Proyecto 3

202201117 – Miguel Ricardo Galicia Urrutia

Resumen

Un proyecto de backend con Flask y un frontend con Django puede ser utilizado para almacenar mensajes importados desde un archivo XML cargado desde el frontend. El backend de Flask se encargaría de procesar el archivo XML y almacenar los mensajes en una base de datos. El frontend de Django proporcionaría una interfaz para cargar el archivo XML y visualizar los mensajes almacenados. El backend de Flask utilizaría una biblioteca de terceros para leer el archivo XML. Una vez que el archivo haya sido leído, el backend procesaría los mensajes y los almacenaría de manera temporal usando una lista enlazada simple.
El frontend de Django proporcionaría una interfaz para cargar los archivos XML de mensajes y la configuracion. El usuario podría seleccionar el archivo XML desde su computadora y cargarlo en la aplicación. Una vez que el archivo haya sido cargado, el frontend mostraría una lista de los mensajes almacenados.

Palabras clave

- Flask
- Django
- Archivo XML
- Frontend
- Backend

Abstract

A backend project with Flask and a frontend with Django can be used to store messages imported from an XML file loaded from the frontend. Flask's backend would take care of processing the XML file and storing the messages in a database. The Django frontend would provide an interface for loading the XML file and viewing the stored messages. Flask's backend would use a third-party library to read the XML file. Once the file has been read, the backend would process the messages and store them temporarily using a simple linked list.
The Django frontend would provide an interface for loading the XML files of messages and configuration. The user would be able to select the XML file from their computer and upload it to the application. Once the file has been uploaded, the frontend would display a list of the stored messages.

Keywords

- Flask
- Django
- XML File
- Frontend
- Backend

Introducción

El análisis de mensajes en redes sociales es una herramienta poderosa para obtener información sobre los temas que se están discutiendo en las redes sociales, así como para identificar tendencias y opiniones. El programa que se presenta en este artículo es un backend para un sistema de análisis de mensajes en redes sociales. El programa está escrito en Python y utiliza el framework Flask.

El programa permite cargar mensajes desde un archivo XML y almacenarlos en una base de datos. El programa también permite obtener los mensajes almacenados, así como hashtags y menciones. El programa puede ser utilizado para analizar mensajes de cualquier red social, siempre que los mensajes estén en formato XML.

El programa puede ser utilizado por empresas, organizaciones gubernamentales y cualquier persona que esté interesada en analizar mensajes de redes sociales. El programa puede ser utilizado para obtener información sobre los temas que se están discutiendo en las redes sociales, así como para identificar tendencias y opiniones.

Desarrollo del tema

Este programa de backend con Flask está diseñado para almacenar mensajes importados desde un archivo XML. El programa consta de cinco rutas principales:

/grabar_mensaje: Esta ruta permite cargar un archivo XML que contiene los mensajes a almacenar. El programa procesará el archivo XML y almacenará los mensajes en una lista.
/get_mensaje: Esta ruta devuelve la lista de mensajes almacenados.
/get_configuracion: Esta ruta permite cargar un archivo XML que contiene la configuración del programa, como las palabras positivas y negativas que se utilizarán para realizar el análisis de sentimientos.
/incializar: Esta ruta reinicia el programa, vaciando la lista de mensajes y la lista de palabras positivas y negativas.
/get_hashtags: Esta ruta devuelve una lista de los hashtags más utilizados en los mensajes almacenados.
/get_menciones: Esta ruta devuelve una lista de las menciones a usuarios más utilizadas en los mensajes almacenados.
Para utilizar el programa, primero se debe cargar un archivo XML que contiene los mensajes a almacenar utilizando la ruta /grabar_mensaje. Una vez que los mensajes hayan sido almacenados, se puede obtener la lista de mensajes utilizando la ruta /get_mensaje. También se puede cargar un archivo XML que contiene la configuración del programa utilizando la ruta /get_configuracion. Para reiniciar el programa, se puede utilizar la ruta /incializar. Para obtener una lista de los hashtags más utilizados en los mensajes almacenados, se puede utilizar la ruta /get_hashtags. Para obtener una lista de las menciones a usuarios más utilizadas en los mensajes almacenados, se puede utilizar la ruta /get_menciones.

Este programa se puede utilizar para almacenar y analizar mensajes de redes sociales, mensajes de correo electrónico o cualquier otro tipo de mensaje que se pueda almacenar en formato XML. El programa es fácil de utilizar y puede ser

personalizado para satisfacer las necesidades específicas de los usuarios.

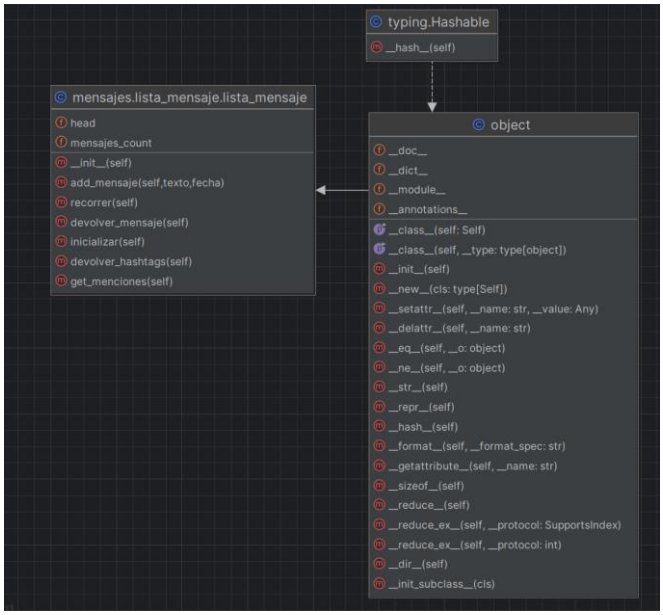


Figura 1. Diagrama de clases para el almacenamiento de mensajes

Fuente: elaboración propia, o citar al autor, año y página.

Django es un framework de desarrollo web de alto nivel escrito en Python. Es un framework muy popular, utilizado por empresas de todo el mundo para crear aplicaciones web.

En proyectos de backend para almacenar mensajes importados desde un archivo XML, Django se puede utilizar para:

Desarrollar la interfaz de usuario (UI) para cargar el archivo XML y visualizar los mensajes almacenados.

Implementar el análisis de sentimientos para determinar el sentimiento positivo o negativo de los mensajes almacenados.

Generar informes sobre los mensajes almacenados, como los hashtags más utilizados o las menciones a usuarios más utilizadas.

Desarrollo de la UI

Django proporciona una serie de herramientas y bibliotecas para el desarrollo de interfaces de usuario. Estas herramientas y bibliotecas se pueden utilizar para crear interfaces de usuario modernas y fáciles de usar.

En el caso de un proyecto de backend para almacenar mensajes importados desde un archivo XML, la UI podría incluir los siguientes elementos:

Un formulario para cargar el archivo XML.

Una lista de los mensajes almacenados.

Una sección para visualizar los detalles de cada mensaje.

Una sección para realizar el análisis de sentimientos.

Una sección para generar informes.

Implementación del análisis de sentimientos

Django no proporciona ninguna funcionalidad integrada para el análisis de sentimientos. Sin embargo, existen una serie de bibliotecas de terceros que se pueden utilizar para implementar el análisis de sentimientos en Django.

Una de las bibliotecas de terceros más populares para el análisis de sentimientos es TextBlob. TextBlob es una biblioteca de Python que proporciona una serie de funciones para el análisis de texto, incluido el análisis de sentimientos.

La función myform_view() es una vista de Django que permite a los usuarios cargar un archivo y enviar datos a un backend de Flask. Cuando se envía el formulario, la función primero verifica si se ha seleccionado un archivo. Si no se ha seleccionado un archivo, la función devuelve una respuesta JSON con un mensaje de error.

Si se ha seleccionado un archivo, la función envía una solicitud POST al backend de Flask con el archivo adjunto. La función utiliza la biblioteca requests para enviar la solicitud. La función también envía los datos del formulario como datos de solicitud.

Si la solicitud POST es exitosa, la función procesa la respuesta JSON del backend de Flask y la devuelve al usuario. Si la solicitud POST falla, la función devuelve una respuesta HTTP con el error.

La función myform_view() también utiliza la función render() de Django para renderizar la plantilla myform.html si el método de la solicitud no es POST.

La clase lista_mensaje tiene los siguientes métodos:

__init__(): Inicializa la lista de mensajes.
add_mensaje(): Agrega un mensaje a la lista.
recorrer(): Recorre la lista y muestra cada mensaje.
devolver_mensaje(): Devuelve una cadena con todos los mensajes de la lista.
inicializar(): Reinicia la lista.
devolver_hashtags(): Devuelve una lista con todos los hashtags de los mensajes de la lista.
get_menciones(): Devuelve una lista con todas las menciones de los mensajes de la lista.
Descripción de los métodos:

__init__(): Este método inicializa la lista de mensajes estableciendo el valor de la variable head a None y el valor de la variable mensajes_count a 0.
add_mensaje(): Este método agrega un mensaje a la lista. El mensaje se agrega al final de la lista.
recorrer(): Este método recorre la lista y muestra cada mensaje.
devolver_mensaje(): Este método devuelve una cadena con todos los mensajes de la lista.
inicializar(): Este método reinicia la lista estableciendo el valor de la variable head a None y el valor de la variable mensajes_count a 0.
devolver_hashtags(): Este método devuelve una lista con todos los hashtags de los mensajes de la lista. El método utiliza una expresión regular para identificar los hashtags.
get_menciones(): Este método devuelve una lista con todas las menciones de los mensajes de la lista. El método utiliza una expresión regular para identificar las menciones.

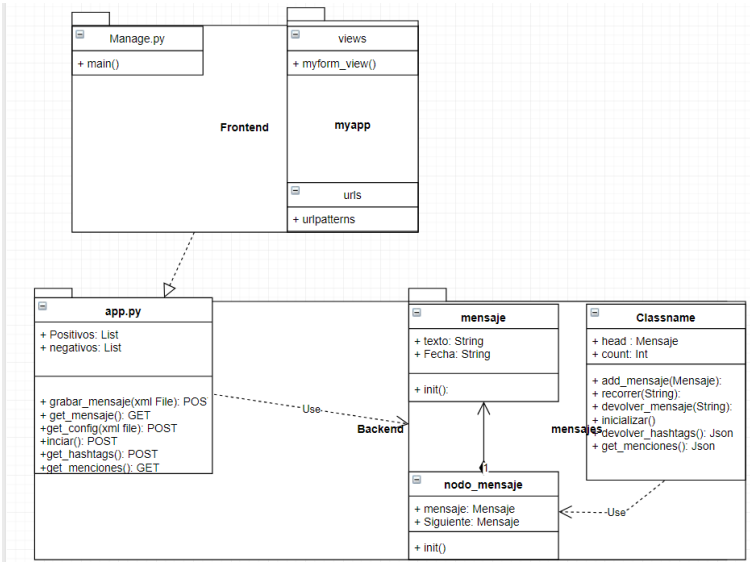
Referencias

Peñate, E. (2023). *IPC2_2S23_CLASSES.* Obtenido de Github: https://github.com/Erwin14k/IPC2_2S23_CLASSES

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.

Anexos

Diagrama de clases



Pagina web