácil para separar en el backend, por lo que quedó como: número::característica.

Al desarrollar la lógica, cuando se obtiene la respuesta de parte de soldai, solamente es necesario separar la respuesta en dos: número, el cual se toma para realizar la petición y la característica, que sirve para parsear la respuesta y obtener los datos necesarios del json que devuelve la api de pokedex.

Al final mediante vuejs, se desarrolla el frontend. Debido al tiempo de desarrollo, se decide usar un componente existente para el chat, con lo cual solamente se adapta a la respuesta de la api desarrollada.
Al final se realizaron unas pruebas y se ha subido a Heroku para probar.

*Nota: Al realizar las pruebas, detecté que cuando se añade un elemento solamente deja poner un sinónimo, esto es, que no se pueden repetir sinónimos. Sin embargo, cuando se evalúa o aprueban las respuestas, si la respuesta no detectó el elemento (me estuvo pasando seguido con bulbasaur) si le daba añadir sinónimo (en este caso bulbasaur) me lo añadia de nuevo, y al entrar a ver el elemento me aparecían repetidos (bulbasaur).

Estructura

****__test__****: contiene los tests que prueban las funcionalidades de la capa de servicios, y los de integración de todos los componentes para la api.

****config****: contiene un archivo con la configuración necesaria para que el servicio pueda trabajar.

****api****: contiene los directorios con los módulos necesarios para que el servicio pueda funcionar via http.

****api.server****: tiene la configuración para expressjs.

****api.routes****: tiene la configuración de las rutas con sus respectivos controladores.

****api.controllers****: tiene las funciones que se le pasan a las rutas, y las funciones tienen los parámetros necesarios para trabajar con expressjs.

****api.services****: tiene la lógica para crear las urls cortas y realizar las consultas necesarias. Esta capa se añade por si en algún momento se hace necesario implementar otro canal como grpc, rpc, etc. Mantiene los controladores que dependen de expressjs limpios.

****api.utils****: contiene algunas funciones de ayuda.

****Dockerfile****: contiene las instrucciones para generar una imagen docker.

API

> GET /v1/query?question=de que tipo charmander

Ejemplo respuesta:

```
{
  "request": 1538325670728,
  "url": "http://localhost:3000/v1/query?question=de que tipo charmander",
  "data": {
     "message": "Es de tipo fire."
  },
  "code": 200
}
```