Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2022.

Aplicación web de Tecnologías Chapines S.A.

202010814 – Luis Enrique Garcia Gutierrez

Resumen

Se crearon las Api y endpoint utilizando Flask donde se llevara a cabo todo el proceso de la petición que haga el usuario desde el fronted. La petición será un xml que contiene los sentimientos positivos, negativos y datos de las empresas. Luego guardara los mensajes que contenga el xml. El usuario desde el fronted podrá solicitar ver un resumen de los mensajes ingresando una fecha o un rango de fecha. Entonces se buscaran los datos en el backend para mostrar el resultado en un xml al usuario.

Palabras clave

Flask, sentimientos, empresas, Django, archivos

Abstract

The Api and endpoint were created using Flask where the entire process of the request made by the user from the fronted will be carried out. The request will be an xml that contains the positive and negative feelings and data of the companies. Then it will save the messages that the xml contains. The user from the fronted can request to see a summary of the messages by entering a date or a date range. Then the data will be searched in the backend to show the result in an xml to the user.

Keywords

Flask, feelings, companies, Django, files

Introducción

Flask y Django son frameworks que se utilizan para crear aplicaciones web. Para esta aplicación se utilizó Flask para desarrollar el backend del programa, donde se llevara a cabo el almacenamiento de los datos del xml de entrada. Se utilizó Django para el desarrollo del fronted del programa, que es la parte externa donde se observara los componentes de la página web y donde el usuario podrá hacer sus peticiones, las cuales se mandaran a las API creadas en el backend con Flask

Desarrollo del tema

La aplicación web cuenta con las siguientes funciones:

- Cargar Archivo
- Peticiones
- Enviar
- Reset
- Ayuda

La aplicación web cuenta con 4 endpoints, que son los siguientes:

1. /ConsultarDatos
   Este endpoint recibirá un archivo xml, para leer las emociones negativas y positivas para guardarlas en un diccionario. También leerá las empresas con su respectivo servicio el cual tiene sus alias en otro diccionario. Luego leerá los mensajes que contenga el archivo xml, clasificara si el mensaje es positivo o negativo. También buscara si esta la empresa y le asignara si es positivo o negativo. Luego creara un archivo xml como respuesta el cual será mostrado en el fronted.

2. /ConsultarFecha
   Primero el usuario indicara la fecha que desea ver, entonces con este endpoint se buscara entre todos los mensajes guardados en la base de datos la fecha que coincida con la que el usuario haya ingresado. Luego leerá ese mensaje para determinar si es positivo o negativo, luego volverá a leer el mensaje y si encuentra la empresa le dará su clasificación al igual si encuentra el servicio o su alias. Entonces devuelve un resumen de los mensajes.

3. /ConsultarRangoFecha
   Primero el usuario indicara el rango de fechas que desea ver, entonces con este endpoint se leerán todos los mensajes que su fecha este dentro del rango establecido por el usuario. Luego se leerá cada mensaje para clasificarlo como positivo y negativo, también se leerá para ver si existe una empresa e indicar si es positivo o negativo. Entonces devuelve un resumen de los mensajes.

4. /ProcesarMensaje
   El usuario podrá ingresar un mensaje y con este endpoint se clasificara el mensaje como positivo, o negativo, también verifica que este el mensaje para clasificarlo como positivo o negativo. Luego devolverá un xml como respuesta con los datos del mensaje.

5. /Gráfica
   Se hará un reporte de todas las peticiones hechas de la clasificación por fecha y clasificación por rango de fechas. Se mostrara

los mensajes totales, positivos, negativos y mensajes neutros por empresa y servicio.

Para el funcionamiento y almacenamiento de los datos de entrada del archivo xml. Se hizo uso de las siguientes clases.

1. Sentimientos
   Esta clase tiene como atributos un diccionario con los sentimientos negativos y positivos encontrados en el archivo xml de entrada. Tendrá como método "Crear diccionario" y e "Get diccionario" el cual retorna el diccionario para poder guardarlo en la clase diccionarios.
2. Empresas
   Esta clase tiene como atributos el nombre de la empresa y un diccionario de servicios en el cual se guardaran los servicios que contenga la empresa. Y tendrá como métodos "Crear diccionario" que ira guardando las empresas del archivo de entrada y "Get empresa" el cual retornara la lista de empresas para guardarlo en la clase Diccionarios
3. Servicios
   Esta clase tiene como atributos el nombre del servicio y la lista de los alias que contenga. Tendrá los métodos "Crear diccionario" el cual crea un diccionario con el nombre del servicio y la lista de sus alias. Y también el método "Get diccionario" el cual retorna al diccionario creado para poder enviarlo a la clase Empresas.
4. Diccionarios
   Esta clase contiene los atributos empresas el cual es un diccionario y sentimientos el cual también es un diccionario. Esta clase contendrá todos los datos del xml de entrada. Con el cual se podrá ver los sentimientos y darle una clasificación a los mensajes.

Para el funcionamiento del backend y algunos endpoint se usaron los siguientes algoritmos:

- Algoritmo del funcionamiento cuando se mande un xml.
  1. Se mandara el xml que contiene los sentimientos positivos, sentimientos negativos, empresas y sus servicios y los mensajes
  2. Guardara los sentimientos negativos, sentimientos positivos y las empresas con sus servicios en diccionarios.
  3. Empezara a leer los mensajes.
  4. Almacenara los datos de los mensajes en una lista.
  5. Creará un xml de salida.

- Algoritmo para guardar los datos del xml entrada
  1. Recorre cada elemento de la raíz.
  2. Si encuentras el tag "diccionario" se inicializara la clase de sentimientos y empresas.
  3. Recorrera cada elemento del tag "diccionario"
  4. Si encuentra el tag "sentimientos positivos" se inicializa una lista de positivos.
  5. Recorrera cada elemento del tag "sentimientos positivos" y se ira creando un diccionario con la lista de positivos y el sentimiento positivo.

6. Si encuentra el tag “sentimientos negativos” se inicializara la lista de negativos.
7. Recorrera cada elemento de “sentimientos negativos” y se ira creando un diccionario con la lista de negativos y el sentimiento negativo.
8. Si al recorrer el tag de cada “diccionario” encuentra el tag “empresas analizar” recorrerá cada elemento del tag “empresas analizar”
9. Se inicializara la clase servicios.
10. Si encuentra el tag “empresa” recorrerá cada elemento del tag “empresa”
11. Si encuentra el tag “nombre” guardara el nombre de la empresa
12. Si encuentra el tag “servicio” inicializa la lista de servicios. Y leerá los servicios que tenga y se creara el diccionario de servicio. Y luego el de empresa
13. De último creara el diccionario general con todos los datos.

- Algoritmo para leer mensajes y guardar datos.
    1. Recorre cada elemento de la raíz “diccionario”
    2. Si encuentra un tag “lista mensajes” recorrerá cada sub elemento del tag “lista mensajes” para empezar a almacenar datos del mensaje
    3. Guarda el mensaje en una variable “mensaje”
    4. Leerá cada carácter del mensaje
    5. Si encuentra “:” y get_lugar es True empezara a guardar cada carácter en una variable para guardar el lugar del mensaje.
    6. Si encuentra un numero y get_fecha es True empezara a guardar cada carácter en una variable hasta que coincida con la expresión regular de la fecha y guardar la fecha del mensaje.
    7. Si encuentra “:” y get_usuario es True empezara a guardar cada carácter en una variable para guardar el usuario del mensaje
    8. Si encuentra “:” y get_red_social es True empezara a guardar cada carácter para guardar la red social del mensaje.
    9. Si get_mensaje es True guardara el mensaje del mensaje.
    10. Hará esto por cada mensaje que venga en el xml de entrada.

- Algoritmo para clasificar mensajes
    1. Leo el mensaje y comparo si cada palabra existe en el diccionario de sentimientos.
    2. Si hay un positivo se aumenta la cantidad de positivos
    3. Si hay un negativo se aumenta la cantidad de negativos.
    4. Comparo la cantidad de positivos y negativos.
    5. Si hay más positivos que negativos el mensaje será positivos
    6. Si hay más negativos que positivos el mensaje será negativo
    7. Si la cantidad de negativos es igual a la cantidad de positivos el mensaje será neutro
    8. Asigno la calificación del mensaje

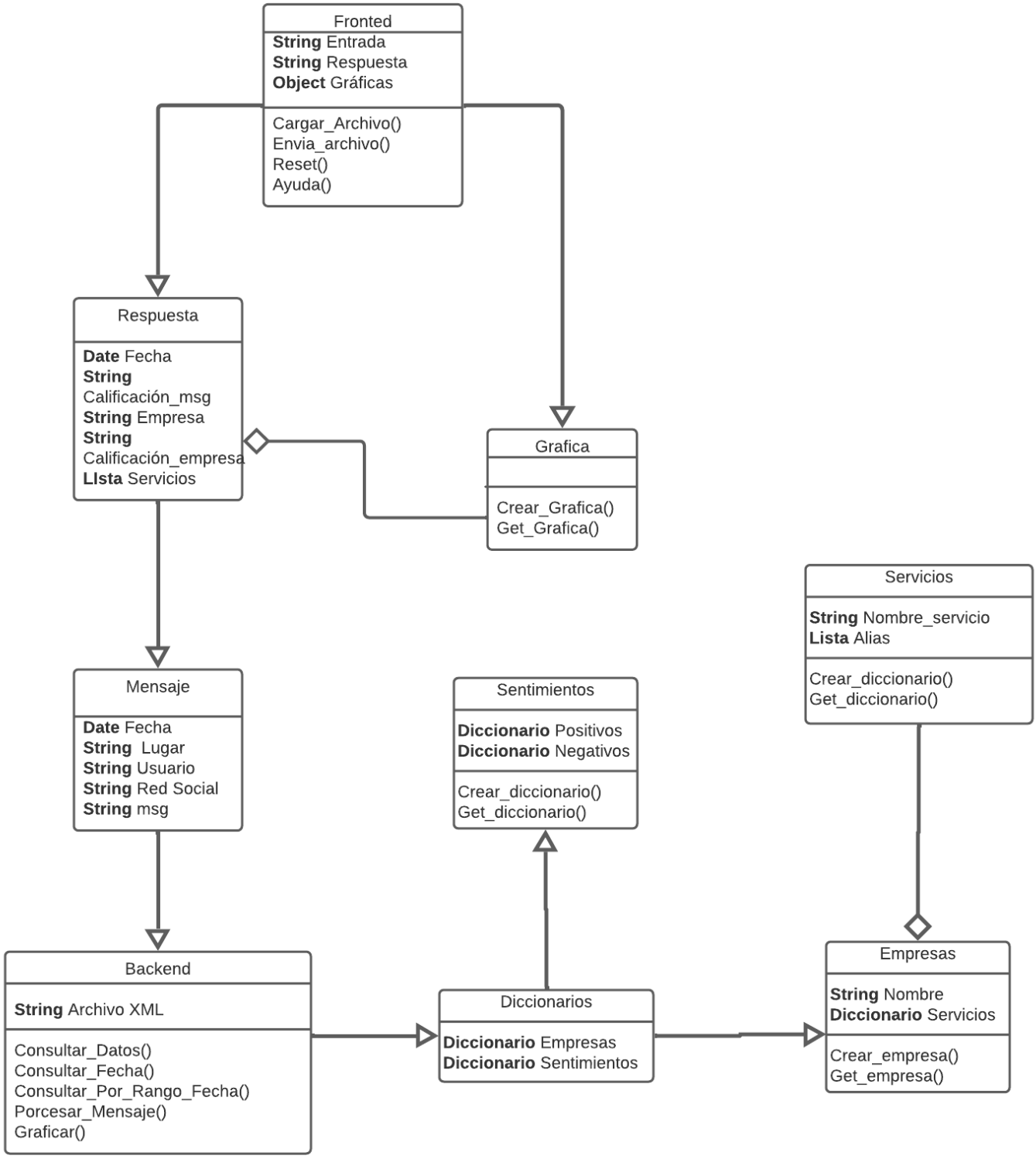
9. Leo otra vez el mensaje
10. Verifico si la empresa existe en el mensaje
11. Si existe la calificación de la empresa será la calificación del mensaje
12. Separo el mensaje en oraciones.
13. Analizo cada oración y contabilizo las palabras positivas y negativas que existan.
14. Verifico si el servicio de la empresa existe en la oración
15. Si existe la calificación del servicio será dependiendo la cantidad de positivos y negativos que haya en la oración.
16. Guardo los datos en una base de datos.

Conclusiones

1. Flask sirve bastante para el manejo del backend de una aplicación web
2. Django sirve bastante para el manejo del fronted de una aplicación web
3. Los archivos xml pueden servir para tener una base de datos de una aplicación web

Referencias bibliográficas

1. Pallets, (2010) Fask desarrollo web, una gota a la vez, https://flask-es.readthedocs.io


Fronted
String Entrada
String Respuesta
Object Gráficas
Cargar_Archivo()
Envia_archivo()
Reset()
Ayuda()
Respuesta
Date Fecha
String
Calificación_msg
String Empresa
String
Calificación_empresa
LIsta Servicios
Grafica
Crear_Grafica()
Get_Grafica()
Servicios
String Nombre_servicio
Lista Alias
Crear_diccionario()
Get_diccionario()
Mensaje
Date Fecha
String Lugar
String Usuario
String Red Social
String msg
Sentimientos
Diccionario Positivos
Diccionario Negativos
Crear_diccionario()
Get_diccionario()
Backend
String Archivo XML
Consultar_Datos()
Consultar_Fecha()
Consultar_Por_Rango_Fecha()
Porcesar_Mensaje()
Graficar()
Diccionarios
Diccionario Empresas
Diccionario Sentimientos
Empresas
String Nombre
Diccionario Servicios
Crear_empresa()
Get_empresa()