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 2021.

GENERADOR Y EDITOR DE IMÁGENES

201901772 – Daniel Reginaldo Dubón Rodríguez

Resumen

El ensayo presenta el desarrollo de una API que brinde servicios utilizando el Protocolo HTTP como medio de comunicación para la transferencia de información entre los servicios, también el desarrollo de una aplicación web que pueda consumir esos servicios proveídos por la API. Para la creación de dicha API se hizo uso del framework Flask debido a que esta permite crear los servicios de una forma sencilla y rápida, para el desarrollo de la aplicación web que consumirá los servicios se uso el framework Django ya que este provee un patron de diseño conocido como modelo-vista-controlador y hace sencilla la construcción de la interfaz gráfica. La solución e implementación de lo antes mencionado fue a través del uso de un lenguaje de programación (Python) para el manejo de la lógica y así crear los distintos algoritmos y estructuras requeridos para el funcionamiento, también se utilizó para la construcción de la interfaz gráfica que permite interactuar de una forma más agradable e intuitiva con la aplicación.

Palabras clave

Python, Django, Flask, Lenguaje de Marcado Extensible (XML), Programación Orientada a Objetos (POO).

Abstract

The essay presents the development of an API that provides services using the HTTP Protocol as a means of communication for the transfer of information between services, as well as the development of a web application that can consume those services provided by the API. For the creation of said API, the Flask framework was used because it allows to create the services in a simple and fast way, for the development of the web application that will consume the services, the Django framework was used since it provides a pattern of design known as model-view-controller and makes the construction of the graphical interface easy. The solution and implementation of the aforementioned was through the use of a programming language (Python) to handle the logic and thus create the different algorithms and structures required for operation, it was also used for the construction of the graphical interface that allows you to interact in a more pleasant and intuitive way with the application..

Keywords

Python, Django, Flask, Extensible Markup Language (XML), Objetct Oriented Programming (OOP), Graphviz.

Introducción

Se realizo el desarrollo de una API que brinda servicios utilizando el Protocolo HTTP como medio de comunicación para la transferencia de información entre los servicios, también el desarrollo de una aplicación web que pueda consumir esos servicios proveídos por la API. Para la creación de dicha API se hizo uso del framework Flask debido a que esta permite crear los servicios de una forma sencilla y rápida, para el desarrollo de la aplicación web que consumirá los servicios se usó el framework Django ya que este provee un patrón de diseño conocido como modelo-vista-controlador y hace sencilla la construcción de la interfaz gráfica. La solución e implementación de lo antes mencionado fue a través del uso de un lenguaje de programación (Python) para el manejo de la lógica y así crear los distintos algoritmos y estructuras requeridos para el funcionamiento, también se utilizó para la construcción de la interfaz gráfica que permite interactuar de una forma más agradable e intuitiva con la aplicación.

Desarrollo del tema

La API permite realizar el trabajo de procesamiento de datos, esta se encarga de recibir la información procesarla y de devolver un resultado que esta a su vez se muestra de forma gráfica.

Se ha optado el uso del lenguaje de programación Phyton debido a que es multiplataforma y por su facilidad de uso y también porque cuenta con frameworks como Flask y Django para la realización de aplicaciones web que para este proyecto es necesario ya que nos facilitaran el trabajo para crearlo, además también porque soporta varios paradigmas de programación.

Para la elaboración de la solución se provee un archivo con extensión y estructura XML, el cual provee información que luego se procesaran y generaran obtendrán datos estadísticos, este archivo tiene la estructura de la siguiente forma:

```
<EVENTOS>
    <EVENTO>
      Guatemala, 15/01/2021
      Reportado por: <”Nombre Empleado 1” xx@ing.usac.edu.gt>
      Usuarios afectados: aa@ing.usac.edu.gt, <bb@ing.usac.edu.gt>
      Error: 20001 - Desbordamiento de búfer de memoria RAM
      en el servidor de correo electrónico.
    </EVENTO>
…
</EVENTOS
```

Figura 1. Estructura del archivo XML dado.

Fuente: Elaboración propia.

Este archivo se procesa y se obtienen los datos esenciales y se obtiene una salida como se le muestra a continuación.

```
<ESTADISTICAS>
    <ESTADISTICA>
    <FECHA> 15/01/2021 </FECHA>
    <CANTIDAD_MENSAJES> 3 </CANTIDAD_MENSAJES>
    <REPORTADO_POR>
        <USUARIO>
            <EMAIL> xx@ing.usac.edu.gt </EMAIL>
            <CANTIDAD_MENSAJES> 1 </CANTIDAD_MENSAJES>
            </USUARIO>
            ...

    <AFECTADOS>
    <AFECTADO> aa@ing.usac.edu.gt </AFECTADO>
    <AFECTADO> bb@ing.usac.edu.gt </AFECTADO>
    …
    </AFECTADOS>
    <ERRORES>
      <ERROR>
        <CODIGO> 20001 </CODIGO>
        <CANTIDAD_MENSAJES> 2 </CANTIDAD_MENSAJES>
      </ERROR>
    <ERROR>
    ...
    </ESTADISTICA>
…
</ESTADISTICAS>
```

Figura 2. Estructura del archivo XML de salida.

Fuente: Elaboración propia.

Debido que los archivo pueden contener errores se vio la necesidad de crear un autómata que reconociera el patrón de las etiquetas evento, si la estructura posee un erro las etiquetas que poseen error deben ser descartadas y debe seguir procesando la información que se le sigue, para esto se creo un autómata que permite guardar todas aquellas etiquetas <EVENTO></EVENTO> que no contengan errores de sintaxis, la estructura del autómata es la siguiente:

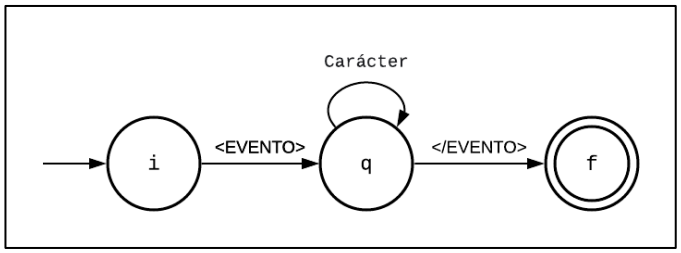

Figura 4. Autómata que obtiene información de las etiquetas evento.

Fuente: Elaboración propia.

Teniendo la información que contienen las etiquetas era necesario también extraer los datos y para ello se utilizaron expresiones regulares para extraerlos ya que estos datos tienen una expresión común que puede ser reconocida y a través de ello extraerlos, a continuación, se muestran las expresiones regulares utilizadas para la extracción de los diferentes datos:

```
Expresión regular para la fecha: \d\d/\d\d/\d\d\d\d

Expresión regular para el usuario que reporta y los afectados:
[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*

Expresión regular para el código de error: \d\d\d\d\d
```

Figura 4. Expresiones regulares para la extracción de la infromación

Fuente: Elaboración propia.

Para saber dónde se tenían que aplicar las expresiones regulares se creo un algoritmo donde se identificaban palabras clave que se anteponían antes de la información que se quería obtener, por ejemplo, para el caso de el usuario que reporta el error, siempre se seguía de la expresión Reportado por.

Obtenidos los datos se necesitaba de una estructura en la cual se guardarían la información del archivo dado, por lo cual se llevó al análisis del uso de un tipo de dato abstracto ya que permite el uso de memoria dinámica ya que el programa lo requiere debido a que no se sabe con certeza la cantidad de datos que puede contener.

Con el paradigma de programación orientada a objetos se emulo los diferentes tipos de datos para almacenarlos en la estructura de datos correspondientes, esto abrió la posibilidad de trabajar más eficientemente y optimización de memoria del programa ya que se tienen varios datos y se necesitan agruparlos entre ellos para su manipulación.

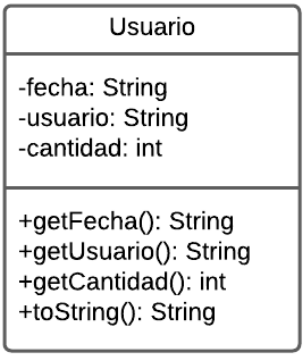

Figura 6. Modelo del tipo de dato Usuario.

Fuente: Elaboración propia

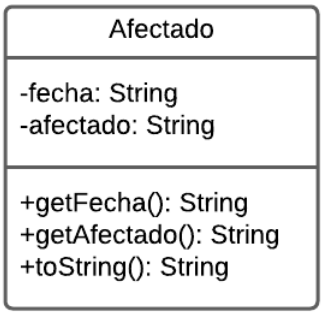

Figura 7. Modelo del tipo de dato Afectado.

Fuente: Elaboración propia

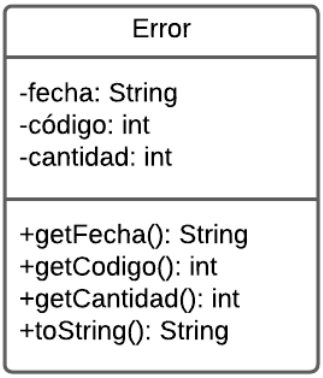

Figura 8. Modelo del tipo de dato Afectado.

Fuente: Elaboración propia

Para el funcionamiento y utilización de la API se crearon los diferentes end-points:

a. '/Archivo', este utiliza el método POST ya que se recibe el archivo cargado desde el front-end el cual será procesado posteriormente, este también genera un archivo XML y escribe la información.

b. '/Estadisticas', este utiliza el método GET ya que se solicita que procese la información antes enviada devuelva la información con la estructura requerida.

c. '/filtro1', este utiliza el método POST ya que desde el fron-end se le envia el parámetro fecha y este realiza una consulta en los datos y regresa todos aquellos usuarios que hayan enviado un mensaje en la fecha especificada.

d. '/filtro2', este utiliza el método POST ya que desde el fron-end se le envia el parámetro código de error y este realiza una consulta en los datos y regresa todos aquellos códigos de errores en las que aparece el error en todas las fechas que se reporto.

e. '/reset', este utiliza el método DELETE ya que este elimina toda la información que esta almacenada en la API.

Conclusiones

Conocer e implementar bien las estructuras de datos permiten manejar de forma muy rápida y eficiente los datos almacenados.

El desarrollo de esta aplicación da un ejemplo claro de cómo es que trabajan a nivel interno la generación y edición de imágenes.

Se debe conocer a profundidad el lenguaje con el que se trabajara la solución ya que de estos dependen cuan accesible puede ser construir los algoritmos que determinaran la solución al problema, también es necesario conocer que paradigmas se pueden utilizar y en base a esto apegarse a una o varias para hacer lo menos tedioso posible el desarrollo de este.

Apéndice

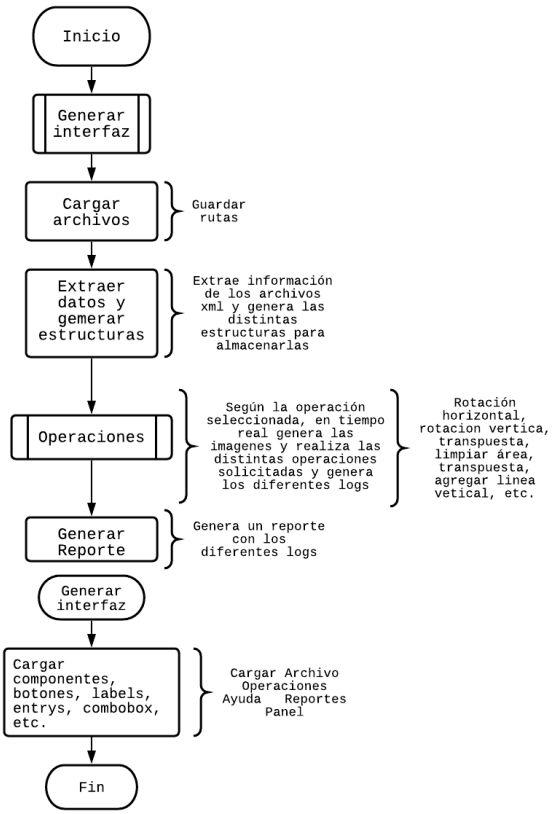

Figura 7. Diagrama general del flujo de la aplicación.

Fuente: Elaboración propia