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.

PROYECTO 2

201801627 – Eduardo René Agustin Mendoza

Resumen

Al consumir los servicios desde internet se tiene en cuenta que a las acciones que nosotros solicitemos tendremos una respuesta por parte de los servidores, pero todo esto tiene un proceso largo el cual de mucha manera se puede presentando, en el presente se tomara en cuenta que todos los objetos utilizados en esto se están almacenando dentro de la misma aplicación ya que serán consumidas en el mismo servidor. En este caso se estará consumiendo la API desde Flask la cual es una librería nata de Python y como Framework se utilizará Django con dichas herramientas se procederá a realizar las peticiones que se soliciten

En este caso nuestro objetivo será poder manipular los diferentes tipos de archivos que se presenten en formato XML los cuales traerán distintos eventos los cuales posteriormente se modificaran para tener de una mejor manera su visualización

Palabras clave

Framework, API, Servidor local, Programación Web, Archivos Cargados

Abstract

When consuming the services from the internet, it is taken into account that to the actions that we request we will have a response from the servers, but all this has a long process which in a lot of way can be presented, in the present it will be taken into account that all the objects used in this are being stored within the same application since they will be consumed in the same server. In this case, the API will be consumed from Flask, which is a natural Python library, and as a Framework, Django will be used with these tools, the requests that are requested will be made.

In this case, our objective will be to be able to manipulate the different types of files that are presented in XML format, which will bring different events which will later be modified to have a better visualization.

Keywords

Framework, API, Local Server, Web programming, Uploaded files.

Introducción

El desarrollo del programa esta orientado en el consumo de una aplicación mediante una página web la cual será la parte visual que el cliente pueda ver (Frontend) donde dicho cliente podrá cargar los archivos y hacer las peticiones a la API (Backend) la cual estará constituida por los endpoints donde vendrán a recaer las peticiones de dicho cliente, al mismo tiempo se tendrán como respuesta las diferentes acciones que desean los clientes y la estructuración de un archivo de salida el cual mostrara los datos en una manera más sencilla de poder visualizar para las peticiones de los clientes.

Así mismo se tendrá como una respuesta un par de gráficas las cuales serán para poder visualizar el número de personas que han participado en los eventos que vendrán plasmados dentro de los archivos que se cargaran con extensión xml lo cual hará que la respuesta sea escrita y mostrada como xml esto para que pueda visualizar con mayor facilidad los datos de las fechas, usuarios, afectados y los errores que se han cometido. Esto se realizo por medio de una limpieza del archivo y una restructuración gracias al uso de Expresiones Regules las cuales brindan la facilidad de poder buscar la información que se desea exponer en este caso, así mismo la aplicación se desarrollo en el entorno de Python y la ayuda de la librería de Flask como backend y Django como framework para el frontend por lo mismo se utilizo la ayuda de una página HTML y por lo consiguiente el uso de CSS y JavaScript para la manipulación de los datos así como él envió y recibimiento de datos.

Desarrollo del tema

Una empresa de software le ha solicitado construir un software que pueda ser consumido desde Internet como un servicio. Este software recibirá un mensaje de la bitácora del software principal de la compañía y producirá una serie de información estadística relacionada. El mensaje que la bitácora enviará contendrá la siguiente información, se presenta en la figura No. 1.

```
FECHA: dd/mm/yyyy
USUARIO: correo electrónico del usuario que genera el error
AFECTADO: lista de correos electrónicos separados por coma
ERROR: código numérico: descripción del error
```

Figura 1. Formato de información cargada en él archivo.

El programa a desarrollar, luego de recibir el mensaje antes mencionado deberá almacenar la información necesaria en un archivo XML que permitirá mostrar la siguiente información, se presenta en la figura No. 2.

```
FECHA: dd/mm/yyyy
 Cantidad total de mensajes recibidos en esta fecha
 Listado de usuarios distintos que reportaron mensajes
 Usuario1 cantidad mensajes generados
 Usuario2 cantidad mensajes generados
 …
 Listado de usuarios afectados
 UsuarioX1
 UsuarioX2
 …
 Listado de errores distintos reportados
 Código numérico del error: cantidad de mensajes recibidos
 Código numérico del error: cantidad de mensajes recibidos
 …
FECHA: dd/mm/yyyy
…
```

Figura 2. Estructura del Archivo de entrada

Teniendo en cuenta que La sumatoria de los mensajes generados en el listado de usuarios que reportaron el mensaje y la sumatoria de los errores reportados deben cuadrar con la cantidad de mensajes recibidos en una fecha dada.

Si el archivo de entrada posee un error de sintaxis con respecto a la estructura XML entonces el evento con el error será ignorado y se continuará con el análisis de los eventos restantes. Dentro de la etiqueta pueden venir 1 o más etiquetas, como se muestra en la figura No.3

```
<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 3. Archivo de Entrada.

Y el archivo de salida tendrá el siguiente formato como se muestra en la figura No.4

```
<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>
      <USUARIO>
        <EMAIL> yy@ing.usac.edu.gt </EMAIL>
        <CANTIDAD_MENSAJES> 2 </CANTIDAD_MENSAJES>
      </USUARIO>
    </REPORTADO_POR>
    <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>
        <CODIGO> 20002 </CODIGO>
        <CANTIDAD_MENSAJES> 1 </CANTIDAD_MENSAJES>
      </ERROR>
      ...
    </ERRORES>
  </ESTADISTICA>
  ...
</ESTADISTICAS>
```

Figura 4. Archivo de Salida.

La arquitectura que se utilizó para realizar dicho proyecto es de la siguiente manera, figura No.5

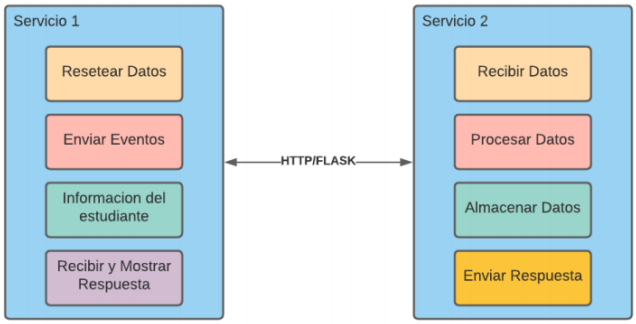

Figura 5. Arquitectura.

Servicio 1 - Frontend.

Este servicio 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 de salida). En la figura No.6 se muestra como es la idea de la aplicación.

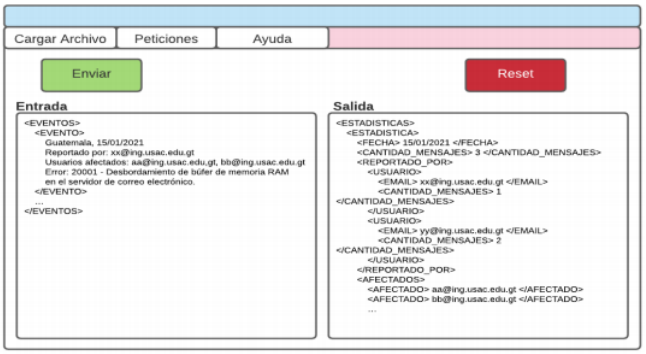

Figura 6. Prototipo de Interfaz.

Componentes:

- Cargar Archivo: Se desplegará una pantalla para gestionar la carga de los archivos de entrada con extensión .xml con uno o varios eventos. Se

especifica en la sección de archivos de entrada y salida.

● Peticiones: En este apartado se debe de tener las siguientes opciones:

❖ Consultar Datos: Al seleccionar esta opción se deben de consultar los datos almacenados en el archivo estadisticas.xml y se mostrarán los datos en el recuadro de texto de salida.

❖ Filtrar información por fecha y usuario que reporta: Al seleccionar esta opción se podrá elegir la fecha por la cual se requiere filtrar y se debe de mostrar gráficamente los usuarios que reportaron errores en esa fecha.

❖ Filtrar por fecha y código de error: Al seleccionar esta opción se podrá ingresar un código de error, se deberá presentar gráficamente el total de mensajes que contienen ese código por cada fecha en donde se hizo el reporte de dicho error.

● Ayuda: desplegará 2 opciones, una para visualizar información del estudiante y otra para visualizar la documentación del programa.

● Botón Enviar: Enviará los eventos del recuadro de texto a la Api para su posterior procesamiento.

● Botón Reset: Este botón mandará la instrucción a la Api para devolver al estado inicial la Api, es decir sin datos.

Servicio 2 – Backend.

Este servicio consiste en una API que brindara servicios utilizando el protocolo HTTP, su funcionalidad principal es procesar los datos recibidos del servicio 1, luego de procesar los datos es necesario que estos sean almacenados en un archivo xml, este archivo está especificado en la sección de archivos de entrada y salida, este servicio también tiene la funcionalidad de devolver los datos que fueron almacenados para que sean mostrados como anteriormente se indica.

Django.

Django fue desarrollado inicialmente entre 2003 y 2005 por un equipo que era responsable de crear y mantener sitios web de periódicos. Después de crear varios sitios, el equipo empezó a tener en cuenta y reutilizar muchos códigos y patrones de diseño comunes. Este código común se convirtió en un framework web genérico, que fue de código abierto, conocido como proyecto "Django" en julio de 2005.
Django es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Desarrollado por programadores experimentados, Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que puedes concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago.
Django te ayuda a escribir software que es:

Completo

Django sigue la filosofía "Baterías incluidas" y provee casi todo lo que los desarrolladores quisieran que tenga "de fábrica". Porque todo lo que necesitas es parte de un único "producto", todo funciona a la perfección, sigue principios de diseño consistentes y

Versátil

Django puede ser (y ha sido) usado para construir casi cualquier tipo de sitio web — desde sistemas manejadores de contenidos y wikis, hasta redes sociales y sitios de noticias. Puede funcionar con cualquier framework en el lado del cliente, y puede devolver contenido en casi cualquier formato (incluyendo HTML, RSS feeds, JSON, XML, etc). ¡El sitio que estás leyendo actualmente está basado en Django!

Flask

Flask es un *"micro"* Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC.

La palabra *"micro"* no designa a que sea un proyecto pequeño o que nos permita hacer páginas web pequeñas, sino que al instalar Flask tenemos las herramientas necesarias para crear una aplicación web funcional, pero si se necesita en algún momento una nueva funcionalidad hay un conjunto muy grande extensiones (plugins) que se pueden instalar con Flask que le van dotando de funcionalidad.

De principio en la instalación no se tienen todas las funcionalidades que se pueden necesitar, pero de una manera muy sencilla se pueden extender el proyecto con nuevas funcionalidades por medio de plugins.

El patrón MVC es una manera o una forma de trabajar que permite diferenciar y separar lo que es el modelo de datos (los datos que van a tener la App que normalmente están guardados en BD), la vista (página HTML) y el controlador (donde se gestiona las peticiones de la app web).

Conclusiones

Los framework nos ayudan con la simplicidad de poder crear una aplicación web ya que nos dan las herramientas necesarias para que todo nuestro entorno funcione de una manera mucho más practica y sencilla teniendo en cuenta que solo se tiene que configurar una parte para poder manejar las vistas y los enlaces de las páginas web.

Tener en cuenta que el consumo de la API es por medio de los protocolos HTTP.

Los métodos utilizados en la siguiente aplicación fueron el GET y POST los cuales sirvieron para poder obtener y enviar información desde el backend hacía el frontend y viceversa.

La facilidad del lenguaje Python facilito la implementación de las expresiones regulares las cuales se necesitaron para poder modificar el archivo de entrada.

Referencias bibliográficas

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.

https://openwebinars.net/blog/que-es-flask/

https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction