Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería Introducción a la programación y computación 2, 2do. Semestre 2023.

PROYECTO 3: Implementación de API

Carnet 202200041 – Daniel Eduardo Velásquez Avila

Resumen

Una empresa de tecnologías esta desarrollando una herramienta que sea capaz de analizar el contenido de redes sociales y establecer el sentimiento de los usuario

Para poder lograr esto se ha creado un servicio capaz de leer mensajes de twitter que pueden contener 2 elementos especiales, a partir de esto se nos dan ciertos parámetros que pueden ser tomados en cuenta al momento de hacer el análisis, entre ellas se encuentran las reglas de que cada mensaje puede contener menciones a uno más usuario, que puede o no contener hashtags, que estos 2 elementos mencionados anteriormente se pueden encontrar en cualquier lugar de la aplicación, los nombres de los usuarios únicamente pueden contener letras números y símbolos

La empresa también ha creado una estrategia para saber si un mensaje tiene sentimiento positivo, negativo, esta aplicación deberá de contener una base de datos en formato XML.

Palabras Clave

- Backend
- Frontend
- Función

Abstract

A technology company is developing a tool that is able to analyze the content of social networks and establish user sentiment.

In order to achieve this it has created a service capable of reading twitter messages that may contain 2 special elements, from this we are given certain parameters that can be taken into account when making the analysis, among them are the rules that each message may contain mentions to one more user, which may or may not contain hashtags, that these 2 elements mentioned above can be found anywhere in the application, the names of users can only contain letters numbers and symbols.

The company has also created a strategy to know if a message has positive or negative sentiment, this application must contain a database in XML format.

KeyWords

- *Backend*
- *Frontend*
- *Function*

Introducción

El Proyecto consta de realizar un software que detecte un numero indeterminado de mensajes que son enviados por medio de un archivo de tipo xml, todos estos datos ya sean de un diccionario o de un archivo de entrada que contiene mensajes deben de ser almacenados en la base de datos, se cuentan con 2 bases de datos que son archivos xml, el primero es especial para los mensajes y el segundo es especial para el diccionario de mensajes.

Para lo anterior descrito se hace uso de diccionarios y listas para su manipulación para mayor facilidad, esta se manejará en diferentes clases de la aplicación, en varias ocasiones se hizo uso de listas dentro de diccionarios, pero también se podía optar por utilizar diccionarios dentro de diccionarios, de este segundo método puede se que pudiera haber manipulado de manera más sencilla la aplicación.

Para las lectura de los archivos y identificar cada uno de los elemento se hizo uso de un analizador que ira detectando los elementos admitidos en la aplicación, al no usar expresiones regulares como tal, si un carácter no encajaba o no se identificaba para el formato únicamente se iba a leer el siguiente carácter, palabra o numero.

Desarrollo del tema

Al iniciar el desarrollo del el proyecto se opto por iniciar por el backend haciendo uso de la aplicación postman para la prueba de funcionamiento del sistema, como primer paso se tenia que configurar los archivos que se utilizarían como bases de datos, ya que dependiendo de la manera en que se utilizaran iba a varias como se hicieran las primeras funcionalidades, entre estas funcionalidades mencionadas se encuentran el añadir archivos hacia la base de datos de mensajes y de diccionarios, para el primer caso que seria la base de datos de mensajes al no saber como realizar el guardado, se opto por primero hacer la verificación de si el archivo estaba vacio y en caso de que no se añadieran los datos, esto se realizo haciendo concatenación de la estructura de un xml hacia una variable que se tenia guardada es ese apartado, una vez que se tenían en variables la estructura de un xml, si el archivo tenia datos y no estaba vacío entonces se obtenia únicamente los datos dentro de las etiquetas mensajes, ignorando completamente esas etiquetas, una vez se tuviera la informacion anterior y la información del archivo entrante esta se concatenaba como se menciona anteriormente en una variable, al finalizar se escribía el archivo en la misma dirección que el original y con el mismo nombre, de esta manera nos encargábamos de que no tuviéramos ningún problema al momento de buscar la base de datos, para el diccionario se opto por realizarlo de manera diferente debido a ciertos errores que se podían tener al momento de realizar la carga, al leer la documentación oficial de xml se encontró que hay una manera de añadir nuevos elementos adentro de un formato xml ya realizado, para este caso no tiene que obtener los archivos ya guardados anteriormente al momento de hacer la carga.

En el primer endpoint del backend se utilizo un método post por lo cual se requeria hacer una verificación de si el request era de método post o método get, esto como una manera de evitar problemas en cuanto a funcionamiento, una vez que se realiza esto se creara una variable, esta será de

tipo form, anteriormente se había realizado de una manera en la cual el request era con un archivo, pero para realizar el archivo de salida de una manera mas fácil se opto por realizarlo de esta segunda manera, una vez que se verifico que el nombre del parámetro coincide con el de postman y se tiene una conexión exitosa entre los 2 componentes se manda a llamar una clase de otro archivo dentro del backend, una vez llamado esta clase se manda a llamar también a una función “añadir” esta función como su nombre lo indica es la que se encarga de añadir los datos entrantes hacia nuestra base de datos, una vez que han sido añadidos se crea una variable que servirá como un archivo temporal, esto se realizo de esta manera para no tener que guardar cada uno de los archivos de salida que se generaba, para esto se especifica la ruta exacta del archivo y en este caso se le añade los datos que fueron obtenidos de la primera función, estos datos son de formato xml y en lugar de mandar un mensaje de tipo json como “Mensaje ingresado de manera éxita” se devolverá un archivo que contiene un pequeño resumen de que se ingreso hacia la base de datos, esto en postman se muestra en la consola que tiene por defecto, pero en el frontend se mostrara por medio de “{{}}” que corresponde a la notación jinja que es la que nos ayuda al momento de querer realizar ciertas funciones dentro de un html ya que este tipo de archivo como tal no tiene forma de hacer esto de manera nativa, en la mayoría de casos se usa un api para facilitar la manipulación de datos o otros componentes como se requieran.

Para el siguiente endPoint al su funcionamiento al parecerse bastante al primero una forma bastante similar de realizarlo, se utilizo el método post haciendo la verificación de los parámetros correspondientes, se manda a llamar la clase de un archivo para posteriormente utilizar la función que tiene almacenada en su interior, esta función al igual que la anterior se mando el contenido del archivo en lugar de mandar el archivo completo como se tenia en mente en la primera forma de realizarlo, se manda a llamar una función de añadir a diccionario, en esta función se ira iterando en cada elemento de el string recibido, al ser un string y no un archivo se tiene que usar una función diferente a la que se utiliza comúnmente que es la “parser”, ya de esta manera nos permitiría poder utilizar las funciones como los son findall sin ningún problema, al ir iterando cada uno de los elementos que debería tener el archivo de entrada por cada mensaje y fecha se llamaran funciones las cuales se encararan de separa inicialmente la fecha dentro de la etiqueta <FECHA> ya que inicialmente esta contiene hora y ubicación y no únicamente la fecha en formato día, mes y año, la segunda función se encarga de verificar en que apartados de el mensaje se cuenta con un símbolo de arroba o con un símbolo numeral, al encontrar estos elementos como inicialmente se nos indico que podía contener el nombre se hace verificación que mientras el carácter analizado este entre los que son aceptado siga iterando, al encontrar otro elemento se detendrá indicándonos que hay un elemento diferente al de un usuario, a partir de en que momento salió del ciclo while y en que momento comenzó luego de encontrar el símbolo de arroba o de numeral podemos determinar cual será el usuario o también el hashtag correspondiente, estos se guardan dentro de un diccionario, dentro de una lista, ya que como se nos especifica en el enunciado este puede contener varios usuarios o hashtags o podría no

contenerlos por lo cual al realizarlo de esta manera evitamos que el programa sufra de un error, este apartado sin ningún problema se pudo haber cambiado a un diccionario, ya que se puede almacenar diccionarios dentro de diccionarios, pero al estar familiarizado de mayor manera con las listas, se prefirió hacer uso de las mismas.
Se realizo también la comprobación de si se encontraban palabras positivas o negativas dentro del mensaje, esto se realizo creando un diccionario antes de empezar a iterar el mensaje, este diccionario cuenta con las palabras positivas y negativas del diccionario, así que cuando encuentra un texto, verifica si no se encuentra en el diccionario mencionado anteriormente en caso de que si se agrega a un contado el sentimiento y en caso que la palabra no coincida se omite y se sigue iterando con los demás elementos.

Para la función de limpiar datos en el caso del mensaje el archivo se queda completamente vacio, y el archivo de los diccionarios solamente se eliminan los elementos que se encuentran en ciertas etiquetas, esto varia por la manera en la cual realizamos el añadir datos a el archivo, este endpoint se realizo con un método post.

Para el funcionamiento de los endpoints de devolver menciones, hashtags, o sentimientos se tiene un funcionamiento bastante similar, primero se manda a llamar una clase para posteriormente mandar a llamar una función en la misma, se especifica dentro de variables el primer y segundo rango que se va a tener para la búsqueda, luego de verificar que coinciden las variables con las keys dentro de el postman se pasa a realizar la función obtener datos, esta función se almacena en una variable ya que al mandar a llamarla devolverá un diccionario que contiene fecha, hashtag, menciones y sentimientos para las fechas correspondientes, luego de obtener las funciones se manda a llamar otra clase y se mandan de parámetro el diccionario, el primer y segundo rango esto para poder filtrar, es necesario para la manipulación de datos que las fechas cambien de tipo, ya que así las podremos evaluar con los operadores “<””>”, esto nos facilitara como trabajamos esto
Al finalizar las función, si se encuentran coincidencias entre las fechas se devolverá la fecha y el dato que necesitamos, ya se mención, hashtag o sentimiento, los demás endpoints se realizan de una manera similar.

Al momento de vincular el backend con el frontend se realizaron ciertas modificaciones, al igual que se tuvo que leer documentación y las formas en las que se trabaja con flask, a diferencia de la api django que fue usada para el frontend si se trabajan más herramientas propias de la api con flask, lo cual hace que el avance del proyecto en ese apartado al igual que siempre al usar una librería nueva sea de una forma más lenta a comparación del ritmo que se llevaba anteriormente.

Conclusiones

Al momento de utilizar una librería nueva, es buena idea que se lea la documentación que esta tiene, ya que así se nos hará más sencillo el saber sobre su funcionamiento y como enfocarlo hacia nuestro programa.

En este proyecto la lectura de la documentación fue de gran ayuda al momento de generar las gráficas o usar diferentes librerías nuevas, al igual que con la manipulación de los archivos XML.

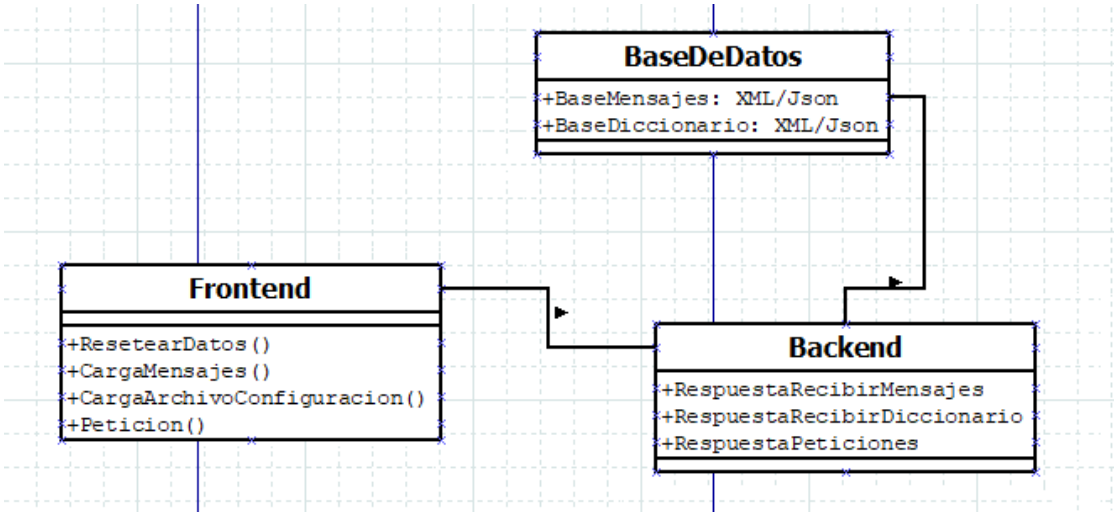
¿Cuáles son las ventajas y desventajas de utilizar listas anidadas en diccionarios en comparación con utilizar diccionarios anidados dentro de diccionarios?

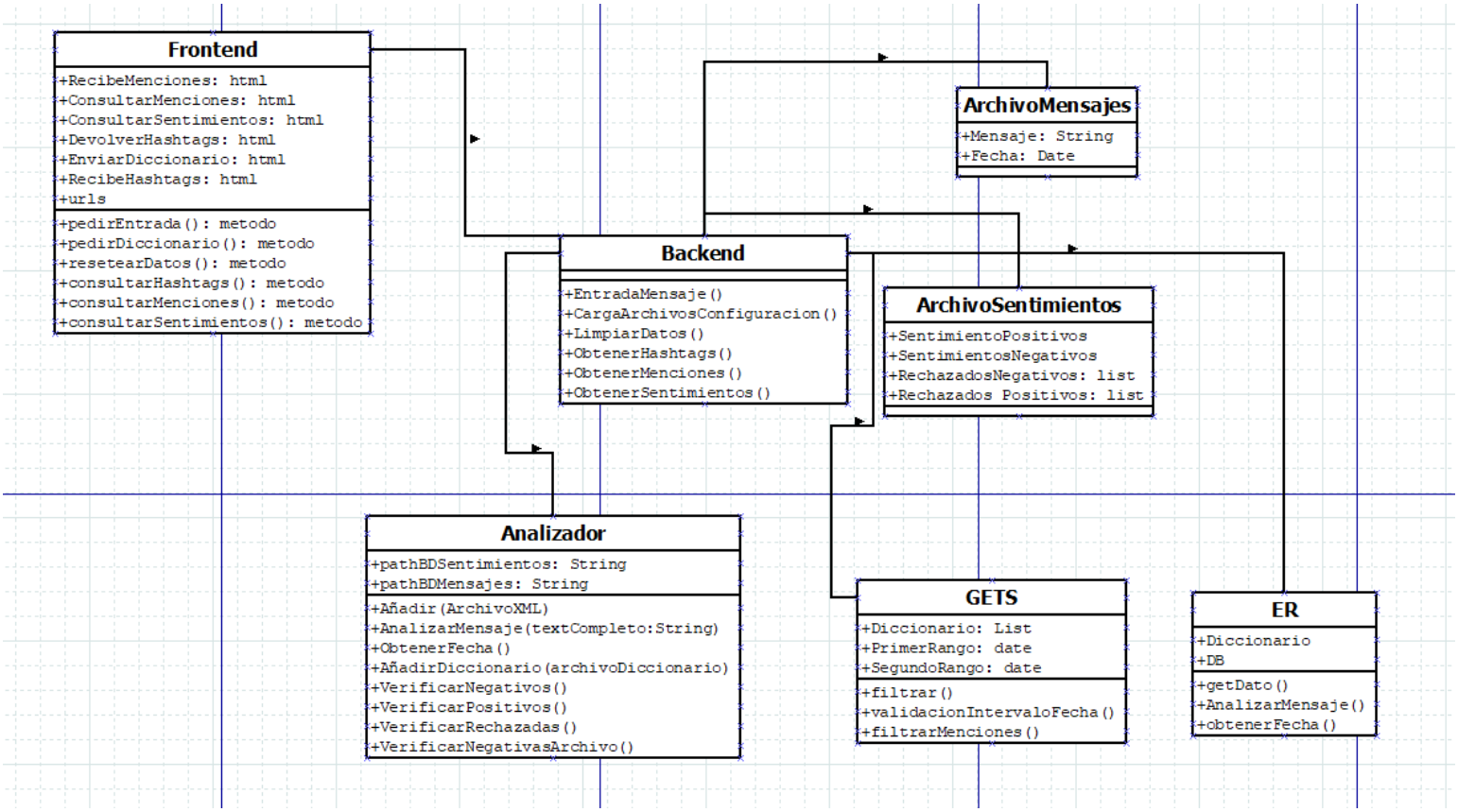
¿Es mejor utilizar una lista o un diccionario?

Referencias bibliográficas

Luis Joyanes Aguilar, (2008), Fundamentos de programación, algoritmos, estructuras de datos y objetos,
McGraw-Hill

Apéndices:


BaseDeDatos
+BaseMensajes: XML/Json
+BaseDiccionario: XML/Json
Frontend
+ResetearDatos()
+CargaMensajes()
+CargaArchivoConfiguracion()
+Peticion()
Backend
+RespuestaRecibirMensajes
+RespuestaRecibirDiccionario
+RespuestaPeticiones


Frontend
+RecibeMenciones: html
+ConsultarMenciones: html
+ConsultarSentimientos: html
+DevolverHashtags: html
+EnviarDiccionario: html
+RecibeHashtags: html
+urls
+pedirEntrada(): metodo
+pedirDiccionario(): metodo
+resetearDatos(): metodo
+consultarHashtags(): metodo
+consultarMenciones(): metodo
+consultarSentimientos(): metodo
ArchivoMensajes
+Mensaje: String
+Fecha: Date
Backend
+EntradaMensaje()
+CargaArchivosConfiguracion()
+LimpiarDatos()
+ObtenerHashtags()
+ObtenerMenciones()
+ObtenerSentimientos()
ArchivoSentimientos
+SentimientoPositivos
+SentimientosNegativos
+RechazadosNegativos: list
+Rechazados Positivos: list
Analizador
+pathBDSentimientos: String
+pathBDMensajes: String
+Añadir(ArchivoXML)
+AnalizarMensaje(textCompleto:String)
+ObtenerFecha()
+AñadirDiccionario(archivoDiccionario)
+VerificarNegativos()
+VerificarPositivos()
+VerificarRechazadas()
+VerificarNegativasArchivo()
GETS
+Diccionario: List
+PrimerRango: date
+SegundoRango: date
+filtrar()
+validacionIntervaloFecha()
+filtrarMenciones()
ER
+Diccionario
+DB
+getDato()
+AnalizarMensaje()
+obtenerFecha()