FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Sistemas distribuidos I (75.74)
Trabajo Práctico 1: Escalabilidad
- Middleware y Coordinación de Procesos -

Integrantes	Padrón	Email
Godoy Dupont, Mateo	105561	mgodoy@fi.uba.ar
Harriet, Eliana	107205	eharriet@fi.uba.ar

Docentes:

Pablo D. Roca - Gabriel Robles - Franco Barreneche

Tomás Nocetti - Nicolás Zulaica

Amazon Books Analyzer

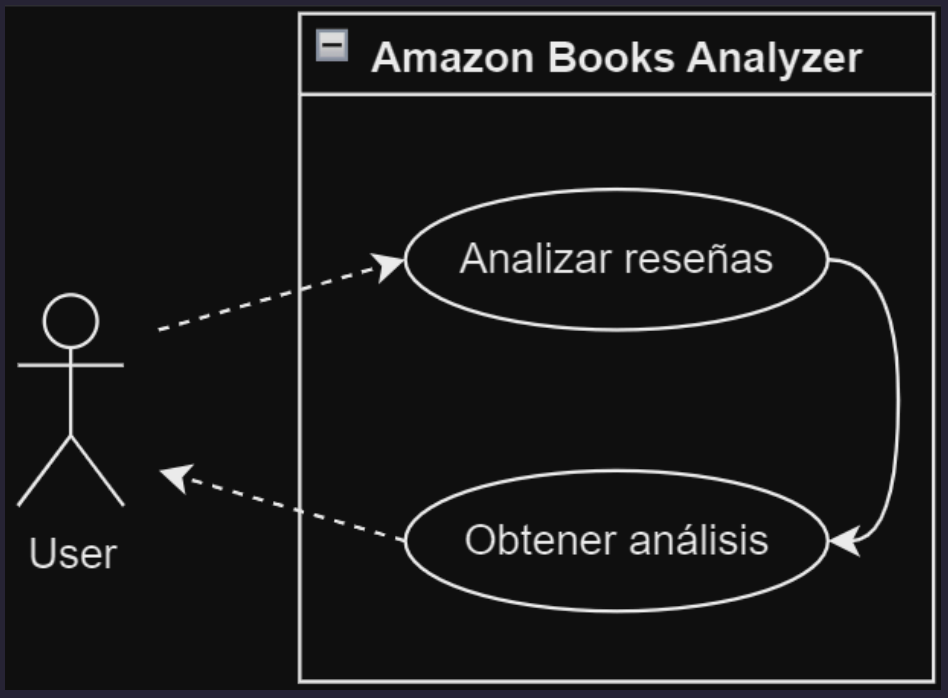
Se provee de un sistema distribuido que analiza las reseñas de los libros en el sitio de Amazon para proponer campañas de marketing. Las reseñas poseen título del libro, texto del comentario y rating. Por cada título de libro, se conoce categoría, fecha de publicación y autores.

El sistema cuenta con la capacidad de resolver las siguientes consultas:

- Título, autores y editoriales de los libros de categoría "Computers" entre 2000 y 2023 que contengan 'distributed' en su título.
- Autores con títulos publicados en al menos 10 décadas distintas
- Títulos y autores de libros publicados en los 90' con al menos 500 reseñas.
- 10 libros con mejor rating promedio entre aquellos publicados en los 90' con al menos 500 reseñas.
- Títulos en categoría "Fiction" cuyo sentimiento de reseña promedio esté en el percentil 90 más alto.

Estas consultas pueden correrse todas al mismo tiempo o seleccionando las de mayor interés. Además, si en un futuro fuera necesario modificar alguna de las consultas, o agregar una nueva, se busca que pueda llevarse a cabo reutilizando partes del sistema proveído. De esta forma se reduciría la complejidad de extender la utilidad del mismo.

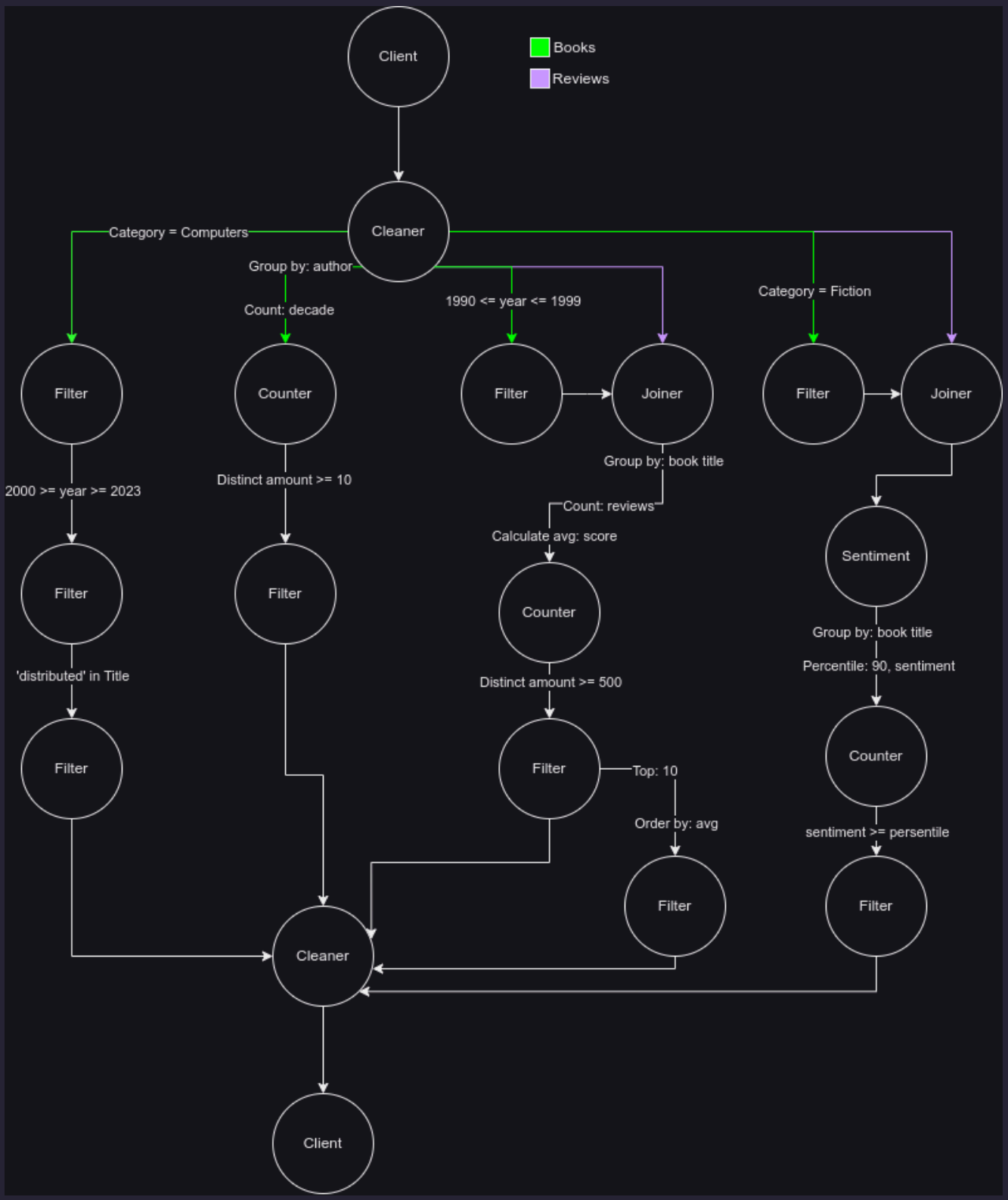
El objetivo principal del sistema es manejar un gran volúmen de datos, por esto mismo es necesario que sus componentes sean escalables.



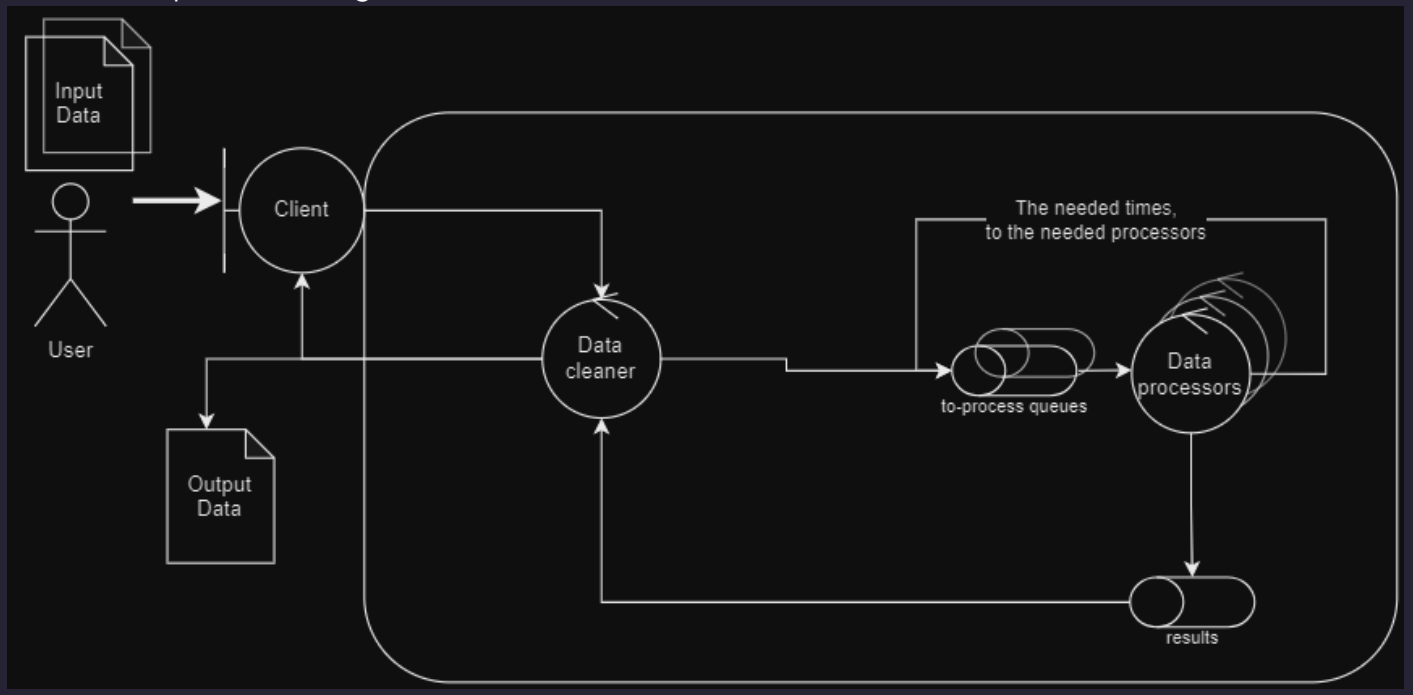
En primer lugar, el usuario puede iniciar el sistema para que comience a hacer el análisis de los datos. Una vez que el sistema comienza a correr, el usuario puede ir constatando el archivo de resultados que se va generando en su computadora. (El sistema no se cerrará hasta completar el análisis).

Esquema de resolución de las consultas

Se provee del siguiente diagrama DAG, en donde se muestra el camino que recorrerá la información para cada una de las consultas.



Respecto a la interacción entre componentes se tiene el siguiente diagrama de robustez, en donde se ve el comportamiento general del sistema:

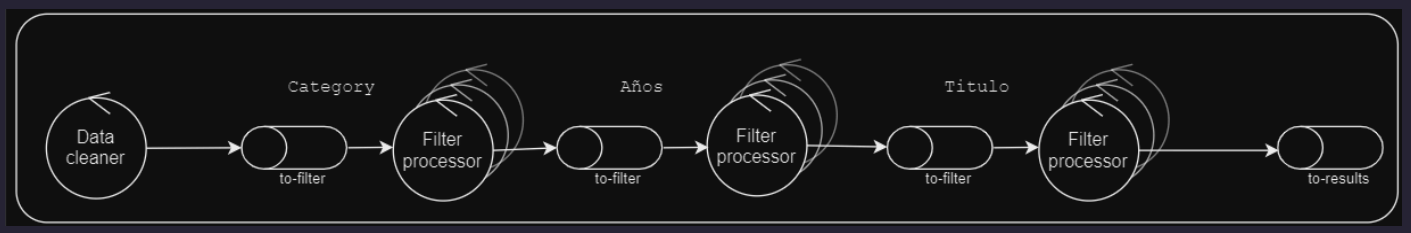


Se puede ver que se tiene una aplicación cliente, a la cual se le pasan los archivos a procesar y a continuación la información proveniente de estos archivos pasa por un nodo de limpieza. Así es como el sistema se queda con la información relevante y descarta datos que no sean íntegros. Los pasos siguientes corresponden a pasar por nodos procesadores de información tantas veces como cada consulta lo requiera y finalmente la información vuelve a limpiarse para enviar al cliente los resultados.

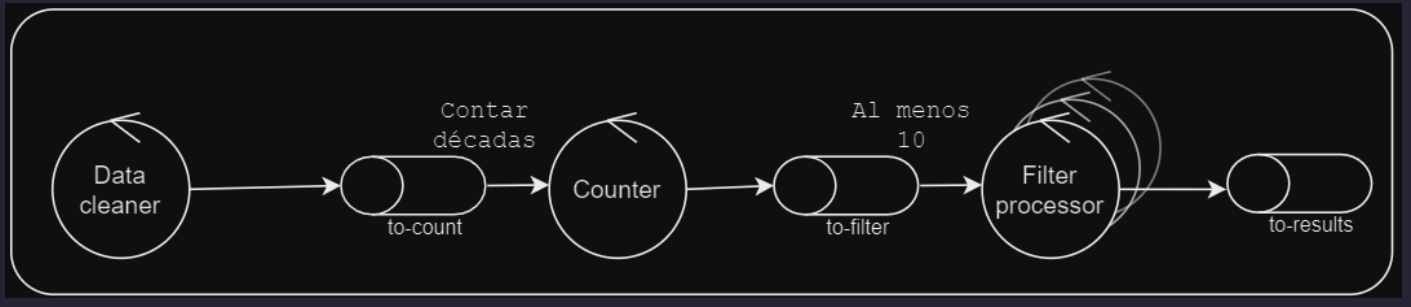
Para cada una de las consultas el paso a paso es el siguiente:

Nota: Cada nodo de tipo X (siendo X un counter, un filter, joiner o sentiment analyzer) corresponde a la misma entidad, pero no son necesariamente nodos diferentes. Se graficó de esta forma para evitar graficar ciclos que puedan confundir.

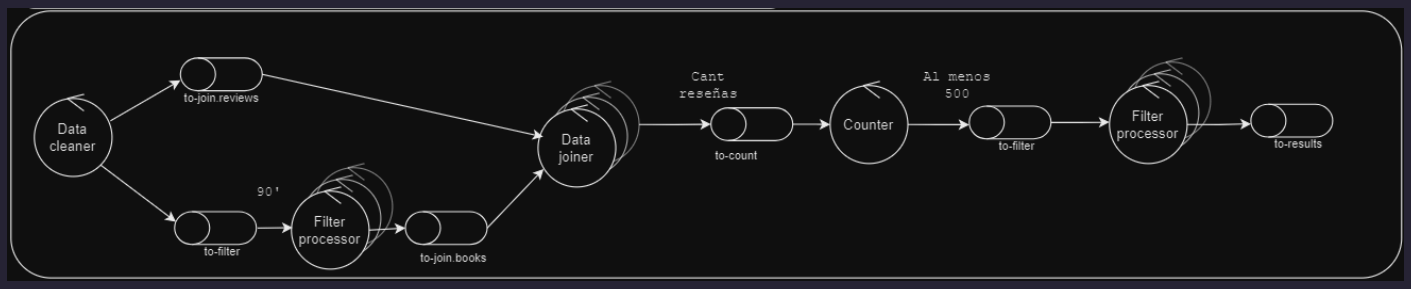
Consulta 1



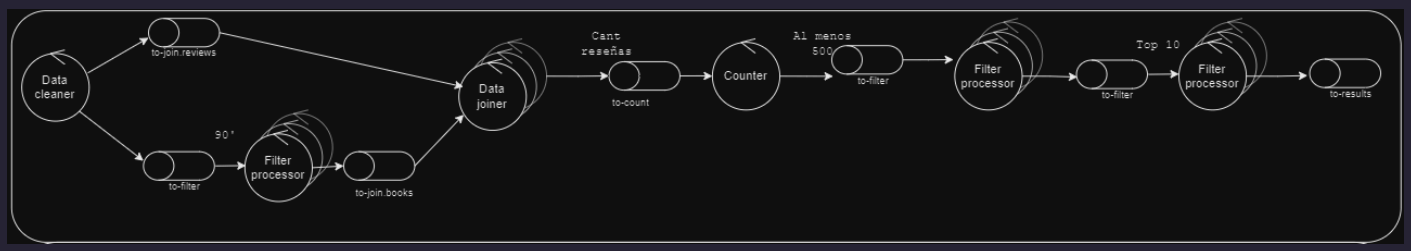
Consulta 2



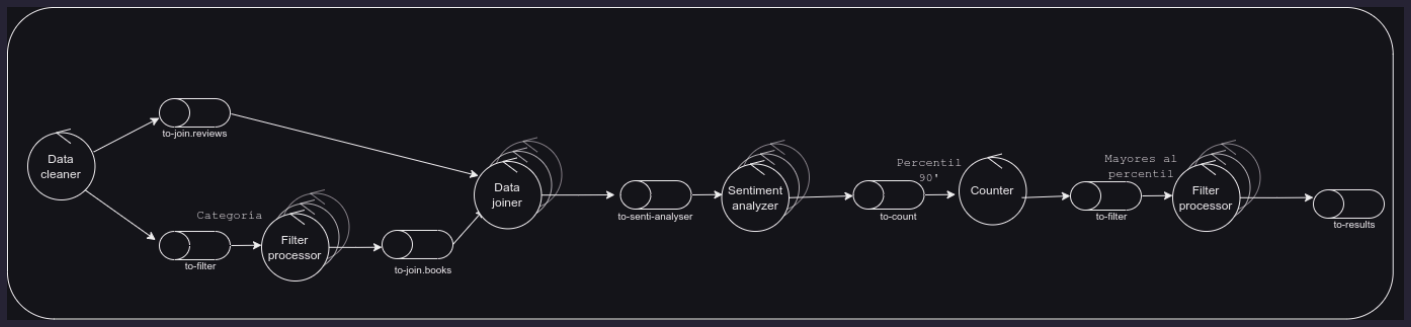
Consulta 3



Consulta 4



Consulta 5



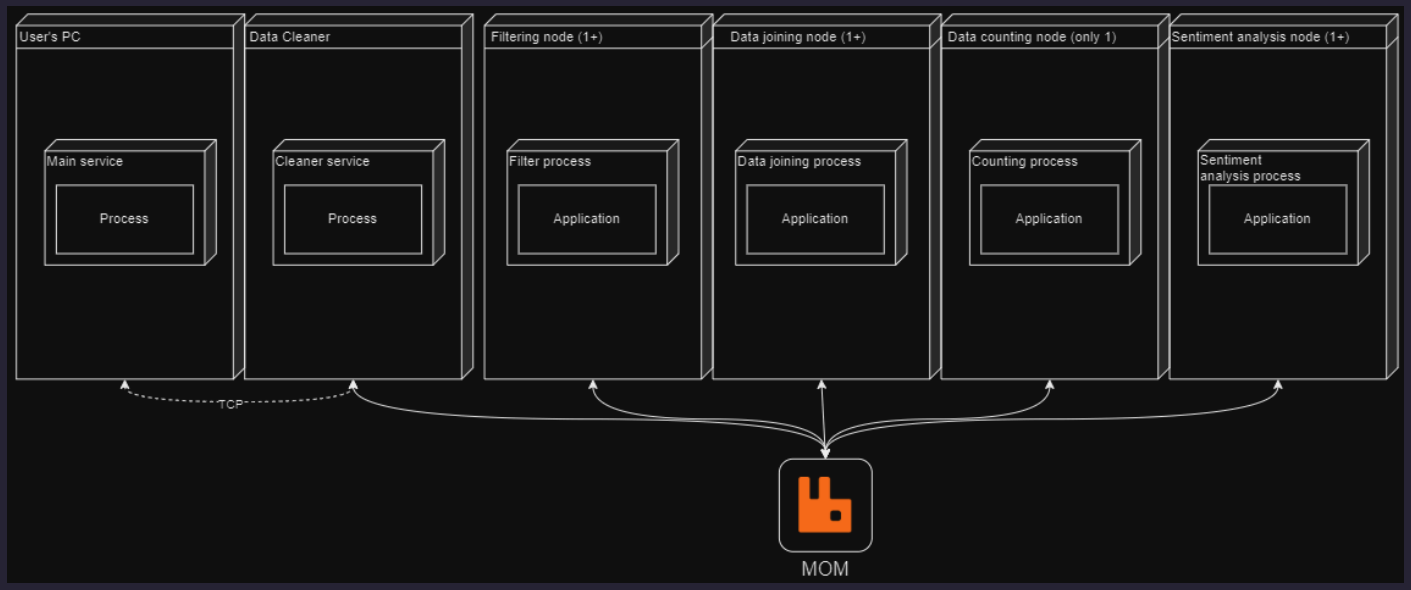
Despliegue del sistema

Como ya se mencionó, el sistema se compone de nodos. Los cuales son los siguientes:

- Client: Este nodo está fuera del servidor, se encuentra en la PC del usuario. Se comunica vía TCP con el servidor, siendo sus interacciones con el DataCleaner.
- Data Cleaner: Es el encargado de limpiar información, dejando pasar lo indispensable.
  En la ida es el encargado de recibir y limpiar la información que envía el cliente, descarta la información que no es correcta (ej.: campos de fecha mal completados) o que no cumple con un mínimo (ej.: libros sin título).
  En la vuelta es el encargado de recibir los resultados, quedarse con las columnas relevantes, y enviarlas al cliente.
  Sumado a sus tareas de limpieza de datos, este nodo es el encargado de otorgarles el formato correcto a los datos tanto desde el cliente al resto del sistema como en el camino inverso.
- Filter: Es el encargado de hacer los filtros correspondientes a las consultas. Es un único filtro que tiene la capacidad para responder en cualquier step correspondiente a filtrado de cualquier consulta. Además, puede instanciarse tantas veces como se lo requiera.
- Joiner: Es el encargado de recibir libros y reviews para luego devolver fragmentos de información con el resultado de la junta de los mismos. Es un único joiner que participa en todas las consultas que lo necesiten y también puede instanciarse la cantidad de veces que se lo requiera.
- Counter: Es el encargado de hacer los conteos correspondientes a las consultas. Es un único contador que tiene la capacidad para responder en cualquier step correspondiente a contar/agrupar de cualquier consulta.

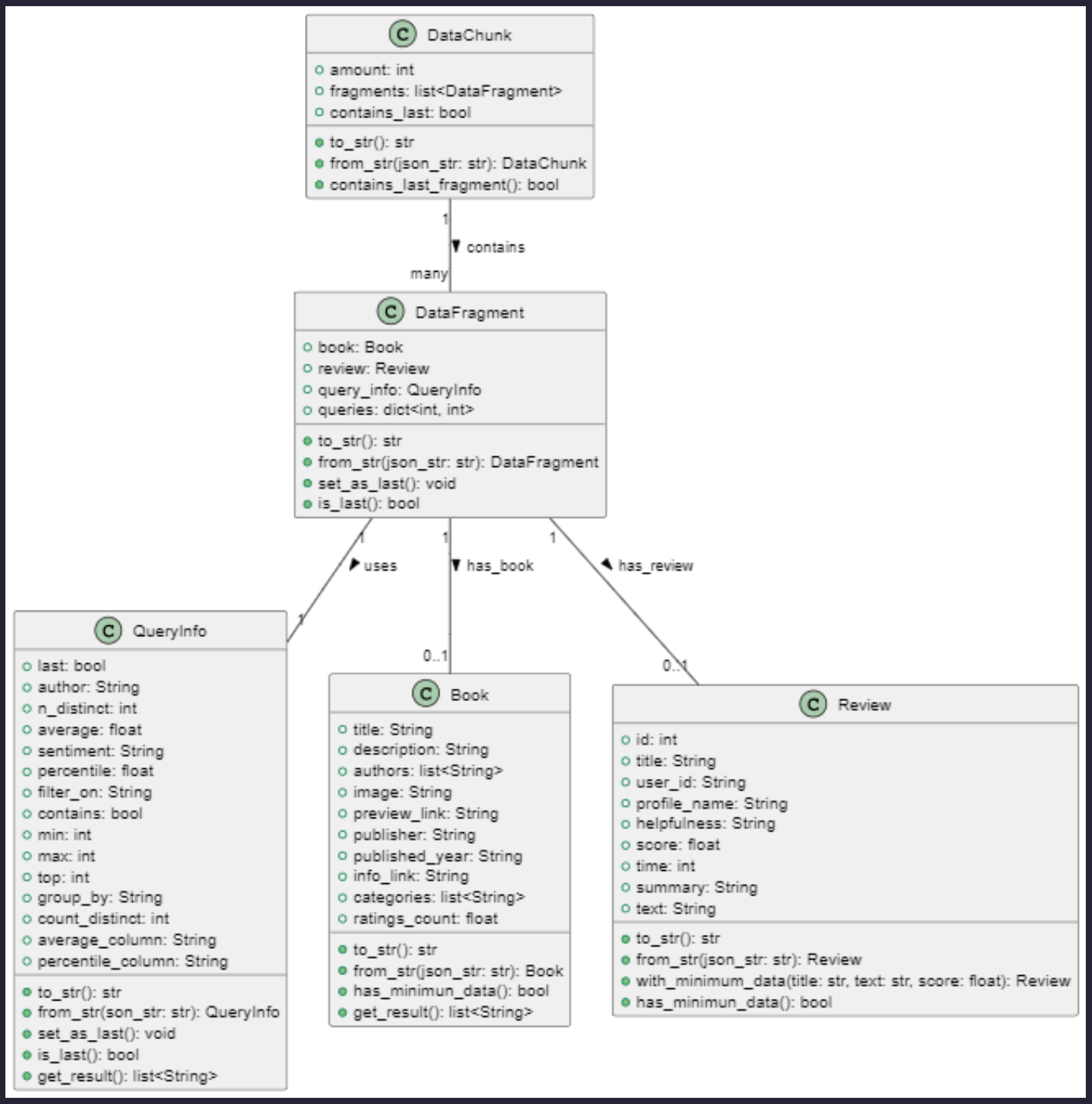
Es un nodo especial, que por limitaciones de diseño no puede ser escalado a más de una instancia. Sin embargo no se vieron consecuencias en tiempo muy severas por esto mismo.

- Sentiment Analyzer: Es el encargado de recibir reviews, tomar su texto principal y hacer un análisis de sentimiento. Puede instanciarse tantas veces como se lo requiera.
- Rabbitmq: Es el nodo en el que se instancia la herramienta utilizada para comunicar a los distintos nodos del servidor.



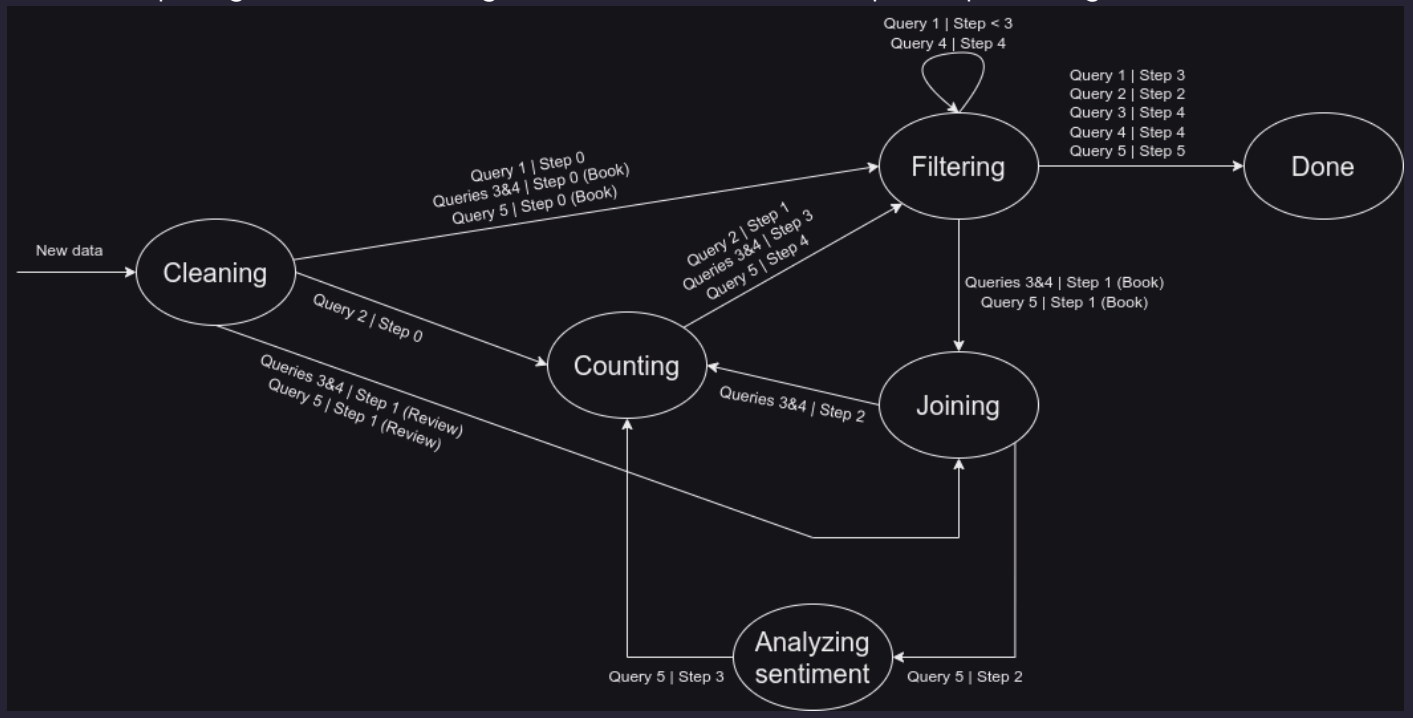
Manejo de los datos

Para trabajar de forma práctica, se crearon distintas estructuras para manejar la información. Se tiene el siguiente diagrama de clases en donde las mostramos:



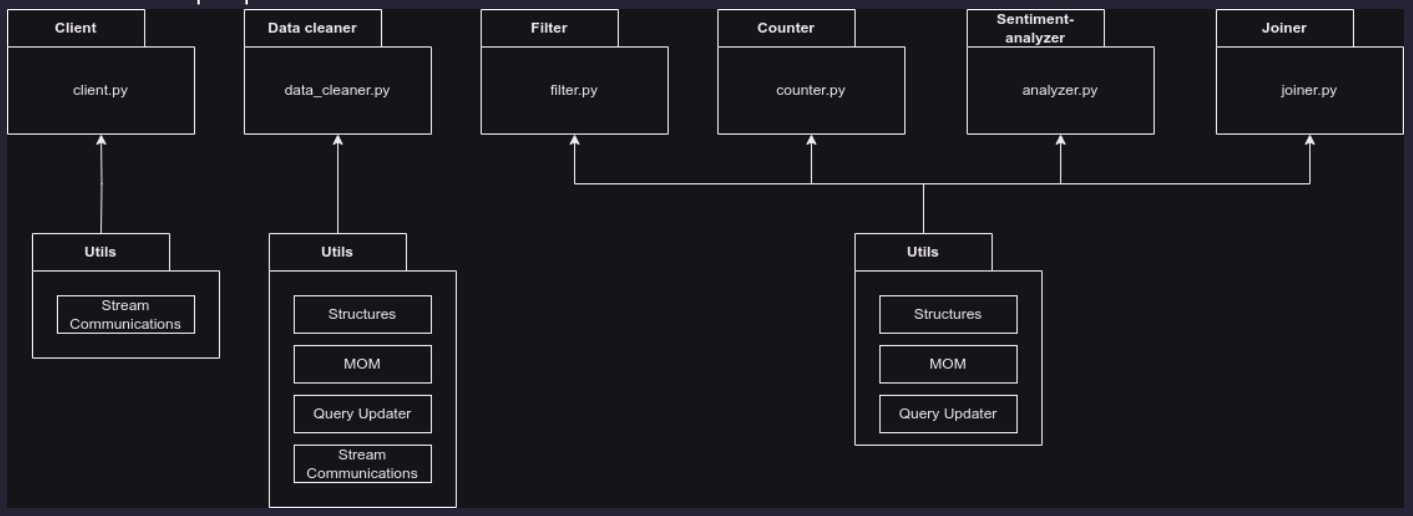
La unidad mínima de información es el DataFragment, en donde se irán haciendo las distintas actualizaciones correspondientes a cada fragmento de datos. El mismo fragmento es el que tiene la información necesaria para definir el siguiente paso en la consulta y almacenará los parámetros necesarios para que el nodo correspondiente pueda procesarlo.

El camino que sigue cada DataFragment al ser actualizado step a step es el siguiente:

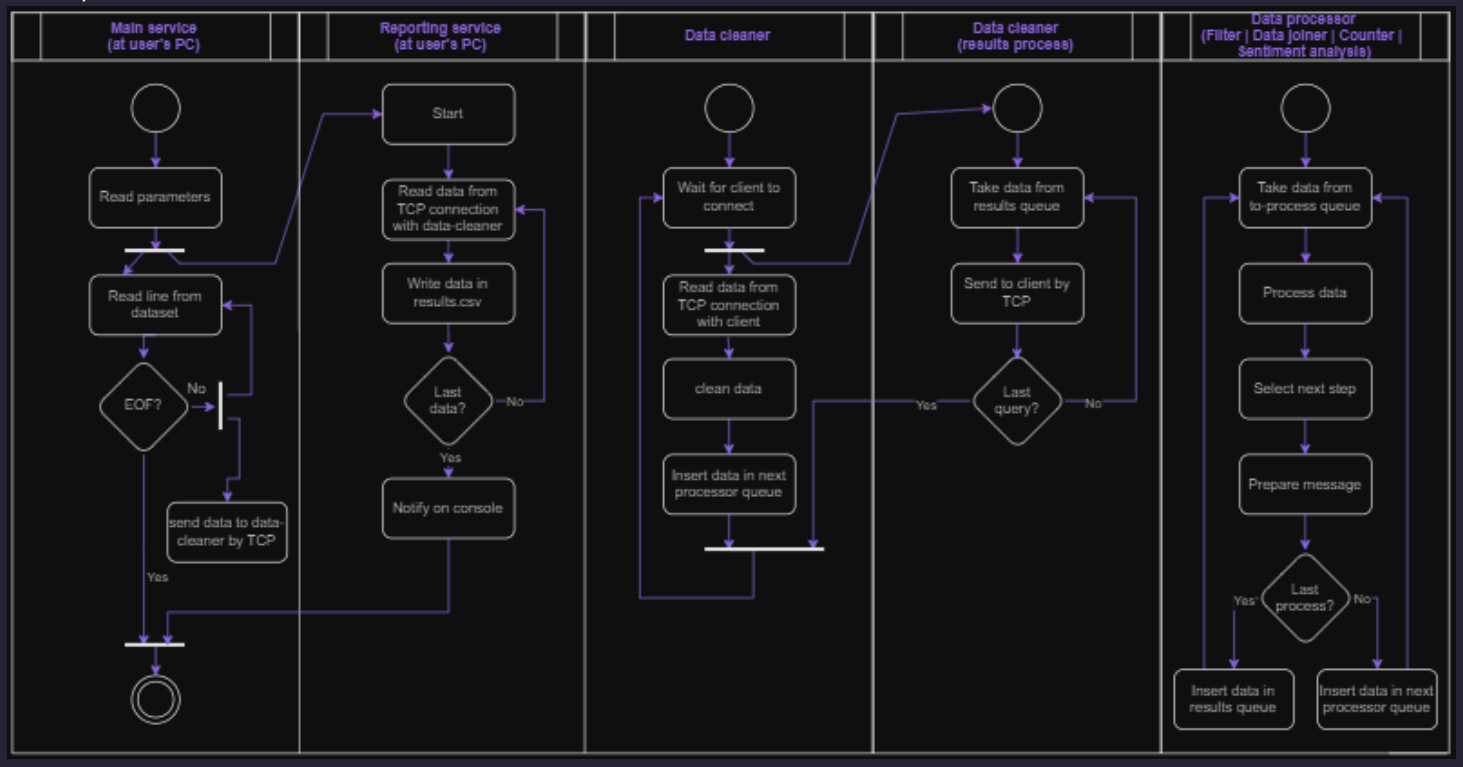


Desarrollo del sistema

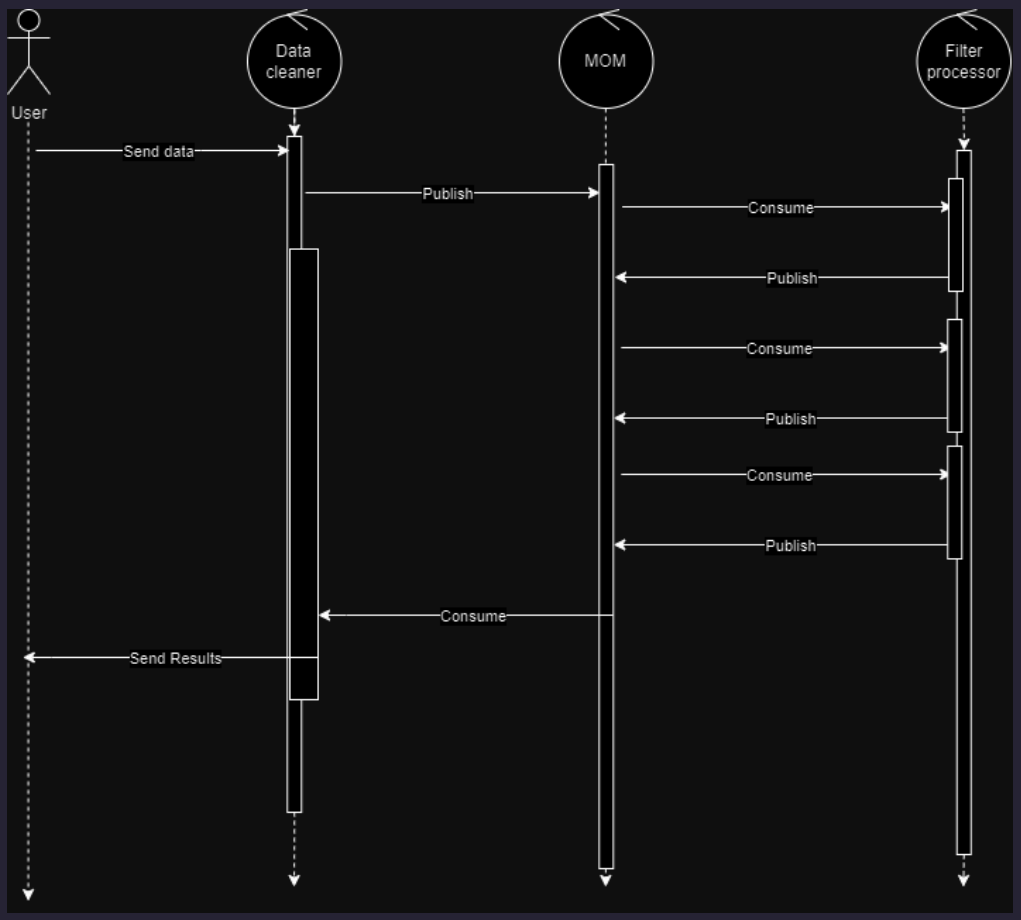
Ya habiendo mencionado los distintos componentes del sistema, en este diagrama se muestra la estructura de paquetes utilizada.



Respecto a los nodos, en este diagrama se muestran el paso a paso del procesamiento llevado a cabo por cada uno.



En el siguiente diagrama de secuencia se muestra un ejemplo de ejecución para la primera consulta (*Título, autores y editoriales de los libros de categoría "Computers" entre 2000 y 2023 que contengan 'distributed' en su título*). En este ejemplo hay un único nodo filter, pero podrían intervenir N nodos de tipo filter.



Ejecución del trabajo:

En lo que respecta a la ejecución del trabajo, se deben cumplir ciertas pre-condiciones. Las mismas consisten en tener ambos archivos de datos en la carpeta "data", es decir, los archivos de "books_data.csv" y "Books_rating.csv". Adicionalmente se debe configurar que queries se desea ejecutar, por defecto el cliente ejecutará las queries 1,2,3,4 y 5 sin embargo esto puede ser personalizado en el archivo de Dockerfile del cliente.
Una vez cumplidas las pre-condiciones, se deben levantar todos los nodos necesarios. Para ello es necesario estar parado en la carpeta "src" y en caso de ejecutarlos individualmente, se deberá ejecutar los comandos respectivos a cada uno de los siete nodos:

- Rabbitmq
- Cleaner
- Counter
- Filter
- Joiner
- Sentiment
- Client

Aquí vale la pena aclarar que los nodos que admiten múltiples instancias deberán ser ejecutados junto al índice de la instancia, es decir, para ejecutar tres instancias del nodo joiner, se deberían ejecutar los comandos:

- Ejemplo:
    - ‘Docker compose up joiner1’
    - ‘Docker compose up joiner2’
    - ‘Docker compose up joiner3’

Visualización del resultado:

Tras la ejecución, se podrán encontrar los archivos de resultados (uno para cada query) en la misma carpeta data donde anteriormente pusimos los csv con los datos.