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.

ARQUITECTURA CLIENTE – SERVIDOR EN PYTHON

201800500 – Anderson Gerardo Zuleta Galdamez

Resumen

Una comunicación entre cliente y un servidor permite una conexión, en la que se centraliza los diversos recursos y aplicaciones con que se cuenta, y que los pone a disposición de los clientes cada vez que estos son solicitados. Es por eso la realización de este proyecto, una solución de una aplicación web que consume los servicios de una API-REST. Para dejar un contexto y se entienda, una API por sus siglas en ingles (Application Programming Interface) y REST por sus siglas en ingles (Representational State Transfer), el cual devuelve los datos que la aplicación web le solicita, creando así la arquitectura cliente-servidor.
En esta arquitectura fue desarrollado en 2 etapas, un backend utilizando Flask y un frontend utilizando Django. La comunicación entre los mismos, es el envio de unos archivos con extensión .csv a django y transformarlo a una estructura xml para luego que el backend reciba la información.

Palabras clave
API, Cliente, Servidor, Conexión, Datos, Backend, Frontend.

Abstract

Communication between the client and a server allows a connection, in which the various resources and applications available are centralized, and which makes them available to clients each time they are requested. That is why the realization of this project, a solution of a web application that consumes the services of an API-REST. To leave a context and be understood, an API for its acronym in English (Application Programming Interface) and REST for its acronym in English (Representational State Transfer), which returns the data that the web application requests, thus creating the client-server architecture.
In this architecture it was developed in 2 stages, a backend using Flask and a frontend using Django. Communication between them is sending files with a .csv extension to django and transforming it into an xml structure so that the backend receives the information.

Keywords
API, Client, Server, Connection, Data, Backend, Frontend.

Introducción

Una API de transferencia de estado representacional, es una interfaz de programación de aplicación, creado por el informático Roy Fielding, la cual se ajusta a los limites de la arquitectura Rest y permite la interacción con los servicios web de Rest. También es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar software de las aplicaciones. Se suele considerarse como el contrato entre el proveedor de información y el usuario. Donde se establece el contenido que se necesita del consumidor y el que requiere el producto.

Para el desarrollo de esta app web, y para el desarrollo de la mismo se uso un framework Django el cual maneja una programación MVC, ya que los templates y Python se desarrolla la lógica. Django, nuestro cliente, nuestro marco web que fomenta el desarrollo rápido y el diseño limpio.

Desarrollo del tema

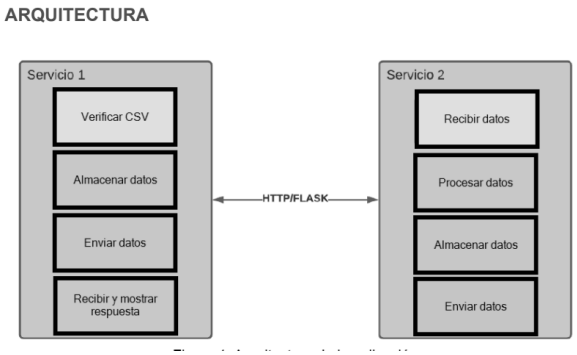

Figura 1. Arquitectura de aplicación

Fuente: Enunciado del proyecto no. 2

a) Servicio 1 – Frontend (Django)

Se utilizo el paradigma orientado a objetos (POO) para resolver el problema del almacenamiento de la información. Utilizando para el desarrollo del diseño, el lenguaje de programación Python, el cual nos provee una gran cantidad de librerías para poder trabajar dependiendo de las necesidades que se tengan, y es muy cómodo a la hora de codificar ya que su sintaxis es bastante simple. Se pueden resolver gran cantidad de problemas con este lenguaje y tiene la característica de ser multiplataforma. Esto quiere decir que una vez que tengamos nuestro proyecto terminado lo podremos ejecutar en cualquier sistema operativo, ya que este cuenta con la máquina virtual de Python, el que se encarga de interpretar el código fuente.

Para el desarrollo de este servicio se utilizo el framework Django, haciendo uso de sus templates y la arquitectura modelo, vista, template. El cual es muy cómodo al trabajar y organizar la información que se desea mostrar al usuario.

El frontend, nuestro cliente. Es el encargado de recibir 4 archivos diferentes con estructuras csv. Por lo que se requiere la traducción a una estructura xml, pero ojo la primera traducción es una parte que se muestra al usuario, para así el usuario pueda ser uso y pueda modificar para previo envió mandarlo a nuestro servidor, a nuestro backend.

- Componentes del cliente.

➢ Carga de archivo: consta de una ventana la cual a travez de forms se seleccionan los archivos con extensión CSV y que realizara la transformación hacia un archivo XML.

  De existir errores en el archivo CSV se muestra una template que avisa al usuario que algunos de los archivos no son posibles las transformaciones.

➢ Peticiones: Template que consta de diferentes opciones.

Opción mejores clientes, muestra al usuario a través de una grafica de barra los clientes que mas han comprado con la cantidad de quetzales gastados.

Opción juegos más vendidos, visualiza una gráfica en forma de pie con los juegos más vendidos con su año de lanzamiento.

Opción clasificación, muestra una grafica con la cantidad de juegos que existen por clasificación.

Opción cumpleaños, Consiste en dedicar a los clientes por medio de las fechas de nacimientos un orden por mes.

Opción listado de juegos, visualiza un listado de los juegos que ofrece.

- Ayuda:

Desplega 2 opciones, una para visualizar la información del creador del proyecto y la otra opción visualiza la documentación del programa.

- Botón enviar:

Envía los eventos del recuadro de texto al servidor por su posterior procedimiento.

Temlates del frontend.

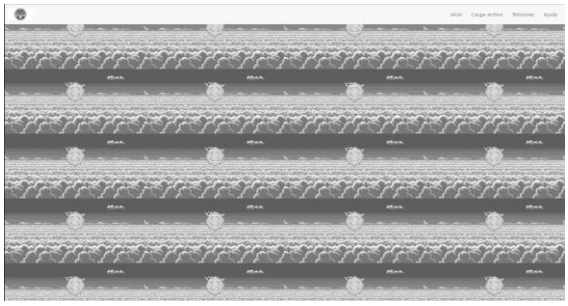

Figura 2. Template de la pagina de inicio

Fuente: elaboración propia.

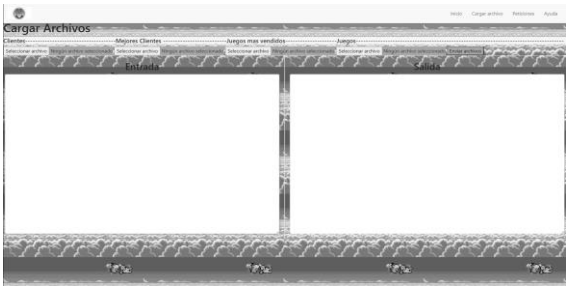

Figura 3. Template de la página de carga.

Fuente: elaboración propia.

b) Servicio 2 – Backend (Flask)

El Backend del sitio web consiste en un servidor, una aplicación y una base de datos. Se toman los datos, se procesa la información y se envía al usuario. Los desarrolladores de Frontend y Backend suelen trabajar juntos para que todo funcione correctamente.
Por ende, para la implementación de este servicio se hizo uso del framework Flask. Consiste en brindar servicios utilizando protocolos HTTP, su funcionalidad principal es procesar los datos recibidos del servicio del frontend, ósea de nuestro cliente Django, para luego de procesar los datos es necesario que estos sean almacenados en un archivo con estructura xml, este servicio también tiene la funcionalidad de devolver los datos que fueron almacenados para que sean mostrados en nuestro frontend. Nuestro servidor no muestra ningún template, como anteriormente dicho, solo almacena datos. Por eso es posible el uso de un servidor 2 que podrá ser consumido desde otro cliente como, por ejemplo, postman.

Conclusiones

La realización estructural de un cliente servidor, hace optima el manejo de información. Es un modelo muy flexible y adaptable al cual se requiera implementar. Esto nos permite aumentar el rendimiento asi como también, el envolvimiento de varias plataformas, base de datos, redes y sistemas operativos que puedan ser de diferentes distribuidores con arquitectura totalmente diferentes y funcionando todo al mismo

tiempo. Además, se puede considerar un sistema ventajoso en cuanto a la seguridad, ya que el servidor controla el acceso de los datos por lo que se necesita que el servidor autorice para que se pueda acceder a él.

Puesto que la utilización de un framework, es necesario aprender la base del mismo. El cual es un lenguaje muy amplio, pero como en este caso Django, muy enriquecido.

Referencias bibliográficas

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

Django documentación https://docs.djangoproject.com/en/3.2/

pypi. Flask. https://pypi.org/project/Flask/

pypi. Django. https://pypi.org/project/Django/

python.org. (s. f.). xml.etree.ElementTree — The ElementTree XML API — Python 3.9.2 documentation. python. Recuperado 5 de abril de 2021, de https://docs.python.org/3/library/xml.etree.elementtr ee.ht ml