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 en **Python** con el *framework* Flask, y los datos serán almacenados utilizando una base de datos remota de 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 soporten *requests* de tipo `GET`, `POST` y `DELETE`.

Detalles Académicos

La forma de desarrollar esta Web API es ejecutando la aplicación en su computador personal, específicamente en *localhost*. Esta aplicación deberán entregarla de forma comprimida y junto a un *readme* para que los ayudantes puedan replicar su ejecución. Además, existe una aplicación base que te van a mostrar en la ayudantía de esta entrega. Los grupos pueden extender esta aplicación según sus necesidades.

Introducción

¡Ha habido un robo de una de las obras de arte más importantes en el mundo! Se trata de la consagración de Napoleón[1], obra de arte que estaba ubicada en el Museo del Louvre. La policía internacional ha interceptado una serie de mensajes entre los sospechosos del robo, pero actualmente no sabe como procesarlos. Ya que tú equipo es experto en obras de artes y en manejo de datos, te han contactado para que los ayudes. La idea es que al final de proyecto, además de tener tu servicio de turismo y arte, puedas encontrar a los criminales.

[1] Sí, se robaron una obra de 10 metros de ancho y salieron por la puerta principal con ella.

Desarrollo de una Web API

Se debe realizar una Web API que exponga en formato JSON los mensajes y la información de los usuarios. Además, se debe poder insertar y borrar mensajes. Se te van a proveer datos de mensajes y usuarios y tú debes cargarlos a una base de datos MongoDB, a la cual se conectará tu aplicación en Flask.

Rutas de tipo `GET`

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

- Una ruta `/messages` que entregue todos los atributos de todos los mensajes en la base de datos.
- Una ruta `/messages/:id` que al recibir el `id` de un mensaje, obtenga toda la información asociada a ese mensaje. El id es un identificador numérico entero, distinto al id alfanumérico que utiliza Mongo.
- Una ruta `/users` que entregue todos los atributos de todos los usuarios en la base de datos.
- Una ruta `/users/:id` que al recibir el `id` de un usuario, obtenga toda la información de ese usuario, junto con los mensajes emitidos por él.
- Una ruta `/messages` que al recibir en la URL los parámetros `id1` e `id2` de dos usuarios, obtenga todos los mensajes intercambiados entre dichos usuarios (en ambas direcciones). Por ejemplo, `/messages?id1=57&id2=35`

Rutas de búsqueda por texto: Además, la API debe permitir obtener mensajes mediante búsqueda por texto. La ruta será `/text-search` y las funcionalidades que se deben soportar son:

- Hacer una búsqueda por texto simple. Esto es, buscar una o varias frases que deseablemente deben estar en el mensaje.
- Buscar por una o varias frases que sí o sí deben estar en el mensaje. Recuerde que para esta frase se utilizan comillas dobles en Mongo.
- Especificar una lista de palabras que no pueden estar en el mensaje.
- Especificar el *id* de un usuario para filtrar aquellos mensajes que este usuario haya emitido y cumplan los criterios anteriores.

Estos 4 criterios serán enviados a la API en el *body* de la *request* mediante un JSON con la siguiente estructura:

```
{
  "desired": ["Hola amigo mío", "Cuánto tiempo ha pasado"],
  "required": ["Vamos a robar", "Robar una pintura"],
  "forbidden": ["broma", "trollear"],
  "userId": 5
}
```

Ruta de tipo `POST`

- Una ruta `/messages` que reciba los atributos del nuevo mensaje mediante un JSON en el *body* de la *request*. Si se reciben todos los atributos y estos son válidos, se debe insertar el nuevo mensaje en la base de datos, asignándole un id numérico único.

Ruta de tipo `DELETE`

- Una ruta `/message/:id` que reciba el *id* de un mensaje y lo elimine de la base de datos.

Detalles adicionales

Conexión a la BD remota de Mongo

La base de datos está escuchando conexiones en el servidor **gray.ing.puc.cl**, puerto 27017. Cada grupo tiene como nombre de usuario y contraseña el texto `grupoXX`, donde `XX` es su numero de grupo. Además, cada grupo tiene su propia *database* con el mismo nombre que su usuario.

Para más detalles sobre cómo conectarse pueden recurrir al tutorial en la Wiki del repositorio. En ese tutorial hay enlaces a las documentaciones que contienen los comandos que soporta la DB. Recuerden cambiar su contraseña para evitar accesos no deseados (la integridad de sus datos es su responsabilidad).

Aclaraciones generales

- La api debe ser desarrollada y ejecutada utilizando un entorno virtual (como pipenv o virtualenv). De lo contrario no será corregida. (Ver tutorial en la Wiki del repositorio)
- Pueden testear su api enviándole solicitudes mediante Postman. Durante el transcurso de la entrega se les proveera una colección de consultas de Postman para que testeen su api.
- Deben respetar el nombre las rutas solicitadas. Otras rutas no serán corregidas.
- Todas las consultas deben retornar siempre en formato JSON.

- Las *request* que lleven contenido en el *body* serán de tipo *application/json*.
- Por regla general, cualquier ruta que reciba `ids` como parámetros debe chequear que el id exista. Si no existe, se debe retornar un error amigable al usuario (no un `INTERNAL SERVER ERROR`).

Aclaraciones sobre la Búsqueda de Texto

- Los 4 criterios deben aplicarse en forma simultánea sobre los mensajes, utilizando la funcionalidad *text search* de Mongo.
- La presencia o ausencia de los 4 criterios es totalmente independiente. Cualquiera podría no estar o estar como una lista vacía. Incluso podría llegar 1 solo criterio.
- Si no llega *body* o el *body* es un diccionario vacío, se deben retornar todos los mensajes.
- Si solo se entregan palabras prohibidas, la busqueda de texto de mongo no entregará resultados, así que uds deben resolver cómo entregar todos los mensajes excepto aquellos que contengan palabras prohibidas.
- No deben preocuparse por las mayúsculas o tíldes, ya que la busqueda será exacta en este sentido.

Aclaraciones sobre las rutas POST y DELETE

- Todos los atributos del nuevo mensaje son requeridos, por lo que si alguno falta en el *body* o bien no llega *body*, el mensaje no debe ser insertado y se debe responder al usuario con un error que indique qué atributo falta o no es válido.
- El ID del nuevo mensaje deben generarlo ustedes antes de insertarlo en la BD. Deben preocuparse de que no coincida con ningún otro.
- Al intentar eliminar un mensaje, debe manejarse el error de que el id no exista.

Instrucciones de Entrega

- La entrega es para el **viernes 19 de junio a las 20:00 horas**.
- Los archivos de su aplicación deben ser entregados en una carpeta comprimida llamada `api.zip`, la que debe ser subida al servidor `bases.ing.puc.cl`, específicamente en la carpeta `Entrega4` 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.