Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la Programación y Computación 2

Inga. Claudia Liceth Rojas Morales
Ing. Marlon Antonio Pérez Türk
Ing. José Manuel Ruiz Juárez
Ing. Edwin Estuardo Zapeta Gómez
Ing. Byron Rodolfo Zepeda Arevalo

Tutores de curso:
Josué Alfredo González
Marvin Daniel Rodríguez
Daniel Arturo Alfaro
Sergio Felipe Zapeta
Erwin Fernando Vásquez

PROYECTO 3

OBJETIVO GENERAL

Desarrollar una solución integral que implemente un API que brinde servicios utilizando el Protocolo HTTP bajo el concepto de programación orientada a objetos (POO) y el uso de bases de datos.

OBJETIVOS ESPECÍFICOS

- Implementar un API a través de lenguaje Python que pueda ser consumida utilizando el protocolo HTTP.
- Utilizar el paradigma de programación orientada a objetos para construir software.
- Utilizar bases de datos para almacenar información de forma persistente.
- Utilizar archivos XML como insumos para la comunicación con el API desarrollado.
- Utilizar expresiones regulares para extraer contenido de texto.

ENUNCIADO

La empresa Tecnologías Chapinas, S.A. está desarrollando una herramienta que sea 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.

A continuación, se muestra un ejemplo de un mensaje en Twitter

> Felicidades @usuario9882!!! Espero que este éxito académico abra muchas puertas en tu vida profesional #FiestaOctubre#

En un mensaje se puede hacer referencia a uno o más usuarios y colocar uno o más Hashtags en cualquier posición del texto.

Reglas y consideraciones del formato para cada mensaje recibido:

- Un mensaje puede o no tener referencias a otros usuarios y puede referir uno o más usuarios.
- Un mensaje puede o no tener hashtags y puede tener uno o más hashtags.
- Los hashtags y las referencias a usuarios pueden estar en cualquier ubicación dentro del texto.
- Los nombres de usuario solo pueden contener letras, números y símbolos “_”.
- Todo Hashtag debe estar entre los símbolos # Ej. #fiesta!!!#

La empresa Tecnologías Chapinas, S.A. ha creado una estrategia para establecer si un mensaje tiene un sentimiento positivo, negativo o neutro a través de la creación de un diccionario de datos que determine palabras que puedan calificar un mensaje como positivo o negativo, en caso de no tener palabras del diccionario de datos específico, o bien, que la cantidad de palabras con sentimientos positivos y negativos sean iguales, entonces, se considera que el mensaje es neutro.

El programa por desarrollar, luego de recibir “n” mensajes de la red social Twitter, deberá almacenar la información necesaria en formato XML, para posteriormente emitir reportes y realizar consultas, además, debe considerar que cada mensaje solamente puede ser clasificado como positivo, negativo o neutro. El archivo de respuesta o salida luego de realizar nuevas cargas de información deberá presentar la siguiente información:

FECHA: dd/mm/yyyy
Cantidad de mensajes recibidos: Número total de mensajes recibidos
Cantidad de usuarios mencionados: Número total de usuarios distintos mencionados en los mensajes
Cantidad de hashtags incluidos: Número total de hashtags distintos incluidos en los mensajes.
…
FECHA: dd/mm/yyyy
…

La estructura completa del archivo de entrada y salida está descrita en la siguiente sección.

ARCHIVOS DE ENTRADA Y SALIDA

Archivo de entrada

El archivo de entrada con mensajes tendrá la siguiente estructura XML:

```
<?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>
```

El archivo de entrada con la configuración del diccionario de sentimientos positivos y negativos tendrá la siguiente estructura XML:

```
<?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>
```

Nota 1: Para el análisis de los archivos de entrada no debe tomarse en consideración mayúsculas o minúsculas, es decir, institución = Institución = INstituCiÓN; y tampoco debe considerar las tildes, es decir, institución = INSTITUCION = insTItucion. Esto aplica para todos los conceptos (usuarios, hashtags, sentimientos positivos, sentimientos negativos, etc.).
Nota 2: El programa a desarrollar podrá cargar “n” archivos de entrada de mensajes y “m” archivos de entrada de configuración, estos datos serán incrementales.

Archivo de salida

Nombre del archivo: resumenMensajes.xml

```
<?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>
```

Nota: El conteo de usuarios mencionados y de hastags incluidos solo debe considerar los usuarios y hashtags diferentes por fecha.

Nombre del archivo: resumenConfig.xml

```
<?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>
```

ARQUITECTURA

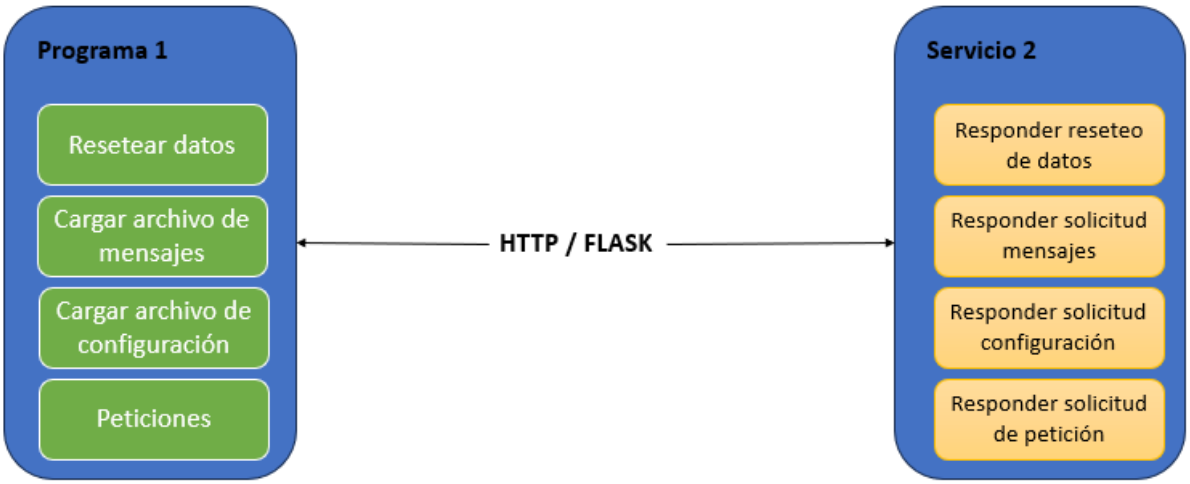

Fig. 1 – Arquitectura general de la aplicación

Programa 1 - Frontend

Este programa consiste en una aplicación Web y consistirá en un simulador de la aplicación principal, contendrá únicamente las funcionalidades necesarias para testear el buen funcionamiento de la API (Servicio 2), en esta aplicación se podrán mostrar los eventos que se procesarán y los datos estadísticos que fueron almacenados en la base de datos XML).

Para realizar el frontend deberá utilizarse el framework **Django**, el cual trabaja con el patrón MVT (Modelo-Vista-Template).

Componentes del Programa 1:

- **Resetear datos:** Esta opción mandará la instrucción a la Api para devolver al estado inicial la Api, es decir, sin datos.
- **Cargar archivo de mensajes:** Se desplegará una pantalla para gestionar la carga de los archivos de entrada con extensión .xml con uno o varios mensajes (ver archivos de entrada de mensajes).
- **Cargar archivo de configuración:** Se desplegará una pantalla para gestionar la carga de los archivos de entrada con extensión .xml con la configuración del diccionario de datos para palabras positivas y negativas (ver archivos de entrada de configuración).
- **Peticiones:** En este apartado se debe de tener las siguientes opciones:
  - **Consultar hashtags:** Al seleccionar esta opción se deben de solicitar un rango de fechas y mostrar la siguiente información de forma amigable:
    Fecha: 01/10/2023
    1. #usac# : 20 mensajes
    2. #fiusac#: 19 mensajes
    3. #bienvenido#: 16 mensajes
    4. …

    Fecha: 02/10/2023
    1. #Guatemala# : 30 mensajes

2. #SelecionNacional# : 26 mensajes
3. …

…

- **Consultar menciones:** Al seleccionar esta opción se deben de solicitar un rango de fechas y mostrar la siguiente información de forma amigable:

  Fecha: 01/10/2023
  1. @user01 : 16 mensajes
  2. @userxx : 10 mensajes
  3. @userab : 9 mensajes
  4. …

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

  …

- **Consultar sentimientos en mensajes**:

  Al seleccionar esta opción se deben de solicitar un rango de fechas y mostrar la siguiente información de forma amigable:

  Fecha: 01/10/2023

  Mensajes con sentimiento positivo : 16 mensajes
  Mensajes con sentimiento negativo : 9 mensajes
  Mensajes neutros : 1 mensaje

  Fecha: 02/10/2023

  Mensajes con sentimiento positivo : 5 mensajes
  Mensajes con sentimiento negativo : 2 mensajes
  Mensajes neutros : 0 mensaje

  …

- **Gráfica:** Se deberán crear gráficas para las tres consultas anteriores mostrando la información resumida para el rango de fechas.
- **Ayuda:** desplegará 2 opciones, una para visualizar información del estudiante y otra para visualizar la documentación del programa.

NOTA 1: Todos los reportes y gráficas deben poder ser generados en formato .pdf
NOTA 2: El Frontend deberá mostrar los datos recuperados del consumo del servicio que se describe en la siguiente sección.

Servicio 2 - Backend

Este servicio consiste en una API que brindará servicios utilizando el protocolo HTTP, su funcionalidad principal es procesar los datos recibidos del programa 1, luego de procesar los datos es necesario que estos sean almacenados en uno o varios archivos xml, algunos de estos archivos están especificados en la sección de archivos de entrada y salida[1], este servicio también tiene la funcionalidad de devolver los datos que fueron almacenados para que sean mostrados como se indica en la sección “Archivos de salida”.

[1] El estudiante puede definir otros archivos que sean útiles para mostrar los resultados solicitados por el frontend.

Para la realización de este servicio debe utilizarse el framework **Flask**. El estudiante deberá definir por su propia cuenta los métodos que necesitará para la realización de este servicio. Esto significa que debe implementar tantos métodos como necesite para consumir la API.

NOTA: Durante la calificación de este proyecto, el Servicio 2 podrá ser consumido desde otro cliente, por ejemplo, Postman y recibirá una respuesta XML con los datos necesarios para cada funcionalidad de la sección “peticiones” del programa 1 - Frontend.

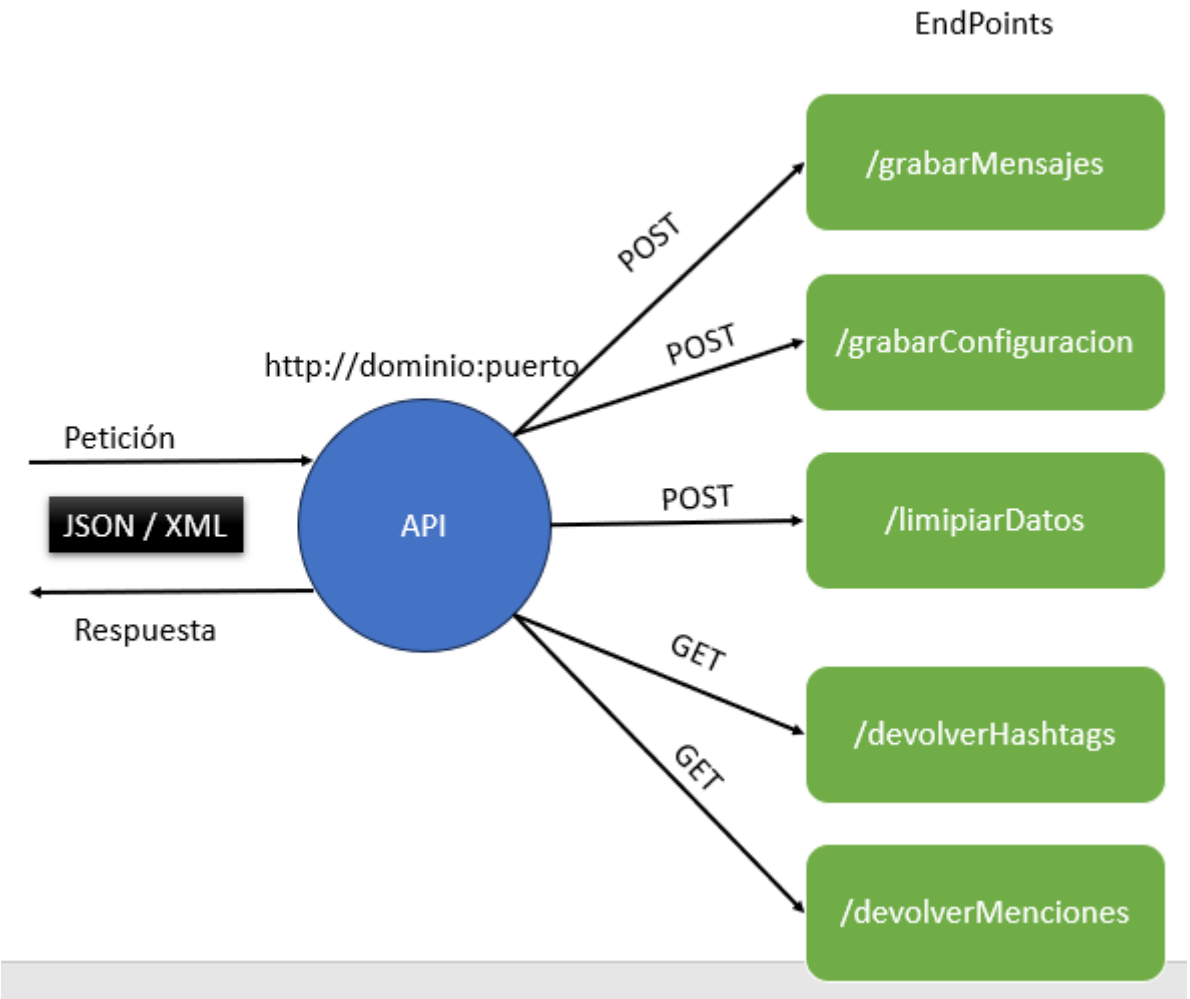

Fig. 4 - Sugerencia de la estructura de un API

CONSIDERACIONES

En las solicitudes de procesamiento del archivo de entrada debe descartarse cualquier información que no sea necesaria para la elaboración de las gráficas y archivos de salida, por ejemplo, si las fechas traen nombres de lugares u otra información, éstos no deben ser considerados errores, solamente debe considerarse la información útil para el proyecto.

Debe utilizarse versionamiento para el desarrollo del proyecto. Se utilizará la plataforma Github en la cual se debe crear un repositorio en el que se gestionará el proyecto. Se deben realizar 4 releases o versiones del proyecto (se recomienda realizar una por semana del tiempo disponible). Se deberá agregar a su respectivo auxiliar como colaborador del

repositorio. El último release será el release final y se deberá de realizar antes de entregar el proyecto en la fecha estipulada.

DOCUMENTACIÓN

Para que el proyecto sea calificado, el estudiante deberá entregar la documentación utilizando el formato de ensayo definido para el curso. En el caso del proyecto, el ensayo debe tener entre 4 y 7 páginas de contenido, este máximo no incluye los apéndices o anexos donde se pueden mostrar modelos y diseños utilizados para construir la solución.

RESTRICCIONES

- Solo se permitirá la utilización de los IDEs discutidos en este enunciado.
- Uso obligatorio de programación orientada a objetos (POO).
- El nombre del repositorio debe de ser **IPC2_Proyecto3_#Carnet.**
- El estudiante debe entregar la documentación solicitada para poder optar a la calificación.
- Se calificará de los cambios realizados en el cuarto release. Los cambios realizados después de ese release no se tomarán en cuenta.
- Para dudas concernientes al proyecto se utilizarán los foros en UEDI de manera que todos los estudiantes puedan ver las preguntas y las posteriores respuestas.
- **NO HABRÁ PRÓRROGA.**

ENTREGA

- La entrega será el **2 de noviembre** a las 23:59 como máximo.
- La entrega será por medio de la UEDI.
- La documentación debe estar subida en el repositorio en una carpeta separada.
- Para entregar el proyecto en UEDI se deberá subir un archivo de texto con el link del repositorio.