UNIVERSIDAD TECNOLÓGICA METROPOLITANA
UTEM del Estado de Chile

Proyecto REST

Computación Paralela y Distribuida

Integrantes:

- Sion-jei Mamani León
- Janira Navarro Quilaqueo
- Shu-yi Wong Baxter

Profesor: Sebastian Salazar

Carrera: Ingeniería Civil en Computación Mención Informática

Código Carrera: 21041

Fecha: Sábado 08 de Agosto de 2020

Índice

Introducción

A continuación se presenta el problema de una aplicación REST, la cual se define como una interfaz entre sistemas que utiliza directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos en distintos tipos de formato (XML, JSON, etc), en el caso de esta API se utilizara json, debido a que es uno de los requerimientos para solución de esta.

También se indica cómo se solucionan las diferentes operaciones que se solicitan de la aplicación REST y las tecnologías que se utilizan para llevar a cabo la API, la cual consiste en entregar las ponderaciones de las carreras de la UTEM para Admisión 2020 y entregar las 10 carreras que se tienen mejor opcion para postular dependiendo de las ponderaciones del postulante.

Problema

La Universidad Tecnológica Metropolitana posee un conjunto de carreras, que tienen una serie de restricciones de ingreso. Por otro lado, los estudiantes postulan a la universidad rindiendo la Prueba de Selección Universitaria que junto al promedio de Notas de Enseñanza Media y al lugar que obtuvieron en la promoción de su cohorte en su etapa de educación secundaria, entregan un puntaje medible para postular a diversas carrera.

Se solicita desarrollar un servicio REST que entregue en función de los puntajes obtenidos por el estudiante, las 10 carreras en las que tiene mayores opciones de ingreso desde la que tiene mejor opción hasta la que tiene menor opción.

El servicio rest debe entregar las siguientes de operaciones:

- Consultar los puntajes de postulación para una carrera específica.
    - Datos de entrada por “Query param” o “Path param”: código de carrera.
    - Verbo para consumir el servicio: GET.
    - Código HTTP de salida exitoso: 200.
    - Datos de salida exitoso en formato JSON:
        - Nombre de Carrera.
        - Código de Carrera.
        - NEM.
        - Ranking.
        - Lenguaje.
        - Matemática.
        - Ciencias Sociales o Ciencias.
        - Puntaje Promedio.
        - Mínimo de Postulación.
        - PSU entre Lenguaje y Matemática.
        - Puntaje Mínimo Ponderado.
        - Vacantes.
        - Primer matriculado 2019.
        - Último matriculado 2019.
    - Código HTTP de salida de error: 4XX.
    - Datos de salida en caso de error en formato JSON:
        - Glosa Descriptiva

- consultar los puntajes de postulación para algunas (una o más) carreras en función de ciertos valores que pueden o no pueden estar presente.
    - Datos de entrada por “Query param” o “Path param”: nombre de la carrera.
    - Verbo para consumir el servicio: GET.
    - Código HTTP de Salida exitoso: 200.
    - Datos de salida exitoso en formato JSON (Como un listado de carreras):
        - Nombre de Carrera.
        - Código de Carrera.
        - NEM.
        - Ranking.
        - Lenguaje.
        - Matemática.
        - Ciencias Sociales o Ciencias.
        - Puntaje Promedio.
        - Mínimo de Postulación.
        - PSU entre Lenguaje y Matemática.
        - Puntaje Mínimo Ponderado.
        - Vacantes.
        - Primer matriculado 2019.
        - Último matriculado 2019.
    - Código HTTP de salida de error: 4XX.
    - Datos de Salida en caso de error en formato JSON:
        - Glosa Descriptiva.

- Consultar en base a los puntajes, las 10 carreras en las que mejores opciones se tiene para postular a la Universidad.
    - Datos de entrada por “Request Body”.
        - Nem.
        - Ranking.
        - Matemática.
        - Lenguaje.
        - Ciencias.
        - Historia
    - Verbo para consumir el servicio: POST.
    - Código HTTP de Salida exitoso: 200.
    - Datos de salida exitoso en formato JSON listado de opciones:
        - Código de Carrera.
        - Nombre de la Carrera.

        - Puntaje de Postulación.
        - Lugar tentativo, en función del primer y último matriculado 2019.
            - Para obtener cada cupo, se usa el puntaje del primer matriculado con el último y dividiéndolo con la cantidad de Vacantes.
            - Ejemplo: 21041
            - (673,65 - 539,35)/130 aproximadamente 1,03 para cada lugar.
    - Código HTTP de salida de error: 4XX.
    - Datos de Salida en caso de error en formato JSON:
        - Glosa Descriptiva.

Además el servicio debe permitir acceso CORS y a su vez debe tener algún mecanismo de
autenticación para consumir el servicio (como por ejemplo API-KEY, JWT, oauth, etc).

Solución

Para la creación de la API REST se utiliza “flask”, ya que es un framework que facilita los servicios web. La API se desarrolla con la información de las 28 carreras pertenecientes a la Utem, las cuales son almacenadas en un diccionario y en el mismo archivo que se desarrolla la API, debido a que no es necesario crear una base de datos y para un rápido manejo de la información.

En la autenticación se utiliza una extensión llamada “flask_httpauth” que permite realizar una autenticación básica de http y consumir el servicio flask, la cual se crean tres usuarios en el mismo archivo con el username y su respectiva contraseña, en donde al momento de consumir el servicio REST se realiza una verificación del usuario y su contraseña. los usuarios creados son: “user1”, “user2”, “user3” con las siguientes contraseñas: “rest1”, “rest2” y “rest3” respectivamente.

Para permitir el acceso CORS se utiliza el caso más simple, el cual se inicializa la extensión “Flask-Cors” y queda con argumentos predeterminados que entrega la extensión en todas las rutas.

Errores

Para el desarrollo de esta API se utilizan cuatro tipos de errores 4XX, los cuales son utilizados con un abort(4XX) en cada función.
Los errores son los siguientes:
Error 400: corresponde a que no se logró procesar la solicitud, por lo general es debido a que los tipos de datos son incorrectos.
Error 401: sucede porque no existe autorización.
Error 404: corresponde a que el recurso solicitado no existe o la url es incorrecta.
Error 405: es porque el método utilizado es incorrecto.

Operaciones

1. Operación

Consultar los puntajes de postulación para una carrera específica:

Esta consulta consiste en responder al cliente en base, al código de carrera que este ingrese y se genere en json los datos de postulación de la carrera específica con un GET.

Principalmente se realiza la búsqueda del código ingresado y se corrobora que el código corresponda a alguna carrera de la universidad. En el caso de ser así, el método retorna la carrera con sus respectivos datos en formato json, de lo contrario, manda el error correspondiente. Para lo anterior se crea una función que contiene la url y el tipo de método que se ingresa, en caso de que el método no sea GET, enviará un error 405.

2. Operación

Consultar los puntajes de postulación para algunas (una o más) carreras en función de ciertos valores que pueden o no estar presentes:

Esta consulta es similar a la anterior, ya que debe entregar en formato json la o las carreras que contengan cierta palabra ingresada por el usuario a través del método GET y que se encuentre en el nombre de la carrera, para lo cual se crea una función que recibe el parámetro ingresado y lo compara con los nombres de las 28 carreras de la universidad.

Para lograr comparar la palabra ingresada con los nombre se separa los nombres de las carreras con un split(), el cual guarda el objeto dividido en una lista, la cual finalmente se compara con la palabra ingresada por el cliente.
En caso de encontrar varias carreras que contengan cierto valor ingresado, se retornan todas en formato json, de lo contrario se realiza un abort() con el error correspondiente.

3. Operación

Consultar las 10 carreras en las que mejores opciones se tiene para postular a la universidad:

Esta última consulta el cliente entrega los puntajes a través de “request body” en formato json y con el método POST, en el cual entrega los puntajes de nem, ranking, lenguaje, matemática, ciencias e historia. La universidad al solicitar el puntaje de ciencias o historia, se opta por elegir la de mayor puntaje.

La función primeramente evalúa el puntaje mínimo para postular en la universidad, en caso de no cumplir con ello no se realiza la consulta, de lo contrario, se realizan las ponderaciones para las 28 carreras de la UTEM con for, y a la misma vez se calcula el lugar tentativo, el cual se obtendrá con el siguiente cálculo:

$$\text{valor tentativo} = \frac{\text{puntaje primer matriculado - puntaje ponderado del cliente}}{\text{valor por lugar}}$$

y que a su vez el “valor por lugar” se obtiene con los siguientes datos dependiendo de cada carrera:

$$\text{valor por lugar} = \frac{\text{puntaje primer matriculado - puntaje ultimo matriculado}}{\text{vacantes}}$$

Finalmente los datos a presentar en la consulta sobre cada carrera se guardan en un diccionario y estos se van agregando a una lista, la cual es ordenada de manera descendente dependiendo del lugar tentativo con un sorted(), para luego guardar en una lista las 10 mejores carreras que tiene para postular y retornarlo en un json.

Tecnologías

La elección de las tecnologías que se utilizaron en el proyecto se basan en la compatibilidad de los objetivos propuestos, asimismo, facilitando la implementación de ello acorde a los conocimientos del equipo de trabajo. Entre las tecnologías de desarrollo y testing se tienen herramientas de alta competitividad y simplicidad, las cuales son:

Lenguaje De Programación Python

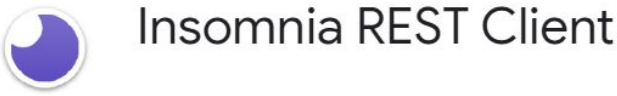
Python es una tecnología que tiene la capacidad de admitir métodos de programación estructurados y funcionales con un formato simple, una sintaxis muy definida y pocas keywords. En cuanto a sus librerías, estas son bastante grandes, variadas, portables y compatibles con Windows, Linux y Mac. Y lo mejor es que se trata de un lenguaje backend de código abierto (opensource), requisito que requiere el desarrollo de este proyecto.

Por otro lado, existen excelentes frameworks y bibliotecas orientadas al desarrollo de APIs REST y sistemas de comunicación basados en HTTP, lo que es perfecto para la construcción de microservicios. De esta manera Python se posiciona en una alternativa muy interesante al ser un lenguaje moderno y muy potente para el desarrollo rápido de estos microservicios.

Desarrollo de API Flask

Flask es un microframework escrito en Python y concebido para facilitar el desarrollo de aplicaciones web, es decir, páginas web dinámicas, APIs, etc. bajo el patrón MVC.
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.
Algunas de las características que incentivaron su uso fueron el depurador y soporte integrado para pruebas unitarias: Si tenemos algún error en el código que se está construyendo se puede depurar ese error y se puede ver los valores de las variables. Además no necesita una infraestructura con un servidor web para probar las aplicaciones sino que de una manera sencilla se puede correr un servidor web para ir viendo los resultados que se van obteniendo. Por otro lado, el controlador recibe todas las peticiones que hacen los clientes y de esta forma se determina que ruta está accediendo el cliente para ejecutar el código necesario.

Pruebas de Desarrollo en Insomnia



Insomnia es un cliente REST multiplataforma, con una interfaz clara y sencilla que sirve para realizar consultas curl contra una API. Algunas de las funcionalidades que posee, facilitan enormemente el trabajo de pruebas. Esta página, nos permite realizar consultas, para comprobar el correcto funcionamiento de nuestra aplicación. Por otro lado, tiene herramientas de autentificación para la variedad de sistemas y protocolos para trabajar con una determinada API REST.

Conclusión

Tener este tipo de conocimiento es una herramienta importante ya que el protocolo de intercambio y manipulación de datos en los servicios de internet cambió por completo el desarrollo de software y más el ámbito de aplicaciones backend. Por este motivo, la mayoría de las empresas o aplicaciones dispone de una API REST para creación de negocio y/o proyectos.

A través del desarrollo de esta aplicación REST, se logró conocer y entender los conceptos de funcionamiento de protocolo HTTP, además del funcionamiento de aplicaciones con protocolo sin estado, mecanismos asíncronos y funcionamiento directamente REST. Asimismo, fue importante comprender el mecanismo de cabecera CORS, gracias a esto es posible permitir el intercambio de recursos desde un dominio distinto a otro, además las posteriores consultas GET y POST, y sus respectivas respuesta a solicitudes.