# Entrega 4: Desarrollo de una Web API utilizando MongoDB y Flask

Esta entrega busca desarrollar una Web API que será consumida por su aplicación en PHP en la siguiente entrega. Esta API deberá ser desarrollada con el *framework* Flask, y su base de datos almacenará información utilizando MongoDB. En concreto, en esta entrega deberán:

- Importar datos a MongoDB.
- Extender un código base de una aplicación en Flask para hacer las consultas deseadas y retornar los JSON respectivos.
- Implementar rutas que realicen *requests* de tipo `GET`, `POST` y `DELETE`.

## Detalles Académicos

En esta entrega, comenzarán a desarrollar una Web API en su computador personal, corriendo la aplicación en *localhost*. Esta aplicación deberán entregarla junto a un *readme* para que los ayudantes puedan replicar su ejecución. Además, existe una aplicación base. Los grupos pueden extender esta aplicación según sus necesidades.

## Introducción

El desarrollo de tu aplicación con servicios de turísmo está funcionando de maravilla. Esto ha permitido que se forme una comunidad muy activa de gente que quiere utilizar la aplicación y compartir sus experiencias. Para incentivar la interacción entre usuarios, te pidieron que integraras un sistema de mensajería para tu aplicación. Para esto, vas a tener que desarrollar una API que permita el intercambio de mensajes entre usuarios.

## Desarrollo de una Web API

Se debe realizar una Web API que exponga en formato JSON los mensajes. Además, se debe poder insertar mensajes y borrar mensajes. Se te van a proveer datos de mensajes, pero

los datos de los usuarios deberás cargarlos tú a la API. Para esto, queda a tu criterio cómo portar los datos desde la base de datos relacional a la base de datos MongoDB.

## Rutas de tipo `GET`

La API debería contar con al menos las siguientes rutas de tipo `GET` para trabajar sobre los mensajes:

- Una ruta que al recibir el `id` de un mensaje, obtenga toda la información asociada a ese mensaje.[1]
- Una ruta que al recibir el `id` de un usuario, obtenga toda la información de ese usuario, junto con los mensajes emitidos por él.
- Una ruta que al recibir el `id` de dos usuarios, obtenga todos los mensajes intercambiados entre dichos usuarios.

Además, la API debe poder permitir obtener mensajes mediante búsqueda por texto. Se debe poder consultar por:

- Agregar una o más frases que sí o sí deben estar en el mensaje.
- Agregar una o más palabras que deseablemente deben estar, pero no son necesarias.
- Agregar un conjunto de palabras que no pueden estar en el mensaje.

Para la búsqueda por texto sobre los mensajes, se debe poder consultar sobre todos los mensajes en la base de datos, o sobre los mensajes emitidos por un usuario en particular.

## Rutas de tipo `POST` y `DELETE`

La API debe tener la siguiente ruta de tipo `POST`:

- Dado dos *ids* de usuarios $i$ y $j$, una ruta que inserte un mensaje a la base de datos que está enviando el usuario $i$ al usuario $j$.

y debe tener la siguiente ruta de tipo `DELETE`:

- Dado un *id* de mensaje, se debe eliminar ese mensaje.

[1] Los mensajes no tienen un *id* en el dataset entregado. Por lo que puede (1) agregar un id al documento o bien (2) utilizar el `_id` de MongoDB.

## Detalles adicionales

El código de su aplicación debe ser subido en una carpeta comprimida a `bases.ing.puc.cl`, al servidor del grupo con el menor índice. Para facilitar la corrección se solicita indicar a los ayudantes en un `readme` toda la información necesaria para comprender de manera más rápida cualquier aspecto de la entrega. Es **obligación** que el `readme` contenga la información para correr la aplicación. Los ayudantes se reservan el derecho a descontar décimas en una entrega en la que se haya dificultado la corrección.