DEPARTAMENTO DE INFORMÁTICA Y COMPUTACIÓN

TRABAJO Nº2

PROYECTO REST

COMPUTACIÓN PARALELA Y DISTRIBUIDA

1° SEMESTRE-2020

PROFESOR: SEBASTIÁN SALAZAR MOLINA

N° DE GRUPO: 9

RICARDO ABDÓN ALISTE GONZÁLEZ

DANIEL IGNACIO CAJAS ULLOA

RODRIGO ANTONIO CARMONA RAMÍREZ

08/08/2020

INTRODUCCIÓN

Dentro del mundo de la programación se encuentra un concepto que ha ganado una notoria popularidad en los últimos años, y también porque genera aportes al desarrollo de aplicaciones. Este término se le conoce como **interfaz de programación de aplicaciones** (API por sus siglas en inglés), que representa un conjunto de funciones o procedimientos utilizados por los programas informáticos para acceder a los servicios del sistema operativo, bibliotecas de software, u otros sistemas, es decir, la capacidad de comunicación entre componentes de software, dado que facilita la relación entre dos aplicaciones para el intercambio de mensajes o datos [1].

Los desarrolladores de este tipo de productos siempre debe tener presente algunas características fundamentales para puntuar la calidad de una API:

- Tiene que ser fácil de usar para el usuario
- Debe presentar estabilidad en su funcionamiento
- Asegurarse de estar documentado
- Garantizar la seguridad de los datos privados del usuario de la aplicación

El enfoque del documento se concentra en APIs de servicio web, que proporciona acceso a un servicio mediante una dirección URL en una red. Es importante que la información proporcionada por una API de servicio web se presente en un formato que sea comprensible para otras aplicaciones, ya que si no se perdería toda la funcionalidad de la API en sí misma. Uno de los sistemas más importantes, el cual es el tema central de este proyecto, trata del funcionamiento de la **transferencia de estado representacional o REST** [2].

El concepto REST (**RE**presentational **S**tate **T**ransfer) es una arquitectura software que se encarga de representar la transferencia de datos [2]. El lanzamiento de este nuevo sistema como protocolo de intercambio y manipulación de datos en los servicios de internet cambió por completo el desarrollo de software a partir de los años 2000.

Este nuevo enfoque de desarrollo de proyectos y servicios web fue definido por Roy Fielding, el padre de la especificación HTTP y uno los referentes internacionales en todo lo relacionado con la Arquitectura de Redes, en su disertación ‘Architectural Styles and the Design of Network based Software Architectures’. Podría decirse que tiene un estilo derivado de otras arquitecturas basadas en redes, lo cual lleva a considerarla más bien como un conjunto de arquitecturas.

Está basado en el principio cliente-servidor, teniendo que tener las peticiones del cliente su estado totalmente representado, donde se apoya en el protocolo HTTP para la transferencia de información entre máquinas. En la figura 1.1 se muestra un esquema de la estructura básica de una arquitectura REST.

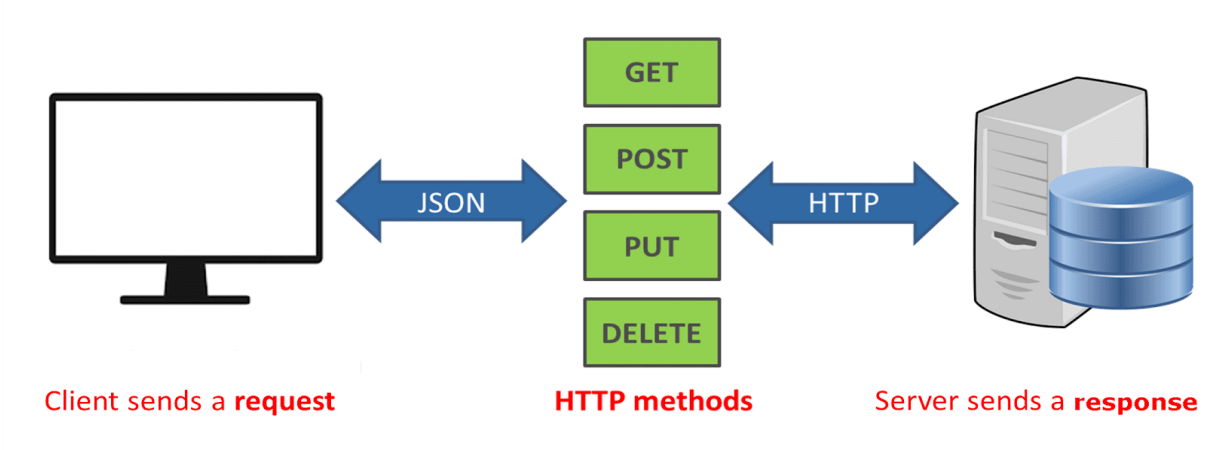

Figura 1.1: Estructura básica de la arquitectura REST

Su función es encargarse de tratar objetos como recursos que pueden ser creados y destruidos, teniendo para ello cuatro métodos básicos: post (para crear), put (para editar), get (para consultar y leer) y delete (para eliminar), los cuales son similares a los métodos CRUD (Create, Read, Update, Delete) de una base de datos. Por dar un ejemplo, un desarrollador trabaja en una aplicación para encontrar a las personas que lo siguen en Twitter, podría solicitar esta información a la API de Twitter: “Dame una lista de todas las personas que me siguen”. No hay diferencias si esa aplicación está en Ruby, Php, C++, u otro lenguaje porque REST transmite y recibe datos en un formato común (normalmente JSON o XML). Esto permite que dos aplicaciones totalmente separadas envíen y reciban datos [3].

Las ventajas que que ofrece REST para el desarrollo de software son las siguientes [1]:

1. Separación entre el cliente y el servidor. El protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.

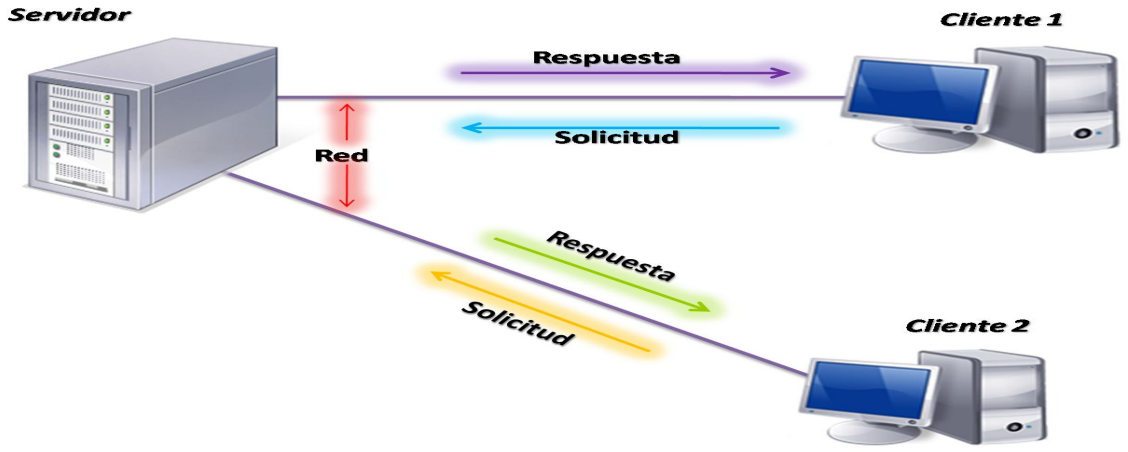

Figura 1.2: Funcionamiento de la arquitectura Cliente-Servidor

2. Visibilidad, fiabilidad y escalabilidad. La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back, lo cual convierte a las aplicaciones en productos más flexibles a la hora de trabajar.

3. La API REST siempre es independiente del tipo de plataformas o lenguajes. Tal como se destacó anteriormente, este tipo de API se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, y esto ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON.

RESOLUCIÓN DEL PROBLEMA

El proyecto REST será desarrollado en el lenguaje de programación Python 3 (de la versión 3.8 en adelante). El principal motivo de su utilización es que este lenguaje ya ha sido utilizado en anteriores proyectos de servicios web, dejando buenas expectativas o sensaciones por su buen rendimiento, pudiendo manejar enormes volúmenes de datos y responder en un periodo de tiempo muy corto. Además, es muy legible, dado que la sintaxis de Python es muy ordenada, muy parecida a la descripción de un pseudocódigo, lo cual facilita la comprensión del código escrito para el programador [4].

También soporta múltiples paradigmas de programación, incluyendo programación estructurada, orientada a objetos y funcional. Proporciona varias librerías y frameworks de gran utilidad en el desarrollo de aplicaciones dedicadas a servicios en línea. Asimismo, se encuentra entre los lenguajes de programación más utilizados, preferidos y relevantes, además de ser uno que posee mayor proyección dentro de los próximos años.

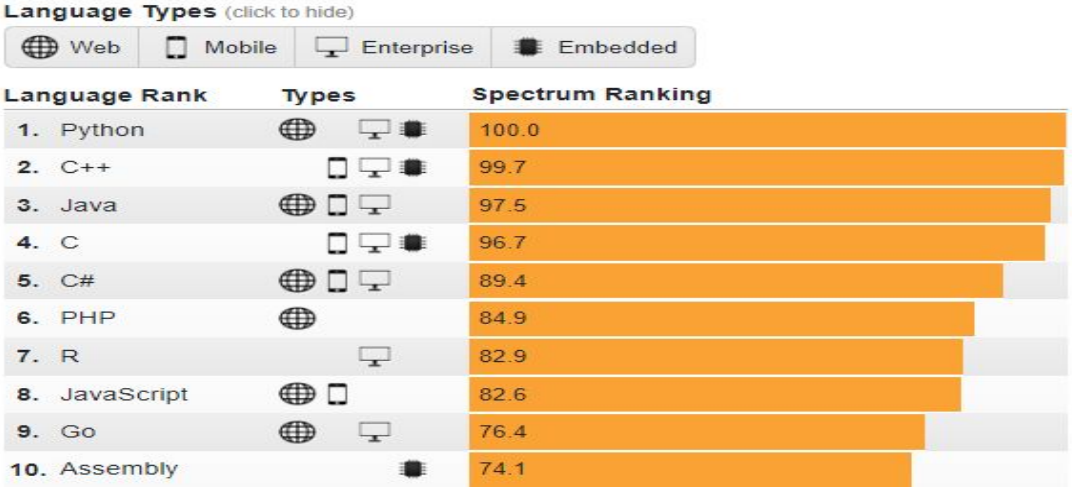

Figura 2.1: Ranking de IEEE Spectrum sobre los lenguajes de programación más utilizados en 2019.

En este programa se requiere obligadamente instalar o descargar de internet las siguientes librerías para crear el servicio de REST:

- Flask: Es un framework minimalista de Python que permite desarrollar servicios web en un corto periodo de tiempo. Incluye un servidor de desarrollo, un depurador y soporte integrado para pruebas unitarias [5]. Se prefirió usar este microframework por sobre otros como Django por ser más recomendable su uso en proyectos de menor envergadura, y por su sencillez a la hora de crear una API y aplicaciones web [8].

- Flask-httpauth: Es una extensión que simplifica el uso de la autenticación HTTP con las rutas de Flask, además de permitir el manejo de sistemas de seguridad para los servicios (en el caso de este trabajo, una verificación de usuario/contraseña) [6].

- Flask-Cors: Una extensión de Flask para manejar el Cross Origin Resource Sharing (CORS) [9], cuyo objetivo es proteger la carga y transferencia de datos entre servidores ajenos y los propios. Todos los datos deben provenir de la misma fuente,

es decir, corresponder al mismo servidor, lo que permite corroborar la fuente de los datos en los casos de servidores externos [10].

La construcción del proyecto REST fue basado en la siguiente estructura de desarrollo:

1.- Navegar por la red con el propósito de encontrar la mejor librería de Python para trabajar en el desarrollo de la API, en este caso, se decidió aplicar Flask por ser un framework apto en aplicaciones pequeñas y para cortos tiempos de construcción.

2.- Se implementó una función donde el estudiante pueda observar cuales son las mejores opciones para estudiar en la universidad, en otras palabras, dicho código fue escrito para comprobar las 10 mejores carreras donde ese alumno tiene el puntaje de postulación más alto.

3.- En caso de que el estudiante tenga una carrera donde tiene el puntaje suficiente para ingresar, el programa generará un bloque de código que le mostrará los datos (nombre, código, su respectiva ponderación para esa carrera, y cual seria su posición en el listado de ingresados) de dicha carrera, y se entregará por pantalla la posición en la que se encuentra comparado a los resultados de los otros postulantes. El orden se aprecia de forma ilustrativa en la siguiente imagen:

NEM	RANKING	MATEMATICAS	LENGUAJE	HISTORIA	CIENCIAS
480	500	400	500	400	450

COD. CARRERA	NOM. CARRERA	PONDERACION	POSICION
21041	Ing. Civil en inf. Men. Info.	490	80
21030	Ing. Informatica	488	90
20040	Bibliotenologo	491	91
22022	Trabajo Social	440	Lista de Espera: -3

Figura 2.2: Ejemplo ilustrativo de las acciones que realiza internamente el sistema

4.- Como se aprecia en la imagen, en caso de que el estudiante no le alcance para ingresar a la carrera que desee, el REST pondrá un mensaje que le indica al individuo que entrará a la lista de espera y en pantalla se mostrará la posición donde se encuentra con respecto a los demás postulantes.

5.- Se implementaron técnicas para detectar fallos cuando el usuario teclea datos incorrectos, por ejemplo, los puntajes ingresados no son valores numéricos, el código de carrera no existe o que no corresponde, etc. Si se llega a dar uno de estos sucesos, el programa los tomará como excepciones que no son válidas por el sistema.

6.- El siguiente paso fue crear el código de la función “career”, que tendrá la responsabilidad de entregar la información de la carrera que al usuario le interese indagar, escribiendo en la aplicación el código correspondiente a dicha carrera. Esta función sólo acepta un código por cada solicitud, es decir, si el usuario ingresa dos o más códigos de carreras, o directamente no digite ningún dato, la API lo toma como una excepción, y se mostrará por pantalla una notificación de que los datos no concuerdan (no son válidos por el sistema).

CODIGO
21041
21030
A03
EPSILON0443
20000

CODIGO	NOMBRE	NEM	RANKING	MATEMATICAS	LENGUAJES	CIENCIAS	HISTORIA	PRIMER MATRICULADO	ULTIMO MATRICULADO
21041	ING. CIVIL EN COMPU. MEN. INFO.	10	10	40	10	20	10	598	510
21030	ING EN INFORMATICA	10	10	35	10	25	10	580	500
EXCEPCION DE ERROR POR CODIGO INVALIDO									
EXCEPCION DE ERROR POR CODIGO INVALIDO									
EXCEPCION DE ERROR POR QUE CODIGO NO CONCUERDA CON REGISTROS									

Figura 2.3: Ejemplo ilustrativo de las respuestas del servidor para servicio Carrer

7.- El posterior servicio que se implementó fue el método “careers”, el cual realiza un proceso similar al de carrer, con la diferencia de utilizar palabras en lugar de códigos, y que este funciona con varias palabras a la vez y entregando varias carreras a la vez (de 1 a “n”, para ser más específicos). En caso de solo ser una frase, se realiza una búsqueda entre los nombres de carreras, donde esa frase concuerde con alguna de los nombres, y en el caso de ser múltiples palabras a buscar, realiza un proceso similar, variando en que registra las carreras que, además de poseer la palabra ingresada, también NO se encuentren ya registrada (si se reciben las palabras “civil” e “inge”, entrega todas las carreras civiles, y las carreras no civiles, pero que lleven inge en el nombre).

PALABRA
INGE

CODIGO	NOMBRE	NEM	RANKING	LENGUAJE	MATEMATICAS	ULTIMO MATRICULADO
21041	ING. CIVIL EN COMPU. MENC. INFO.	10	10	10	40	520
21030	ING. INFORMATICA	15	15	10	30	500

PALABRA
INGE
ARQUI

CODIGO	NOMBRE	NEM	RANKING	LENGUAJE	MATEMATICAS	ULTIMO MATRICULADO
21041	ING. CIVIL EN COMPU. MENC. INFO.	10	10	10	40	520
21030	ING. INFORMATICA	15	15	10	30	500
22022	ARQUITECTURA	10	15	15	35	498

Figura 2.4: Ejemplo ilustrativo de las respuestas del servidor para servicio Carrer

8.- Hacer pruebas y testeos del código escrito, corrigiendo los errores que se presenten durante la ejecución del programa (hipotéticamente hablando), para que así el servicio esté funcionando sin mayores problemas, y que los resultados que se entreguen por pantalla sean los valores esperados.

9.- Para comprobar el rendimiento de la API, se recurre al software llamado Postman, una plataforma dedicada al desarrollo de servicios web.

10.- Realizar la documentación correspondiente de la versión definitiva del código escrito en el archivo del lenguaje Python.

CONCLUSIÓN

REST permite crear servicios y aplicaciones de una forma más simple y convencional, dado que es capaz de transmitir prácticamente ***cualquier tipo de datos***, en otras palabras, no está restringido por el formato de transmisión o impone el uso de ciertas tecnologías, su propia concepción determina que un servicio de estas características debe ser un sistema uniformemente accesible [7], donde los recursos se desacoplan de su representación de forma que puedan ser accedidos en una variedad de formatos, como por ejemplo XML, HTML, texto plano, PDF, JPEG, JSON, etc. Por lo tanto, se puede considerar que es una arquitectura estandarizada y adecuada para crear APIs para servicios orientados a Internet.

Se puede destacar la flexibilidad que proporciona, sin importar el lenguaje o plataforma que el usuario intente acceder al servicio. Esto permite que una misma API REST sea consumida por varios clientes independientemente de la naturaleza de estos. Dicha característica proporciona fiabilidad y escalabilidad, ya que separa por completo al cliente del servidor, aunque cabe destacar que el intercambio de información de las respuestas tiene que realizarse en un formato con soporte (por lo general JSON o XML).

Al usar un lenguaje como Python, se logró desarrollar un programa más optimizado de lo esperado, ahorrando enormes cantidades de líneas de código en el proceso. También este proceso ha sido fructífero gracias a la librería Flask, un framework bastante útil para la creación de aplicaciones de menor magnitud, pudiendo acortar los tiempos de desarrollo de este proyecto. Además, la sintaxis ha permitido crear algoritmos bien ordenados, evitando malas prácticas de programación que no hubiera sido posible en otros lenguajes como JavaScript.

BIBLIOGRAFÍA

[1] BBVA, [En línea]. Available: https://bbvaopen4u.com/sites/default/files/ebook/bbva-open4u-ebook-101-apis-espok.pdf. [Último acceso: 17 Julio 2020].

[2] S. Plaza, N. Ramírez y C. Acosta, «E-Prints Universidad Complutense de Madrid,» [En línea]. Available: https://eprints.ucm.es/38686/1/Memoria_API%20de%20servicios%20web%20de%20accesibilidad.pdf. [Último acceso: 17 Julio 2020].

[3] Aprendiendo Arduino, [En línea]. Available: https://aprendiendoarduino.wordpress.com/2019/10/27/api-rest/. [Último acceso: 17 Julio 2020].

[4] A. Marzal, I. Gracia y P. García, «http://mmc.geofisica.unam.mx/,» 2014. [En línea]. Available:http://mmc.geofisica.unam.mx/femp/Herramientas/Lenguajes/Python/s93.pdf. [Último acceso: 12 Junio 2020].

[5] J. Muñoz, «OpenWebinars,» 17 Noviembre 2017. [En línea]. Available: https://openwebinars.net/blog/que-es-flask/. [Último acceso: 17 Julio 2020].

[6] Flask-HTTPAuth, [En línea]. Available: https://flask-httpauth.readthedocs.io/en/latest/. [Último acceso: 17 Julio 2020].

[7] Universidad de Alicante, [En línea]. Available: http://www.jtech.ua.es/j2ee/restringido/cw/sesion11. [Último acceso: 17 Julio 2020].

[8] D. Coss, «Nearsoft Jobs,» 1 Junio 2018. [En línea]. Available: https://blog.nearsoftjobs.com/crear-un-api-y-una-aplicaci%C3%B3n-web-con-flask-6a76b8bf5383. [Último acceso: 17 Julio 2020].

[9] Flask-Cors, [En línea]. Available: https://flask-cors.readthedocs.io/en/latest/. [Último acceso: 17 Julio 2020].

[10] Ionos, «Digital Guide,» 12 Diciembre 2019. [En línea]. Available: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/cross-origin-resource-sharing/. [Último acceso: 17 Julio 2020].

INFORMACIÓN ADICIONAL

- https://exploreflask.com/en/latest/views.html
- https://docs.python.org/3/tutorial/errors.html