API – Django – Flask

2011-14493 – Victor Ronaldo Gomez Lara

Resumen

El término API es una abreviatura de Application Programming Interfaces, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.
Así pues, podemos hablar de una API como una especificación formal que establece cómo un módulo de un software se comunica o interactúa con otro para cumplir una o muchas funciones. Todo dependiendo de las aplicaciones que las vayan a utilizar, y de los permisos que les dé el propietario de la API a los desarrolladores de terceros.

Qué es Django

Django es un framework web diseñado para realizar aplicaciones de cualquier complejidad en unos tiempos muy razonables.
Está escrito en Python y tiene una comunidad muy amplia, que está en continuo crecimiento.

Por qué usar Django

Los motivos principales para usar Django son:

Es muy rápido: Si tenéis una startup, tenéis prisa por terminar vuestro proyecto o, simplemente, queréis reducir costes, con Django podéis construir una aplicación muy buena en poco tiempo.

Viene bien cargado: Cualquier cosa que necesitéis realizar, ya estará implementada, sólo hay que adaptarla a vuestras necesidades. Ya sea porque hay módulos de la comunidad, por cualquier paquete Python que encontréis o las propias aplicaciones que Django trae, que son muy útiles.

Es bastante seguro: Podemos estar tranquilos con Django, ya que implementa por defecto algunas medidas de seguridad, las más clásicas, para que no haya SQL Injection, no haya Cross site request forgery (CSRF) o no haya Clickjacking por JavaScript. Django se encarga de manejar todo esto de una manera realmente sencilla.

Es muy escalable: Podemos pasar desde muy poco a una aplicación enorme perfectamente, una aplicación que sea modular, que funcione rápido y sea estable.

Es increíblemente versátil: Es cierto que en un principio Django comienza siendo un Framework para almacenar noticias por sitios de prensa, blogs y este estilo de webs, pero con el tiempo ha ganado tanta popularidad que se puede usar para el propósito que queráis.

Qué es Flask

A la hora de realizar desarrollo en el Back-end, tenemos una diversidad de opciones para elegir. Si se inclinan por Python como lenguaje, seguramente habrán usado o escuchado hablar sobre Django, uno de los frameworks para desarrollo Web más conocidos para este lenguaje. En esta oportunidad quiero hablarles sobre otra opción. Les voy a contar qué es Flask y porqué podríamos elegirlo para nuestro próximo proyecto.
Flask el micro framework más famoso para Python
¿Flask es un framework o un micro framework? Seguramente si hemos buscado en Internet sobre este proyecto lo habremos encontrado de ambas formas.
En este sentido, podríamos pensar que el segundo es una variante del primero. Y que, en definitiva, Flask es un framework “minimalista”, que no requiere de otras dependencias para hacer cosas básicas. También

nos ofrece una curva de aprendizaje muy baja para comenzar a entenderlo.
Todo esto hace que, si decidimos comenzar a desarrollar soluciones Web con Python, Flask puede ser un muy buen punto de entrada.

¿Qué podemos hacer con Flask?

La respuesta básica puede ser muy sencilla: desarrollar el lado servidor de una aplicación Web.

¿Cómo instalo Flask? En primer lugar debemos tener instalado Python (https://www.python.org/). Luego, desde la consola o línea de comando podremos ejecutar el gestor de paquetes: pip install Flask.

Palabras clave
API
Framework

Abstract

The term API is an abbreviation for Application Programming Interfaces, which in Spanish means application programming interface. It is a set of definitions and protocols used to develop and integrate application software, allowing communication between two software applications through a set of rules. Thus, we can speak of an API as a formal specification that establishes how a software module communicates or interacts with another to fulfill one or many functions. All depending on the applications that are going to use them, and the permissions that the API owner gives to third-party developers.

What is Django

Django is a web framework designed to make applications of any complexity in a very reasonable time. It is written in Python and has a very large community, which is constantly growing.

Why use Django

The main reasons for using Django are: It is very fast: If you have a startup, you are in a hurry to finish your project or simply want to reduce costs, with Django you can build a very good application in a short time. It comes well loaded: Whatever you need to do, it will already be implemented, you just have to adapt it to your needs. Either because there are community modules, for any Python package that you find or the applications that Django brings, which are very useful. It is quite safe: We can rest easy with Django, since it implements by default some security measures, the most classic, so that there is no SQL Injection, there is no Cross site request forgery (CSRF) or there is no JavaScript Clickjacking. Django handles all this in a really simple way. It is very scalable: We can go from very little to a huge application perfectly, an application that is modular, works fast and is stable. It is incredibly versatile: It is true that at first Django begins as a framework for storing news for press sites, blogs and this style of webs, but over time it has gained so much popularity that it can be used for whatever purpose you want. When it comes to back-end development, we have a variety of options to choose from. If you are inclined to Python as a language, you have surely used or heard about Django, one of the most popular web development frameworks for this language.

What is Flask

This time I want to talk to you about another option. I'm going to tell you what Flask is and why we could choose it for our next project. Flask the most famous micro framework for Python Is Flask a framework or a micro framework? Surely if we have searched the Internet about this project we will have found it both ways. In this sense, we could think that the second is a variant of the first. And that, in short, Flask is a "minimalist" framework, which does not require other dependencies to do basic things. It also offers us a very low learning curve to begin to understand it. All this means that, if we decide to start developing Web solutions with Python, Flask can be a very good entry point. What can we do with Flask? The basic answer p It can be very simple: develop the server side of a Web application. How do I install Flask? First of all we must have Python installed (https://www.python.org/). Then, from the console or command line we can run the package manager: pip install Flask. Keywords API Framework

Keywords

API
Framework

Introducción

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:

```
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
```

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:

```
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
…
```

ARQUITECTURA

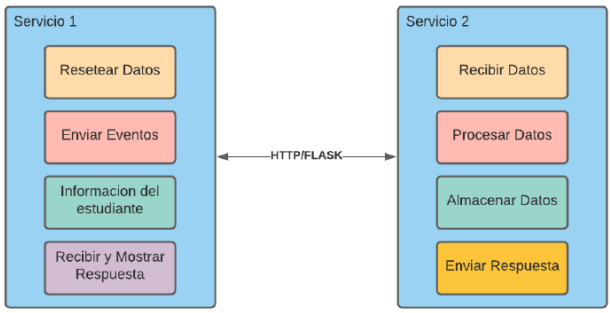


Figura 1: Arquitectura de la aplicación

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).

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

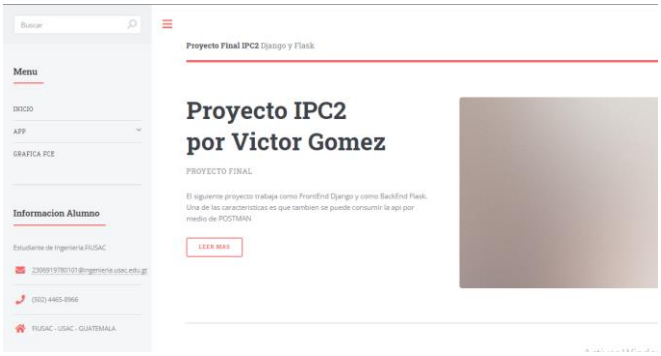
para visualizar la documentación del programa.
● Botón Enviar: Enviará los eventos del recuadro de texto a la Api para su posterior procesamiento.
● Botón Reset: Este botón mandará la instrucción a la Api para devolver al estado inicial la Api, es decir sin datos.

Servicio 2 – Backend

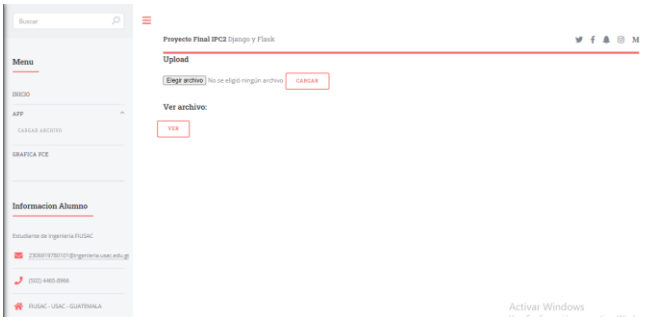
Este servicio consiste en una API que brindara servicios utilizando el protocolo HTTP, su funcionalidad principal es procesar los datos recibidos del servicio 1, luego de procesar los datos es necesario que estos sean almacenados en un archivo xml, este archivo está especificado en la sección de archivos 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.

NOTA: Durante la calificación de este proyecto, el Servicio 2 podrá ser consumido desde
otro cliente, por ejemplo, postman.

Anexos

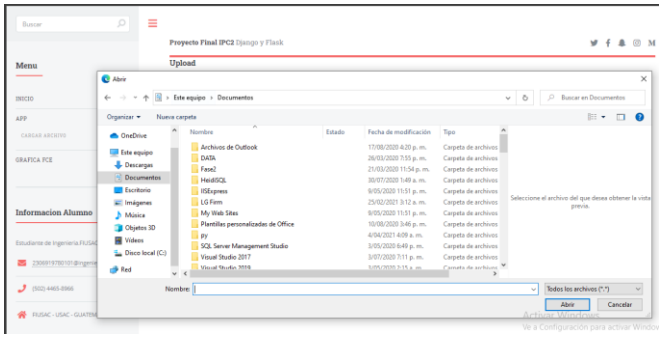


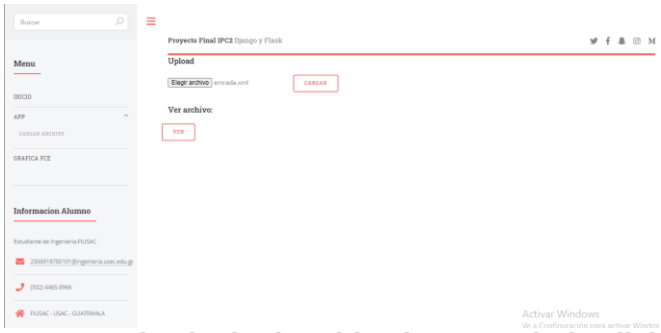
Pantalla Inicial



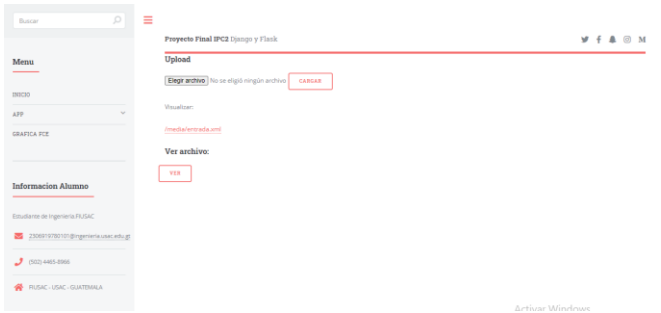
Pantalla de Carga de Archivo

En la siguiente imagen primero dar le clik a elegir archivo para poder seleccionar el archivo a trabajar.



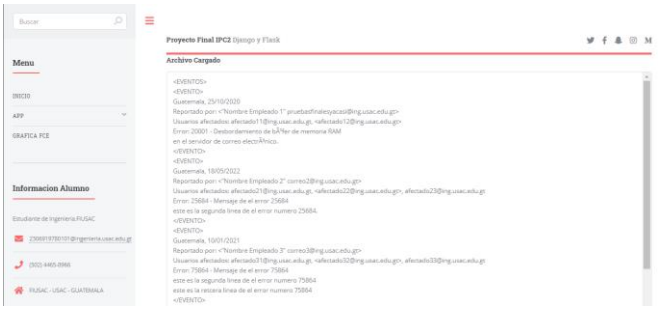


Despues de elegir el archivo hay que darle click en la opción de cargar.



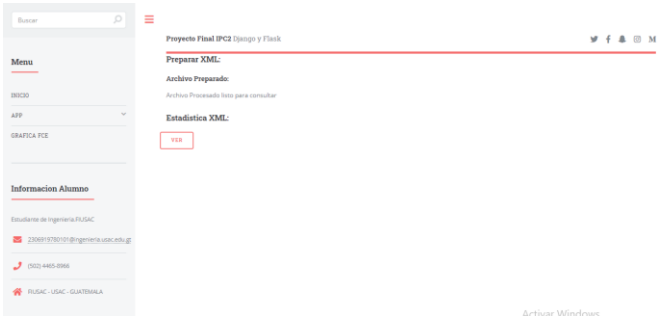
Se puede visualizar el archivo que fue cargado

Y después le damos clic en el botón ver



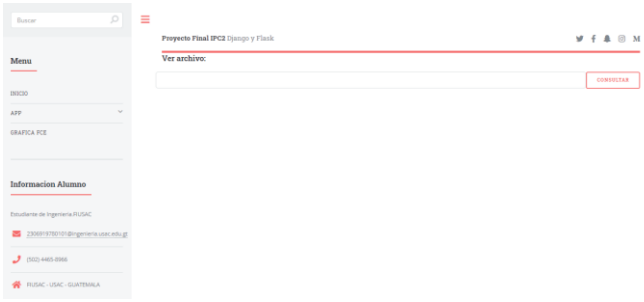
Visualizamos el archivo cargado

Bajamos y le damos click en enviar



Acá ya tenemos el archivo listo para poder generar el xml de estadísticas.

En la imagen anterior podemos visualizar el archivo ya como estadísticas.



Acá ingresamos el código de error para sacar la gráfica.

Conclusiones

- Se desarrolló una api que se pueda consumir desde diferentes clientes
- Se desarrolló una app en ambiente web
- Se aplicaron las diferencias entre backend y frontend
- Uso de expresiones regulares para poder modificar y limpiar el archivo de entrada
- Trabajo de archivos XML dentro de una API
- Visualizar las estadísticas.

Referencias bibliográficas

Curso Django Desde Cero | CRUD completo y Bases de Datos - YouTube

Reemplazar y sobrescribir en lugar de agregar (qastack.mx)

python — Analizando XML en Python con regex (it-swarm-es.com)

regex — ¿Coincidir con los saltos de línea -\n o\r\n? (it-swarm-es.com)

Cómo utilizar expresiones regulares (regex) en Python | robologs

Online Python Compiler - Python Examples

20.5. xml.etree.ElementTree — The ElementTree XML API — Python 3.7.0a2 documentation

django - Invalid block tag: 'static' - Stack Overflow

python - How do I call a Django function on button click? - Stack Overflow