.UBAfiuba
FACULTAD DE INGENIERÍA

Sistemas Distribuidos
75.74

Trabajo Práctico:
Coffee Shop Analyzer

Nombre	Padrón
Baldi, Tomás	108317
Katta, Gabriel	105935
Bellido, Santiago	106449

Índice

1. Alcance

El trabajo consiste en el diseño de un **sistema distribuido** para analizar información transaccional de una cadena de cafeterías en Malasia.

El sistema debe cumplir los siguientes **requerimientos funcionales**:

1. Filtrar transacciones realizadas en 2024 y 2025, entre las 06:00 y las 23:00 horas, con monto mayor o igual a 75.
2. Identificar productos más vendidos y los que más ingresos generaron, mes a mes durante 2024 y 2025.
3. Calcular el *Total Payment Value (TPV)* por semestre y por sucursal, considerando únicamente transacciones dentro del rango horario establecido.
4. Obtener la fecha de cumpleaños de los tres clientes con mayor cantidad de compras en cada sucursal.

A continuación se presentarán los Diagramas correspondientes a la propuesta de solución:

2. Contexto

2.1. Diagrama de Secuencia

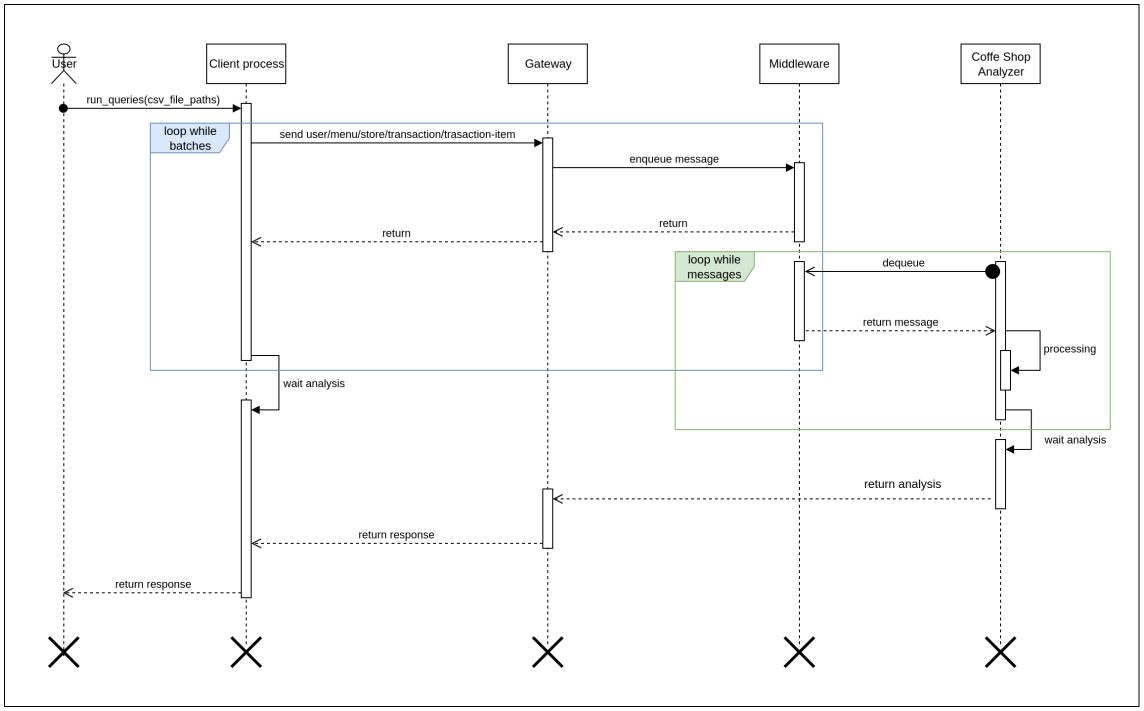

Figura 1: Diagrama de Secuencia del sistema.

En el diagrama se observa que, una vez iniciada la consulta por parte del *User*, el proceso principal del cliente envía la información obtenida de los archivos CSV en *batches* hacia el *Gateway*. Este componente encola los mensajes a través del *Middleware*. El *Coffee Shop Analyzer* desencola la información para ejecutar los *pipelines* y procesar la consulta. Finalizado el análisis, el resultado retorna al *User* a través del *Gateway*.

2.2. DAG

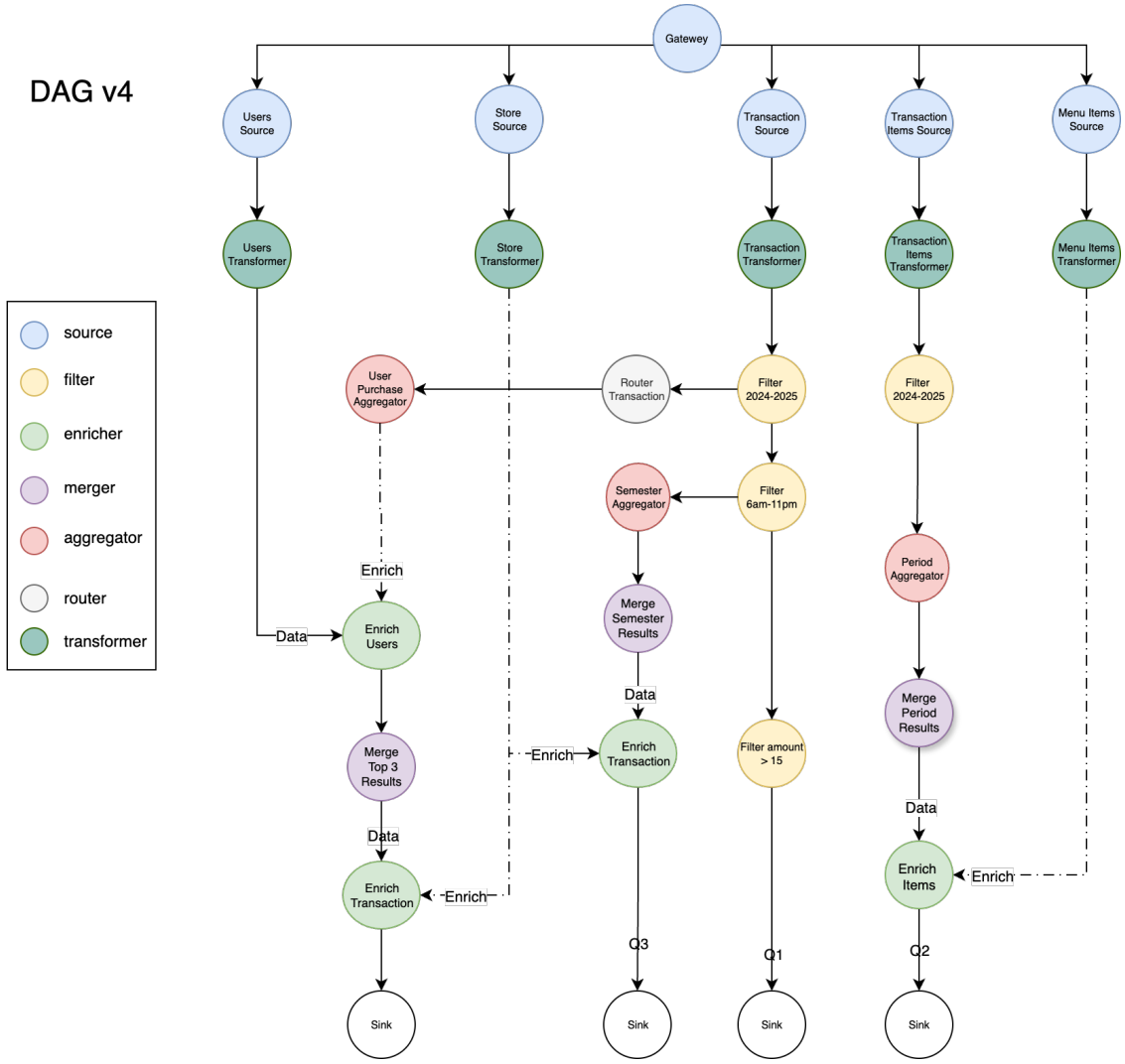

Figura 2: Pipeline de procesamiento (DAG).

Componentes del *pipeline*:

- **Transformers**: encargado de pasar las lineas csv del batch a entidades en un nuevo batch
- **Sources**: colas con la información enviada por el `client_process`, pueden contener batches o registros indivuales.
- **Aggregators**: acumulan en buffer y construyen estructuras que sintetizan información.
- **Filters**: aplican condiciones para filtrar datos.
- **Mergers**: fusionan resultados parciales de nodos anteriores.
- **Enrichers**: combinan datos de dos fuentes para generar una única fuente enriquecida.

3. Contenedores

3.1. Diagrama de Despliegue

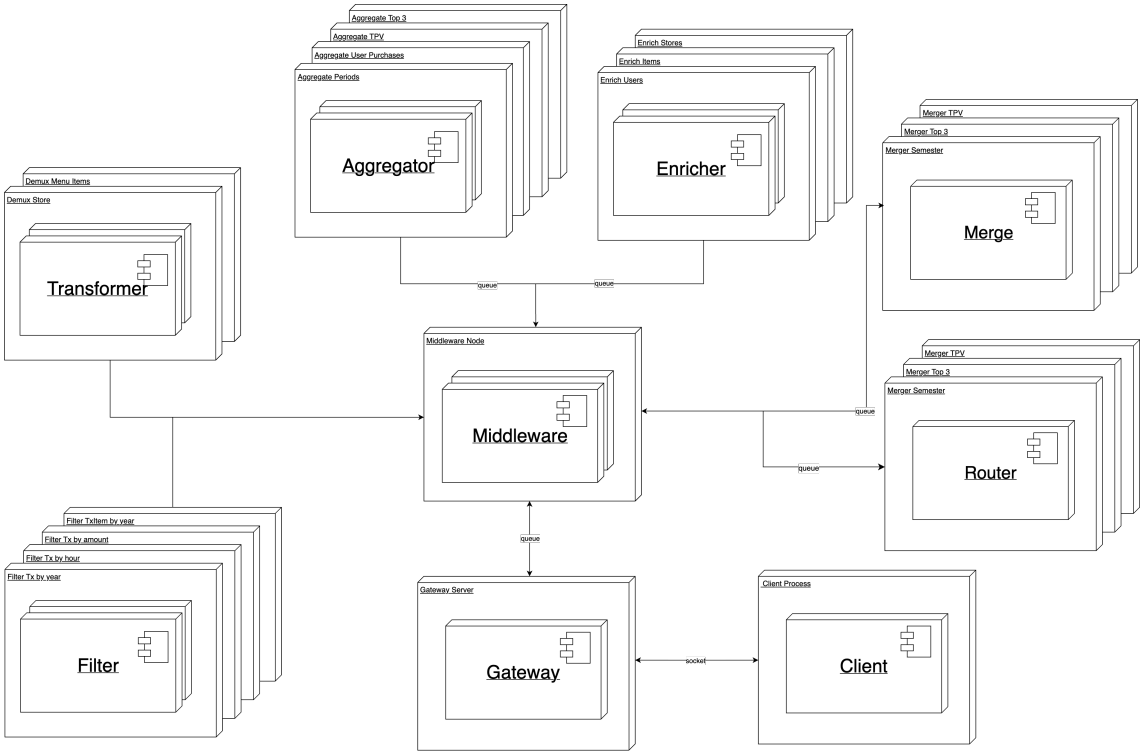

Figura 3: Comunicación de alto nivel entre elementos del sistema.

La mayoría de los componentes se comunican mediante el *Middleware* (RabbitMQ). El *Gateway* y el *Client* manejan su comunicación por sockets TCP.

Para simplificar, se agrupan componentes cuyos *workers* realizan operaciones similares sobre distintos dominios:

- **Transformer**: múltiples *transformer* (User, Transaction, Transaction Item, Store, Menu Item).
- **Router**: routea Transactions a diferentes destinos.
- **Filter**: múltiples *workers* (Amount, Hourly, Yearly).
- **Aggregator**: múltiples *workers* (Top 3, TPV, User Purchases, Periods).
- **Enricher**: múltiples *workers* (Stores, Items, Users).
- **Merge**: *worker* único por dominio (TPV, Top 3, Semester) por estado compartido.
- **Middleware**: coordina la comunicación asíncrona (RabbitMQ).
- **Gateway y Client**: interfaz de acceso al sistema.

4. Componentes

4.1. Diagrama de Robustez

El diagrama muestra la distribución de instancias y cómo trabajan en conjunto para responder *queries*. Grupos como filtros, enriquecedores y agregadores usan múltiples *workers* por ser *stateless*, mientras que los *mergers* requieren instancias únicas para juntar resultados.

Ver Figura 4 en la página siguiente

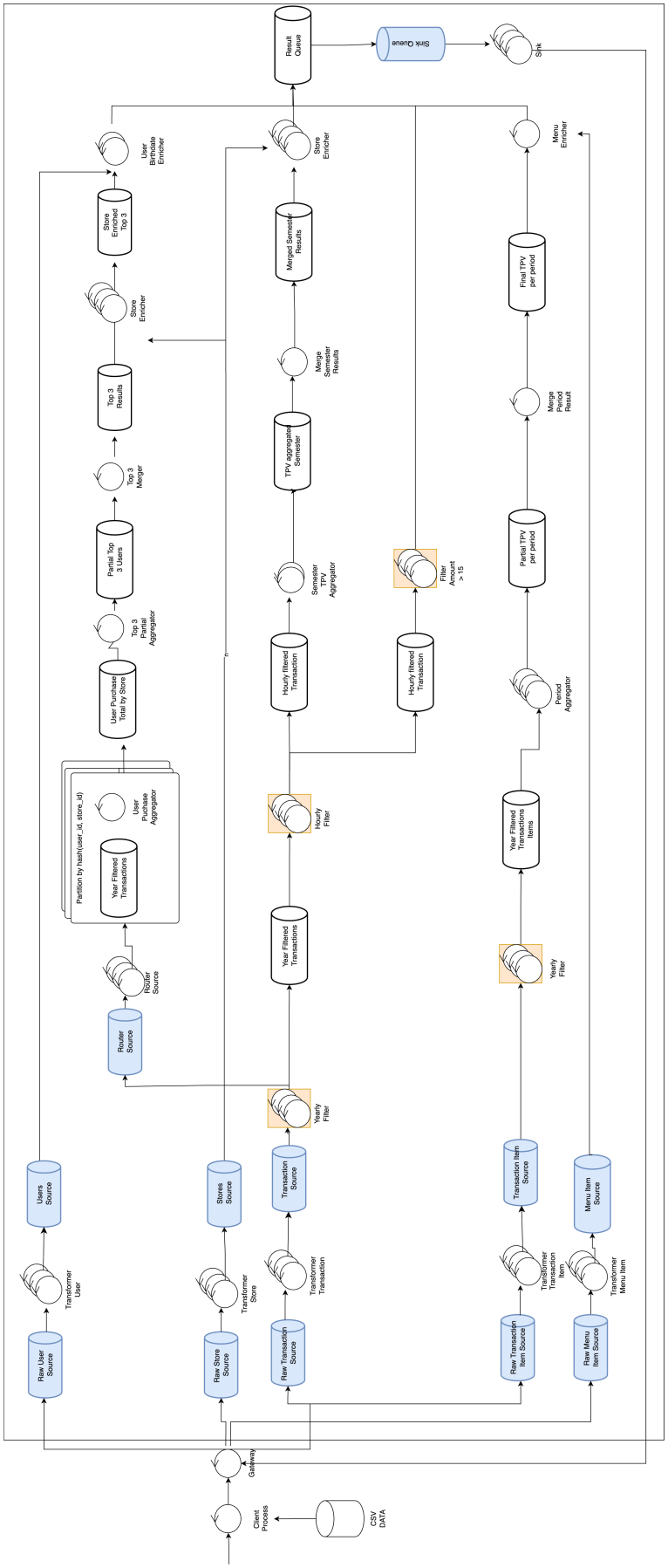

Figura 4: Diagrama de robustez: distribución de instancias del pipeline.

4.2. Diagrama de Actividad

El **User Purchase Aggregator** (múltiples instancias, shardeado por (*user_id*, *store_id*)) recibe transacciones filtradas (años 2024–2025) y construye una estructura interna con la cantidad de compras por usuario y tienda. Al finalizar, sus resultados para esos pares son definitivos.

Luego, el **Top 3 Aggregator** (shardeado por *user_id*) arma el top 3 parcial por tienda. Por último, un *worker* **Merge Top 3 Result** fusiona todos los parciales para obtener el top 3 final por tienda.

Ver Figura 5 en la página siguiente

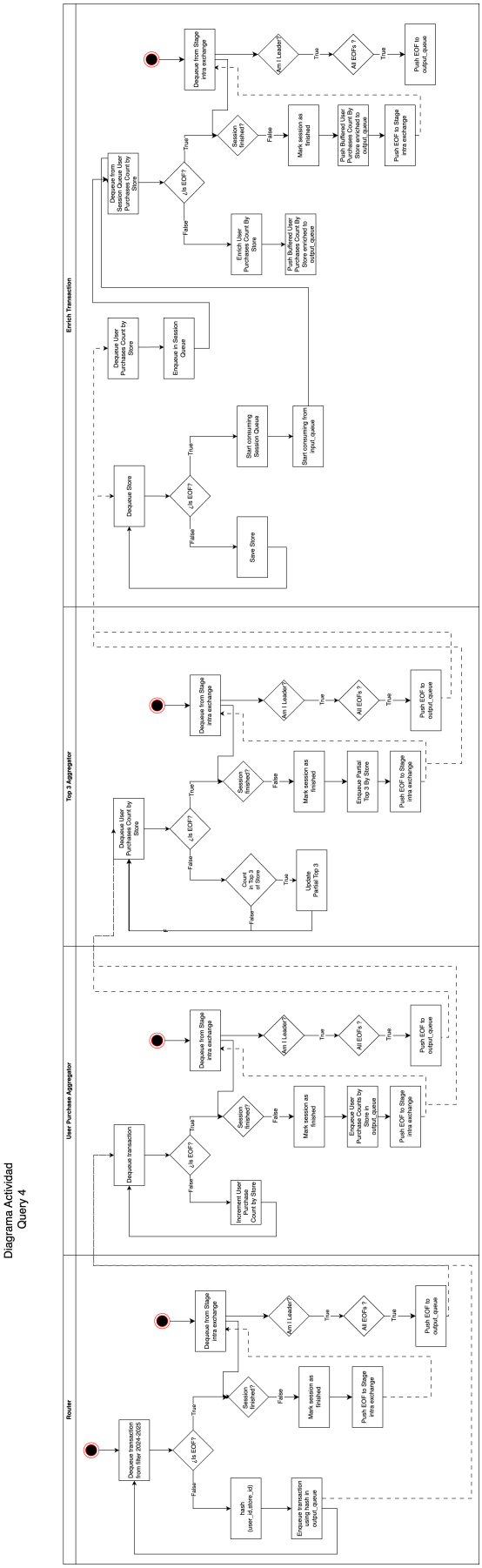

Figura 5: Diagrama de actividad

5. Código

5.1. Diagrama de Paquetes

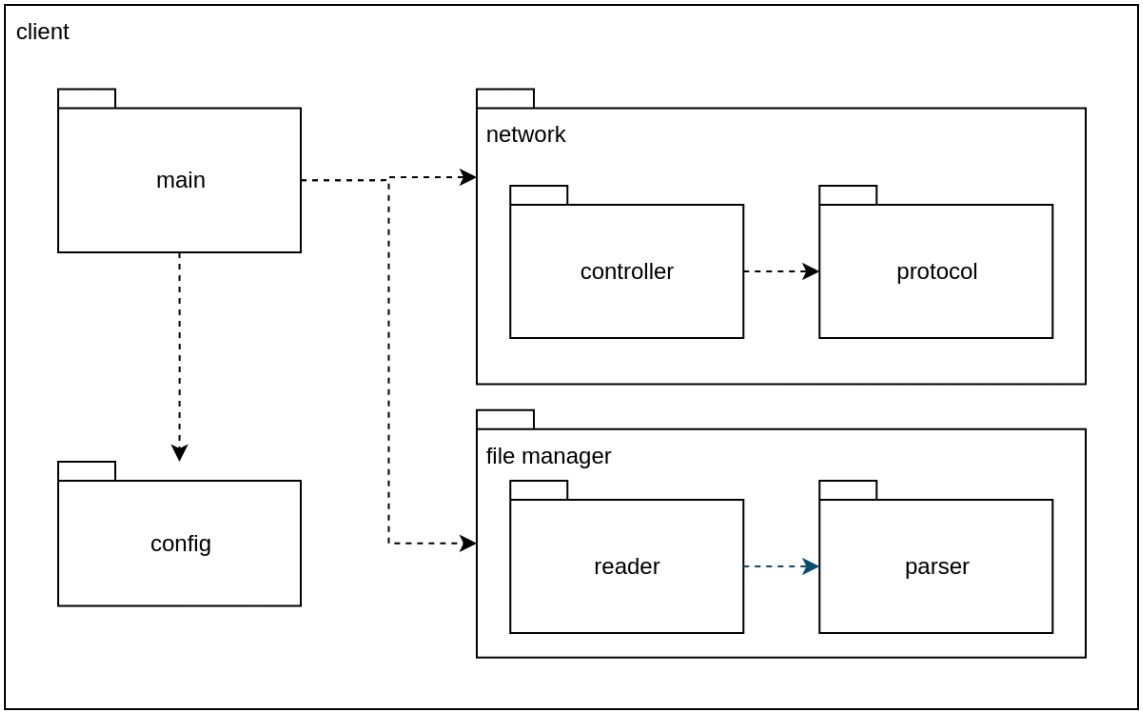

Figura 6: Diagrama de paquetes: Vista Cliente

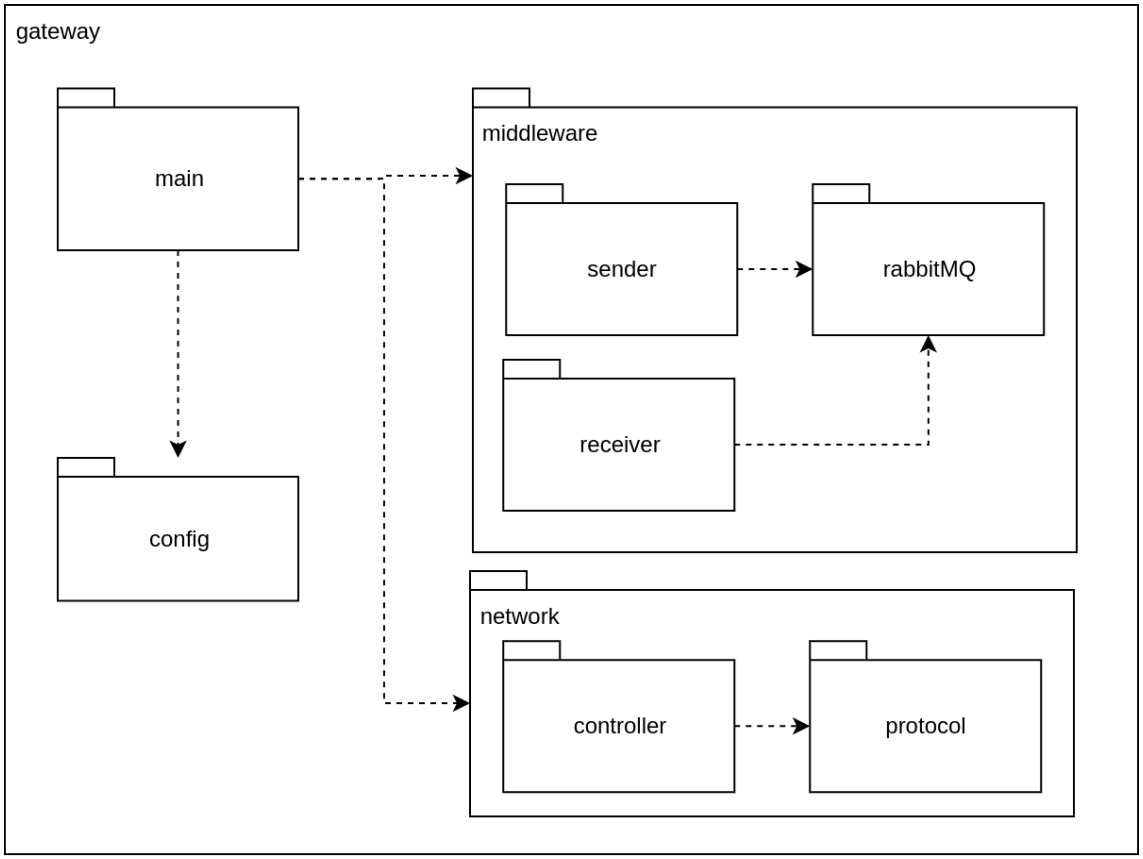

Figura 7: Diagrama de paquetes: Vista Gateway

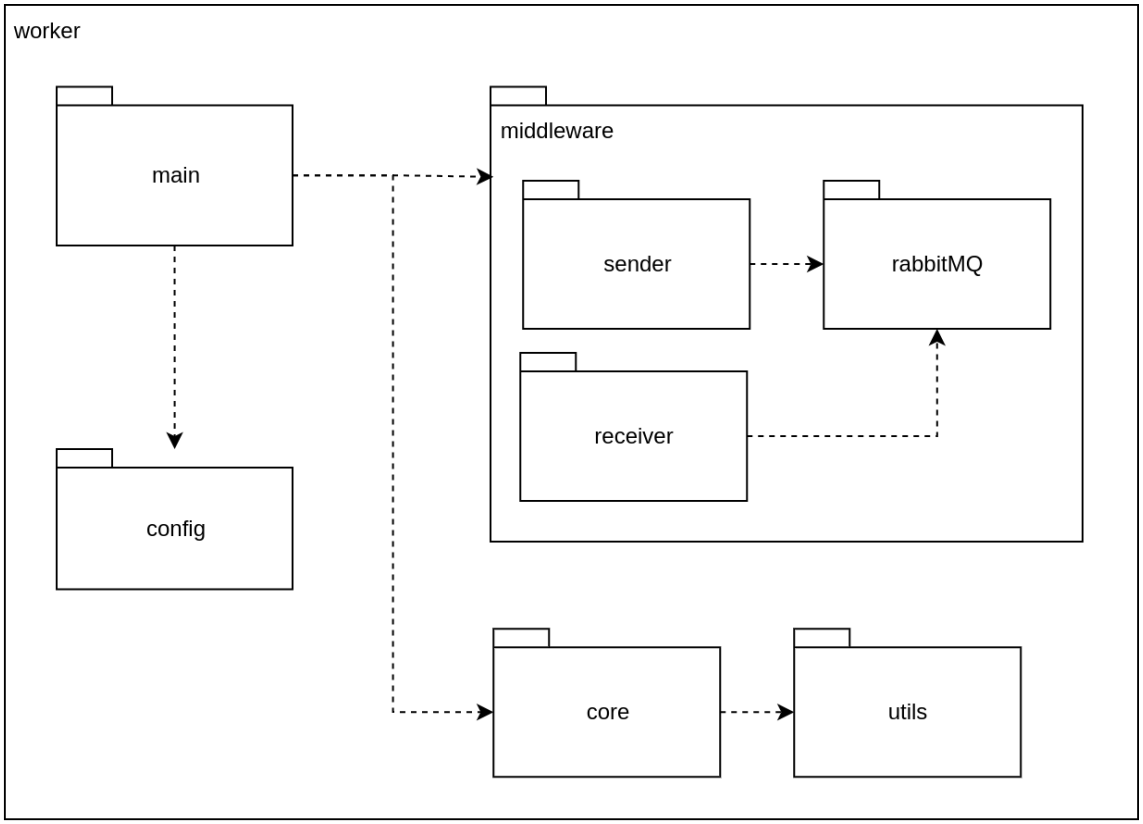

Figura 8: Diagrama de paquetes: Vista Worker

El diagrama muestra agrupación y dependencias entre módulos de cada componente.

Client

- **Main**: usa `Config`, `Network` y `FileManager`.
- **Config**: provee configuración para el funcionamiento de componentes.
- **Network**: gestiona comunicación con *Gateway*; usa `Controller` y `Protocol`.
- **FileManager**: maneja CSV de entrada; submódulos `Reader` y `Parser`.

Gateway

- **Main**: usa `Config`, `Network` y `Middleware`.
- **Config**: configuración de componentes.
- **Network**: comunicación con *Client*; usa `Controller` y `Protocol`.
- **Middleware**: integra RabbitMQ; submódulos `Sender` y `Receiver` para encolar y desencolar mensajes con *Workers*.

Workers

- **Main**: usa `Config`, `Middleware` y `Core`.

- **Config**: configuración de componentes.
- **Middleware**: submódulos `Sender`, `Receiver` y `RabbitMQ`; coordina comunicación con *Gateway* y otros *Workers*.
- **Core**: lógica específica de cada *Worker*; usa `Utils`.
- **Utils**: utilidades para `Core`.

6. División de Tareas

Integrante 1

- **Filters**: *workers* tipo `Filter`.
- **Enrichers**: *workers* tipo `Enricher`.
- **Cliente**: cliente que se comunica con el *Gateway* para solicitud de análisis.

Integrante 2

- **Gateway**: recibe archivos y envía al *pipeline* de análisis.
- **Mergers**: *workers* tipo `Merge`.

Integrante 3

- **Aggregators**: *workers* tipo `Aggregator`.
- **Middleware**: módulos de middleware para *Workers* y *Gateway* (RabbitMQ o ZeroMQ).