Arquitectura
Servicio de geolocalización de documentos territoriales
DocLoc

Contexto

Para el equipo de Investigación y Desarrollo (I+D) de Sophia Language Technologies, quienes proporcionan bases de datos de documentos territoriales para sus clientes, están en la necesidad de un software accesible a través de una API. Este software permite procesar documentos textuales, como noticias de prensa, y calcular coordenadas de latitud y longitud para localizar el documento en función de su contenido y significado, como nombres de comunas u otras entidades geográficas. El software utiliza modelos de lenguaje y técnicas de procesamiento automático del lenguaje, y se integrará en la arquitectura de software propietaria de la empresa.

Objetivo

El objetivo del presente documento es explicar de manera sencilla, la arquitectura para la solución propuesta de tal manera que sea entendible por todos los futuros desarrolladores que quieran seguir el desarrollo de la API una vez terminado el MVP.

Diagramas

Secuencia

Muestra la secuencia de interacciones entre objetos a lo largo del uso de la API

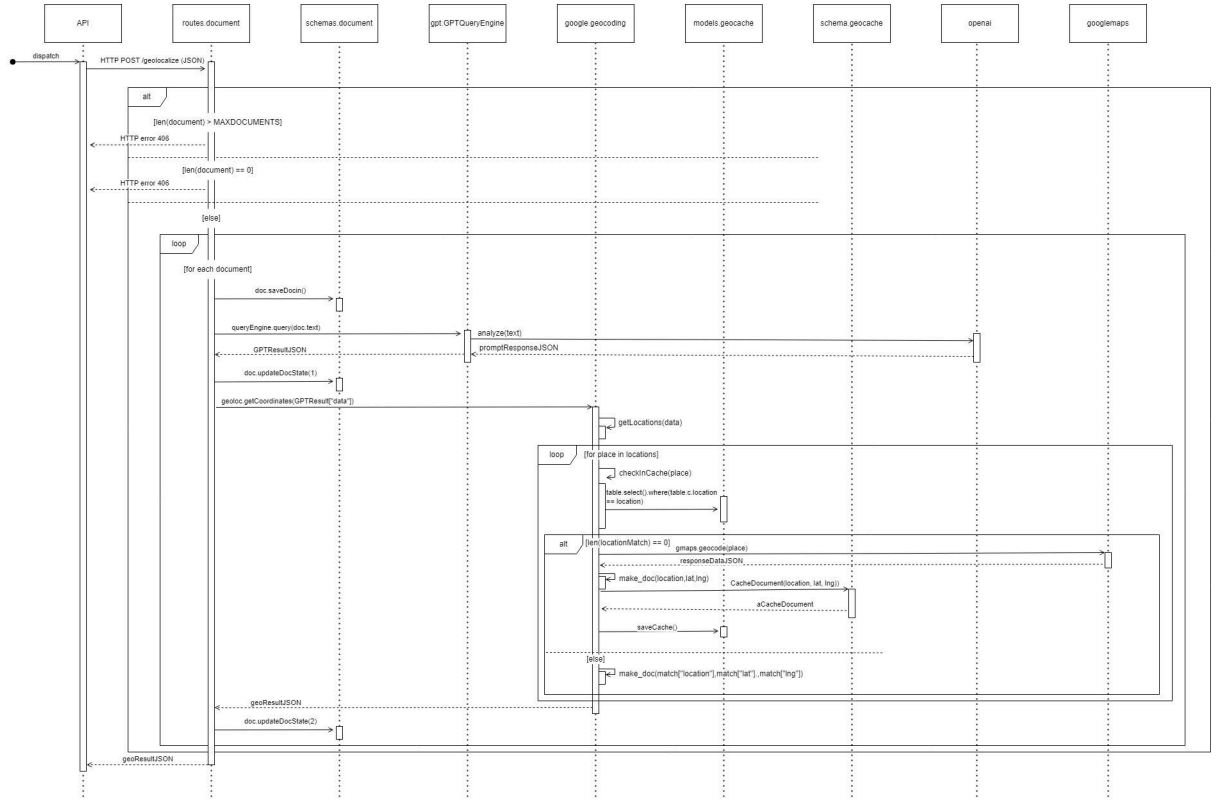

Diagrama de secuencia. Figura 1

Clases

Estructura estática de nuestra API, se denotan las relaciones entre nuestras clases (Document, GPTQueryEngine, Geocoding y CacheDocument) y sus atributos/métodos.

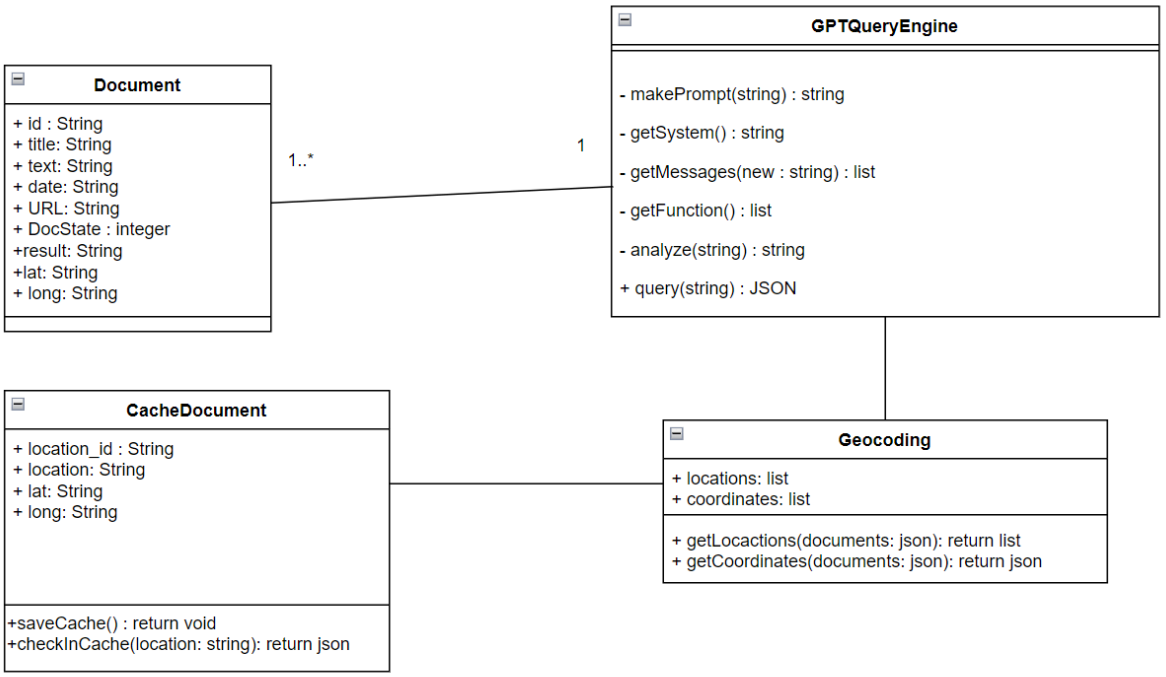

Diagrama de clases. Figura 2

GPTQueryEngine:

Clase encargada de procesar el cuerpo del documento utilizando gpt 3.5 turbo 4k (con soporte de function-calling). Esta analizará el documento y encontrará las localidades mencionadas y un breve resumen del evento relacionado a la mención.
En la clase se encuentran las siguientes funciones:

- **-query:** Única función pública de la clase. Permite realizar peticiones utilizando el cuerpo del texto y devolviendo un JSON.
- **-analyze:** Ensambla la petición utilizando diversas funciones, luego la envía y retorna el resultado organizado.
- **-getFunction:** Almacena la función para utilizar function-calling en la petición
- **-getMessages:** Ensambla y retorna el mensaje para la petición.
- **-makePrompt:** Retorna el cuerpo de texto que va como mensaje de usuario en la petición.
- **-getSystem:** Retorna el rol del sistema en la petición.

CacheDocument:

Clase encargada de definir la estructura del “caché de localidades”, esto es porque:
Para encontrar la latitud y longitud de una localidad, se utiliza la API de Google que consume créditos que se traducen en dinero.
Para aminorar este problema, se creó una base de datos que funciona como caché de localidades, de esta manera, lugares que ya han sido consultados estarán en la base de datos y no será necesario volver a enviar una petición a la API de google.

- **location_id:** Id único y autoincremental que se asigna al ser guardado en la base de datos.
- **location:** Nombre de la localidad.
- **lat:** Latitud de la localidad.
- **long:** Longitud de la localidad.
- **+saveCache:** Función encargada de guardar la localidad en la base de datos de caché.
- **+checkInCache:** Función encargada de verificar si la localidad que se quiere buscar está guardada en la base de datos, si está, entonces entrega su latitud y longitud, si no, la busca en la API de google.

Document:

Se centra en el objeto documento que será tratado.
donde se le asigna un id único que se asigna al ser guardado en la base de datos.

- **id:** Id único y autoincremental que se asigna al ser guardado en la base de datos.
- **title:** Título tomado de la noticia que se quiere tratar.
- **text:** Cuerpo de la noticia que se quiere tratar.
- **date:** Fecha en la que el documento fue escrapeado.
- **URL:** Enlace de la noticia que se quiere tratar.
- **DocState:** Estado actual del documento, donde:
    - 0 : Documento inicial recibido y guardado en la base de datos correctamente.
    - 1: Documento tratado por GPTQueryEngine.
    - 2: Documento Geolocalizado (Clase Geocoding)
- **result:** Se inicializa en nulo, y al ser tratado por GPTQueryEngine, se añade un json con las localidades encontradas y con un breve resumen de donde se encuentra.
- **lat:** Se inicializa en nulo y al ser tratado por Geocoding se rellena con la latitud encontrada en el documento.
- **long:** Se inicializa en nulo y al ser tratado por Geocoding se rellena con la longitud encontrada en el documento.

Componentes

Interacción entre los componentes de nuestra API, denota las interacciones y organización en ejecución.

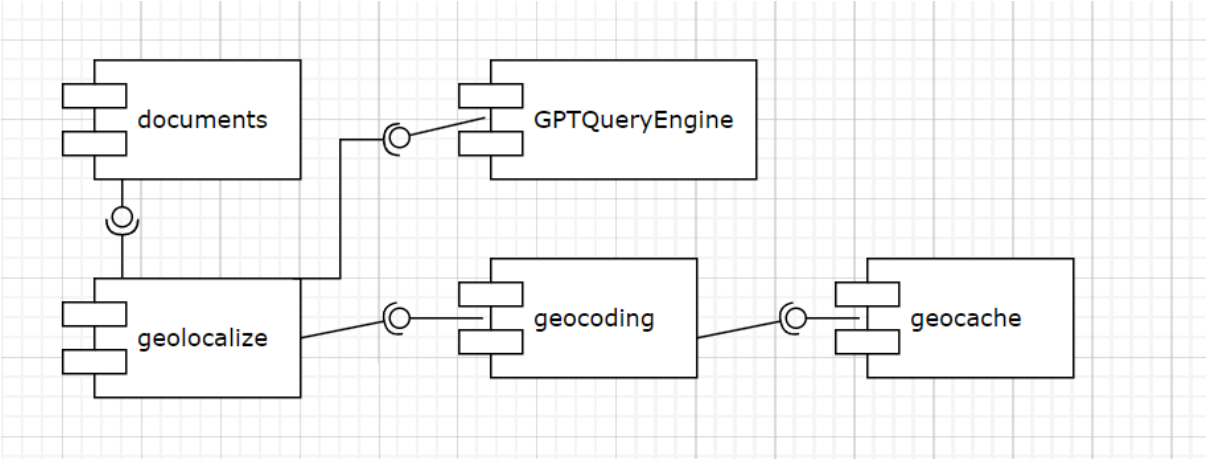

Diagrama de componentes. Figura 3

Persistencia

Modelado de cómo los objetos se almacenarán en la base de datos

Documents			
PK	**id**	int	NOT null
	title	varchar(255)	NOT null
	text	text	NOT null
	date	varchar(255)	NOT null
	url	varchar(255)	NOT null
	state	int	NOT null
	result	text	null
	lat	varchar(50)	null
	long	varchar(50)	null

geoCache			
PK	**id**	int	NOT null
	location	varchar(255)	NOT null
	lat	varchar(50)	NOT null
	long	varchar(50)	NOT null

Diagrama de persistencia. Figura 4

Despliegue

Nuestra arquitectura física del sistema, aquí mostramos como los componentes se desplegaran en el hardware elegido.

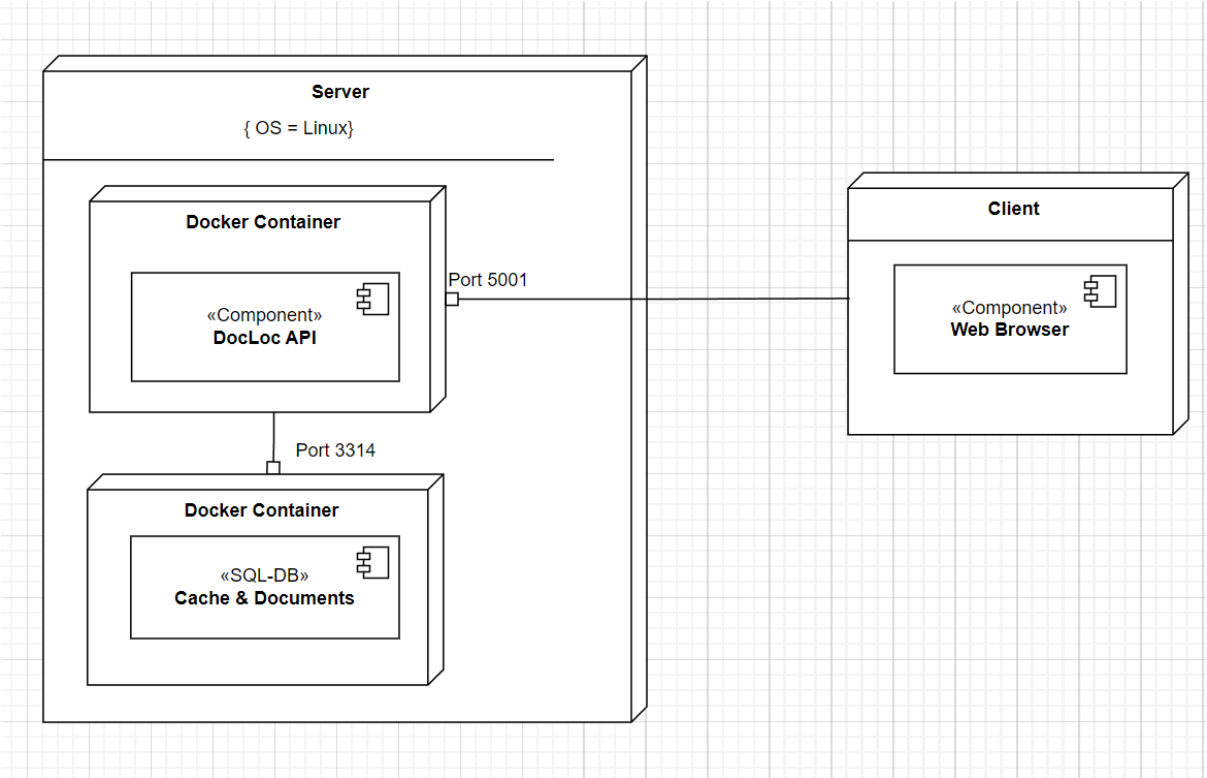

Diagrama de despliegue. Figura 5