Trabajo Práctico N.º 2

Sistemas Distribuidos
1er Cuatrimestre 2022

Alumno: Klein, Santiago - 102192

Índice

1. Diagramas Arquitectura

1.1. DAG

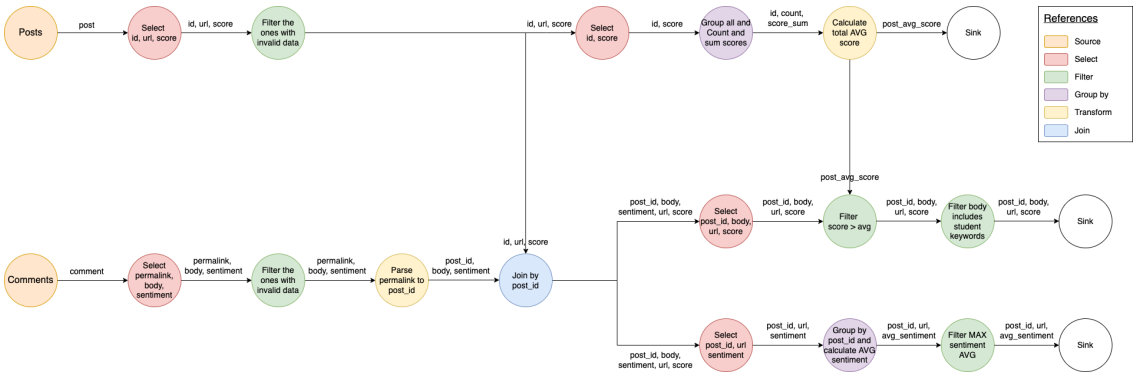

Figura 1: DAG del sistema

1.2. Diagrama de robustez

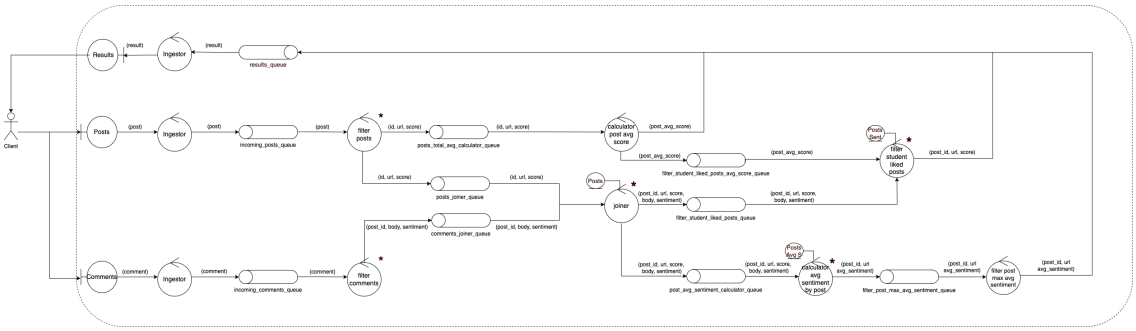

Figura 2: Diagrama de robustez del sistema

1.3. Diagrama de despliegue

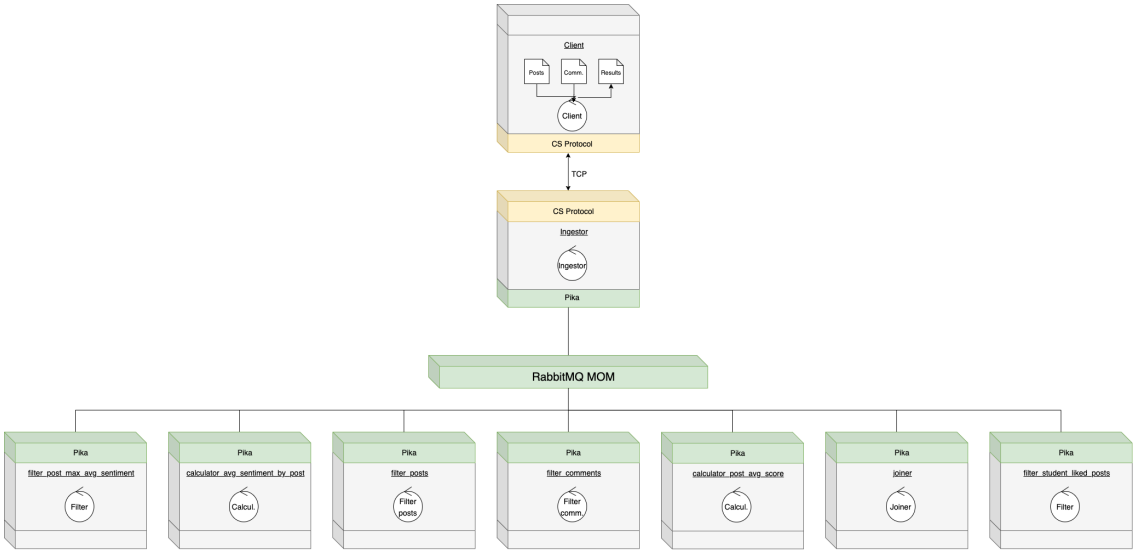

Figura 3: Diagrama de despliegue del sistema

1.4. Diagrama de secuencia

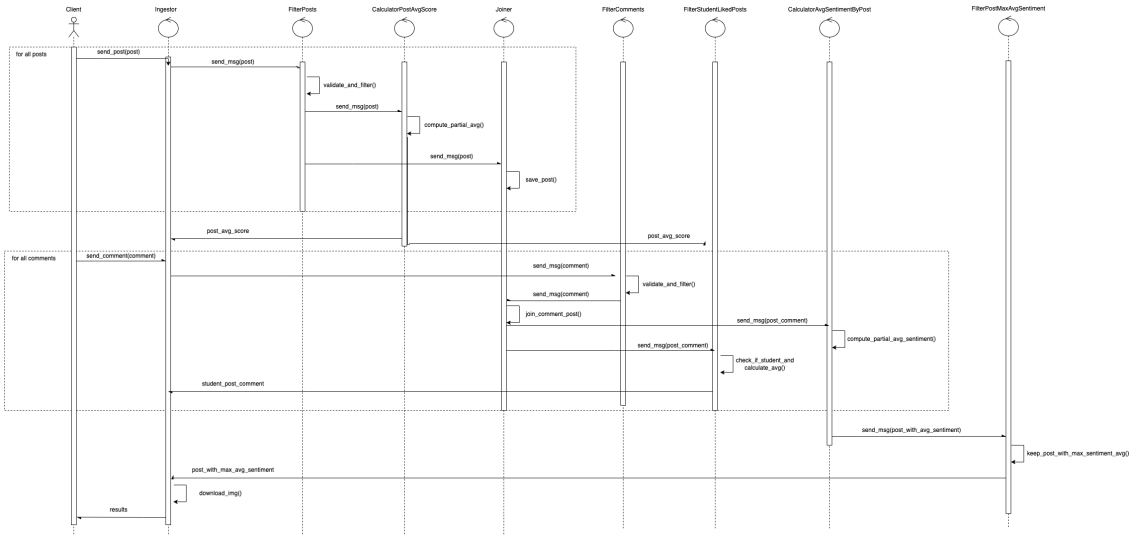

Figura 4: Diagrama de secuencia del sistema simplificado. Se omiten algunos detalles.

1.5. Diagrama de actividades

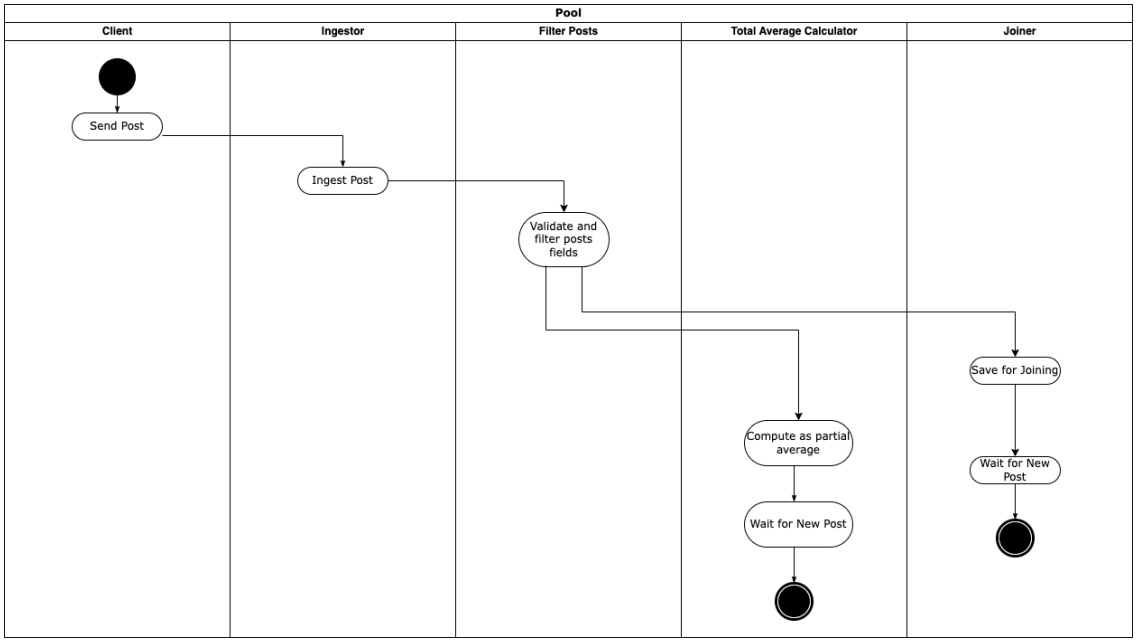

Figura 5: Diagrama de actividades del sistema ante el envío de un Post por parte de un cliente.

1.6. Diagrama de paquetes

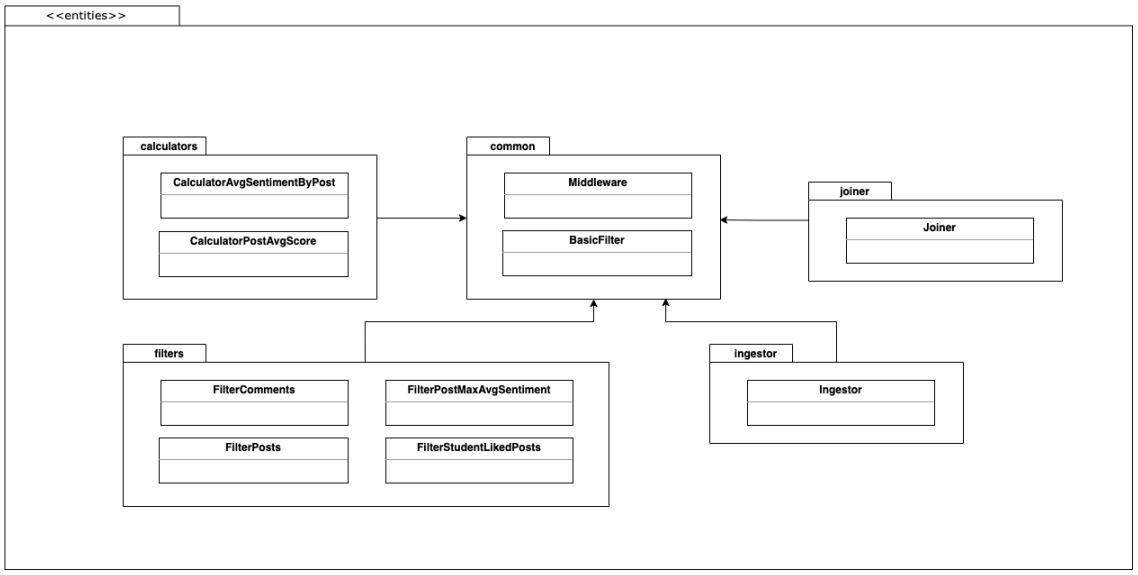

Figura 6: Diagrama de paquetes del sistema

2. Puntos de mejora

- El ingestor efectúa un trabajo secuencial, procesando primero todos los posts, luego todos los comments y por último los resultados. Debido a esto, los resultados se encolan en el MOM y permanecen allí en tanto y en cuanto el ingestor no termine de procesar los posts y comments. Se podría utilizar otro hilo del ingestor (u otra entidad) para ir consumiendo y enviando los resultados a medida que se generan.
- Así como se implementó el BasicFilter para encapsular el comportamiento de la mayoría de los filtros que consumen de una cola, realizan un procesamiento a través de la callback y envían el resultado a un exchange, se podría haber desarrollado otro filtro genérico que consuma primero de una cola, obtenga un resultado, y luego consuma de otra cola, para finalmente enviar los resultados a un exchange (para los casos de filter_student_liked_posts y joiner, que dependen de dos colas).
- Tanto el protocolo de comunicación entre cliente y servidor, y el protocolo de los mensajes del broker son subóptimos, dado que se utiliza json como formato. Esto podría optimizarse, reduciendo la cantidad de información en tránsito y aumentando la performance. Considerando que el client y el ingestor son el cuello de botella de funcionamiento del sistema (con un CPU rondando el 100 %), este es un factor importante de mejora.
- El archivo de configuración del pipeline (pipeline.json) puede ser optimizado en un formato que contenga menos información redundante. Además, los atributos de cada exchange no se están utilizando (fueron agregados para una optimización que no se alcanzó a ejecutar), y podrían utilizarse para enviar a cada exchange solamente la información que necesita, reduciendo el tráfico de información innecesaria.
- Actualmente, cada entidad del pipeline conoce la configuración del mismo, y lo primero que realiza es configurar todas las colas y los exchanges, y hacer los binds correspondientes. Esto se puede realizar de manera segura porque son operaciones idempotentes, mas una optimización posible es crear un script o servicio que se encargue de la configuración del pipeline en el comienzo de la ejecución, o bien que cada servicio configure solamente lo que va a usar, pero se optó por el enfoque actual para simplificar y evitar problemas.
- No está implementado un cierre graceful entre client e ingestor, dado que el client primero manda todo (sin recibir nada), por lo que no detecta que el cliente le manda un paquete de END al recibir la signal SIGTERM. Se podría modificar el protocolo para enviar un ACK/END entre envío de mensajes entre cliente y servidor, y de esa manera cerrar gracefully.