FACULTAD DE INGENIERIA
Universidad de Buenos Aires

TP Diseño - Coffee Shop Analysis

[75.74] Sistemas Distribuidos I
Segundo cuatrimestre de 2025

Avecilla, Ignacio	105067	iavecilla@fi.uba.ar
Avila, Gaston	104482	gavila@fi.uba.ar
Muñoz, Juan Martín	106699	jmmunoz@fi.uba.ar

Índice

1. Alcance

El presente informe presenta la documentación de un sistema distribuido flexible, robusto y escalable, capaz de resolver las consultas otorgadas por la catedra con una cantidad de unidades de procesamiento mayor o igual a uno.

Las consultas a resolver son:

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 cant) 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.

2. Arquitectura

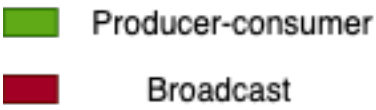
2.1. Vista Física

Se muestra la enteridad del sistema y las conexiones existentes entre las diversas entidades del sistema.

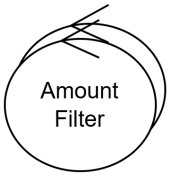
2.1.1. Diagrama de Robustez

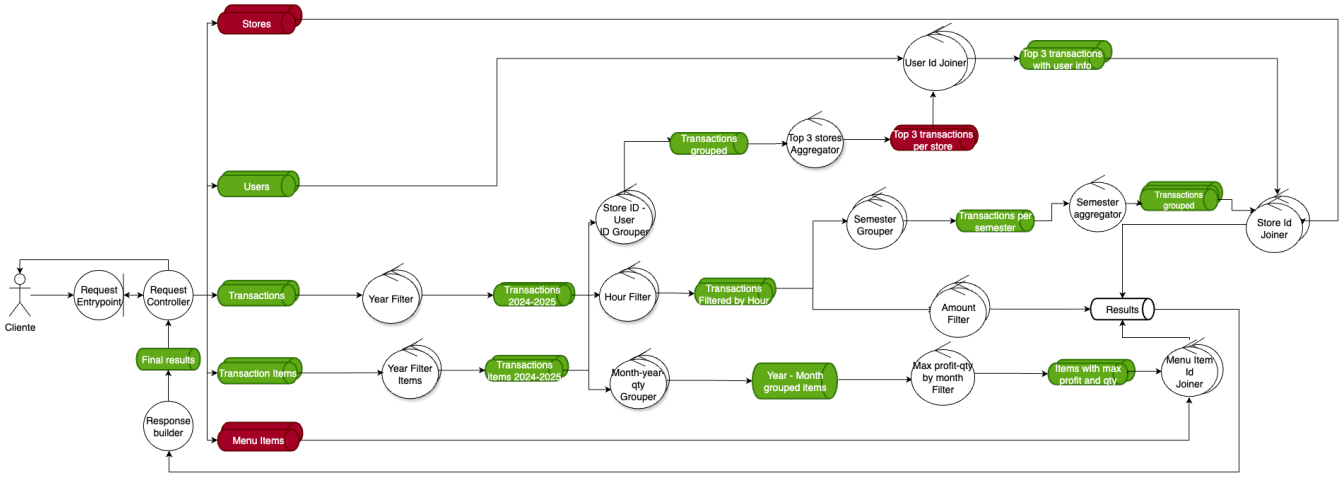
Se visualizan todos los componentes que interactuan en nuestro diseño, desde la interacción inicial del cliente, hasta la finalización del procesado de todas las consultas.

- Los trabajadores se comunican a través de la inserción y consumo de datos en diversas colas, las cuales pueden ser accedidas por multiples trabajadores concurrentemente. Se distinguen 2 tipos de cola segun como se distribuyen los mensajes:



- Las entidades que pueden ser escaladas a multiples unidades de computo se representan de la siguiente forma:





Cada worker tiene asociado una cola de input y una cola de output, salvo los aggregators que obtienen resultados de muchos workers en una unica cola y se encargan de agregar los resultados.

Las colas marcadas como tipo "broadcast"se basan en multiples colas cada una de ellas conectadas a un worker de salida, el worker que se encarga de producir en esta cola iterara cada una de ellas dejando el mismo dato en todas ellas.

Por otro lado el manejo de EOF (end of file) se realiza mediante un mensaje especial que indica a los workers que no habra mas datos a procesar. El request handler encola ese mensaje en todas las colas de entrada y los sucesivos workers iran propagando el mensaje de EOF a medida que terminan su trabajo, una vez que todos los EOF correspondientes lleguen a las colas de resultados entonces el Response Builder arma la respuesta y la envia al cliente indicando que una query ya fue completamente procesada

Todos los resultaados seran insertados en diferentes colas, una para cada consulta, el Response builder sera el encargado de leer de ellas y mandar una unica respuesta para la consulta correspondiente

A continuación, se muestran cuatro extractos del diagrama de robustez, destacando los aspectos más relevantes de cada consulta.

Consulta 1

Es la más simple. Requiere tres filtros encadenados (**Year Filter**, **Hour Filter** y **Amount Filter**). Cada etapa puede escalarse mediante un esquema **producer–consumer**: los workers consumen mensajes de la cola, procesan la entrada y deciden si reenviarla o descartarla.

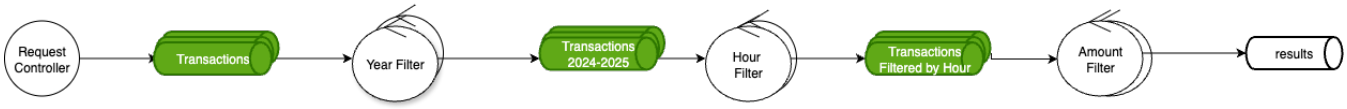


Figura 1: Diagrama de robustez - Consulta 1

Consulta 2

Incorpora las colas de tipo **fan-out (broadcast)** para mandar un mismo mensaje a multiples workers:

- **Transactions with profit and total amount**: En este caso contenemos una unica cola que va a tener todos los resultados parciales de nuestro groupers segun la data que pudieron obtener de la cola anterior, en este caso el mas profit and quantity filter sera el encargado de no solo filtrar los resultados si no tambien agregarlos para tener los resultados finales
- **Menu Items** (fan-out): los datos se difunden a todos los `Menu Item Joiner`. Estos almacenan en memoria la lista de ítems (pocos y estáticos) y luego consumen de la cola **Items with max profit and qty** (producer–consumer) para realizar los joins. Como el total de items del menú siempre son 8 entonces podemos cargarlos en memoria sin problemas.

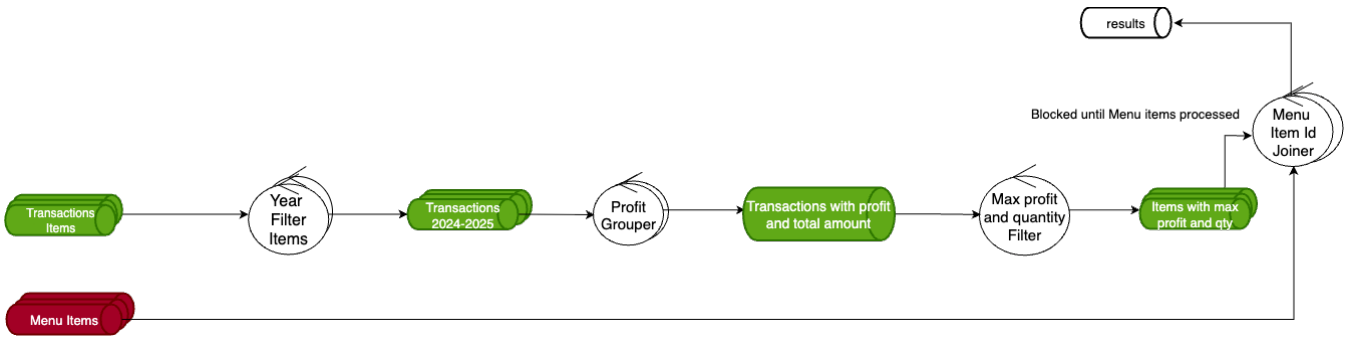


Figura 2: Diagrama de robustez - Consulta 2

Consulta 3

Utiliza una cola de tipo **broadcast**:

- **Transactions per semester**: Siguiendo la logica anterior tenemos un único aggregator que se encarga de obtener los datos parciales de los groupers y los agrega en un solo resultado final, el cual sera enviado a la cola de **Transactions grouped**.
- **Stores** (fan-out): los datos se difunden a todos los `Store ID Joiner`. Estos almacenan en memoria la lista de ítems (pocos y estáticos) y luego consumen de la cola **Transactions grouped** (producer–consumer) para realizar los joins. Como el total de items del menú siempre son 10 entonces podemos cargarlos en memoria sin problemas.

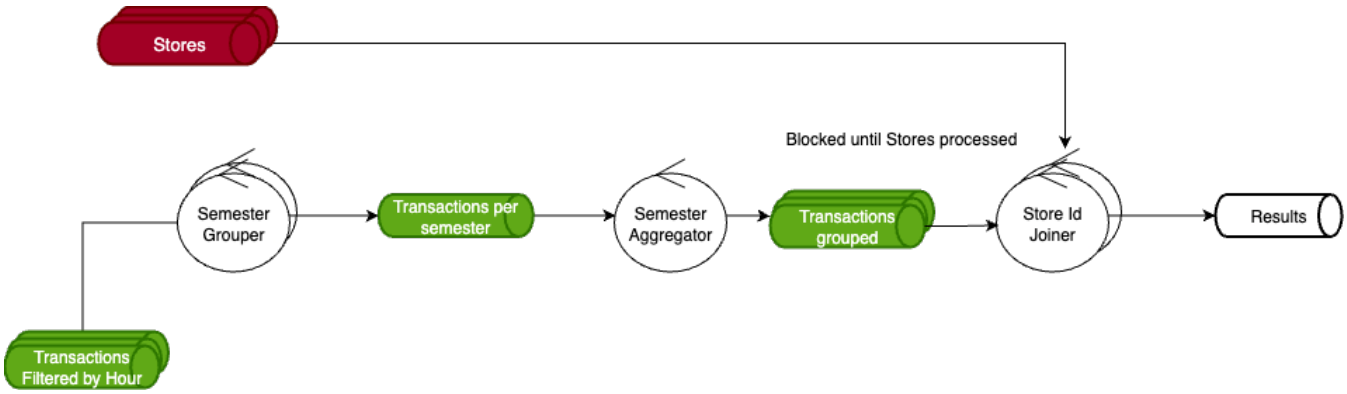


Figura 3: Diagrama de robustez - Consulta 3

Consulta 4

Requiere dos colas de tipo **broadcast**:

- **Transactions per Store ID - User ID**: Al igual que en los casos anteriores, tenemos un agregator que recibe los datos parciales de los groupers y los agrega en un solo resultado final, el cual sera enviado a la cola de **Top 3 transactions per store**.
- **Top 3 transactions per store**: Estas colas se comportan como si fuera un fan-out exchange donde todos los resultados llegan a todos los workers, ya que estos resultados nunca son muy grandes (solo 3 resultados por cada sucursal) y al tenerlo en memoria, los workers pueden realizar los joins de manera correcta.

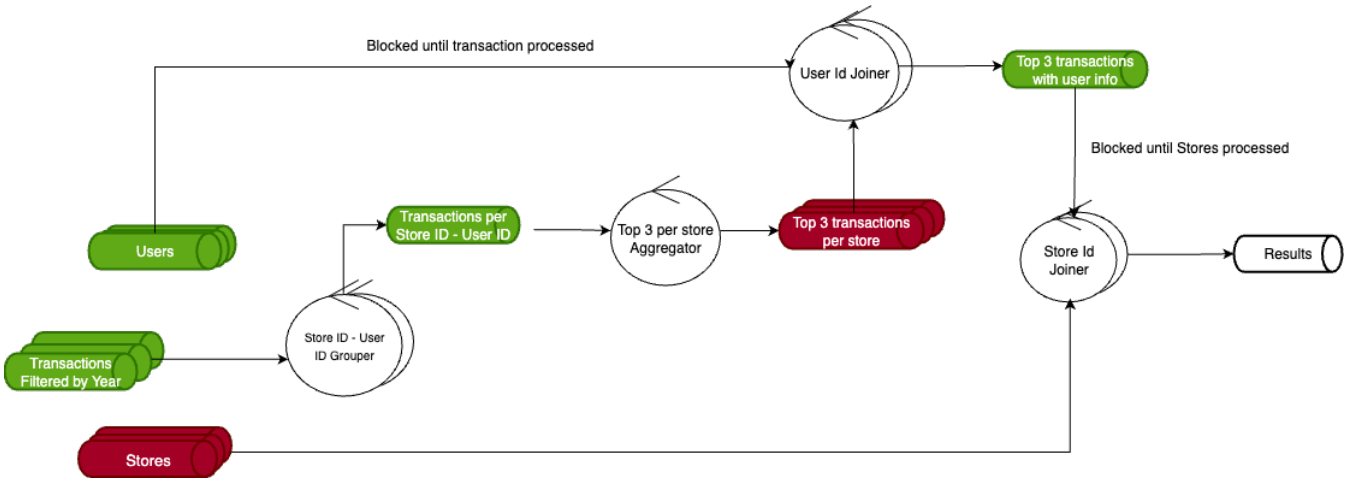
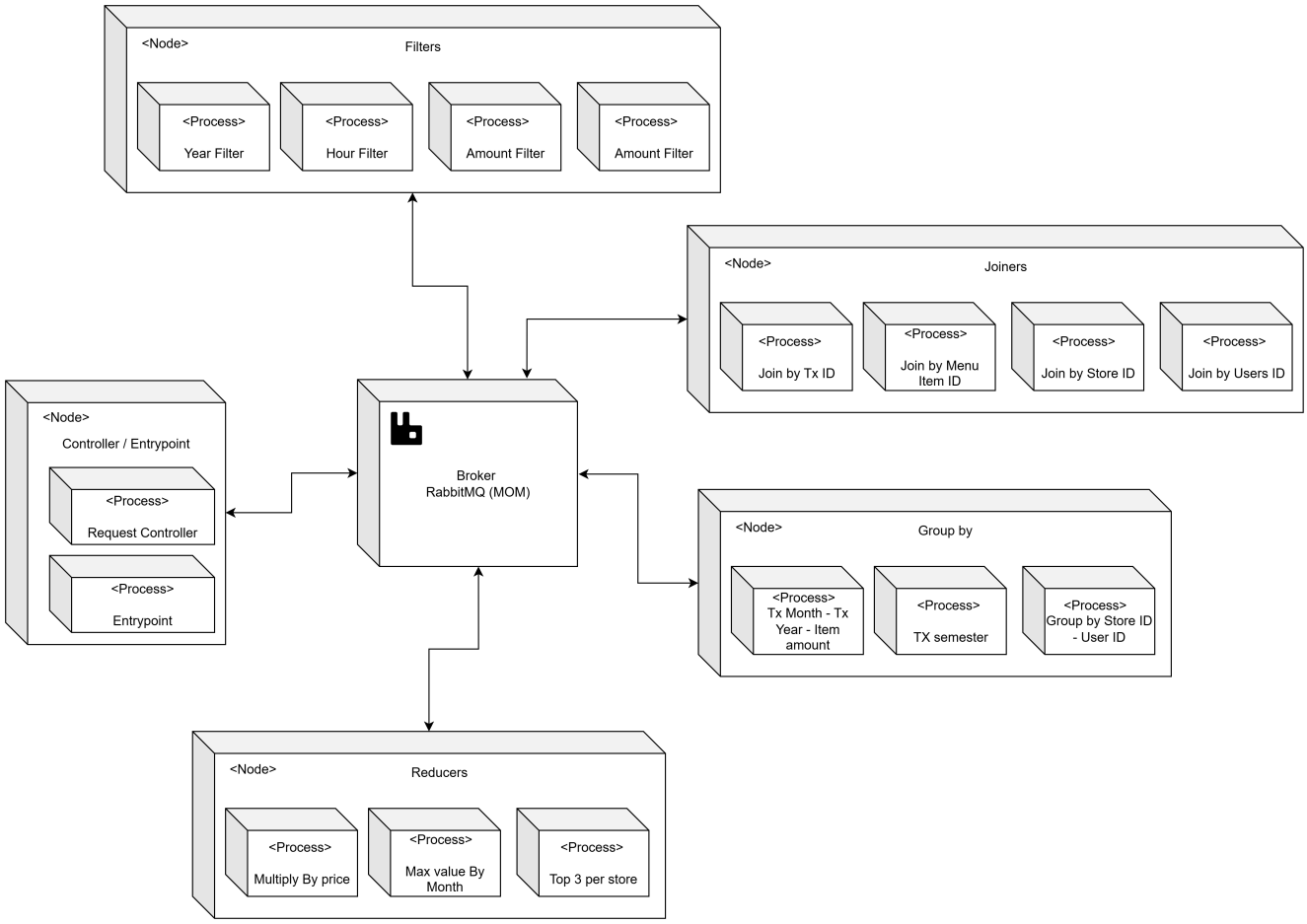


Figura 4: Diagrama de robustez - Consulta 4

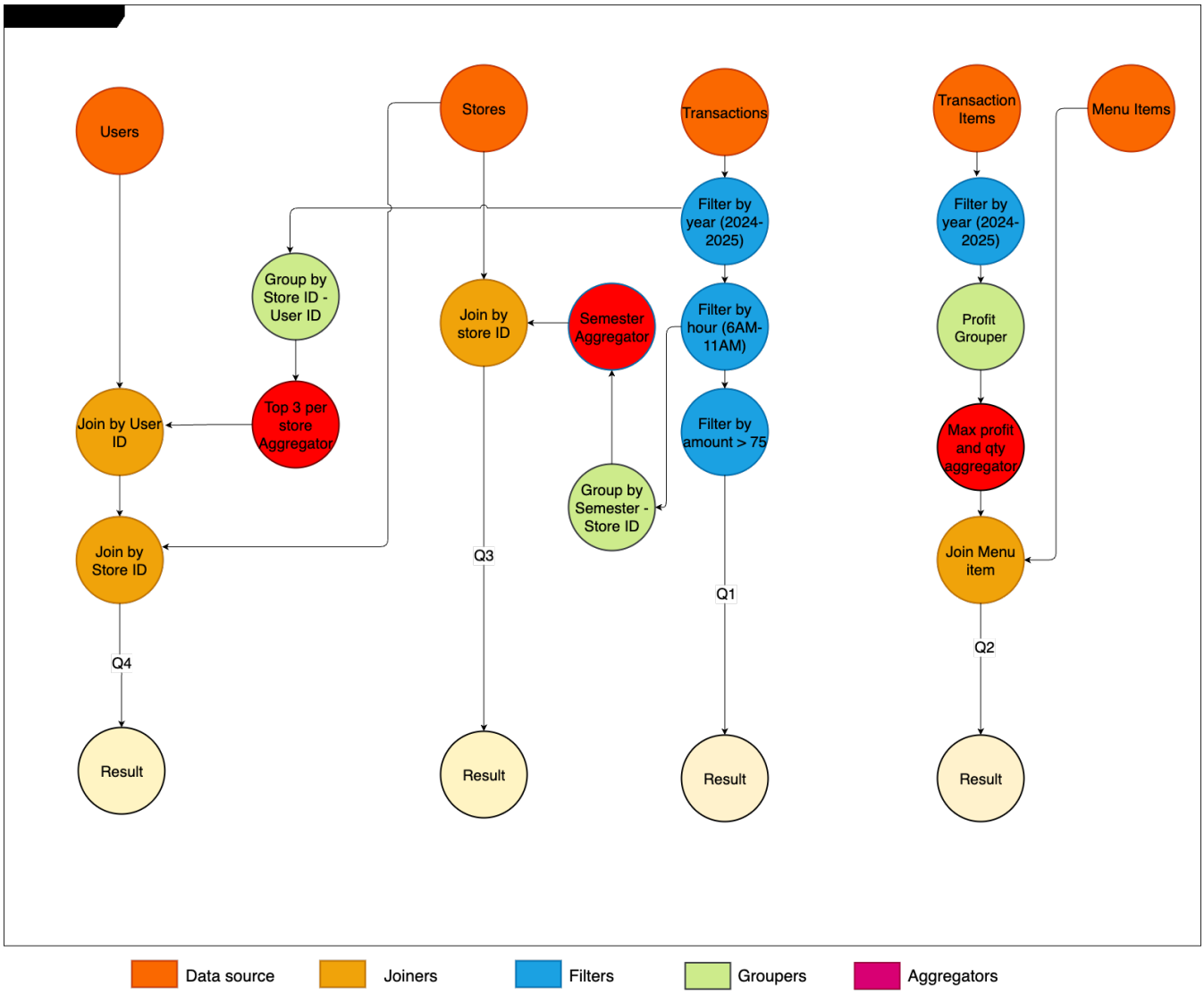
2.1.2. Diagrama de Despliegue

Podemos ver como los toda comunicación interna es realizada mediante RabbitMQ (el middleware).


<Node>
Filters
<Process>
Year Filter
<Process>
Hour Filter
<Process>
Amount Filter
<Process>
Amount Filter
<Node>
Joiners
<Process>
Join by Tx ID
<Process>
Join by Menu Item ID
<Process>
Join by Store ID
<Process>
Join by Users ID
<Node>
Controller / Entrypoint
<Process>
Request Controller
<Process>
Entrypoint
Broker
RabbitMQ (MOM)
<Node>
Group by
<Process>
Tx Month - Tx Year - Item amount
<Process>
TX semester
<Process>
Group by Store ID - User ID
<Node>
Reducers
<Process>
Multiply By price
<Process>
Max value By Month
<Process>
Top 3 per store

2.2. Vista Lógica

2.2.1. DAG



Se visualiza un directed acyclic graph que muestra el flujo de datos siendo atravezado por los distintos componentes, comenzando de arriba hacia abajo. Los trabajadores (workers) se dividen en agrupadores (groupers), acumuladores (joiners), agregadores (aggregators) y filtradores (filters). Las fuentes de información inicial, y donde se almacena finalmente lo procesado son a nivel de implementación colas. Los aggregators son los nodos encargados de obtener data de los groupers y acumularla en un solo resultado final segun sea necesario, de esta manera los groupers pueden recibir cualquier tipo de data y agrupar en base a los datos que le lleguen.

2.3. Vista de Desarrollo

Aquí podemos visualizar como esta planeada la arquitectura del sistema desde la perspectiva del código. Se divide el sistema en distintos modulos para Cliente, Worker, Request Controller, Response builder y Middleware.

2.3.1. Diagrama de Paquetes

Se muestran los distintos modulos a implementar:

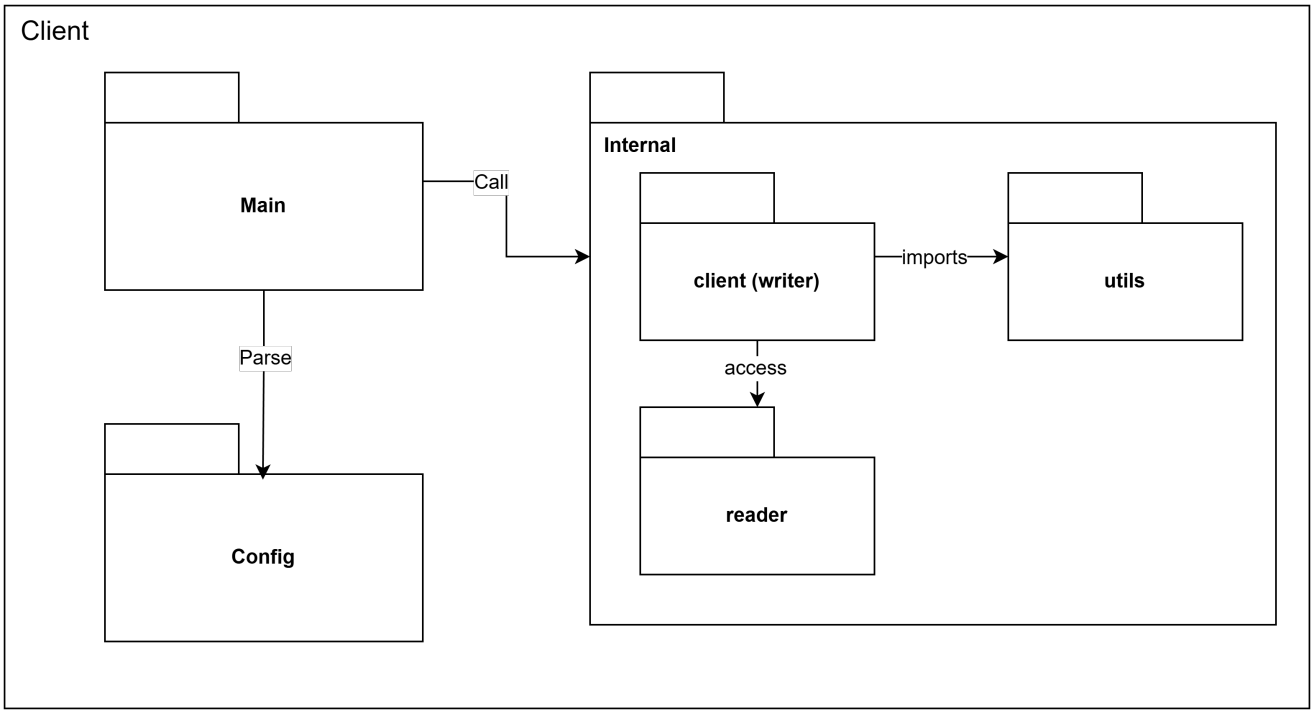


Figura 5: El cliente tiene la responsabilidad de comunicarse con el Request controller, siguiendo los parametros especificados en su configuración.

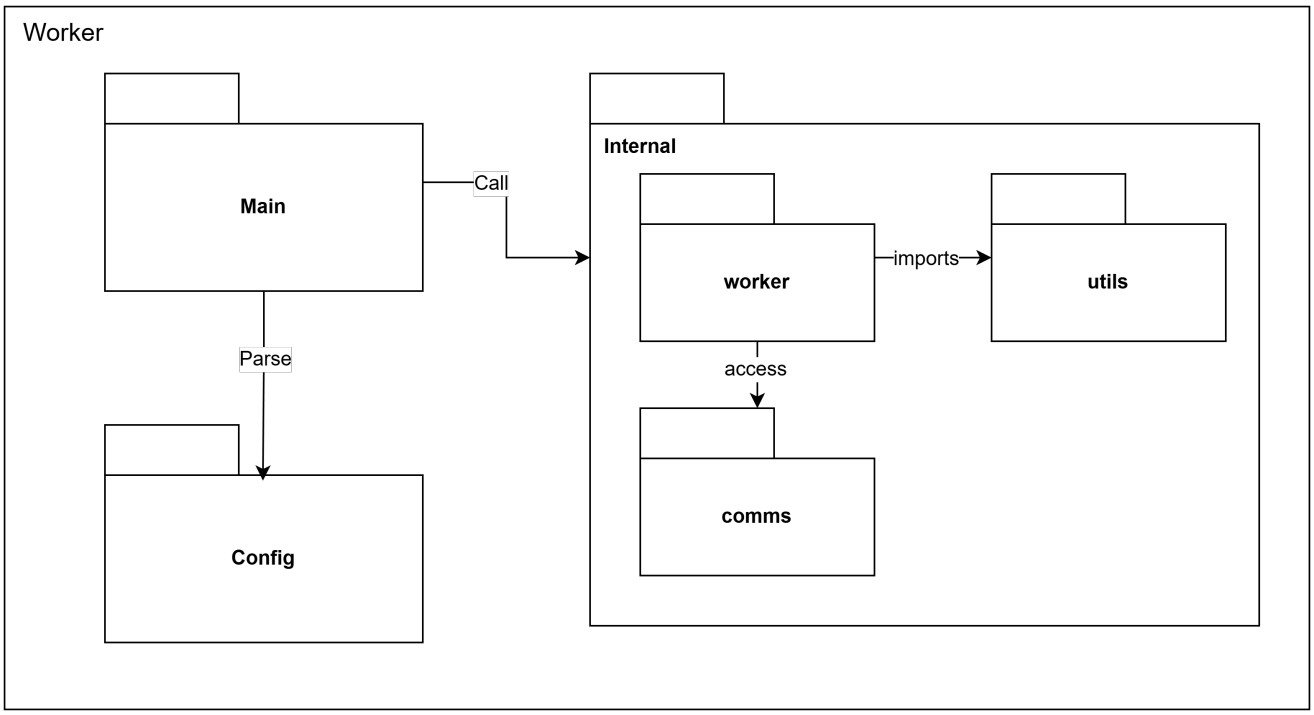


Figura 6: Los workers tienen la tarea de filtrar, acumular o agrupar los diversos tipos de datos que lean de las colas a las que estes subscriptos.

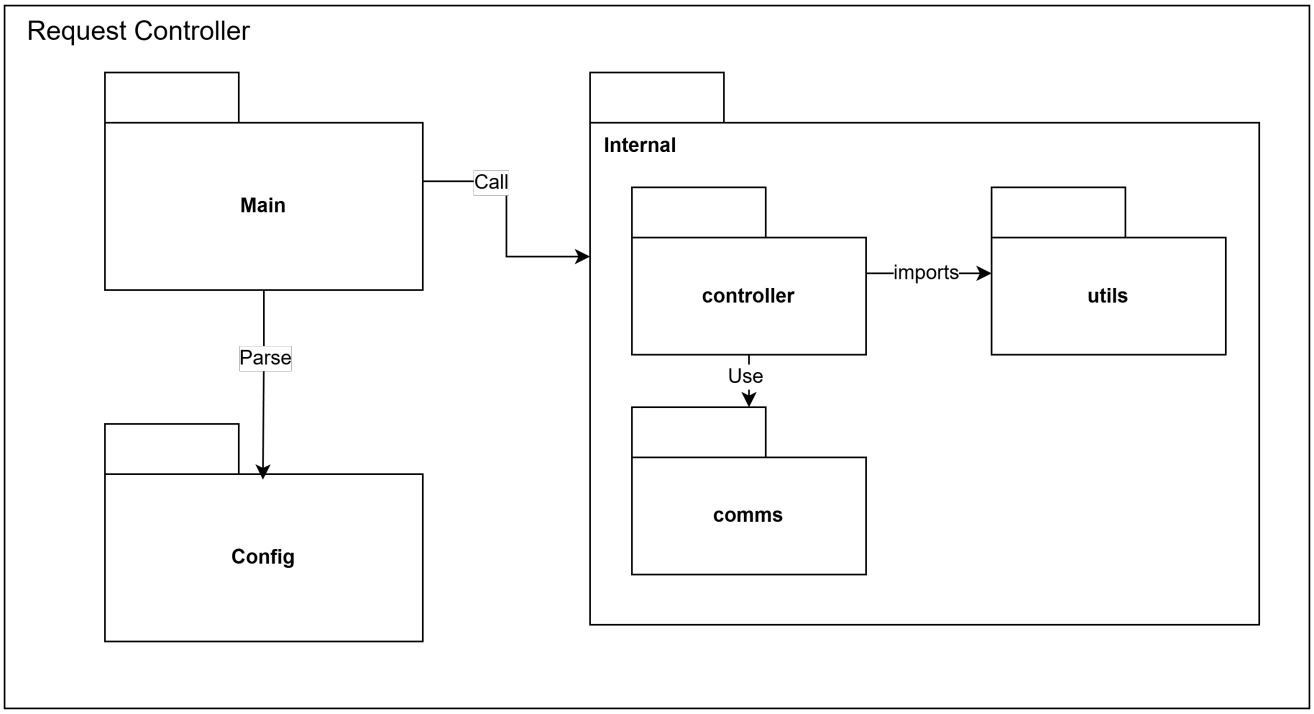


Figura 7: La tarea del request controller es interpretar los mensajes del cliente, para poder derivarlos a las diferentes colas de RabbitMQ.

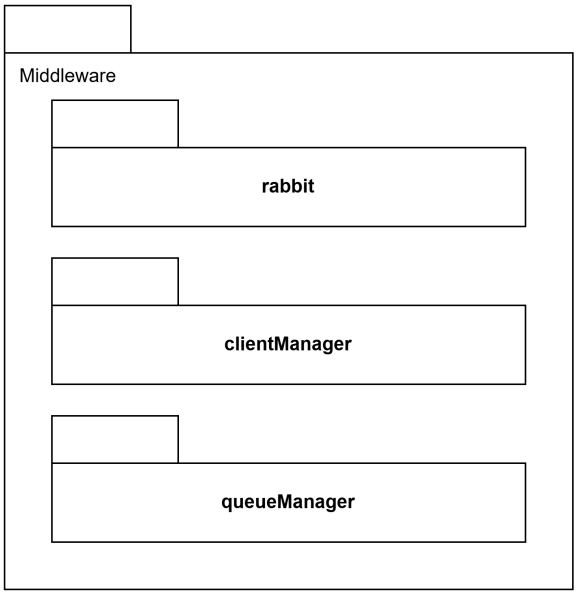
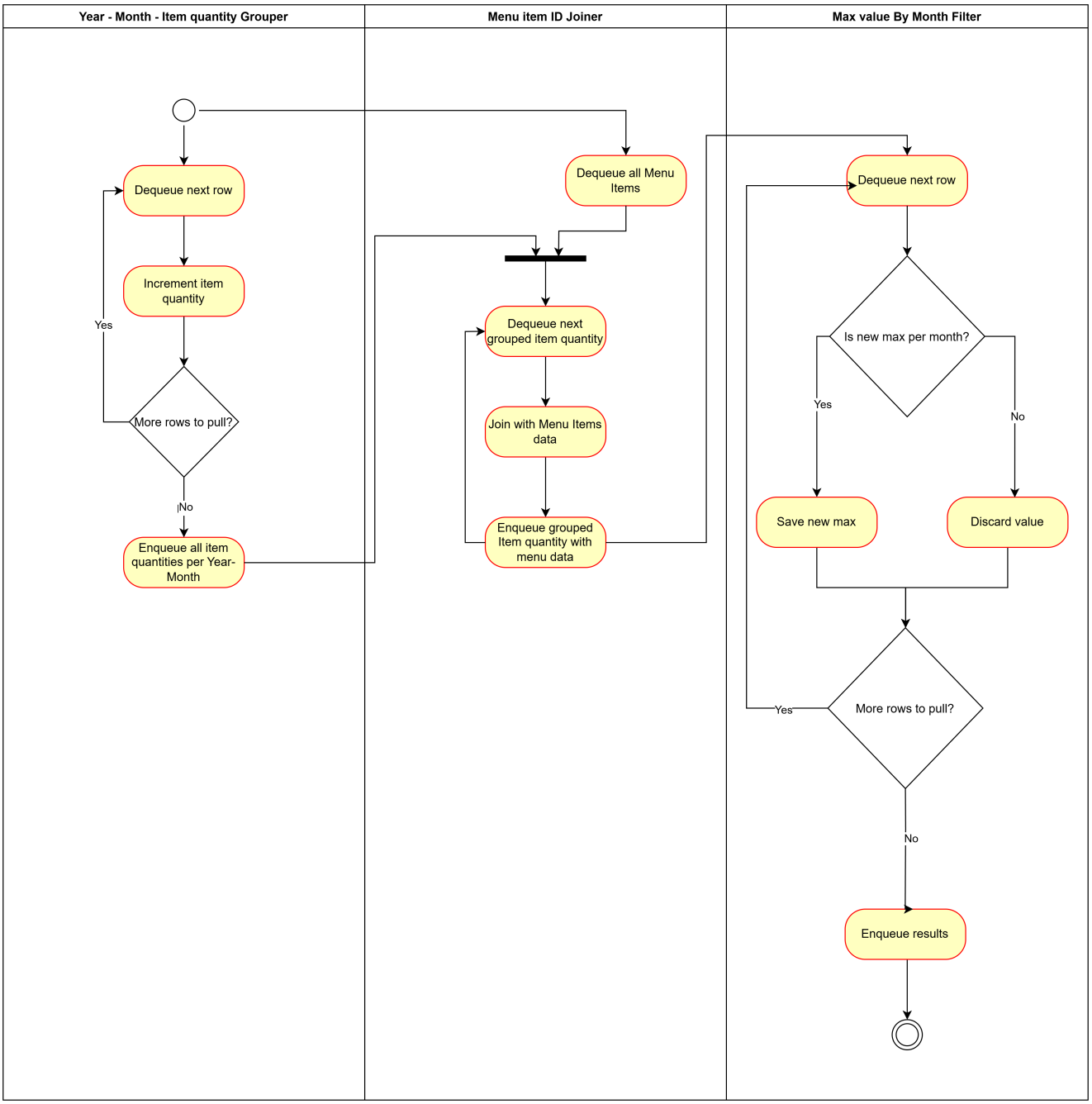


Figura 8: El Middleware tiene un manejo de cola y cliente, asi como también las librerias necesarias para el interfazado con RabbitMQ

2.4. Vista de Procesos

Se representa la interacción entre los componentes del sistema, su forma de comunicación de principio a fin. Es posible visualizar la concurrencia del sistema, asi como también la escalabilidad y distribuición de tareas.

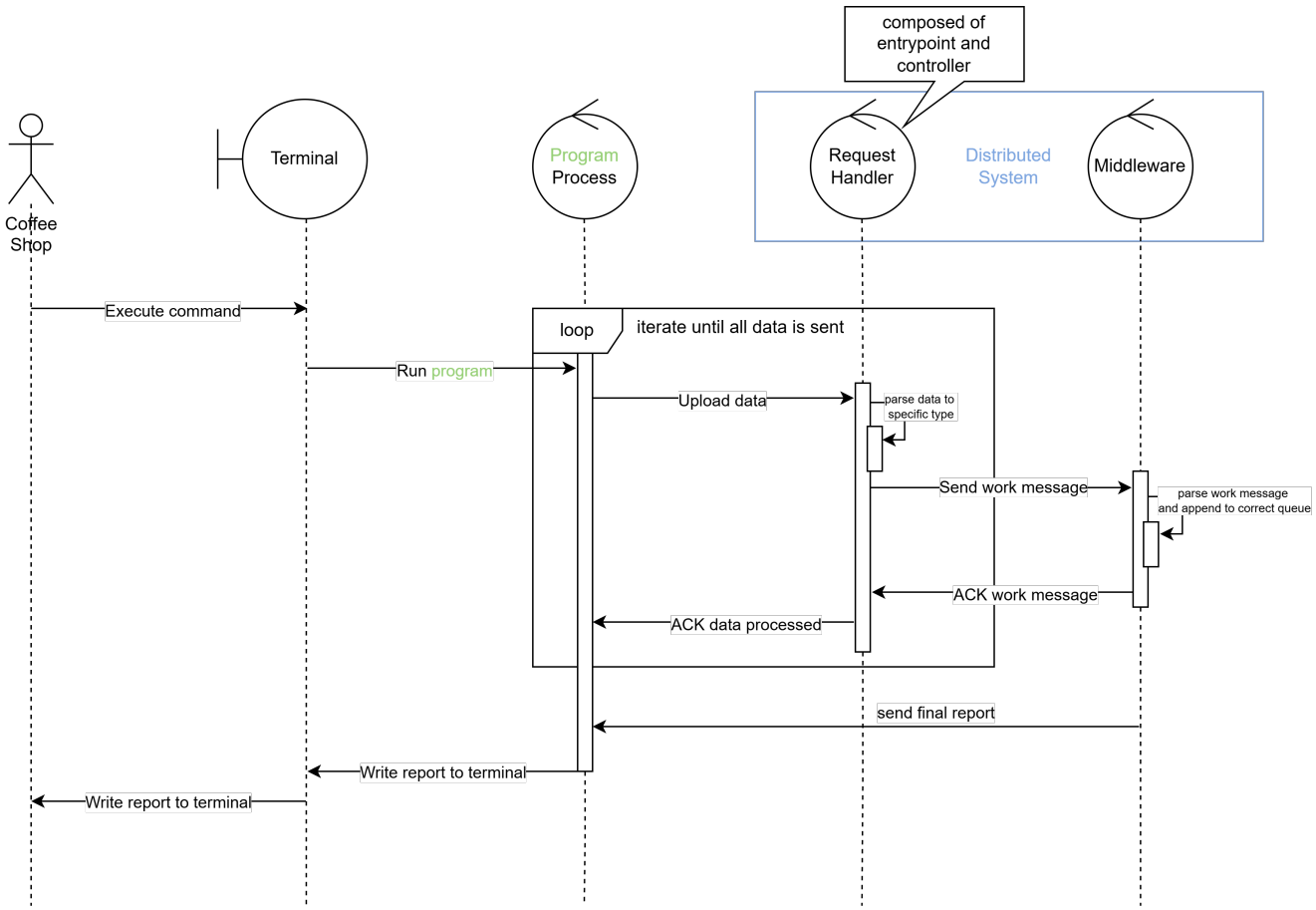
2.4.1. Diagrama de Actividad



Se muestra el parcialmente el proceso de obtener los productos más vendidos y que más ganancias han generado, para cada mes, en este caso se representa solamente la parte de mayor complejidad* de obtención, la de mayor valor y no solo mayor cantidad de ventas.

***Se considera más complejo debido a que los pasos necesarios para obtener este dato es superior a la de calcular el producto más vendido en cantidad.**

2.4.2. Diagrama de Secuencia



Podemos ver como el cliente desde la terminal inicializa el programa, el cual establece conexión con el sistema mediante el Request Handler. La comunicación entre ellos consistira en enviar todos los datos necesarios en bucle, cada mensaje transmitido por el cliente incluye un header informando el tipo de archivo transmitido, el tamaño del payload, y un bit que indica si hay datos pendientes por transmitir posteriores a ese mensaje. Una vez que ya no haya nada más que enviar, el Middleware le enviará el programa el resultado final, el cual sera mostrado por terminal al usuario.