

Trabajo Práctico N°1: Diseño - Coffee Shop Analysis

Sistemas Distribuidos - 75.74

2º Cuatrimestre 2025

Integrantes

Estudiante	Padrón	Correo electrónico
Juarez Goldemberg, Mariana Noelia	108441	mjuarezg@fi.uba.ar
Orsi, Tomas Fabrizio	109735	torsi@fi.uba.ar
Starnone, Bruno Luciano	108018	bstarnone@fi.uba.ar

Índice

[Alcance](#)

[Arquitectura](#)

[Escenarios - Casos de uso](#)

[CU-01: Listar transacciones filtradas](#)

[CU-02: Ranking mensual de productos](#)

[CU-03: TPV por semestre y sucursal](#)

[CU-04: Top clientes por sucursal](#)

[Vista física](#)

[Diagrama de robustez](#)

[Diagrama de robustez - Query N°1](#)

[Diagrama de robustez - Query N°2](#)

[Diagrama de robustez - Query N°3](#)

[Diagrama de robustez - Query N°4](#)

[Diagrama de robustez - Vista completa](#)

[Vista lógica](#)

[DAG](#)

[Diagrama de despliegue](#)

[Vista de procesos](#)

[Diagramas de secuencia](#)

[Diagrama de secuencia para flujo cliente-servidor](#)

[Diagrama de secuencia para la consulta N°1](#)

[Diagrama de secuencia para la consulta N°2](#)

[Diagrama de secuencia para la consulta N°3](#)

[Diagrama de secuencia para la consulta N°4](#)

[Diagramas de actividades](#)

[Vista de desarrollo](#)

[Formato paquete](#)

[Mientras el sistema procesa la información, el Client permanece en un loop de espera hasta recibir todas las respuestas. Una vez procesadas, los resultados viajan de regreso al Client a través del Sender, y finalmente se muestran en pantalla al usuario \(o se genera un archivo\).](#)

[Diagrama de paquetes](#)

[División de tareas](#)

[Adiciones de entrega N° 4](#)

[Tolerancia a fallos](#)

[Monitoreo de nodos](#)

[Reinicio del nodo](#)

[Recuperación del nodo](#)

[Algoritmo de consenso para líder de Watchdogs](#)

[Chaos Monkey](#)

[Monitoreo de performance con dataset completo](#)

Alcance

Se busca diseñar un sistema distribuido que procese y analice información de ventas de una cadena de cafeterías en Malasia, a partir de datasets transaccionales (ventas, clientes, productos y sucursales) enviados por los clientes de la aplicación.

El análisis debe responder preguntas clave de negocio en distintos niveles. Se deben obtener resultados para 4 consultas, las cuales se detallan en los casos de uso.

Arquitectura

Escenarios - Casos de uso

CU-01: Listar transacciones filtradas

- Dado un cliente que posee el dataset de ventas, cuando ejecuta el sistema, obtiene un listado de las mismas filtradas por fecha, horario y monto.

CU-02: Ranking mensual de productos

- Dado un cliente que posee el dataset de ventas, ítems y productos, cuando ejecuta el sistema, obtiene para cada mes los productos más vendidos (nombre + cantidad) y los productos que más ingresos generaron (nombre + monto)

CU-03: TPV por semestre y sucursal

- Dado un cliente que posee el dataset de ventas y sucursales, cuando ejecuta el sistema, obtiene el TPV (Total Payment Value) por cada semestre y sucursal sobre operaciones en un rango horario determinado.

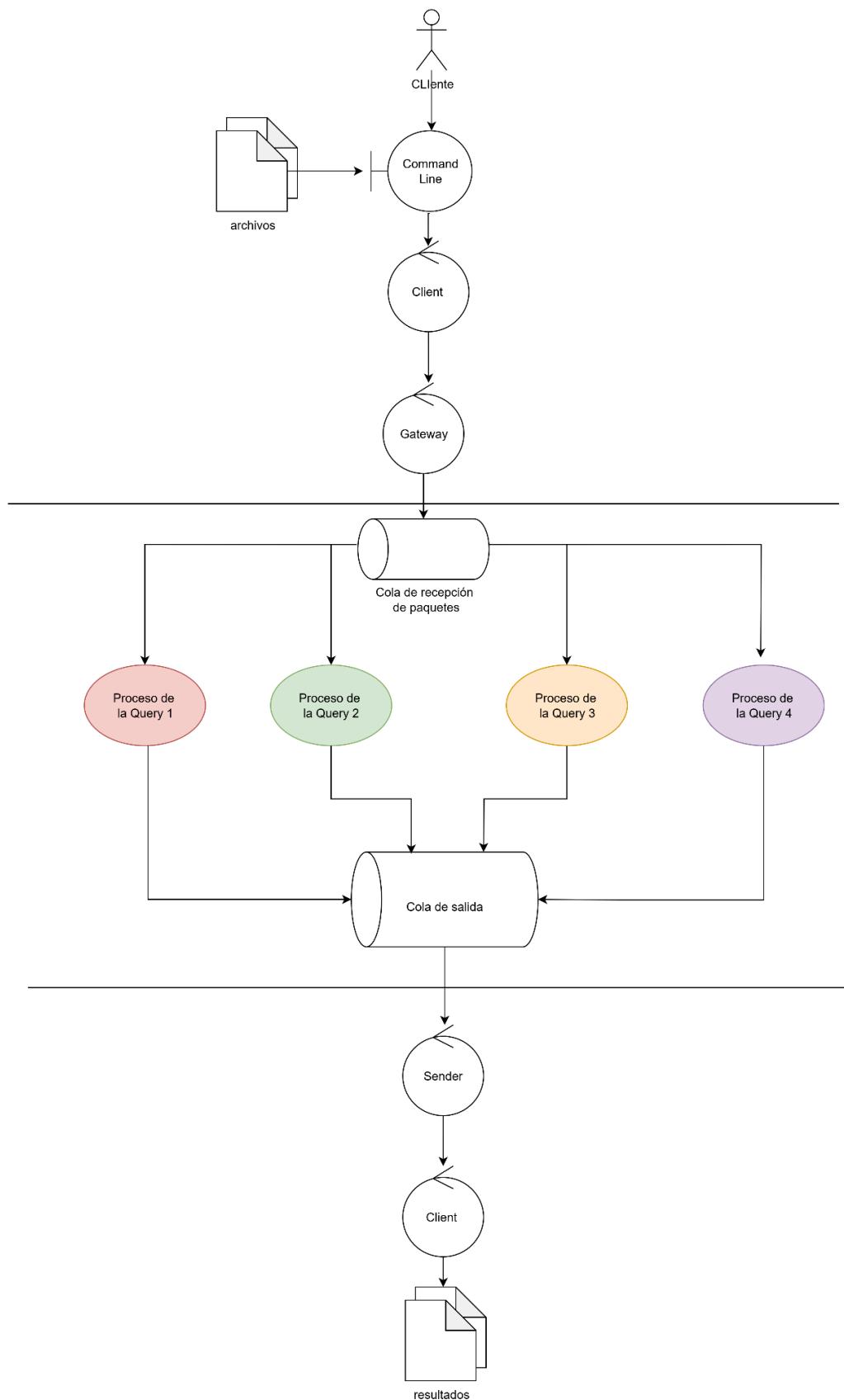
CU-04: Top clientes por sucursal

- Dado un cliente que posee el dataset de ventas y clientes, cuando ejecuta el sistema, obtiene para cada sucursal la fecha de cumpleaños de los 3 clientes con mayor cantidad de compras en un período definido.

Vista física

En esta vista encontramos todos los componentes físicos del sistema. De esta manera definimos responsabilidades y mostramos un escenario de interacción entre los componentes.

Diagrama de robustez



El proceso que se realiza para cada consulta, lo vemos por separado en los siguientes diagramas. Tenemos en cuenta que tendremos workers que pueden reutilizarse para más de una consulta, para el entendimiento de estos casos los sepáramos.

Cambios entrega 4: Se adjunta link al diagrama detallado completo

[Diagrama de robustez completo y corregido](#)

Diagrama de robustez - Query N°1

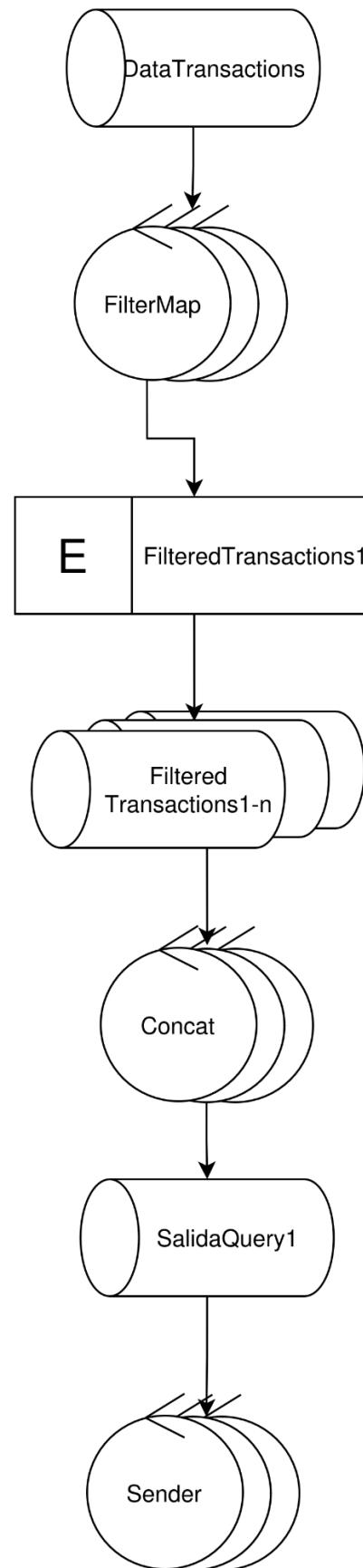


Diagrama de robustez - Query N°2

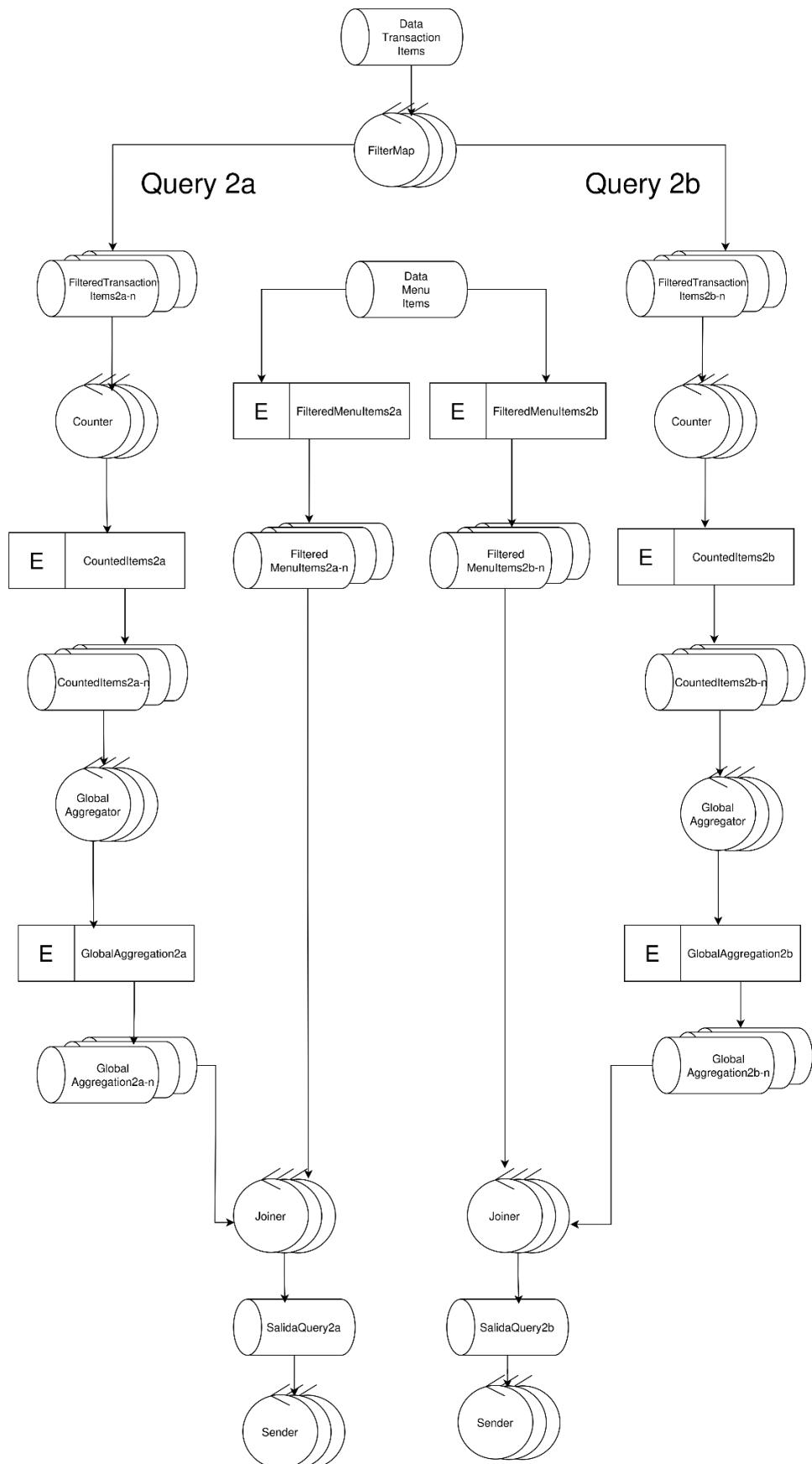


Diagrama de robustez - Query N°3

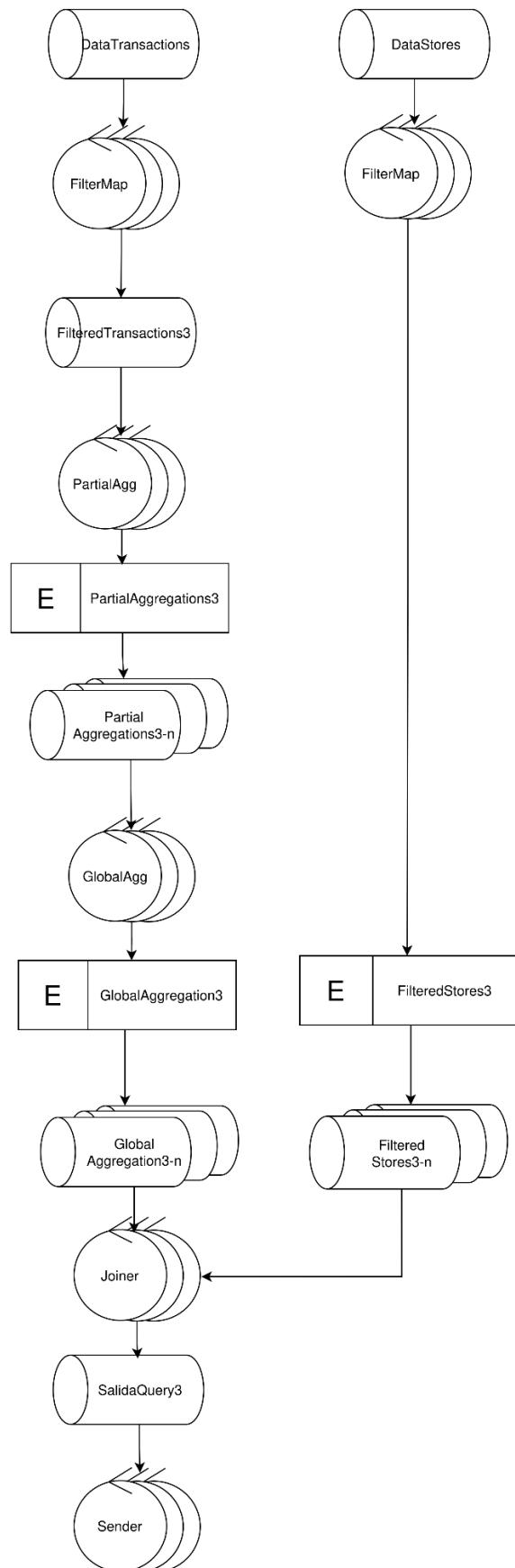


Diagrama de robustez - Query N°4

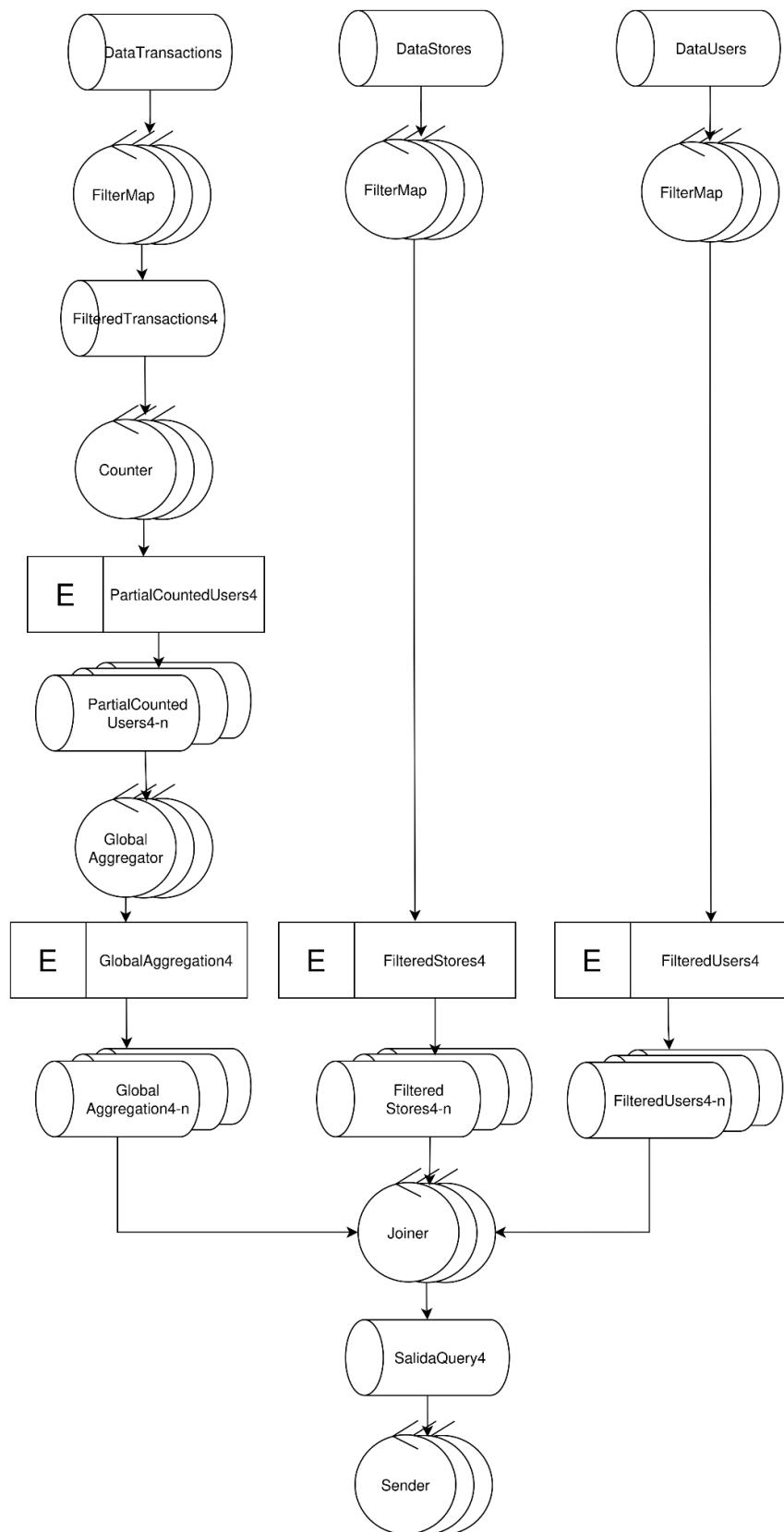


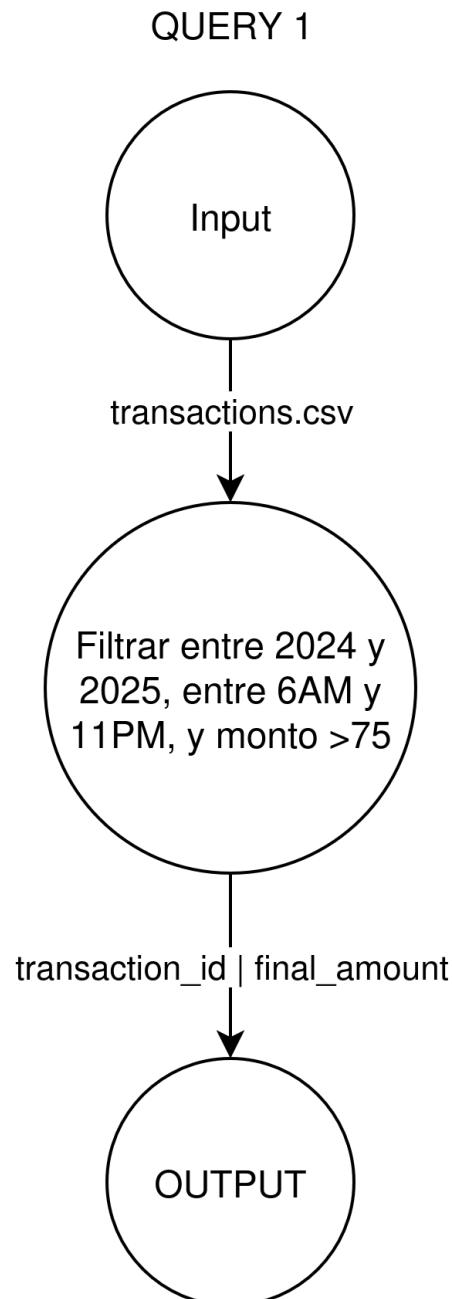
Diagrama de robustez - Vista completa

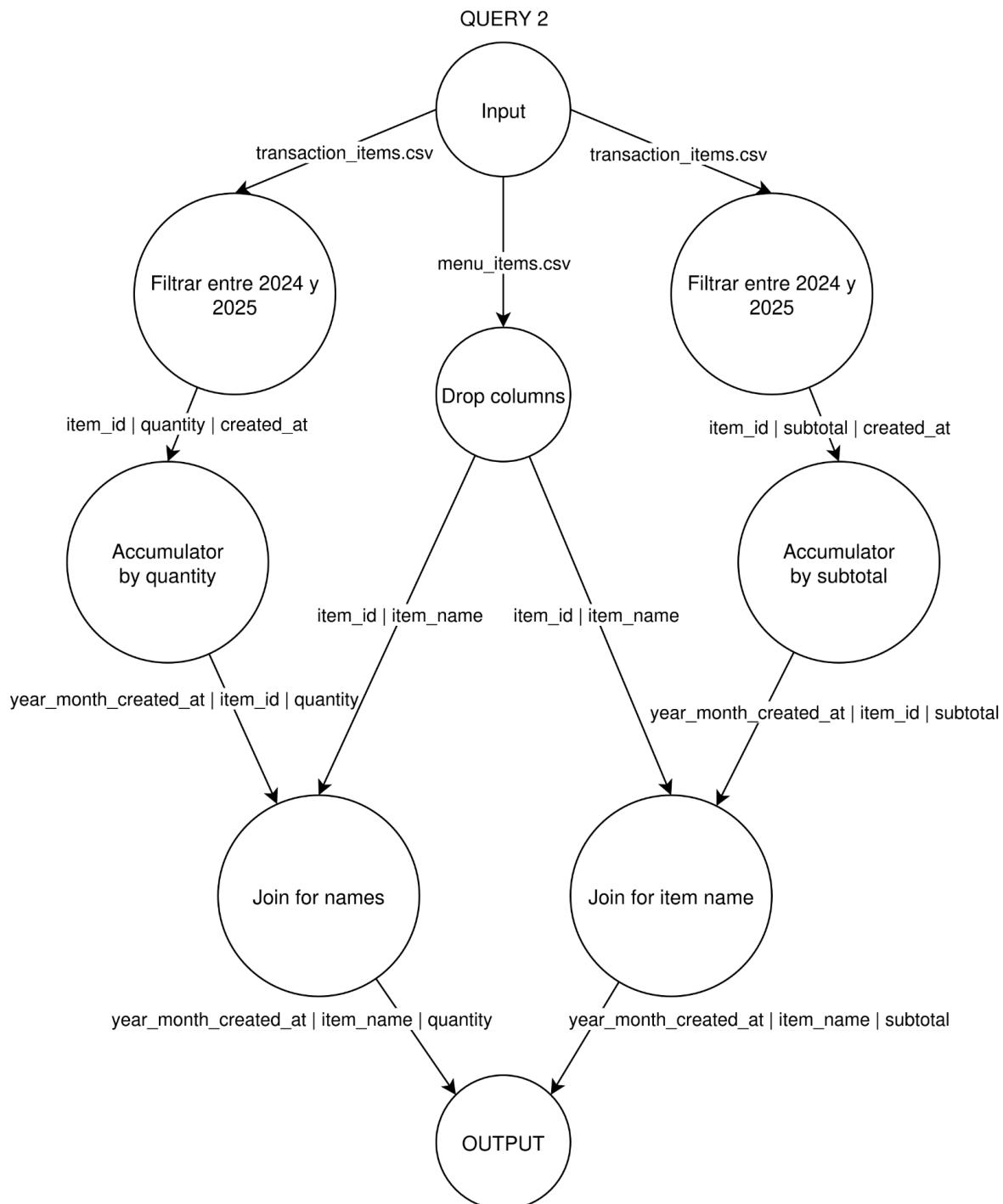
El diagrama de robustez creció a un punto que no se aprecia dentro del documento. Se provee un enlace para acceder a la [imagen almacenada en github](#).

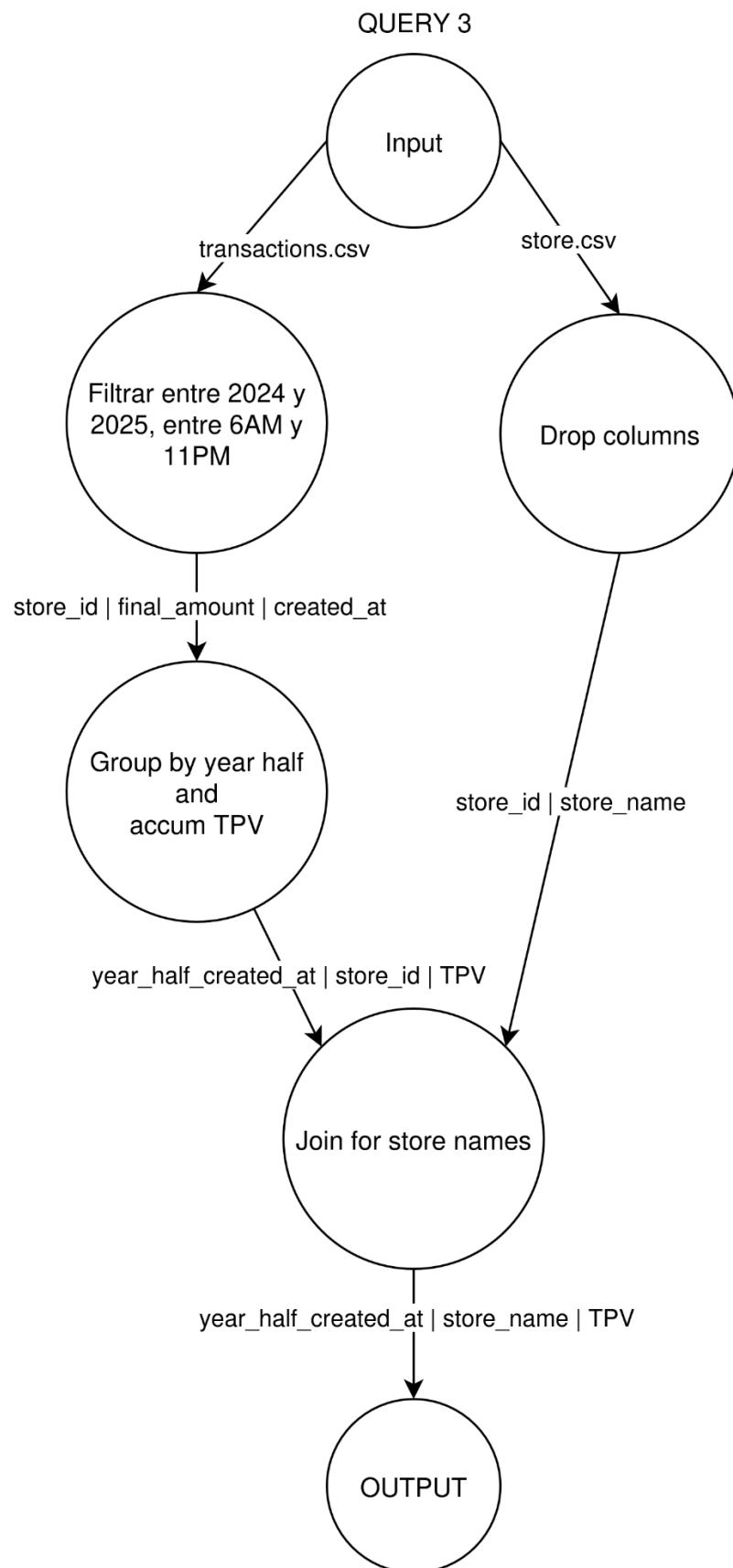
Vista lógica

En esta vista se observa la estructura y funcionalidad del sistema. Contamos con un DAG (Directed Acyclic Graph) de cada consulta para mostrar el flujo de datos y un diagrama de despliegue para mostrar la distribución de procesos en distintas computadoras.

DAG







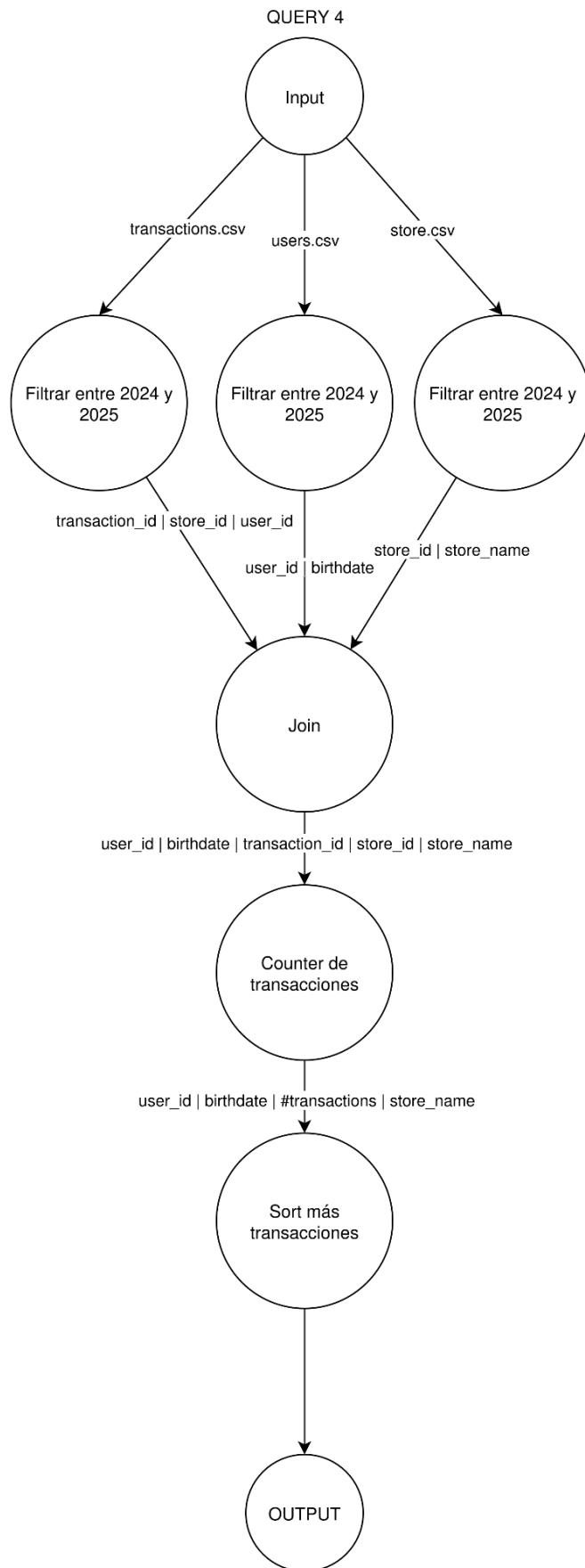
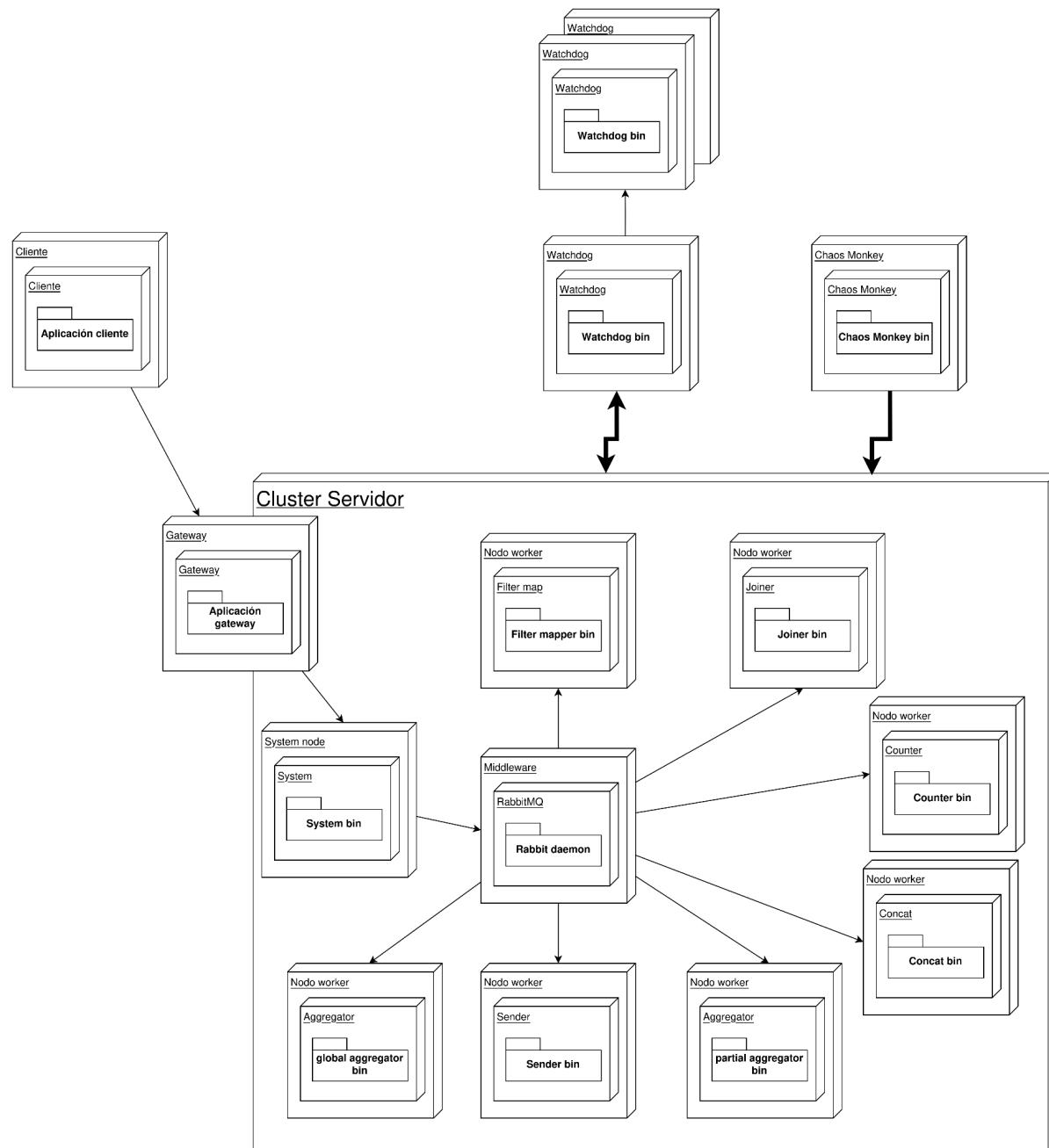


Diagrama de despliegue

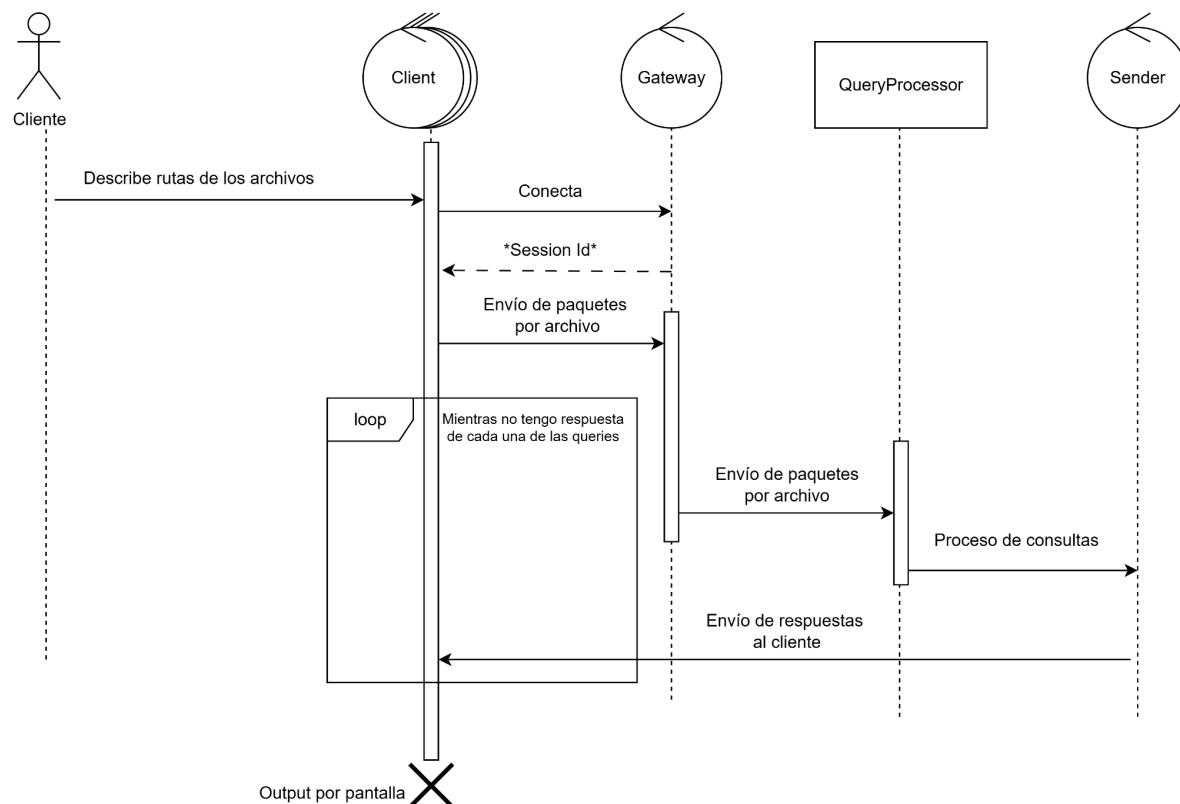


Vista de procesos

En esta vista se representan los aspectos dinámicos del sistema y cómo se comporta el sistema en ejecución.

Diagramas de secuencia

Diagrama de secuencia para flujo cliente-servidor



El Cliente indica las rutas de los archivos al módulo Client, que se conecta al Gateway y recibe un Session Id que identificará la sesión de procesamiento.

Luego, el Client divide los archivos en paquetes y los envía al Gateway, que los reenvía al QueryProcessor para procesar las consultas.

Diagrama de secuencia para la consulta N°1

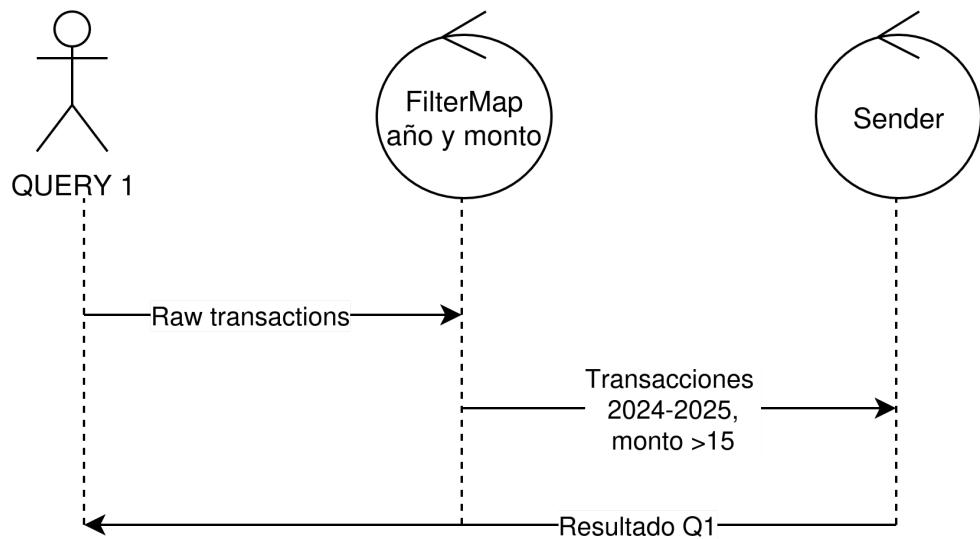


Diagrama de secuencia para la consulta N°2

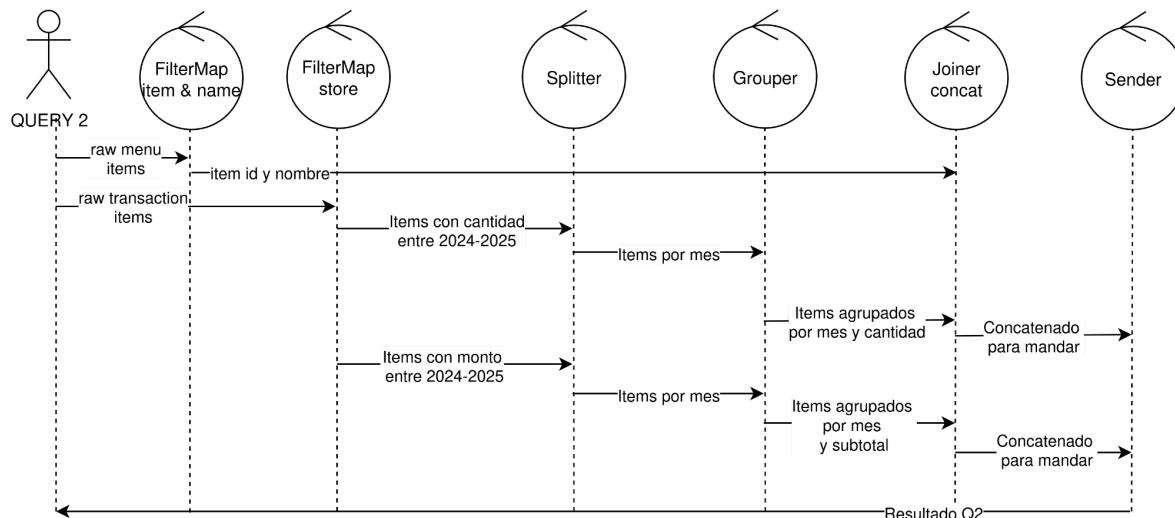


Diagrama de secuencia para la consulta N°3

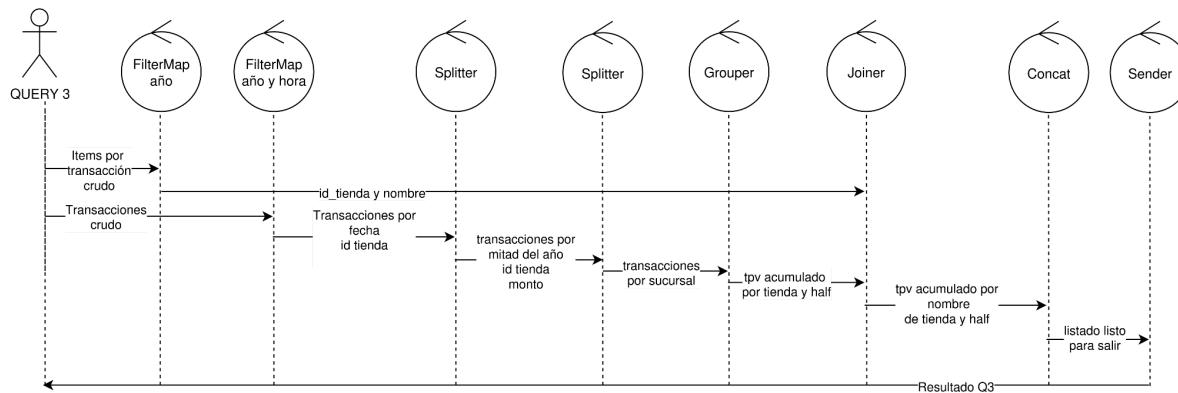
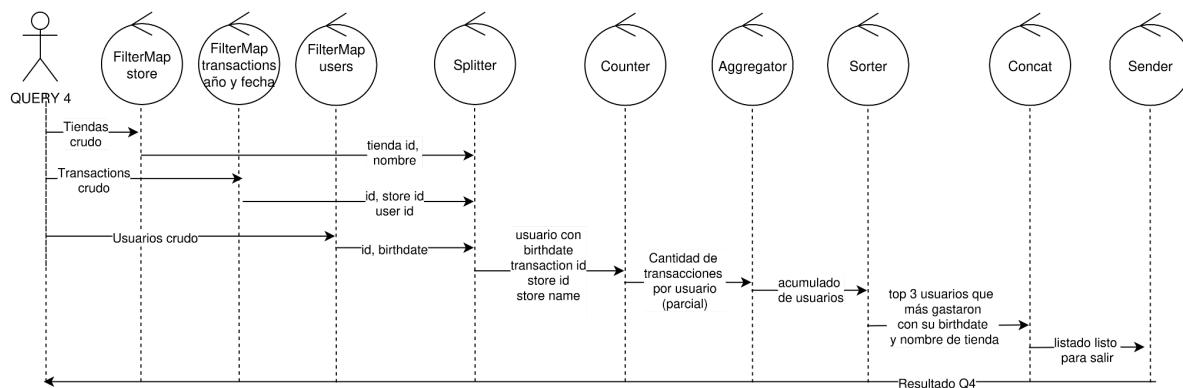
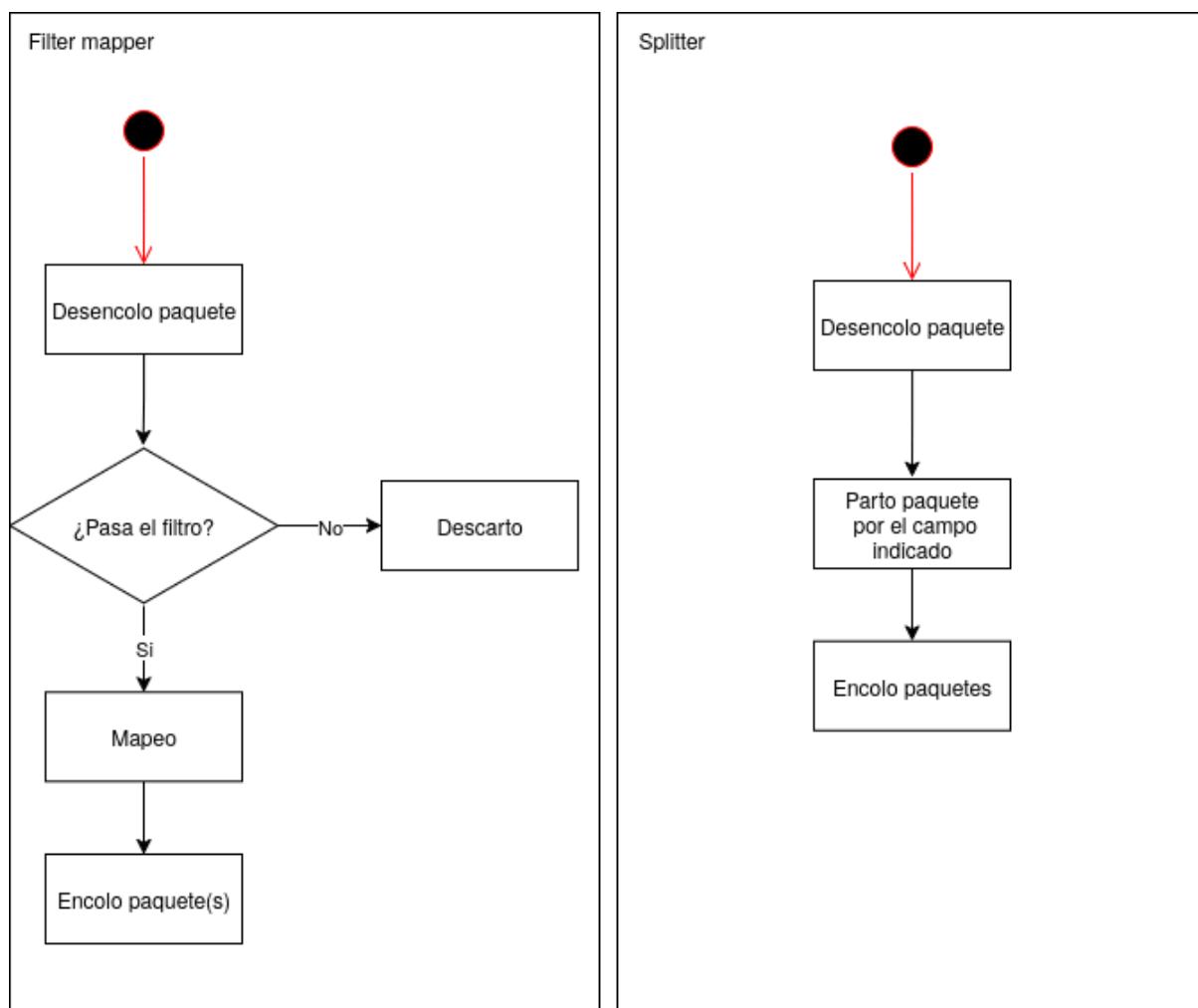


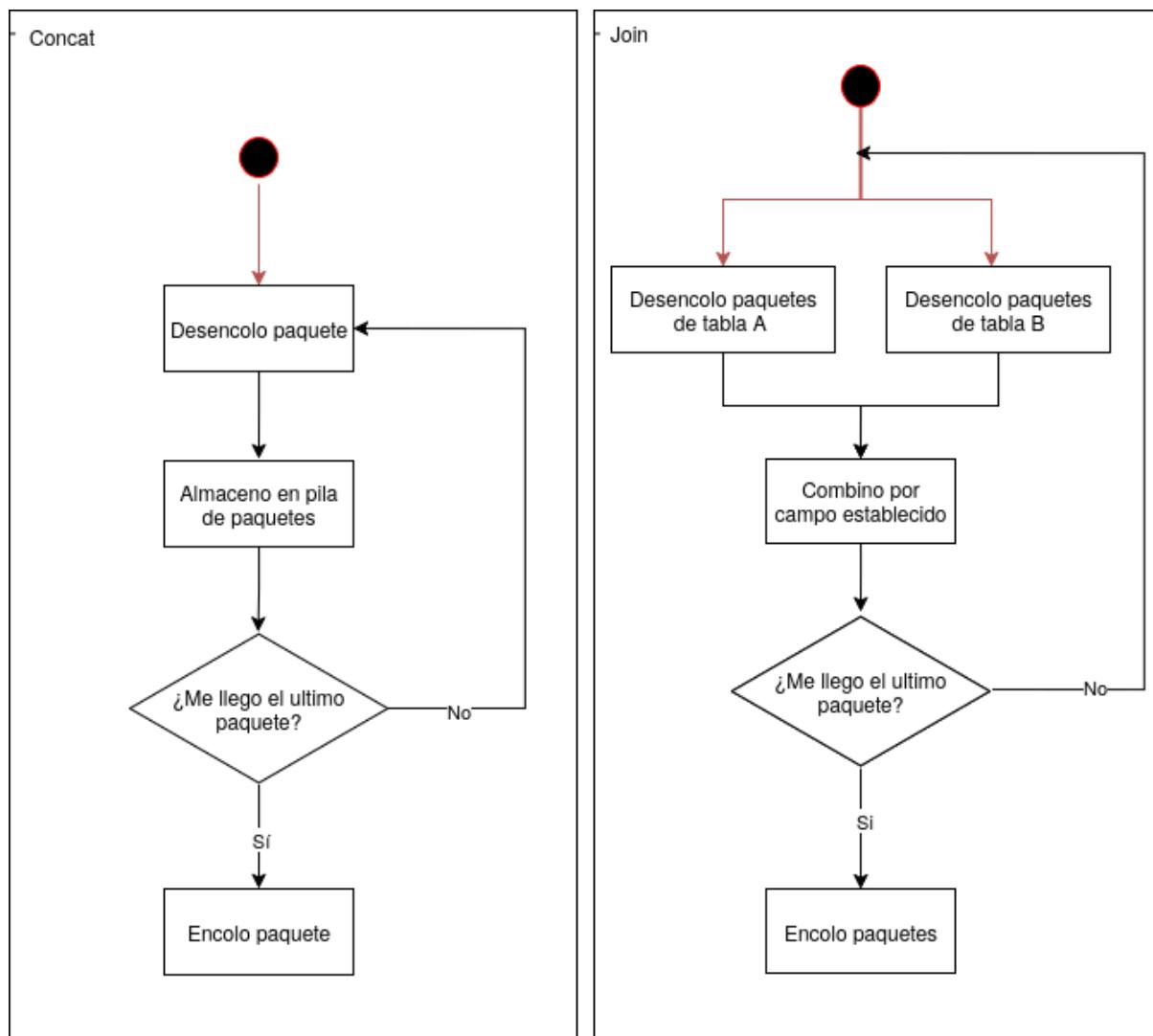
Diagrama de secuencia para la consulta N°4

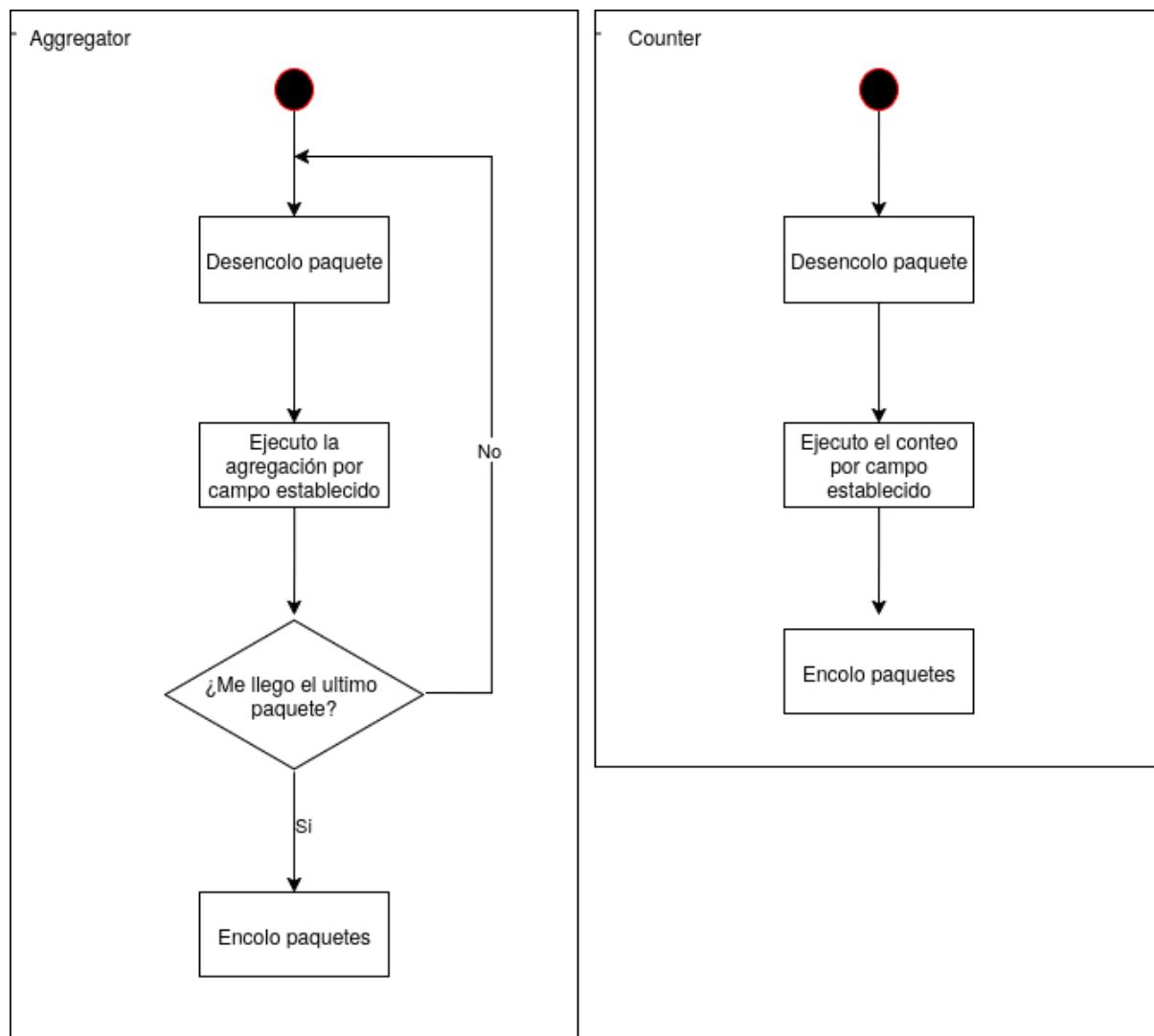


Diagramas de actividades

Cada uno de los diagramas representa el diagrama de actividades de cada uno de los workers; cada uno realiza una tarea específica y distinta.



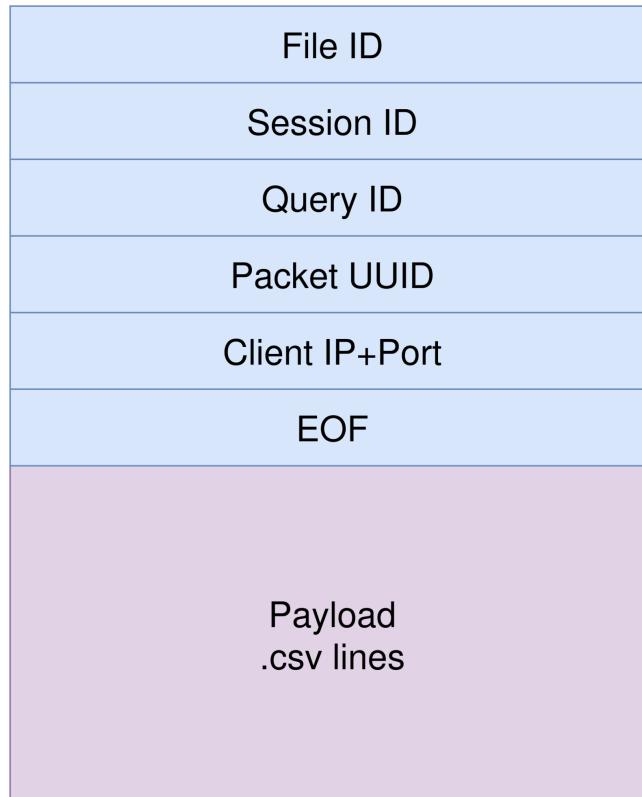




Vista de desarrollo

Se presenta el diagrama de paquetes de la organización de los módulos a desarrollar, con foco en la separación de incumbencias (procesamiento de datos de comunicación).

Formato paquete

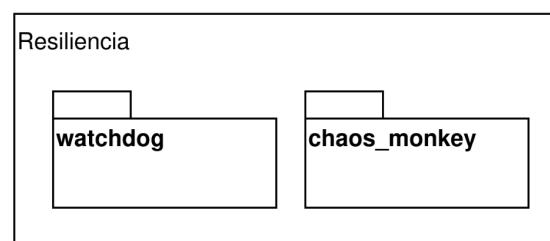
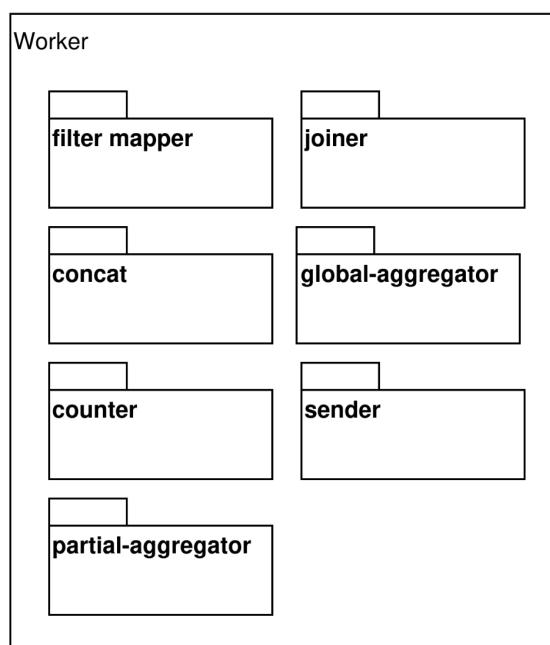
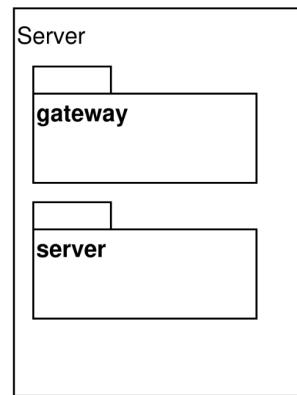
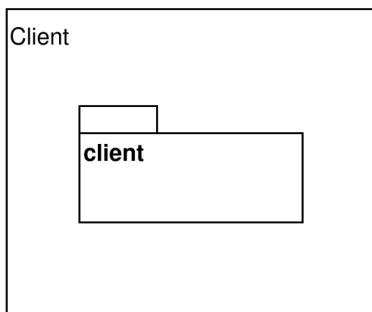


El formato de los paquetes es el siguiente:

- FileID: identifica al archivo al cual pertenecen los registros del payload
- SessionID: identificador único por sesión de procesamiento. Este identificador existe para poder distinguir los pedidos de procesamiento de un mismo cliente.
- QueryID: identificador de la query pedida.
- Packet UUID: identificador del segmento del paquete. Este identificador sigue el formato de versionado semántico (X.Y.Z) y es usado para la reconstrucción de paquetes.
- ClientIP + ClientPort: dirección donde el cliente va a quedarse escuchando a la espera de los resultados.
- EOF: campo que indica que este paquete contiene el último registro del archivo correspondiente.
- Payload: contiene los registros del csv.

Mientras el sistema procesa la información, el Client permanece en un loop de espera hasta recibir todas las respuestas. Una vez procesadas, los resultados viajan de regreso al Client a través del Sender, y finalmente se muestran en pantalla al usuario (o se genera un archivo).

Diagrama de paquetes



División de tareas

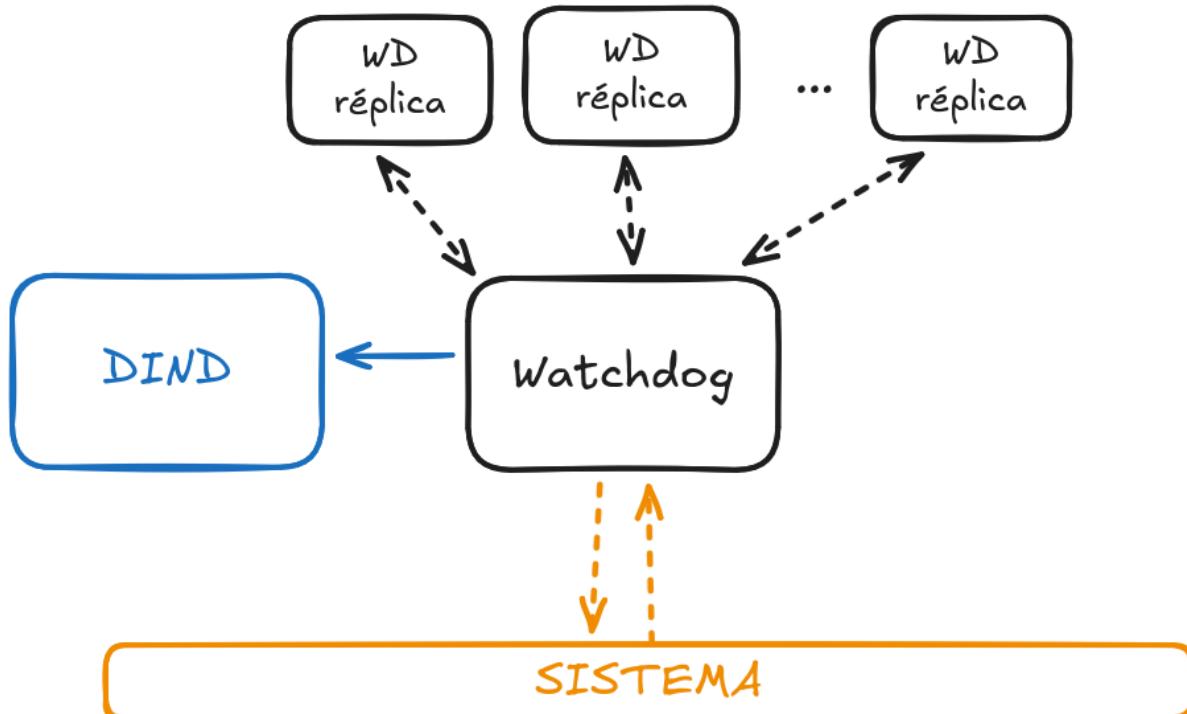
Actividad	A cargo
Cliente	Mariana
Gateway	Fabrizio
Nodo Rabbit	Bruno
Filter Mapper	Bruno
Splitter	Mariana
Joiner	Mariana
Counter	Bruno
Sorter	Fabrizio
Cutter	Bruno
Aggregator	Mariana
Concatenador	Fabrizio
Sender	Fabrizio
Dockerización y automatización	Bruno
Mejora de protocolo y paquetes	Fabrizio
Datos de testing y corrección de resultados	Mariana

Adiciones de entrega N° 4

Tolerancia a fallos

Las adiciones que permiten la tolerancia a fallos consisten en un nodo Watchdog que monitorea los *workers*. Cuando detecta que un *worker* cayó, utiliza el bind del socket a docker-in-docker para reiniciarlo.

Asimismo, el watchdog tiene réplicas que monitorean su estado. Cuando se cae el watchdog principal, se dispara una elección de líder que determina la réplica que ocupará su lugar.



Monitoreo de nodos

El monitoreo del *watchdog* a los *workers* se realiza utilizando un esquema de *healthchecks*. El *watchdog* principal envía, por UDP, un mensaje a cada uno de los *workers* y espera su respuesta.

Las direcciones son conocidas de antemano y se estableció que el puerto a utilizar para intercambiar los mensajes es el 1958.

Si no se detecta una respuesta del *worker* se reintenta hasta 3 veces antes de darlo por caído y reiniciarlo.

Reinicio del nodo

Para el reinicio se utiliza la técnica de *docker-in-docker* donde se hace un *bind* del socket de Docker en el host al socket en el contenedor del *watchdog*, consiguiendo poder enviar comandos de Docker programáticamente desde el código del *watchdog* y lograr las recuperaciones de nodos.

Recuperación del nodo

Para que un nodo recupere su estado al momento de haber caído, se implementa un algoritmo que funciona con una ventana y flushes a disco para mantener el estado del trabajo ante una caída.

El algoritmo utiliza una ventana móvil de paquetes. Cuando la ventana se llena, se dispara el procesamiento de los paquetes y se guarda el resultado en disco.

Para asegurar la atomicidad del guardado en disco, se utiliza la técnica de *Atomic File Rename*, en la cual se aprovecha la atomicidad provista por el kernel de Linux para renombrar un archivo. Se crea un nuevo archivo temporal, en este archivo se realizan los cambios y luego se hace un renombramiento. Si el renombramiento falla o el nodo cae, el archivo original sigue existiendo. Si el renombramiento es exitoso, se asegura la escritura del nuevo archivo.

Luego de bajar a disco los resultados, se borran los archivos de la ventana del disco para liberar el espacio.

Ante una caída, el nodo puede recuperarse haciendo uso de los archivos en disco y seguir su funcionamiento.

Cabe destacar que los nodos de tipo *joiner* tienen un tratamiento especial en este sentido, ya que no pueden realizar trabajo parcial sin tener la totalidad de una de las partes del *join*. Dado que establecimos que el joiner recibe pocos paquetes, el recibidor de paquetes del joiner no implementa una ventana pero se guarda todos los paquetes que recibe, sin utilizar una ventana para borrarlos.

Algoritmo de consenso para líder de Watchdogs

Ante la caída del Watchdog principal, otro debe tomar su lugar para continuar su monitoreo. Generalmente el Watchdog se acompaña de 2 copias más, teniendo 3 en total con una como líder.

Cuando el sistema recién se inicia, los nodos se ponen de acuerdo para elegir un líder mediante la implementación de un algoritmo *bully*.

Este algoritmo consiste en delegar la elección del líder a un proceso superior (decidido por algún criterio particular) y esperar que el nodo que cumple el criterio asuma el rol y lo comunique al resto.

En el caso de esta implementación, el criterio *bully* es que será líder aquel nodo cuyo nombre de servicio tenga el sufijo superior. Por ejemplo, al solicitar al sistema que el watchdog tenga 3 réplicas, aparecen los siguientes contenedores: *watchdog_1*, *watchdog_2* y *watchdog_3*.

Según el criterio explicado arriba, se inicia una elección de líder y el *watchdog_3* ocupará el lugar de líder. Si este cae, el primero de los otros dos que lo detecte, iniciará una elección de la cual resultará ganador el *watchdog_2* y se podrá continuar el monitoreo.

Además de ocupar su lugar, el nuevo líder intenta recuperar el nodo caído para seguir teniendo dos réplicas disponibles.

Chaos Monkey

Se creó un proceso encargado de generar “caos” entre los contenedores para comprobar que funciona la recuperación ante caídas de manera aleatoria. Para lograr la aleatoriedad, el proceso recorre un listado de los contenedores levantados, avanzando cada determinado tiempo. Al llegar a un nodo, se corre una probabilidad basada en los parámetros para decidir si se envía un SIGKILL a ese contenedor o no.

Se tienen dos parámetros de configuración:

- **Threshold:** un número entre 0 y 100 que refleja la probabilidad de que el *monkey* mate a un proceso de la lista.
- **Seed:** entrada para el pRNG para lograr cierta predictibilidad en el orden que caen los procesos.

Monitoreo de performance con dataset completo

Por motivos de practicidad y tiempos, no se logra correr el dataset completo en el tiempo asignado para la demostración sincrónica. Por ello se deja constancia de las distintas configuraciones y su comportamiento con el dataset completo.

Configuración de workers	Se incluyen caídas	Resultado temporal
1 de cada uno + 1 cliente	No	57m34.945s
3 de cada uno + 3 clientes	Sí	112m10.583s

Las pruebas indican que se tarda mucho en procesar todo el dataset habiendo añadido caídas y persistencia en disco, teniendo en cuenta que en experimentos de entregas anteriores, estas configuraciones solían estar en el orden de los 20-30 minutos.