ANALIZADOR DE SENTIMIENTOS EN MENSAJES "TECNOLOGÍAS CHAPINAS, S.A."

201403745 – Linda Madelin Fabiola Quelex Sep

Resumen

El proyecto pertenece al "Laboratorio de Introducción a la Programación y Computación 2" de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, donde se requiere desarrollar un analizador de mensajes, el cual determine si para una empresa se clasifique un mensaje como positivo, negativo o neutro, de acuerdo a los diccionarios establecidos en un archivo XML.

Para la creación del analizador de sentimientos en mensajes se utilizó la arquitectura de software cliente-servidor, donde para el desarrollar el backend se utilizó el framework Flask y para el desarrollo del frontend se utilizó el framework Django.

Para el diseño de la solución se utilizó el lenguaje de programación Python, tipos de datos abstractos, paradigma de programación orientada a objetos, manejo de archivos XML, HTML, CSS, entre otros aspectos técnicos que se detallan en el ensayo.

Palabras clave
Flask, Django, Python, XML, API

Abstract

The project belongs to the "Laboratory of Introduction to Programming and Computing 2" of the Faculty of Engineering of the University of San Carlos of Guatemala, where it is required to develop a message analyzer, which determines if a company classifies a message as positive, negative or neutral, according to the dictionaries established in an XML file.

For the creation of the sentiment analyzer in messages, the client-server software architecture was used, where the Flask framework was used to develop the backend and the Django framework was used for the frontend development.

For the design of the solution, the Python programming language, abstract data types, object-oriented programming paradigm, handling of XML files, among other technical aspects that are detailed in the essay, were used.

Keywords
Flask, Django, Python, XML, API.

Introducción

El presente ensayo es sobre el proyecto 3 del Laboratorio de Introducción a la Programación y Computación 2, de la Facultad de Ingeniería, de la Universidad de San Carlos de Guatemala.

El proyecto fue nombrado como "Analizador de sentimientos en mensajes" de la empresa Tecnologías Chapinas, S.A.

Para la realización del proyecto se utilizó el lenguaje de programación Python, el paradigma de programación orientada a objetos, la arquitectura de software cliente-servidor, para el desarrollo del frontend se utilizó HTML, CSS y el framework Django, y para el desarrollo del backend se utilizó el framework Flask, y se utilizaron listas para el almacenamiento de la información.

La aplicación web permite la carga de un archivo XML el cual brinda la información para realizar el análisis de mensajes. La aplicación retorna si un mensaje es positivo, negativo o neutro para una empresa y/o servicio, según los diccionarios establecidos en el archivo de entrada.

Desarrollo del tema

Para el desarrollo del proyecto 3 del Laboratorio de Introducción a la Programación 2 (LABIPC2) se desarrolló e implementó lo siguiente:

1. **Backend (API):**
   Se desarrolló utilizando el framework Flask, donde se definieron los siguientes endpoints a través del puerto 4000:
   a. **/analizardatosxml:**
      De acuerdo a los protocolos HTTP se utilizó el método POST para realizar esta petición, este endpoint permite la lectura del archivo XML que enviado por el frontend, también permite la carga de información a las listas para el manejo y manipulación de los datos, y finalmente almacena los resultados obtenidos del análisis en un archivo XML denominado "ARCHIVO_SALIDA.xml", de definieron las etiquetas de acuerdo al enunciado del proyecto.

```
30  @app.route('/almacenardatosxml', methods=['POST'])
31  def alamacenar_datos_xml():
32      print('\n','****ENCABEZADOS****','\n')
33      print(request.headers)
34      print('****ARCHIVO XML***','\n')
35      xml=request.data.decode('UTF-8')
36      print (xml)
37      raiz=ET.XML(xml)
38      contador=0
```

Figura 1. Enpoint: /analizardatosxml.

Fuente: elaboración propia, mayo 2022.

   b. **/reset:**
      Utiliza el método POST y realizar el vaciado de información de las listas como simulación de vaciado de una base de datos.

```
410  @app.route('/reset', methods=['POST'])
411  def reset():
412      uno=manager.vaciar_positivos()
413      dos=manager.vaciar_negativos()
414      tres=manager.vaciar_mensajes()
415      cuatro=empresas.vaciar_lista_empresa()
416      cinco=msg.vaciar_lista_msg()
417      seis=positivos.vaciar_lista_positivos()
418      siete=negativos.vaciar_lista_negativos()
419
420      return jsonify ({'msg':'LISTAS VACÍAS','lista empresas':cuatro,
```

Figura 2. Enpoint: /reset.

Fuente: elaboración propia, mayo 2022

c. /retornar

Utiliza el método GET, para retornar al backend una muestra de que la información fue cargada a las listas exitosamente, el retorno de información es con estructura JSON, donde se trasladan los datos de las listas como diccionarios.

```
401  @app.route('/retornar', methods=['GET'])
402  def retornar_cont_xml():
403      c3=manager.get_mensajes()
404      c2=manager.get_sentimientos_negativos()
405      c =manager.get_sentimientos_positivos()
406      c4= empresas.mostrar_empresas_json()
407
408      return jsonify(c3,c2,c,c4),200
```

Figura 3. Enpoint: /retornar.

Fuente: elaboración propia, mayo 2022

1.1. Clase: Manage

Esta clase contiene los métodos de las listas, entre ellas agregar, retornar, mostrar, entre otros ser utilizados en los endpoints.

1.2. Main:

Contiene los endpoints y las librerías y frameworks siguientes: ElementTree, Flask, JSON, RE, DOM, request, urllib, entre otros, todo lo anterior fue utilizado para el procesamiento de la información enviada por el backend.

2. Frontend:

Para el desarrollo del frontend se utilizó el framework Django, el diseño se desarrolló utilizando HTML y CSS, la estructura del frontend es la siguiente:

a. **Views**:

Contiene los métodos para la atención de los request y establecer comunicación con el backend y los templates, entre ellos:

- **Home:** retorna a la información básica del proyecto.
- **Infoestudiante**: muestra datos del estudiante como nombre y carné
- **Docu:** muestra en PDF la documentación técnica del proyecto
- **Pruebamsg**: permite el procesamiento de un texto en XML para ver el funcionamiento de la aplicación, consume el endpoint /pruebamensaje.
- **reportePDF:** muestra en el navegador el contenido de archivo XML de salida.
- **reset:** consume el endpoint /reset y vacía el archivo XML de salida.
- **Consulta:** muestra en un cuadro de texto la información almacenada en el XML de salida.
- **cargaMasiva**: consume el endpoint /cargardatosxml del backend y muestra la información del XML a procesar en un cuadro de texto y el análisis realizado que se almacena en un archivo XML de salida en otro cuadro de texto.

b. Urls:

Contiene la definición los path para el acceso a los métodos establecidos en las views.

c. Templates:

Contiene todos los archivos HTML que se describen a continuación:

- **Base.html:** se utilizó Bootstrap para el dar estilo al arhivo, este contiene la barra de

navegación de la página web, se utilizó Jinja para el uso de plantillas y mejorar la experiencia de usuario, en la barra de navegación se encuentra: inicio, cargar archivo, peticiones y ayuda.

Figura 4. Barra de navegación.
Fuente: elaboración propia, mayo 2022

- **Index.html**: es la página inicial que muestra información básica del proyecto.
- **Carga.html**: este template muestra la opción de seleccionar un archivo con extensión XML para que pueda ser procesado por el backend, contiene los botones de: enviar, reset y Generar PDF.

Figura 5. Template-Carga.
Fuente: elaboración propia, mayo 2022

- **Consulta.html**: en esta página se muestra un cuadro de texto y la opción de generar PDF con la información almacenada en el XML de salida:
- **Datosestudiante.html:** esta página muestra los datos del estudiante como párrafo.
- **Docu.html:** esta template muestra en PDF el Ensayo del Proyecto.
- **Pruebamensaje.html**: muestra dos cuadros de texto, el primero para ingreso de un fragmento de texto de un archivo XML, muestra otro cuadro de texto para mostrar el resultado del análisis en el backend.

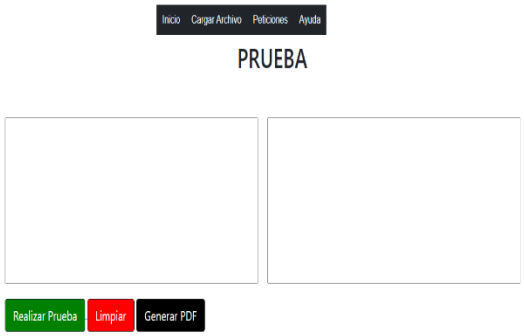


Figura 5. Template-pruebamensaje.
Fuente: elaboración propia, mayo 2022.

- **reportePDF.html**: abre y muestra en el navegador el archivo PDF con el resultado almacenado en el archivo XML de salida.

d. static:

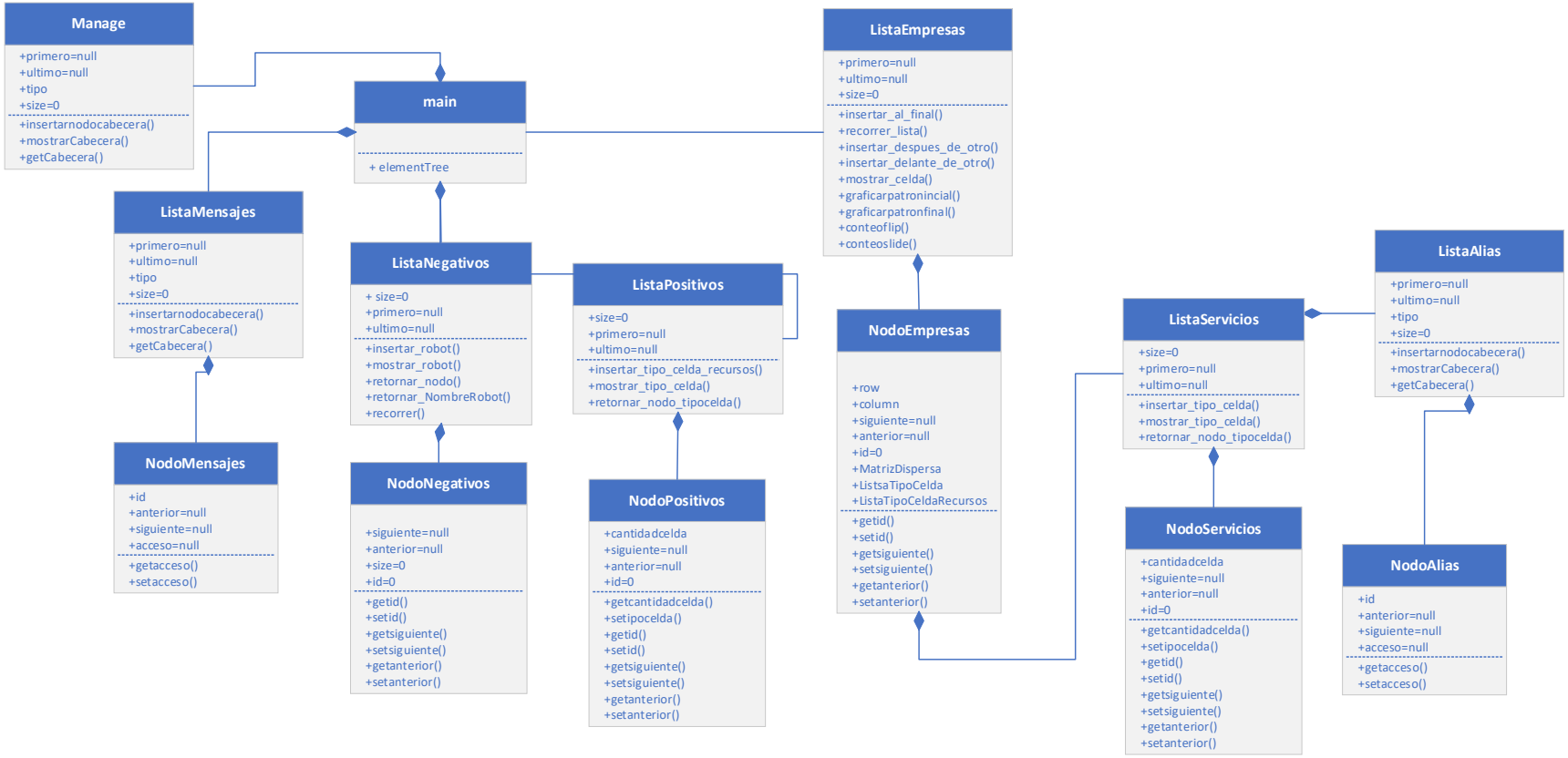
Dentro de esta carpeta de encuentra los estilos para los archivos html con el archivo denominado “style.css”

Referencias bibliográficas

1. Python - xml.etree.ElementTree - *La API XML ElementTree - Source code:Lib/xml/etree/ElementTree.py* El módulo xml.etree.ElementTree impleme - Español. (2021). Runebook.dev. https://runebook.dev/es/docs/python/library/xml.etree.elementtree
2. The web framework for perfectionists with deadlines | *Django*. (2022). Djangoproject.com. https://www.djangoproject.com/
3. Welcome to Flask — *Flask Documentation* (2.1.x). (2022). Palletsprojects.com. https://flask.palletsprojects.com/en/2.1.x/

APÉNDICE

1. Diagrama de Clases del Proyecto



FUENTE: diagrama de clases proyecto 3, elaboración propia, mayo 2022.

2. Arquitectura de software: cleinte-servicdor