PROYECTO 3 IPC2 " API "

201901844 – Oswaldo Antonio Choc Cuteres

Resumen

La empresa Tecnologías Chapinas, S.A. desarrolló una api que es capaz de analizar contenido de redes sociales y establecer el sentimiento de los usuarios respecto a los mensajes emitidos en la red social.

Para lograr este fin, la empresa Tecnologías Chapinas, S.A. ha creado un servicio capaz de leer mensajes de Twitter que en esencia son textos que pueden contener 2 elementos especiales:

Una referencia a otro usuario de la red utilizando el símbolo @ y el nombre del usuario.

Un texto entre símbolo # para definir un tema o hashtag para el mensaje.

La Facultad de Ingeniería implemento el diseño de la API con objetivo principal de cumplir con las necesidades requeridas y así mismo para el uso adecuado del cliente.

Palabras clave

Api, usuario, hashtag, servicio, cliente, redes sociales.

Abstract

The company Tecnologías Chapinas, S.A. developed an API that is capable of analyzing social network content and establishing user sentiment regarding messages broadcast on the social network.

To achieve this goal, the company Tecnologías Chapinas, S.A. has created a service capable of reading Twitter messages that are essentially texts that can contain 2 special elements:

A reference to another user on the network using the @ symbol and the user's name user.

A text between # symbol to define a topic or hashtag for the message.

The Faculty of Engineering implements the design of the API with the main objective of meeting the required needs and also for the appropriate use of the client.

Keywords

Api, user, hashtag, service, client, social networks.

Introducción

Es importante atender el trabajo del frontend y el backend con una API pues es una de las dinámicas más encontradas en el trabajo con programas de código, conexión con APIs y desarrollo de aplicaciones. Ambos conceptos revelan características importantes acerca del lenguaje computacional y cómo trabajar con él. Por lo tanto, a continuación, se mencionará cómo es el trabajo frontend y backend en esta API.

Tomando las características de los requerimientos para esta API que necesita la empresa de Tecnologías Chapinas, lo cual requiere de una herramienta capas de poder analizar mensajes de Twitter y poder procesar la información de sentimientos encontrados en dichos mensajes, toda esta información deberá ser procesada por la aplicación mediante un formato de archivo XML, para poder procesar cada instrucción y descifrar los sentimientos que hay en el mismo, este archivo de entrada es muy importante para el funcionamiento del programa, ya que de lo contrario no podrá mostrar información al cliente.

Partiendo de esta idea principal se describirá el proceso necesario para llegar a ese resultado, dividiendo el proceso en dos partes, lo que es la funcionalidad del backend y la funcionalidad el frontend.

Desarrollo del programa

Cuando estamos usando una API nos encontramos con varios elementos dentro del programa para desarrollar una aplicación que conecte con ella. Este trabajo se divide en dos partes muy conocidas dentro del sector de la programación. Estos son el campo fronted y el campo backend. Este trabajo mancomunado es importante, pues ambos campos del programa se juntan en un solo sitio. Por lo tanto, el programa tendrá dos caras: la de Django en el cliente y la de Flask en el campo del servidor para conectar frontend con backend.

Backend

Es una parte trascendental dentro del desarrollo de esta API se encarga de todos los procesos necesarios para que la parte del cliente se ejecute de forma correcta. El backend ejecuta procesos y funciones que no son visibles para el usuario final. Como se describió anteriormente el backend se realizó en la plataforma de Flask con el siguiente dominio http://127.0.0.1:3050 donde según lo solicitado tiene las siguientes peticiones o funciones:

- Grabar mensaje
- Eliminar datos
- Peticiones
- Mostrar datos

Grabar mensaje:

Es una petición dentro de el backend que tiene como dominio el siguiente “/grabarMensajes” con un método tipo “POST” el cual su función principal es poder leer los archivos de entrada que le cliente va a mandar desde el servicio, ya sea desde el frontend o utilizar postman.

Esta petición recibe lo que es un archivo XML el cual almacena los sentimientos que se quiera ingresar y también se puede mandar los mensajes que el cliente dese procesar, esta información al momento de recibirlo se guarda en la base de datos lo cual para esta API se

utilizó una base de datos en archivos con formato XML, lo cual se almacena en la información del frontend para que se pueda acceder a ella cuando se deseé, todo este proceso se utilizaron métodos, funciones y clases para poder realizarlo, así mismo la lectura de los datos se implementó expresiones regulares para obtener las siguiente información:

- Fecha
- Hora
- Hashtags
- Usuarios

Los archivos de entrada deben de tener el siguiente formato:

```
<?xml version="1.0"?>
<MENSAJES>
 <MENSAJE>
  <FECHA> Guatemala, 15/01/2023 15:25 hrs. </FECHA>
  <TEXTO> Bienvenido a USAC @estudiante01 @estudiante02,
       es un gusto que seas parte de esta institución
       #bienvenidaUSAC#
  </TEXTO>
 </MENSAJE>
 ...
</MENSAJES>
```

Figura 1. Archivo de entrada mensajes

Fuente: Enunciado proyecto

```
<?xml version="1.0"?>
<diccionario>
  <sentimientos_positivos>
    <palabra> bueno </palabra>
    <palabra> excelente </palabra>
    <palabra> cool </palabra>
    <palabra> satisfecho </palabra>
    ...
  </sentimientos_positivos>
  <sentimientos_negativos>
    <palabra> malo </palabra>
    <palabra> pésimo </palabra>
    <palabra> triste </palabra>
    <palabra> molesto </palabra>
    <palabra> decepcionado </palabra>
    <palabra> enojo </palabra>
    ...
  </sentimientos_negativos>
</diccionario>
```

Figura 2. Archivo de entrada diccionario

Fuente: Enunciado proyecto

Al momento de leer estos archivos de entradas se generan automáticamente los archivos de salidas que son las configuraciones o resumen de la información que se acaba de ingresar. Que tiene el siguiente formato:

```
<?xml version="1.0"?>
<MENSAJES_RECIBIDOS>
 <TIEMPO>
   <FECHA> 15/01/2023 </FECHA>
   <MSJ_RECIBIDOS> 1 </MSJ_RECIBIDOS>
   <USR_MENCIONADOS> 2 </USR_MENCIONADOS>
   <HASH_INCLUIDOS> 1 </HASH_INCLUIDOS>
 </TIEMPO>
 ...
</MENSAJES_RECIBIDOS>
```

Figura 3. Archivo de salida mensajes

Fuente: Enunciado proyecto

```
<?xml version="1.0"?>
<CONFIG_RECIBIDA>
 <PALABRAS_POSITIVAS> 4 </PALABRAS_POSITIVAS>
 <PALABRAS_POSITIVAS_RECHAZADA> 0 </PALABRAS_POSITIVAS_RECHAZADA>
 <PALABRAS_NEGATIVAS> 6 </PALABRAS_NEGATIVAS>
 <PALABRAS_NEGATIVAS_RECHAZADA> 0 </PALABRAS_NEGATIVAS_RECHAZADA>
</CONFIG_RECIBIDA>
```

Figura 4. Archivo de salida diccionario

Fuente: Enunciado proyecto

En los archivos de salida se generan lo que es un resumen de los datos almacenados, el archivo de salida de mensajes muestra la fecha del mensaje y además muestra los mensajes recibidos en esa fecha, la cantidad de usuarios y hashtags que fueron mencionados.

En el archivo de salida del diccionario se muestran las palabras positivas, negativas, positivas rechazadas y negativas rechazadas las cuales se generan al momento de leer el diccionario y encontrar palabras positivas que ya están en el conjunto de palabras negativas lo cual por lo tanto se almacenan en el conjunto de palabras positivas rechazadas y así mismo pasa con las palabras negativas rechazadas.

Eliminar datos:

Esta petición tiene como dominio el siguiente "/limpiarDatos" con un método tipo "POST" el cual su función principal es poder eliminar los datos almacenados en la base de datos, ya que, si se desea usar la API desde cero, se recomienda eliminar los valores en la base de datos ya que por defecto al inicial el servicio se recolecta la información que hay almacenada anteriormente por el cliente.

Peticiones:

En petición se deriva en tres tipos diferentes los cuales son:

- Petición de hashtags
- Petición de menciones
- Petición de sentimientos

Sus dominios son los siguientes:

- / devolverHashtags
- / devolverMenciones
- / devolverSentimientos

Las peticiones tienen un método tipo "POST" cada una de estas peticiones se les pide como información esencial una fecha de inicio y una fecha de fin que deseé realizar la consulta, una vez ingresado estos datos el servicio podrá ser capas de poder mostrar la información solicitada en ese rango de fechas y poder mostrársela al cliente de la siguiente forma.

Petición Hashtags ejemplo:

Fecha: 01/10/2023
1. #usac# : 20 mensajes
2. #fiusac#: 19 mensajes
3. #bienvenido#: 16 mensajes
4. …

Fecha: 02/10/2023
1. @user03 : 22 mensajes
2. @user01 : 16 mensajes
3. …

Así mismo con las demás peticiones se mostrará el mismo formato con los datos indicados.

Mostrar datos

En petición se deriva en siete tipos diferentes los cuales son:

- Mostrar Mensajes
- Mostrar palabras positivas
- Mostrar palabras negativas
- Mostrar palabras positivas rechazadas
- Mostrar palabras negativas rechazadas
- Mostrar configuración mensajes
- Mostrar configuración diccionario

Sus dominios son los siguientes:

- / DB_Mensajes
- / DB_PalabrasPositivas
- / DB_PalabrasNegativas
- / DB_PalabrasPositivasRechazadas
- / DB_PalabrasNegativasRechazadas
- / ResumenMensajes
- / ResumenConfig

Las peticiones tienen un método tipo “GET” las cuales se encargan de obtener la información almacenada en la base de datos, por medio de un archivo en formato XML.

Frontend

Esla parte visible de la API con la que los usuarios pueden interactuar directamente. Es la parte donde el cliente puede realizar las peticiones anteriormente descritas, como se explicó anteriormente el frontend se desarrolló en Django el cual tiene un acceso directo a la parte del backend, todo esto se realizó bajo el siguiente dominio http://127.0.0.1:8000/ .

Lógica del programa

Sabiendo que esta aplicación de software fue diseñada en dos partes lo que es el backend y el frontend se relación según la siguiente lógica

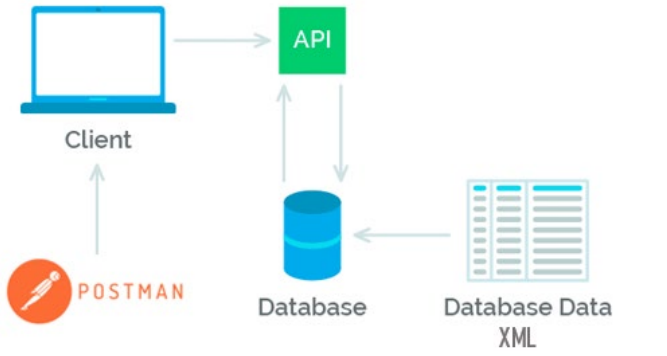


Figura 5. Lógica api
Fuente: Enunciado proyecto

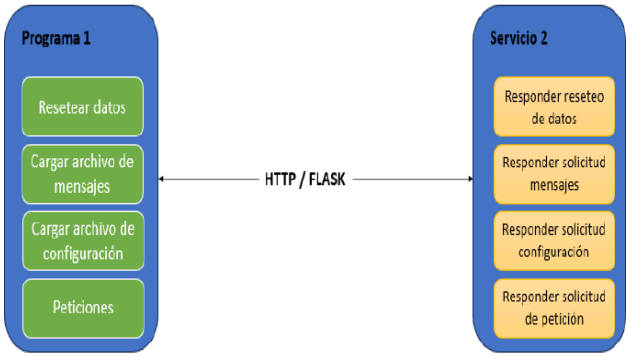


Figura 6. Relación
Fuente: Enunciado proyecto

Como se logra observar en las figuras 5 y 6, la lógica que se tiene es la relación que hay entre el programa 1 que es la parte de Django y el programa dos que es la parte de Flask que a su vez tiene una relación con la base de datos que son formatos XML, cada petición debe de

responder a lo solicitado según el dominio del backend. Lo cual tiene su lógica de la siguiente forma:

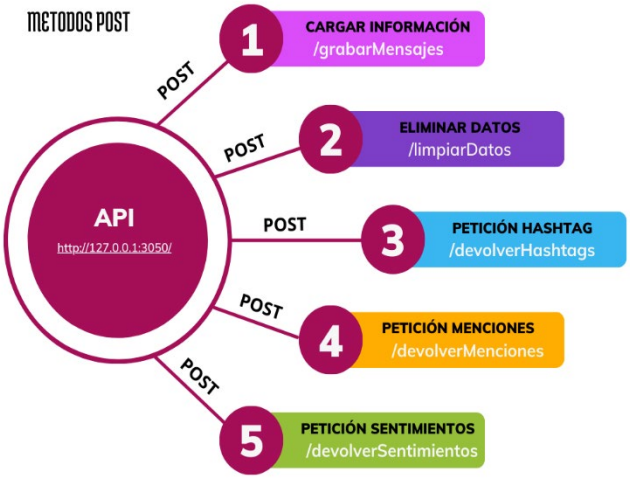


Figura 7. Arquitectura Backend
Fuente: Elaboración propia

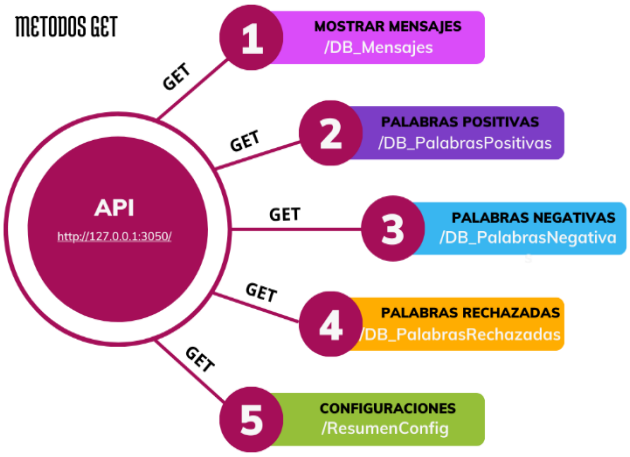


Figura 8. Arquitectura del backend
Fuente: Elaboración propia

Funcionamiento de programa

El funcionamiento del programa se base prácticamente en la parte del frontend y como el cliente puede realizar las peticiones. La parte del visual del programa es la siguiente:

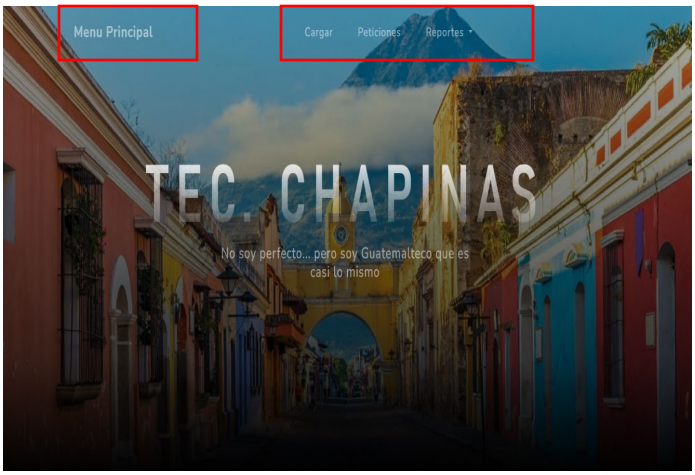


Figura 9. Menú principal
Fuente: Elaboración propia

Esta es la parte que el cliente mira y tiene las siguientes opciones:

- Menu principal
- Cargar
- Peticiones
- Reportes

Menu principal

Esta opción te devuelve al menú principal que tiene el siguiente dominio http://127.0.0.1:8000/ donde se encuentra una breve descripción de la página y las operaciones que se pueden realizar.

Cargar

Esta opción se mostrará un formulario para poder seleccionar y subir los archivos que se desea procesar, como se describió en el backend solamente recibirá archivos en formato XML y tiene el siguiente dominio

http://127.0.0.1:8000/cargarArchivo/ . Una vez ingresado los archivos se mostrará un mensaje que se realizó correctamente la carga.

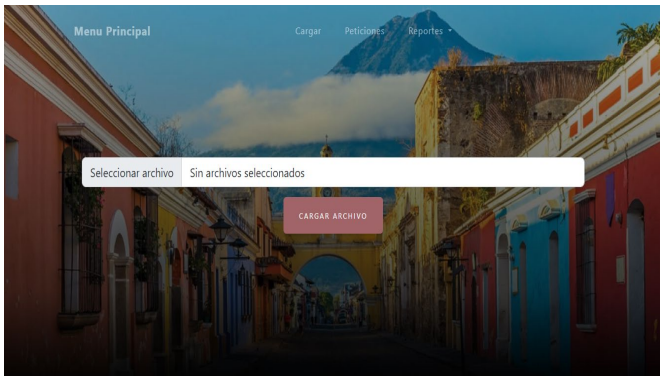


Figura 10. Cargar archivo
Fuente: Elaboración propia

Peticiones

Esta opción tiene el siguiente dominio http://127.0.0.1:8000/peticiones/ y se mostrará un formulario para poder ingresar la fecha de inicio y la fecha de fin que se desea consultar, posterior a eso se tiene que seleccionar en que tipo de petición desea realizar, una vez ingresada la fecha que tiene el siguiente formato dd/mm/yyyy se podrá visualizar en la parte inferior los resultado y las gráficas de que uno de los resultados.

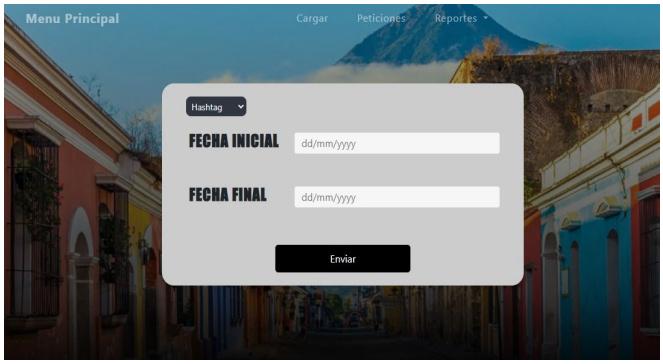


Figura 11. Peticiones
Fuente: Elaboración propia

Reportes

Esta opción tiene el siguiente submenú:

- Reporte configuración
- Reporte diccionario
- Eliminar datos

Al seleccionar reporte configuración se podrá visualizar un archivo pdf donde se encuentra la configuración que desea visualizar.

La opción eliminar datos como se describió anteriormente en el backend tiene la función principal de eliminar los datos almacenados en los archivos XML que sirven como base de datos para las peticiones, una vez ingresado se mostrará un mensaje que fueron eliminados los datos correctamente.

Conclusiones

El término API que significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

Este proyecto dio la oportunidad de poder trabajar con una aplicación que es un tema que actualmente se esucha mucho en el área de la tecnología la mayoría de las empresas o paginas que consultamos todos los días viene derivado de una API y el trabajar con dos framework como Flask y Django es una gran oportunidad de poder familirizarse con toda esta rama de aplicaciones web e ir conociendo un poco de las grandes cosas que se pueden llegar a realizar, también fue un gran conocimiento poder aprender un poco de como funciona una base de datos a pesar que solo se trabajaron con archivos.

Referencias bibliográficas

Bienvenido a Flask — Documentación de Flask (2.3.x). (s.f.). Bienvenido a Flask — Documentación de Flask (2.3.x). https://flask-es.readthedocs.io/

Rosita Wachenchauzer, Margarita Manterola, Maximiliano Curia, Marcos Medrano, & Nicolás Paez. (2006). *Algoritmos de Programación con Python*. Diseño y programación web (libros, tutoriales y vídeos sobre HTML, CSS, JavaScript, PHP). https://uniwebsidad.com/libros/algoritmos-python

Anexos

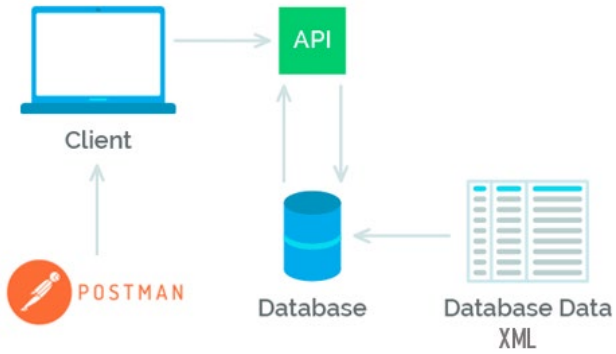
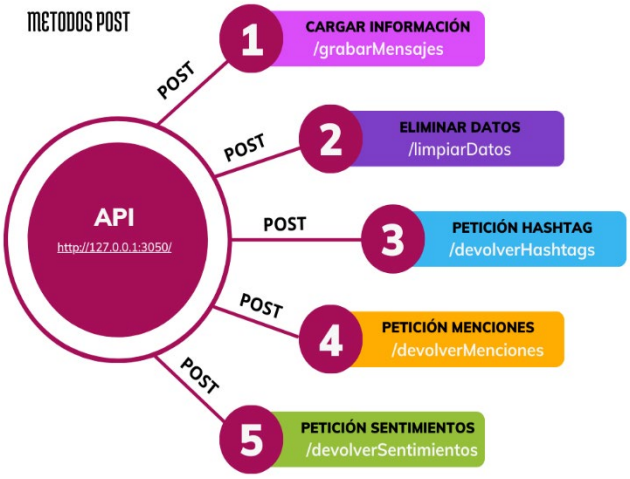


Figura 12, Lógica del programa
Fuente: Elaboración propia



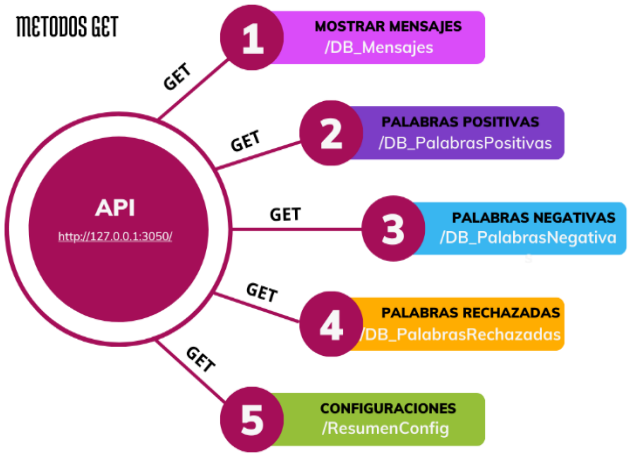


Figura 13, Arquitectura backend
Fuente: Elaboración propia

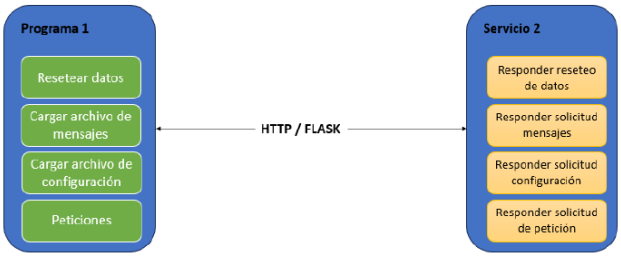


Figura 14, Relación
Fuente: Enunciado proyecto