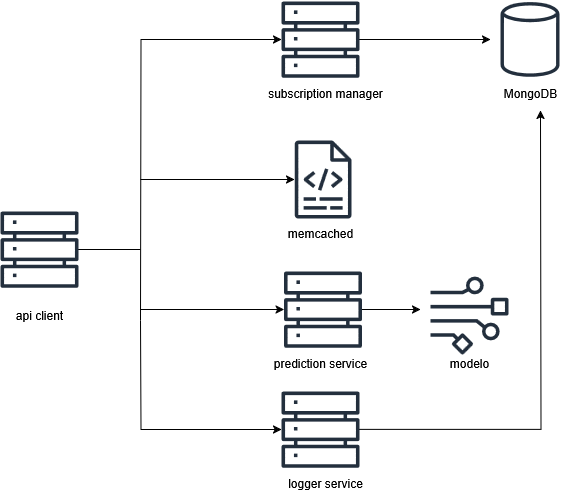
# Toma de decisiones Topicos 2 2024

## Integrante: Gamal Lautaro Lascano Hassen

Diagrama de arquitectura:



El diseño de este sistema consiste de cuatro servicios y dos componentes. Los cuales se explicarán a continuación.

Api Client: Este servicio funcionara como un API Gateway, el cual manejara los requests que llegan al sistema, y funcionara como un rate limiter, y como un autentificador con respecto al resto del sistema. Al realizar un request contra los endpoints expuestos para este servicio, se realizará un logging de ellos, y se procedera a enviar al logger service.

Subscription Manager: Se encargará de manejar los usuarios o tokens, y nos permitira saber el nivel del usuario requerido. Si no se encuentra en la base, ocurrira un 404 not found.

Memcached container: Es un componente que implementa un key-value store, que en nuestro caso funcionara como una cache. Tiene dos usos. El primero, es el de almacenar los usuarios que han hecho requests recientemente, y se guardaran en la cache. Con esto, tenemos acceso rapido al nivel de subscripcion de cierta api key. El segundo uso es de rate limiter, y se implemento de la siguiente forma. Al realizar la primera request, se agrega la api key con un prefijo que identifica al mecanismo de rate limiting, y se guarda en la cache con un ttl de 1 minuto. Al realizar requests subsiguientes, se usará la funcion incr, la cual incrementa el valor de esta key. Al llegar al valor establecido por su subscripcion, se activará el rate limiter.

Logger Service: Este servicio logeara cada una de las operaciones que pueden pasar en la api client en el momento que suceda una request. Al hacer el llamado a la funcion de log en Api Client, se grabará el execution time, la api key que ejecuto la funcion, el nivel de subscripcion usado, el REST method usado, el nombre del endpoint, y el status code que va a ser devuelto.

MongoDB container: Base de Datos No Relacional. Se encargará de almacenar los api keys, junto con su nivel de subscripcion, en conjunto a todos los logs grabados.

## Ejecucion del Projecto

Nuestra aplicación se podra ejecutar utilizando el comando “docker-compose up” estando localizado en el directorio raiz de la aplicación.

## Entrenamiento del Modelo

Para realizar el entrenamiento del modelo, se debera tener instalada en su instalacion de python la librería de pykeen. Se debera ejecutar el notebook train\_transH.ipynb y asegurarse de que se esta usando la librería compatible con CUDA si se esta ejecutando en una placa grafica geforce RTX o GTX, ya que los tiempos de ejecucion disminuyen una gran cantidad usando CUDA para entrenar el modelo.

## Funcionamiento del servicio

El servicio toma un array de enteros, y se hara la comparacion usando los ids de propiedades a comparar. El resultado nos dara la probabilidad de que ambas propiedades sean similares.