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.

OPTIMIZACIÓN Y VISUALIZACIÓN DE DATOS EN DESARROLLO DE SOFTWARE: UN ENFOQUE INTEGRAL

202200174 – Andres Alejandro Quezada Cabrera

Resumen

La investigación propone el diseño de un sistema dividido en dos partes: un frontend web utilizando Django y un backend basado en una API con Flask. El frontend ofrece funcionalidades como resetear datos, cargar archivos XML de mensajes y configuraciones, consultar hashtags, menciones y sentimientos en mensajes, además de generar gráficas y reportes en PDF. Por otro lado, el backend, mediante la API, procesa datos del frontend y los almacena en archivos XML específicos. El sistema facilita la interacción con datos y estadísticas, permitiendo un análisis exhaustivo de mensajes en un rango de fechas dado.

Palabras clave
XML, frontend, backend, Flask, Django

The research proposes the design of a system divided into two parts: a web frontend using Django and a backend based on a Flask API. The frontend offers functionalities such as resetting data, loading XML files of messages and configurations, querying hashtags, mentions, and sentiment in messages, as well as generating graphs and PDF reports. On the other hand, the backend, through the API, processes data from the frontend and stores it in specific XML files. The system facilitates interaction with data and statistics, allowing for a comprehensive analysis of messages within a given date range.

Keywords
XML, frontend, backend, Flask, Django

Introducción

En esta investigación, se presenta un enfoque innovador para el diseño de un sistema web que consta de un frontend y un backend, utilizando las tecnologías Django y Flask, respectivamente. El frontend proporciona una interfaz amigable para interactuar con datos, permitiendo a los usuarios cargar archivos XML, consultar hashtags, menciones y sentimientos en mensajes, y generar informes gráficos en formato PDF. Por otro lado, el backend, a través de una API basada en Flask, procesa los datos del frontend y los almacena en archivos XML específicos. Este enfoque modular y eficiente facilita el análisis detallado de los mensajes en un período de tiempo determinado, ofreciendo una solución completa para el manejo y visualización de datos en aplicaciones web.

Desarrollo del tema

Importación de Librerías:
El script importa varias bibliotecas estándar de Python, como xml.etree.ElementTree, re, datetime, json, matplotlib.pyplot, io, xml.dom.minidom y os.
Clase Mensaje_DAO:

La clase Mensaje_DAO es el núcleo del script y contiene múltiples métodos para procesar mensajes y generar estadísticas.

Métodos Relevantes:
Inicialización y Procesamiento:

__init__(): Inicializa listas para mensajes, sentimientos positivos y negativos, y un diccionario para tildes.

procesar_mensaje(): Lee mensajes y configuraciones desde archivos XML, procesa los mensajes y los almacena en la lista self.mensajes.

procesar_old_config(): Procesa configuraciones antiguas y almacena palabras positivas y negativas en listas separadas.

Métodos para Estadísticas:

get_hash_date_range(fecha_inicio, fecha_final): Obtiene hashtags dentro de un rango de fechas y devuelve un JSON con recuento de hashtags.

get_menciones_date_range(fecha_inicio, fecha_final): Obtiene menciones dentro de un rango de fechas y devuelve un JSON con recuento de menciones.

get_sentimientos_date_range(fecha_inicio, fecha_final): Calcula la cantidad de mensajes con sentimientos positivos, negativos y neutros en un rango de fechas y devuelve un diccionario con los resultados.

graficar_hashtags(fecha_inicial, fecha_final), graficar_menciones(fecha_inicial, fecha_final), graficar_sentimientos(fecha_inicial, fecha_final): Generan gráficos de hashtags, menciones y sentimientos respectivamente en formato PDF.

Métodos para Contar Elementos en Mensajes:

contar_msj_recibidos(fecha), contar_user_mencionados(fecha), contar_hashtags(fecha): Cuentan mensajes recibidos, usuarios mencionados y hashtags en un día específico respectivamente.

resumen_mensajes(): Crea un resumen XML de mensajes con detalles sobre mensajes recibidos, usuarios mencionados y hashtags incluidos por fecha.
Métodos para Contar Palabras Positivas/Negativas:

contar_palabras_positivas(),
contar_palabras_negativas(): Cuentan palabras positivas y negativas en los mensajes procesados.
contar_palabras_positivas_rechazadas(),
contar_palabras_negativas_rechazadas(): Cuentan palabras positivas y negativas según configuraciones antiguas.
Métodos de Resumen Configuración:

resumen_config(): Crea un resumen XML de la configuración actual y antigua, mostrando palabras positivas y negativas.
Método para Resetear Datos:

resetear_datos(): Reinicia las listas y diccionarios, elimina archivos XML y maneja excepciones si los archivos no existen.
Notas Adicionales:
El código utiliza expresiones regulares para manipular cadenas y extraer información específica de los mensajes.
Los resultados se presentan en formato JSON y XML para análisis y visualización.

Tecnologías Utilizadas:
Flask: El código utiliza el framework Flask para construir una API web.
Bibliotecas Python: Se utilizan varias bibliotecas de Python, como os, shutil, datetime, Flask, Response para manipular archivos, fechas y HTTP responses.
Operaciones con Archivos: El código maneja la lectura y escritura de archivos XML para mensajes y configuraciones.

Funcionalidades Principales:
Carga y Almacenamiento de Archivos:

La API permite cargar archivos XML conteniendo mensajes y configuraciones.

Los archivos son guardados en el servidor para su posterior procesamiento.
Estadísticas de Mensajes:

Hashtags y Menciones: La API puede devolver estadísticas de hashtags y menciones dentro de un rango de fechas especificado.

Sentimientos: Proporciona estadísticas sobre mensajes con sentimientos positivos, negativos y neutros en un período de tiempo determinado.
Generación de Gráficos:

La API puede generar gráficos en formato PDF para visualizar el recuento de hashtags, menciones y sentimientos en un rango de fechas dado.
Resúmenes en Formato XML:

Proporciona resúmenes en formato XML sobre mensajes recibidos, usuarios mencionados, hashtags incluidos y configuraciones utilizadas.
Resetear Datos:

Ofrece una funcionalidad para resetear todos los datos, incluyendo mensajes y configuraciones, a su estado inicial.
CORS Habilitado:

CORS (Cross-Origin Resource Sharing) está habilitado para permitir solicitudes desde diferentes dominios.

Endpoints de la API:

/api/grabarMensajes/ (POST): Permite cargar archivos XML de mensajes y configuraciones al servidor.
/api/devolverHashtags/<fecha_inicial>/<fecha_final> (GET): Devuelve estadísticas de hashtags dentro del rango de fechas especificado.
/api/devolverMenciones/<fecha_inicial>/<fecha_final> (GET): Devuelve estadísticas de menciones dentro del rango de fechas especificado.
/api/devolverSentimientos/<fecha_inicial>/<fecha_final> (GET): Devuelve estadísticas de sentimientos dentro del rango de fechas especificado.

/api/graficarHashtags/<fecha_inicial>/<fecha_final> (GET): Genera y descarga un gráfico de hashtags en formato PDF.
/api/graficarMenciones/<fecha_inicial>/<fecha_final> (GET): Genera y descarga un gráfico de menciones en formato PDF.

/api/graficarSentimientos/<fecha_inicial>/<fecha_final> (GET): Genera y descarga un gráfico de sentimientos en formato PDF.

/api/resumenMensajes/ (GET): Devuelve un resumen en formato XML de mensajes recibidos, usuarios mencionados y hashtags incluidos.

/api/resumenConfig/ (GET): Devuelve un resumen en formato XML de las configuraciones actuales y antiguas.

/api/resetearDatos/ (POST): Permite resetear todos los datos almacenados en la aplicación.

Notas Adicionales:
El código utiliza el paradigma REST para las operaciones CRUD (Create, Read, Update, Delete) sobre los datos.
La aplicación maneja adecuadamente los errores de lectura y escritura de archivos, y proporciona respuestas HTTP apropiadas.

Tecnologías Utilizadas:
Django: El código utiliza el framework Django para construir la interfaz web.

Funcionalidades Principales:
Cargar Archivos:

La aplicación permite a los usuarios cargar archivos XML conteniendo mensajes y configuraciones.
Los archivos son enviados a la API Flask para su procesamiento.

Consultar Estadísticas:

Los usuarios pueden consultar estadísticas de hashtags, menciones y sentimientos para un rango de fechas específico.
Se envían solicitudes a la API Flask para obtener los datos requeridos.
Generar Gráficos:

Los usuarios pueden generar gráficos en formato PDF para visualizar datos de hashtags, menciones y sentimientos.
Se envían solicitudes a la API Flask para generar los gráficos.

Acceder a Resúmenes:
Los usuarios pueden acceder a resúmenes en formato XML sobre mensajes recibidos y configuraciones utilizadas.
Se envían solicitudes a la API Flask para obtener los resúmenes.

Resetear Datos:
Existe una funcionalidad para resetear todos los datos almacenados en la API Flask.
Se envía una solicitud POST a la API para llevar a cabo el reseteo.

Manejo de Archivos y Respuestas:
La aplicación maneja el envío y recepción de archivos, así como las respuestas de la API, de manera adecuada.

Páginas de Ayuda:
La aplicación incluye páginas de ayuda para orientar a los usuarios.
Endpoints de la API Utilizados:
/grabarMensajes/ (POST): Se utiliza para cargar archivos XML de mensajes y configuraciones.

/devolverHashtags/<fecha_inicial>/<fecha_final> (GET): Para obtener estadísticas de hashtags en un rango de fechas.
/devolverMenciones/<fecha_inicial>/<fecha_final> (GET): Para obtener estadísticas de menciones en un rango de fechas.

/devolverSentimientos/<fecha_inicial>/<fecha_final> (GET): Para obtener estadísticas de sentimientos en un rango de fechas.

/graficarHashtags/<fecha_inicial>/<fecha_final> (GET): Para generar un gráfico de hashtags en formato PDF.

/graficarMenciones/<fecha_inicial>/<fecha_final> (GET): Para generar un gráfico de menciones en formato PDF.

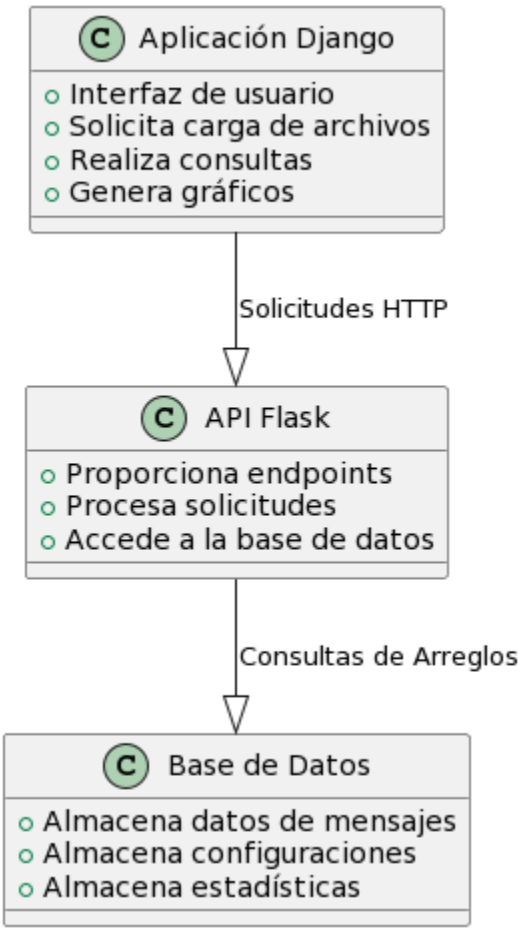
/graficarSentimientos/<fecha_inicial>/<fecha_final> (GET): Para generar un gráfico de sentimientos en formato PDF.

/resumenMensajes/ (GET): Para obtener un resumen en formato XML de mensajes recibidos y estadísticas relacionadas.

/resumenConfig/ (GET): Para obtener un resumen en formato XML de las configuraciones utilizadas.

/resetearDatos/ (POST): Para resetear todos los datos almacenados.

Diagrama



Referencias bibliográficas

"Clean Code: A Handbook of Agile Software Craftsmanship" - Robert C. Martin

"Design Patterns: Elements of Reusable Object-Oriented Software" - Erich Gamma

"The Pragmatic Programmer: Your Journey to Mastery" - Andrew Hunt, David Thomas