FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Sistemas Distribuidos I
TP Diseño

[75.74] - Cátedra Roca
Segundo cuatrimestre de 2025

Nombre	Padrón	Correo Electrónico
Jonathan David Rosenblatt	104105	jrosenblatt@fi.uba.ar
Carolina Pico	105098	cpico@fi.uba.ar
Julián García Sánchez	104590	jgarcias@fi.uba.ar

Índice

1. Enunciado

Se solicitó un sistema distribuido que pueda realizar un análisis de datos de ventas en una cadena de negocios de cafés en Malasia. En base a información transaccional de ventas, clientes, tiendas y productos ofrecidos se debe obtener:

1. Transacciones (Id y monto) realizadas durante 2024 y 2025 entre las 06:00 AM y las 11:00 PM con monto total mayor o igual a 75.
2. Productos más vendidos (nombre y cantidad) y productos que más ganancias han generado (nombre y monto), para cada mes en 2024 y 2025.
3. TPV (Total Payment Value) por cada semestre en 2024 y 2025, para cada sucursal, para transacciones realizadas entre las 06:00 AM y las 11:00 PM.
4. Fecha de cumpleaños de los 3 clientes que han hecho más compras durante 2024 y 2025, para cada sucursal.

Donde además se deben cumplir los siguientes requerimientos no funcionales:

- El sistema debe estar optimizado para entornos multicomputadoras
- Se debe soportar el incremento de los elementos de cómputo para escalar los volúmenes de información a procesar
- Se requiere del desarrollo de un Middleware para abstraer la comunicación basada en grupos.
- Se debe soportar una única ejecución del procesamiento y proveer graceful quit frente a señales SIGTERM.

Finalmente para esta entrega presentaremos un diagrama 4+1 incluyendo diagramas de robustez, despliegue, actividades, paquetes, secuencia, DAG y una división final de las tareas entre los integrantes del grupo.

2. Vistas 4+1

2.1. Vista Lógica

La estructura del trabajo se organiza en tres niveles principales: el **cliente**, el **gateway** y los nodos de procesamiento que hacen uso de la clase **Middleware**.

En primer lugar, la clase Client representa el punto de inicio del sistema, ya que se encarga de procesar y enviar los archivos, guardar reportes en archivos locales y recibir los resultados obtenidos.

La clase Gateway funciona como intermediario entre el cliente, mediante un protocolo tcp y los nodos de procesamiento, mediante un middleware. Sus responsabilidades incluyen procesar mensajes los mensajes recibidos por el cliente, recolectar reportes desde el pipeline y coordinar el inicio o detención del consumo de datos. De esta manera, actúa como punto central de comunicación y control.

La clase Middleware encapsula operaciones relacionadas con el consumo de datos (start_consuming, stop_consuming) mediante las Queues y los Exchange de RabbitMQ. Esta clase es utilizada por el gateway y los nodos de procesamiento, para facilitar la comunicacion entre ellos, publicando y consumiento mensajes mediante el.

A partir de allí, se definen distintos tipos de para implementar funciones específicas.

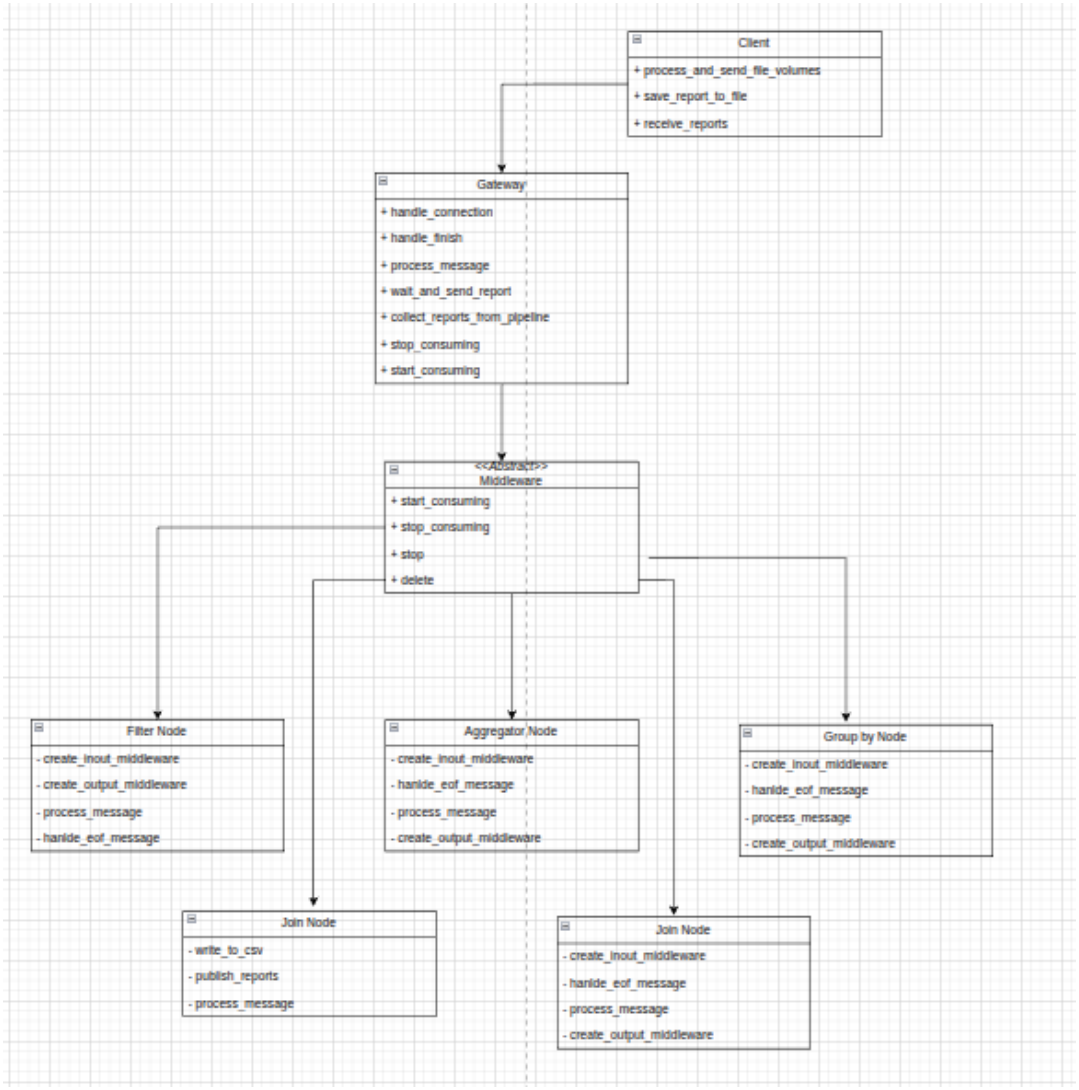

Figura 1: Diagrama de Clases.

Figura 2: Diagrama de Clases Nodo Filter.

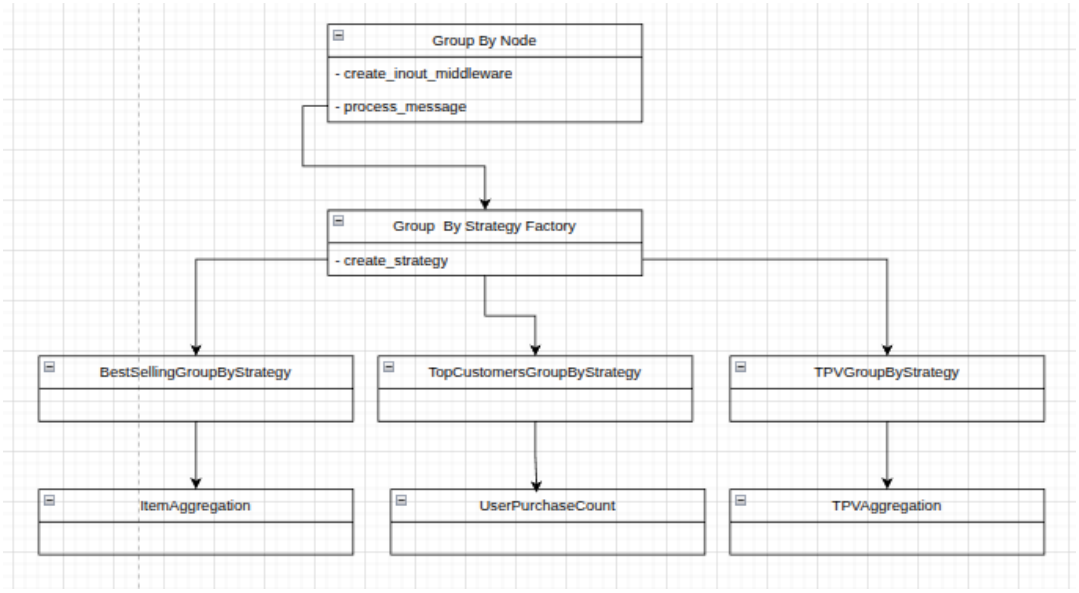

Figura 3: Diagrama de Clases Nodo Group By.

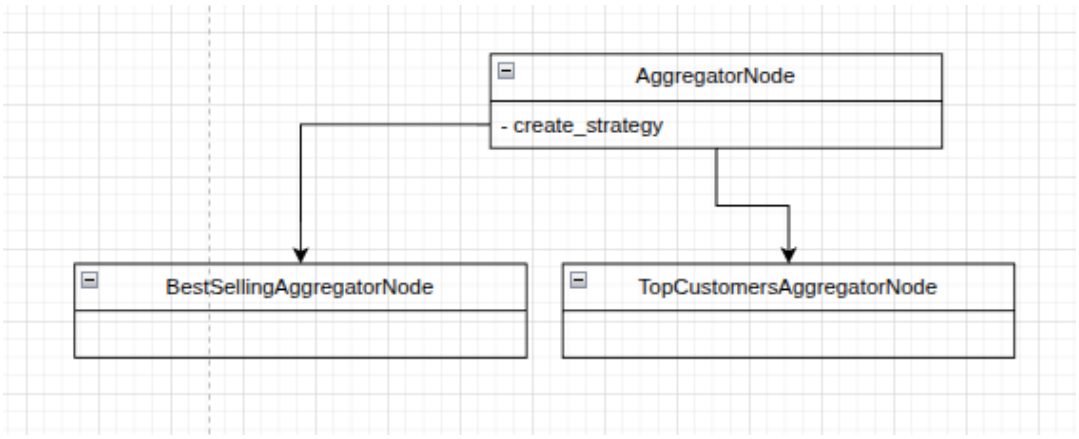

Figura 4: Diagrama de Clases Aggregator.

El diagrama presentado representa un DAG que describe el flujo de procesamiento de datos a partir de los archivos de origen. A partir de esta fuente, se aplican distintos filtros iniciales. Los filtros de año se dividen entre los que procesan archivos de transactions y los que procesan archivos de transaction_items, pero ambos filtran sobre el campo transaction_id, considerando la fecha de creación (created_at). Luego, los filtros de mes y monto siguen el flujo de filtrado mediante los datos proveidos por los nodos que procesan transacciones, para resolver la query 1. Para las siguientes queries, los datos filtrados se someten a diferentes operaciones de agrupamiento:

- Agrupaciones por fecha de creación (created_at) segmentadas por semestre y por año (2024 y 2025).
- Agrupación por tienda (store_id) y usuario (user_id).
- Agrupación por fecha (created_at) y transacciones (transaction_id).

A partir de estas agrupaciones se ejecutan cálculos de ranking, donde se identifican los valores principales (como el top 1). Los resultados parciales son enviados al nodo join, que consolida los

distintos cálculos con los archivos correspondientes, recivido desde el csv source. Finalmente, cada resultado de una query es enviada al nodo Generate Report.

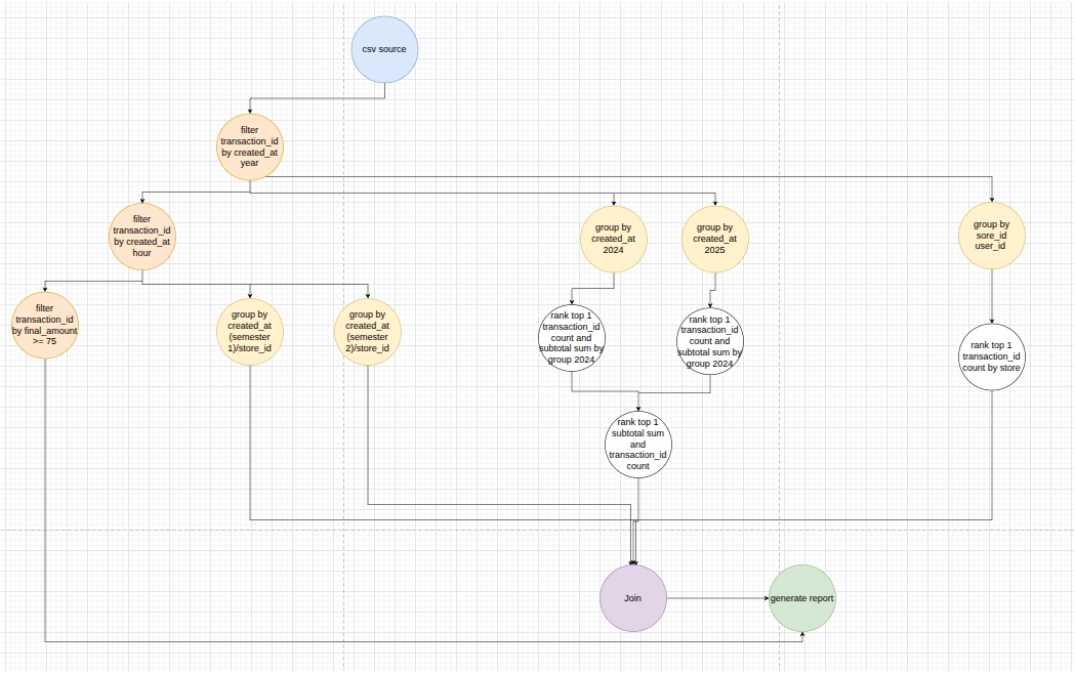

Figura 5: DAG.

2.2. Vista Desarrollo

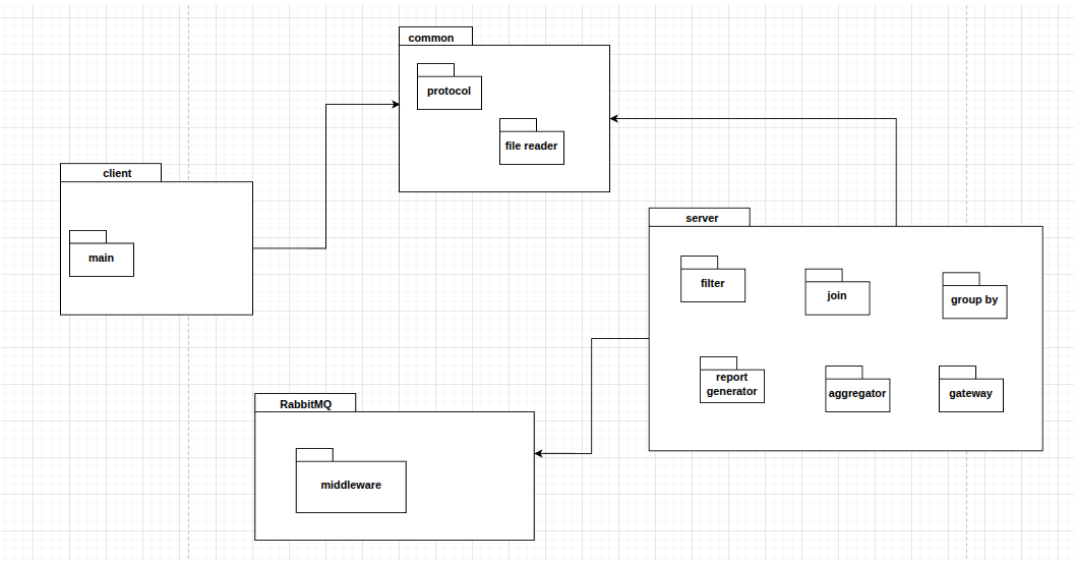

Figura 6: Diagrama de paquetes.

2.3. Vista Procesos

El siguiente diagrama de secuencia demuestra el distinto manejo de los EOFs entre los distintos tipos de nodos.

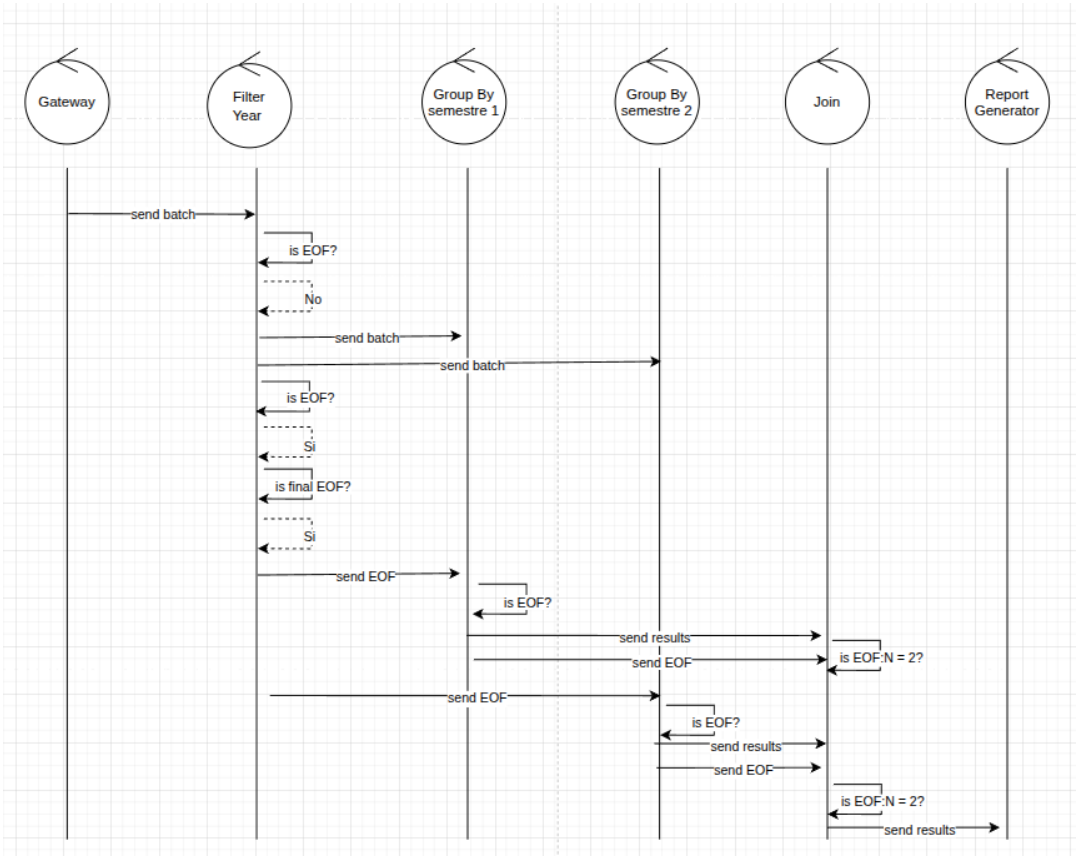

Figura 7: Diagrama de secuencia.

El flujo de propagación de EOF comienza con la recepción de un mensaje, que se evalúa para determinar si el lote corresponde a un EOF. En caso de que no sea un EOF, el nodo procede a procesar la línea CSV y posteriormente enviar el batch filtrado al siguiente nodo del pipeline.

Cuando el lote es identificado como EOF, el worker realiza una verificación adicional para determinar si se trata del último EOF recibido de todos los nodos involucrados. Si aún no se han recibido todos los EOF, el nodo propaga el EOF al siguiente nodo del grupo, sumando EOF:N+1 al contador y finalizando su escucha de la cola. En cambio, si se confirma que es el último (EOF:N = total de nodos del grupo), el EOF se propaga hacia los nodos siguientes, tambien finalizando el consumo de la cola.

Todos los nodos escalables implementan el mismo modelo de propagación de EOF.

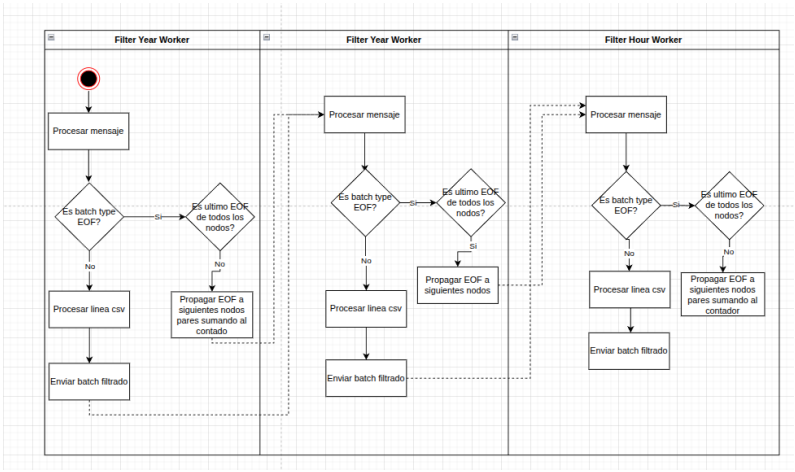

Figura 8: Diagrama de actividad para el envio de EOF.

El diagrama de actividades modela el funcionamiento del Best Selling Aggregator. Aunque todos los nodos comparten la misma lógica básica de procesamiento, existen dos modos de operación diferenciados: intermediate y final. En ambos casos, el flujo comienza con la recepción de un mensaje. Si el lote no corresponde a un EOF, el nodo procede a procesar la línea CSV y guardar las líneas filtradas. Cuando el lote es identificado como EOF, se realiza una verificación adicional para determinar si se trata del último EOF recibido desde todos los nodos del grupo. La diferencia esencial radica en el tratamiento del EOF y el envío de resultados: En el modo Intermediate, el nodo guarda los resultados parciales a medida que recibe mensajes. Cuando detecta el EOF, lo propaga a los siguientes nodos pares dentro del mismo grupo, incrementando un contador que coordina la finalización del conjunto (EOF:N+1) y procede a enviar sus resultados parciales al nodo final En el modo Final, el nodo envía los resultados consolidados solo al recibir el EOF definitivo y, en lugar de propagarlo dentro del mismo grupo, lo propaga a los nodos de la etapa siguiente del pipeline, cerrando así el ciclo de procesamiento.

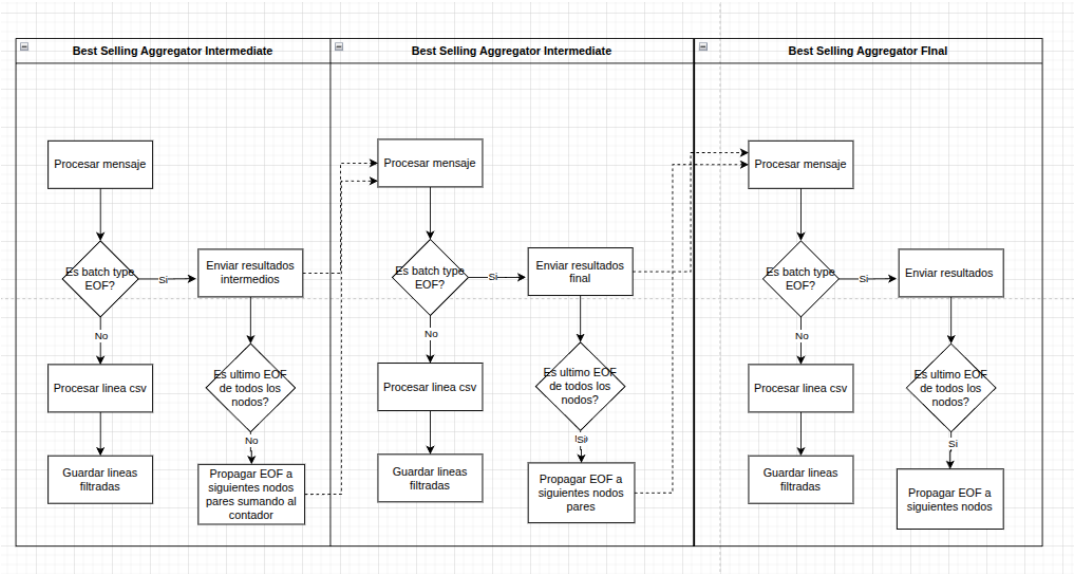

Figura 9: Diagrama de actividad entre nodos intermedios y final.

2.4. Vista Física

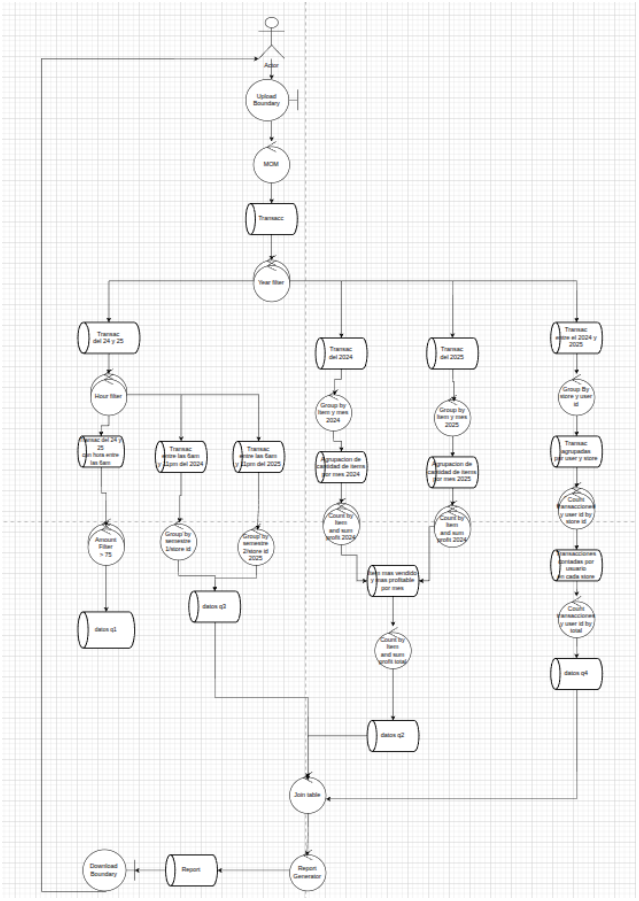

Figura 10: Diagrama de Robustez.

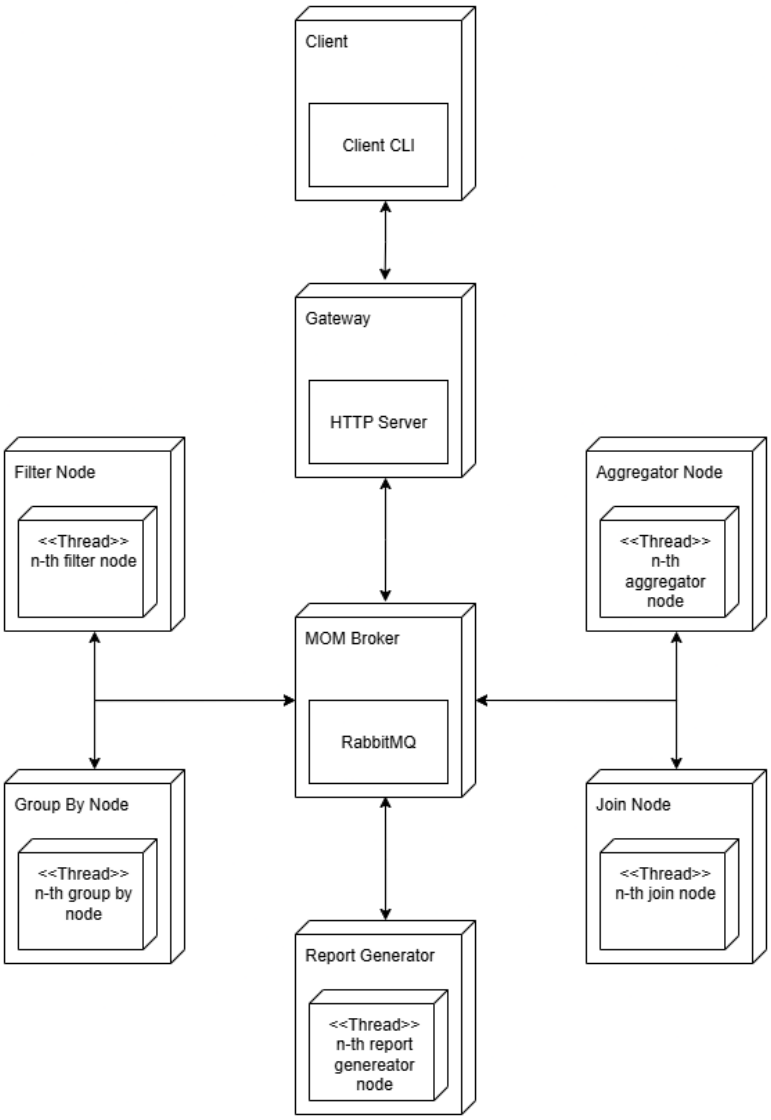

Figura 11: Diagrama de Despliegue.

2.5. Casos de Uso

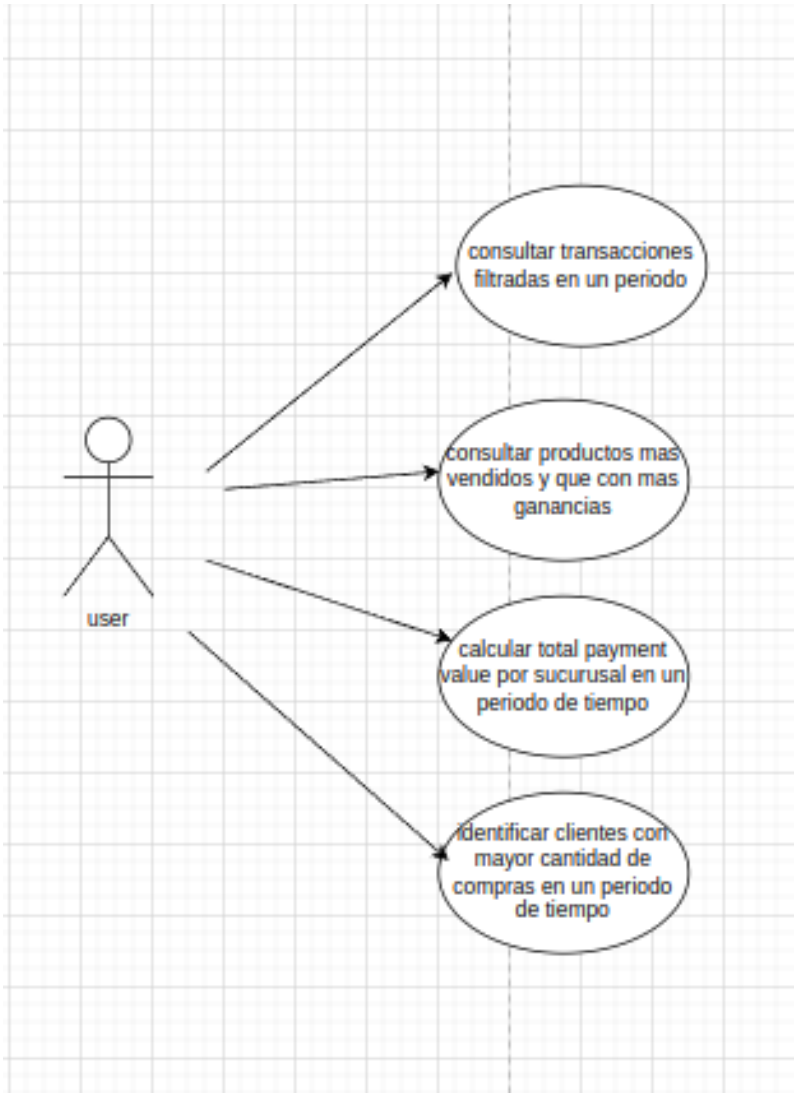

Figura 12: Diagrama de Casos de Uso.

3. División de Tareas

Nombre	Tareas
Cliente	Julián García Sánchez
Middlware	Jonathan David Rosenblatt / Carolina Pico
Filter Node	Carolina Pico / Julián García Sánchez
Group By Node	Carolina Pico / Julián García Sánchez
Join Node	Carolina Pico
Aggregator Node	Carolina Pico
Report Generator	Julián García Sánchez