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 3

201602734 – Wilfred Alejandro Barrios Ola

Resumen

El presente proyecto académico presenta, mediante el uso del lenguaje de programación Python y los frameworks Flask y Django, la idea principal del proyecto es dar un ejemplo de la implementación de una página web realizada con MVT y un API con diferentes métodos HTTP para su correcto consumo.

Adicional a esto se acompaña de material gráfico para que sea más sencilla la comprensión de los temas a tratar a continuación. De esta manera facilitar el entendimiento del uso de un API, programación orientada a objetos, frameworks Flask y Django.

Palabras clave
API, Flask, Django, Http, Framework

Abstract

The present academic project was created using Python as the programming language used to create a flask API and a web app with Django. The main idea of the project is giving an example of how to implement an API in a Web App created using Django with MVT.

Also, the present inform comes with graphic material to make easier the comprehension of the next topics. In that way make easier the understating of how to use an API, OOP, Flask and Django frameworks.

Keywords
API, Flask, Django, Http, Framework

Introducción

El desarrollo de un sistema web orientado a la arquitectura cliente-servidor está compuesta principalmente por 2 componentes esenciales.

La Web App que en este caso ha sido desarrollado con Django utilizando el patrón de diseño MVT y un API desarrollado con Flask.

De esta manera se dividen los trabajos correspondientes para cada elemento. El Web App se encarga únicamente de hacer uso de la información proporcionada y previamente procesada por el API y el API se encarga de manejar toda la información, sistemas de bases de datos, filtros así como procesar correctamente la información antes de enviarla al Web App o al cliente desde el cual se esté consumiendo el REST API.

MARCO TEÓRICO

REST API

Un API, acrónimo para application programming interface, es un conjunto de reglas que le indican a una aplicación o un grupo de dispositivos como pueden conectarse y comunicarse entre ellos.

Una REST API es simplemente un API que obedece los principios de REST, acrónimo para representational state architectural style.

Principios de diseño REST

Los principios de diseño REST son 6 y se enumeran a continuación:

1. Interfaz uniforme: Todas las respuestas proporcionadas por el API deben tener el mismo formato y estructura, independientemente del tipo de petición que se esté realizando.

2. Desacoplamiento cliente-servidor: Tanto el cliente como el servidor deben ser completamente independientes el uno del otro. Esto quiere decir que deben poder funcionar sin la necesidad de incluír al otro excepto en los casos que el otro está encargado de determinado proceso.

3. Statelessness: Un API no maneja estados, por lo tanto en cada petición debe enviarse la información completa. No habrá información previamente guardada para ayuidar a la petición que necesitamos enviar, toda la información necesaria para poder procesarla debe ser enviada.

4. Caché: Cuando sea posible, los recursos deben poder guardarse en caché, ya sea en el cliente o en el servidor, para no realizar peticiones innecesarias al servidor, ya que esto llevaría a un exceso de tráfico en el servidor, mayor necesidad de ancho de banda y generaría lentitud en el procesamiento al tener múltiples usuarios realizando peticiones de manera simultanea.

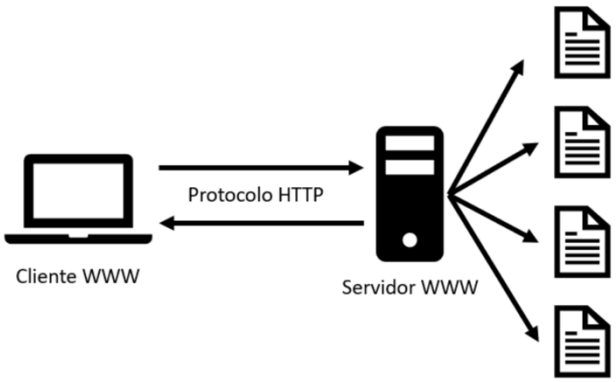
5. Arquitectura por capas: El API está realizado por capas, cada una comunicándose con otra para obtener toda la información que el cliente necesita sin arriesgar la seguridad.

6. Código bajo demanda (opcional): Cuando sea necesario debe ser posible ejecutar código específico desde una petición y este código debe ejecutarse únicamente cuando sea demandado. Un ejemplo de esto pueden ser los Java Applets.

HTTP

Http, acrónimo para hypertext transfer protocol, es un protocolo que se ejecuta en la capa de aplicación para la transmisión de documentos hipermedia, como HTML.

HTTP fue diseñado para la comunicación entre los navegadores y servidores web, aunque este protocolo puede ser utilizado para otros propósitos también. El protocolo HTTP obedece al modelo cliente-servidor, el cual indica que un cliente establece una conexión, realizando una petición a un servidor para posteriormente esperar una respuesta del mismo.



FLASK FRAMEWORK

Flask es un microframework desarrollado para Python basado en Werkzeug que permite crear aplicaciones web de todo tipo rápidamente.

Flask permite crear aplicaciones web, REST API's y demás de manera muy rápida en función de los conocimientos que se tengan sobre el lenguaje de programación Python.

DJANGO FRAMEWORK

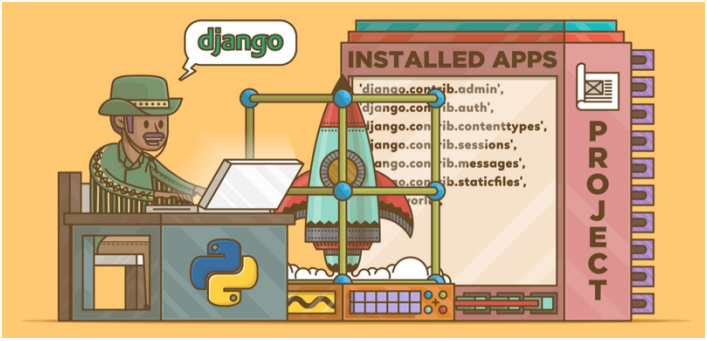
Django es un framework web de alto nivel desarrollado para python que ayuda en el desarrollo rápido y limpio de una aplicación web.

De acuerdo con su página web, Django hace más fácil construir mejores aplicaciones web, más rápido y con menos código.

Django ya te da un servidor web básico y es completamente de código abierto.

Django se nos describe con los siguientes puntos:

1. Ridículamente rápido: Django fue creado para ayudar a desarrolladores en la construcción de aplicaciones web permitiendo que se enfoquen en lo importante.
2. Completamente listo: Django incluye docenas de extras para manejar tareas comunes en el desarrollo de aplicaciones web. Desde la autenticación de usuarios hasta administración de contenido.
3. Tranquilizadoramente seguro: Django se toma la seguridad muy en serio y ayuda a los desarrolladores a evitar errores comunes, como inyección SQL.
4. Excesivamente escalable: Algunos de los sitios más utilizados del planeta usan la habilidad de Django para, rápidamente y de manera muy flexible, escalar para soportar las demandas de tráfico pesado.
5. Increíblemente versátil: Django es utlizado por compañías, organizaciones y gobiernos. Desde administración de contenido hasta redes sociales y computación científica.



CONCLUSIONES

Hoy en día se pueden encontrar muchas herramientas para el desarrollo de aplicaciones web. Y una gran herramienta son los REST API que facilitan la integración de la información en el cliente sin importar del lenguaje de programación que se ha utilizado para desarrollar ambas plataformas, pues son totalmente independientes.

Esto da inicio a una gran cantidad de tecnologías, como las serverless, cloud-computing y demás.

Para el desarrollo de aplicaciones es posible utilizar diferentes lenguajes de programación ya que gracias a los frameworks existentes facilitan el proceso de preparación del entorno para preocuparnos por lo más importante, desarrollar.

Desde frameworks como Django y Flask que son desarrollados para Python, hasta Express que es utilizado con Node.js o Spring que es trabajado de la mano con Java EE.

Independientemente de qué framework y qué lenguaje de programación sea elegido para desarrollar un sistema, habrá algo de lo que no vamos a salvarnos: las peticiones HTTP.

Se utilizará el protocolo HTTP para poder comunicarnos de la aplicación web al servidor, esto es ya la manera de trabajar que tiene un REST API que está sirviendo información a un cliente, sin importar si este es una aplicación web desarrollada por nosotros o únicamente es un cliente como POSTMAN desarrollado para conectarnos a API y llevar nuestras pruebas a cabo.

REFERENCIAS BIBLIOGRÁFICAS

IBM (2021) REST APIs
https://www.ibm.com/cloud/learn/rest-apis

Mozilla Contributors (2021) HTTP
https://developer.mozilla.org/es/docs/Web/HTTP

Flask (2021) Flask
https://flask.palletsprojects.com/en/2.0.x/

Django (2021) Django Overview
https://www.djangoproject.com/start/overview/