Facultad de Ingeniería de la Universidad de Buenos Aires

Sistemas Distribuidos I (TA050)

Trabajo Práctico Nº1

Coffee Shop Analysis - Diseño

Integrantes	Padrón
Landi, Agustina	107850
Solari Vazquez, Federico	106895
Slepowron Majowiecki, María Mercedes	109454

Índice

1. Diseño

1.1. Vista de Escenarios

Casos de Uso

El sistema distribuído propuesto se diseñó con el objetivo de responder 5 consultas sobre un conjunto de datos asociado a ventas en una cadena de negocios de Cafés en Malasia. En la figura a continuación se presenta el caso de uso que modela la resolución de la consulta del cliente. Cada una de las consultas en particular se incluyen dentro del caso de uso general “Realizar consulta sobre datos” porque el sistema procesa y resuelve todas las consultas en conjunto.

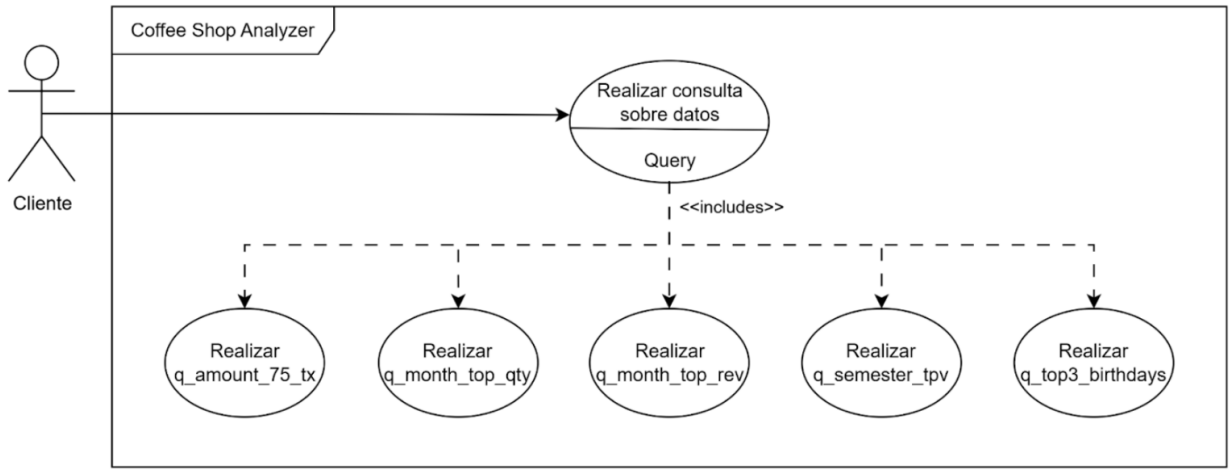


Figura 1: Caso de Uso General.

Realizar consulta q_amount_75_tx

Resumen: El cliente recibe la respuesta sobre las 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.

Casos Borde:

- No hay transacciones que superen ese monto entre esas horas: se devuelve la lista vacía.
- Falta un archivo mensual: se procesan los que están y si no hay ninguno se retorna la lista vacía.
- Si alguna línea está mal formateada (rango horario fuera de 00:00–23:59, o monto negativo, por ejemplo), se ignora esa línea.

Realizar consulta q_month_top_qty

Resumen: El cliente recibe la respuesta sobre el nombre y cantidad de los productos más vendidos para cada mes en 2024 y 2025.

Casos Borde:

- No hay transacciones registradas para un par año-mes: se devuelve la lista vacía.
- Si dos o más productos tienen la misma cantidad de ventas ganancia máxima el sistema retorna uno solo; el que alfabéticamente, según su nombre, sea menor.
- Si no se encuentra el nombre de producto en según su id de item, se lo descarta.
- Si falta un archivo mensual: Se procesan los que están y si no hay ninguno se retorna la lista vacía.
- Si alguna línea está mal formateada (id de item inválido, fecha mal formateada por ejemplo), se ignora esa línea.

Realizar consulta q_month_top_rev

Resumen: El cliente recibe la respuesta sobre el nombre y cantidad de los productos que más ganancias han generado para cada mes en 2024 y 2025.

Casos Borde:

- No hay transacciones registradas para un par año-mes: se devuelve la lista vacía.
- Si dos o más productos tienen la misma ganancia máxima el sistema retorna uno solo; el que alfabéticamente, según su nombre, sea menor.
- Si no se encuentra el nombre de producto en según su id de item, se lo descarta.
- Si falta un archivo mensual: Se procesan los que están y si no hay ninguno se retorna la lista vacía.
- Si alguna línea está mal formateada (id de item inválido, fecha mal formateada por ejemplo), se ignora esa línea.

Realizar consulta q_semester_tpv

Resumen: El cliente recibe la respuesta sobre el 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.

Casos Borde:

- No hay transacciones registradas entre las 6:00 y las 23:00hrs: se devuelve la lista vacía.

- Falta un archivo mensual: Se procesan los que están y si no hay ninguno se retorna la lista vacía.
- Si una línea tiene formato no esperado (fecha inválida, monto no numérico), se ignora esa línea.
- Si el monto final de una transacción es un valor nulo o negativo se ignora esa fila.

Realizar consulta q_top3_birthdays

Resumen: El cliente recibe respuesta sobre la fecha de cumpleaños de los 3 clientes que han hecho más compras durante 2024 y 2025, para cada sucursal.

Casos Borde:

- Si una sucursal tiene menos de 3 clientes con compras se devuelve el top N de clientes, siendo N menor a 3 y mayor a 0.
- Si no hay transacciones para esa sucursal, se ignora la sucursal.
- Si el dato de user_id es nulo, esa fila se ignora.
- Si alguna línea del dataset contiene un formato inesperado, la línea se ignora.

1.2. Flujo de Procesamiento de Datos

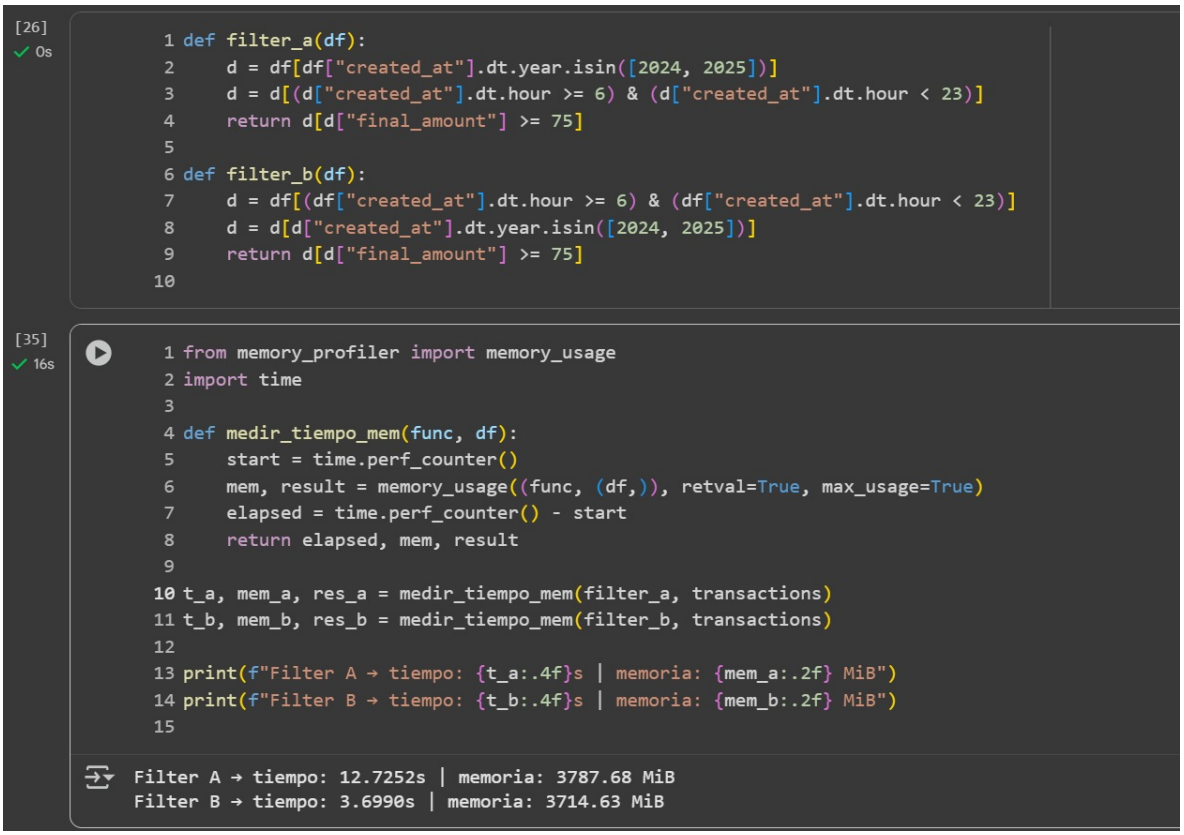
A continuación, se modela en un **DAG (Directed Acyclic Graph)** el flujo de procesamiento de datos enviados por el cliente. Estas figuras detallan todas las etapas necesarias para transformar la información provista por el usuario, en formato de archivos csv, en los resultados consolidados de las consultas del usuario.

El flujo comienza con la ingestión de los siguientes archivos: transactions.csv, transaction_items.csv, menu_items.csv, stores_csv y users.csv, provistos por el usuario. Cada nodo del DAG representa una acción específica (por ejemplo, Filter, Join, Map, Reduce), que se aplica sobre los datos, mientras que las aristas definen la secuencia de ejecución y las dependencias entre tareas.

Consideraciones que tuvimos en cuenta al diagramar las consultas:

- Como primera etapa de procesamiento para cada archivo se encuentra FILTER, que es el responsable de quitar columnas no deseadas y filas con valores faltantes. Asi como también el casteo de las variables para nuestra convenciencia (como por ejemplo los id a Int64 y las fechas a Datetime).
- Para las queries 1, 3 y 4, analizamos las queries con Google Colab y comparamos tiempos de ejecución y uso de memoria. Probamos dos estrategias: filtrar primero

por hora y despues por año y filtrar primero por año y después por hora. Resultó ser más eficiente filtrar primero por hora y después por año.



```
[26]
✓ 0s
```

```python
def filter_a(df):
    d = df[df["created_at"].dt.year.isin([2024, 2025])]
    d = d[(d["created_at"].dt.hour >= 6) & (d["created_at"].dt.hour < 23)]
    return d[d["final_amount"] >= 75]

def filter_b(df):
    d = df[(df["created_at"].dt.hour >= 6) & (df["created_at"].dt.hour < 23)]
    d = d[d["created_at"].dt.year.isin([2024, 2025])]
    return d[d["final_amount"] >= 75]

```

```
[35]
✓ 16s
```

```python
from memory_profiler import memory_usage
import time

def medir_tiempo_mem(func, df):
    start = time.perf_counter()
    mem, result = memory_usage((func, (df,)), retval=True, max_usage=True)
    elapsed = time.perf_counter() - start
    return elapsed, mem, result

t_a, mem_a, res_a = medir_tiempo_mem(filter_a, transactions)
t_b, mem_b, res_b = medir_tiempo_mem(filter_b, transactions)

print(f"Filter A → tiempo: {t_a:.4f}s | memoria: {mem_a:.2f} MiB")
print(f"Filter B → tiempo: {t_b:.4f}s | memoria: {mem_b:.2f} MiB")

```

```
Filter A → tiempo: 12.7252s | memoria: 3787.68 MiB
Filter B → tiempo: 3.6990s | memoria: 3714.63 MiB
```

Figura 2: Análisis orden de filtrado por tiempo de ejecución y memoria.

Sin embargo, como una de las queries no tiene como limitacion el horario, filtrar primero por hora resultaría en pérdida de datos para esa query. Por lo tanto, decidimos dejar el filtrado de años primero, de modo que ese FILTER pueda reutilizarse.

- Para la query 4, decidimos dejar el JOIN como última operación ya que tomaría menos datos, debido a que los datos que no figuran en el top 3 de compradores por tienda, ya fueron descartados anteriormente.

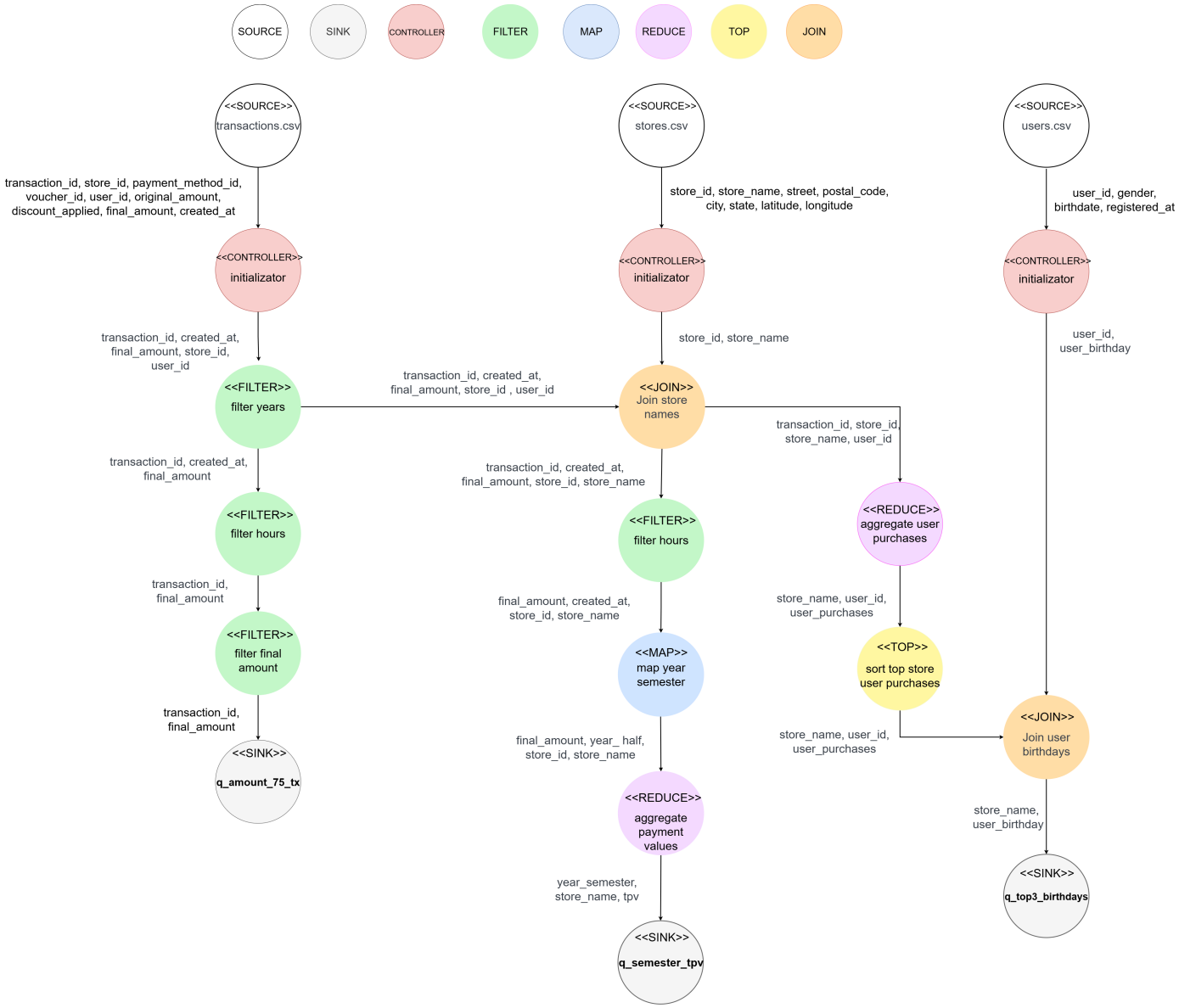


Figura 3: DAG para queries 1, 3 y 4.

Figura 4: DAG para query 2.

1.3. Vista de Procesos

Diagrama de Secuencia

Un diagrama de secuencia describe las interacciones entre los distintos componentes

de un sistema, permitiendo visualizar el orden y flujo de los mensajes entre las entidades involucradas.

Decidimos presentar tres fases del proceso de comunicación entre el cliente y el sistema. El primero de ellos muestra el proceso de conexión entre las entidades. El segundo diagrama describe el flujo de ejecución del programa una vez que la conexión ha sido establecida. Finalmente, el tercer diagrama muestra el proceso de cierre de la conexión.

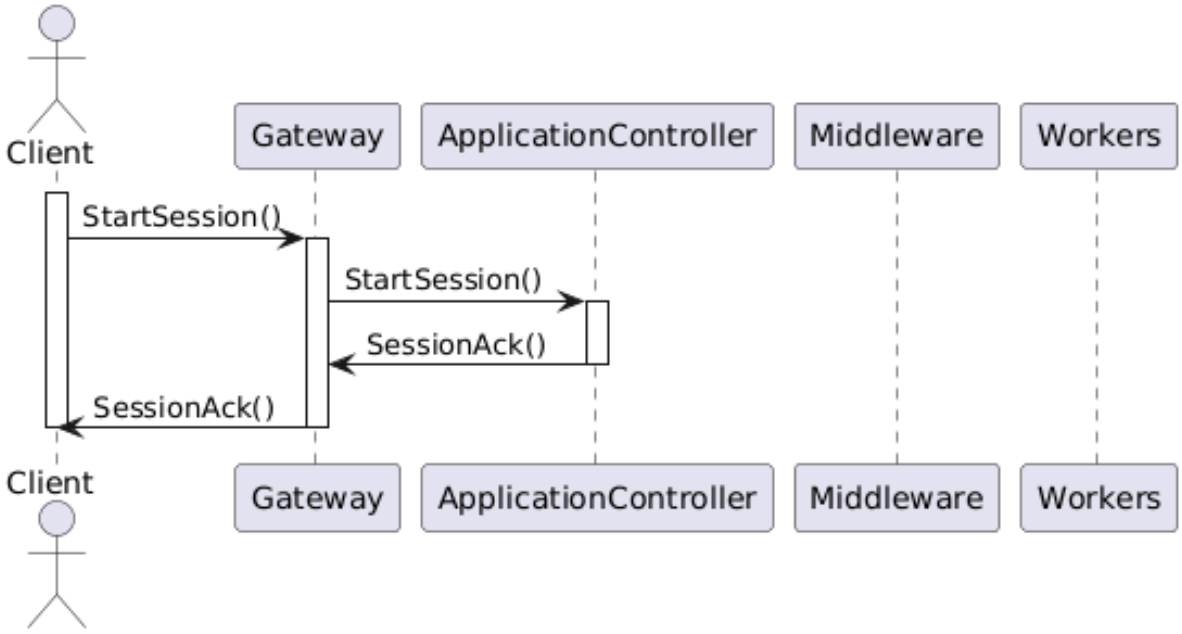


Figura 5: Establecimiento de conexión entre el cliente y el sistema.

Una vez establecida la conexión entre el cliente y el sistema, dado que el tamaño de los archivos del cliente no se conoce de antemano, éstos son enviados en batches.

Se inicia entonces un loop en el que el cliente envía estos batches al Gateway, quien los reenvía al ApplicationController. Este último se encarga de inicializar los datos, eliminando columnas innecesarias, manejando valores nulos y ajustando los tipos de datos según sea necesario. Una vez procesados, los batches inicializados son enviados al Middleware, que los distribuye a los Workers. Cada Worker procesa los datos de manera secuencial, realizando las operaciones esperadas para cada query.

Una vez que el cliente envia su batch final, envía un mensaje de .endOfFile"para indicar que no quedan mas datos restantes. Este mensaje viaja desde el cliente hasta los Workers, pasando por el Gateway, ApplicationController y Middleware. Los Workers procesan el final del archivo y devuelven los resultados parciales al Middleware, que los reenvía al ApplicationController. Finalmente, el ApplicationController genera las respuestas finales y se envían al cliente, cerrando el ciclo de procesamiento de datos.

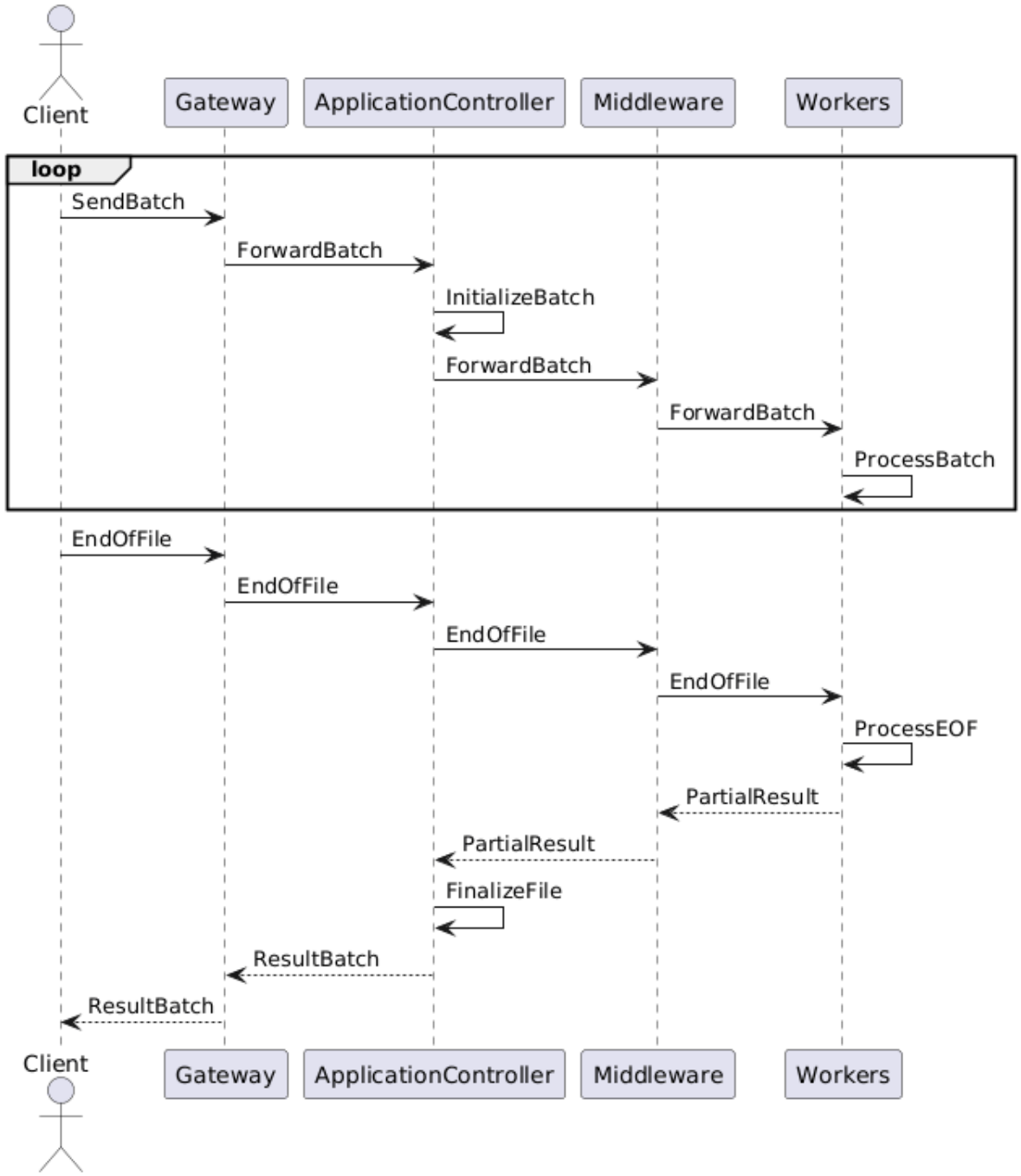


Figura 6: Flujo principal del sistema.

Una vez que todos los batches fueron procesados y el cliente recibió sus resultados, éste comienza el cierre de conexión.

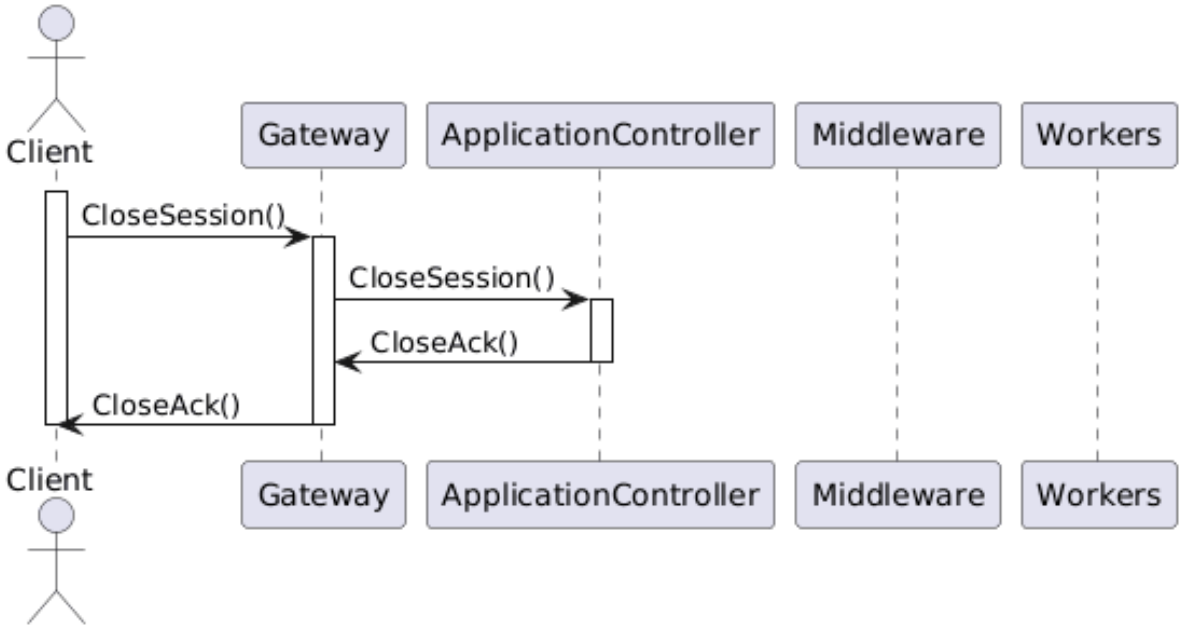


Figura 7: Cierre de conexión entre el cliente y el sistema.

Diagrama de Actividades

En este diagrama observamos la representacion de un flujo de actividades desde el cliente hasta los workers. El proceso comienza con la sincronización entre el Cliente y el Application controller. Una vez establecida la conexión, el Cliente procede a leer y enviar los datos en batches, es decir, fragmentos de información que se transmiten de manera secuencial para no saturar el sistema.

El Application controller recibe cada batch y verifica si se alcanzó el fin del archivo (EOF). Si no es el caso, reenvía los batches hacia el Middleware, encargado de distribuirlos a los Workers.

Los Workers procesan cada batch de manera independiente y paralela, aplicando las operaciones definidas en la consulta. Al finalizar, generan un Result_Batch que es enviado nuevamente al Middleware.

El Middleware recibe esos resultados parciales y los reagrupa. Si todavía quedan más resultados pendientes, continúa consolidando; en caso contrario, envía los resultados al Application controller. El Application controller gestiona la recolección de esos resultados parciales y, una vez que tiene suficientes datos o se llega al EOF, comienza a enviar los resultados al Cliente. Mientras tanto, el Cliente permanece en espera activa, recibiendo y ensamblando los Result_Batches para reconstruir la respuesta final.

El ciclo de envío y recepción continúa hasta que ya no quedan más batches. En ese punto, el Application controller notifica la finalización del proceso y devuelve al Cliente todos los resultados consolidados.

Finalmente, el Cliente ensambla la respuesta completa y el flujo de actividades termina, liberando recursos y dejando al sistema listo para procesar nuevas solicitudes.

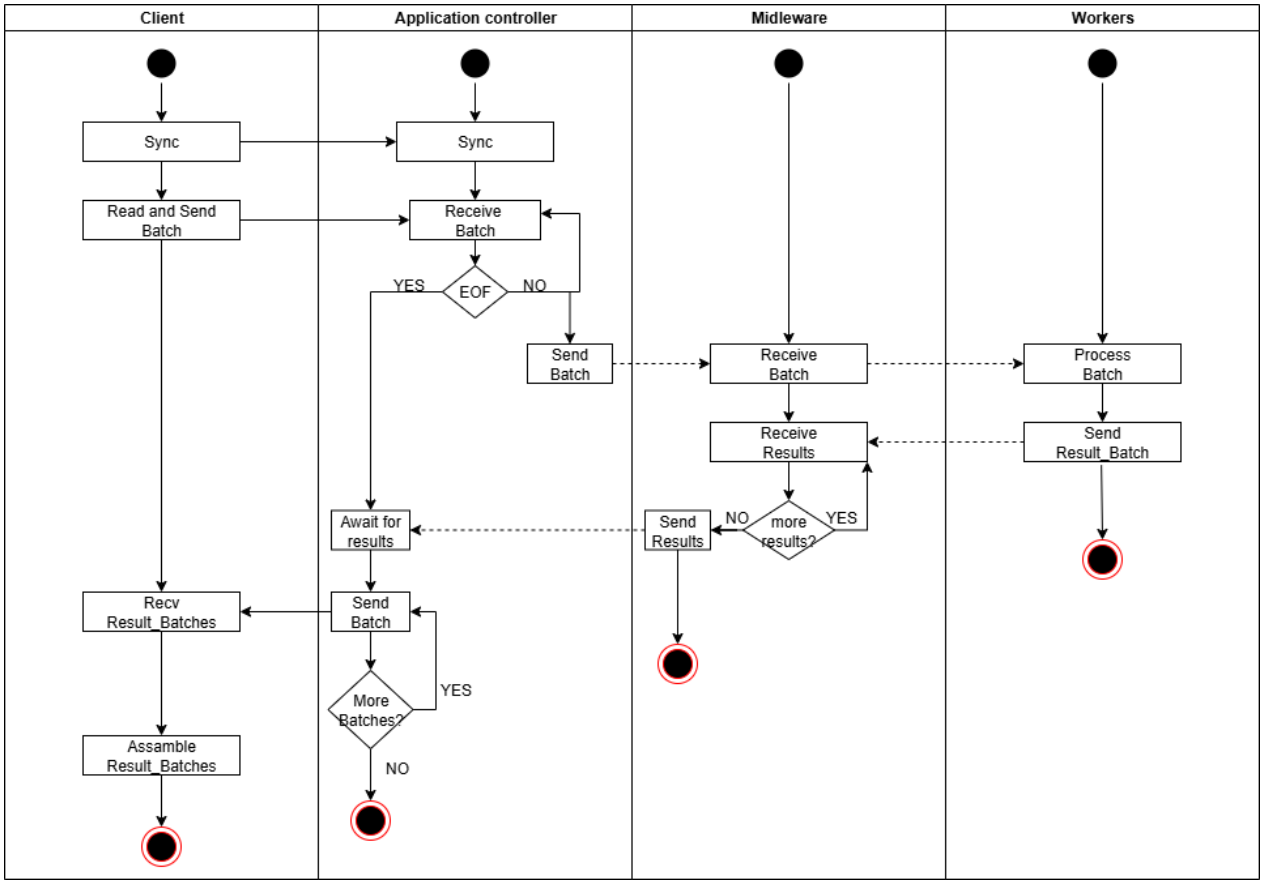


Figura 8: Diagrama de Actividades.

1.4. Vista de Desarrollo

Diagrama de Paquetes

Se organizó el sistema en múltiples paquetes, cada uno con una responsabilidad específicas para que el sistema pueda llevar a cabo las tareas de procesamiento y análisis de las consultas de usuario.

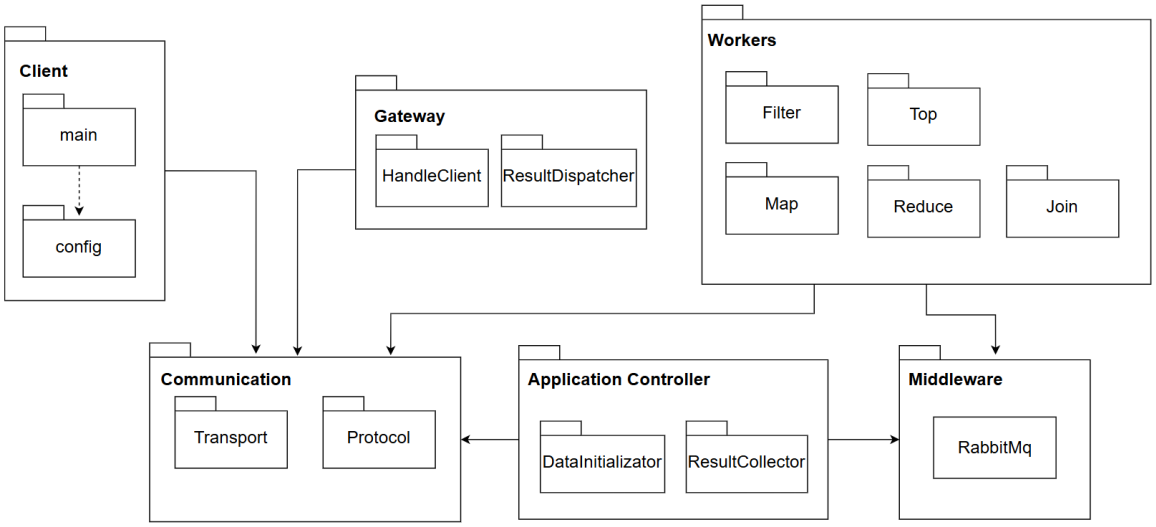


Figura 9: Diagrama de Paquetes.

Los nodos detallados en la figura, cumplen con las siguientes funciones y características:

- Client: Representa al usuario que utilizará el sistema. Contiene los componentes necesarios para que un cliente pueda realizar sus consultas. Inicia su ejecución desde el módulo main, cargando su configuración inicial del subpaquete config. Desde config, se pueden definir los lags de las queries que el cliente consultara, el tamaño de batch de envío y tecepción de datos y endpoints, por ejemplo. A su vez, el cliente cuenta con una librería de comunicación que utilizara para conectarse al gateway del sistema. Desde DataSender se realiza el streaming de archivos (divididos en chuncks), y luego con ResultsReciever se escuchan por resultados relacionados a las consultas realizadas.

- Gateway: Este es el punto de entrada principal de los usuarios al sistema. A través del gateway, se establecen las comunicaciones con usuarios. Se encarga de autenticas, enrutar y devolver resultados al cliente. El modulo Router permite balancear contra instancias del Application Controller que tomarán los datos del cliente. Por otro lado, el ResultDispatcher se encarga de tomar las respuestas del sistema y enviarlas al ResultReceiver del cliente.

- Communication: En este módulo se abstrae la lógica de transporte para el envío y recepción de mensajes, y el protocolo utilizado, donde se especifíca el formato y tamaño previamente acordado por las partes para el envío de la información.

- Application Controller: El gateway se comunica con este nodo, el cual se encarga de limpiar y formatear la información que se recibió del cliente, para realizar las queries correctamente. Una vez inicializada la data, esta es reenviada al middleware para que los workers conectados a este realicen el procesamiento de la información. El

Application Controller es también quien recibe los datos ya procesados de los workers y genera las respuestas finales que se reenviaran al cliente como resultados. Las respuestas son reenviadas al ResultDispatcher del Gateway.

- Middleware: En este módulo se abstrae la lógica del middleware de RabbitMQ. Gestiona la distribución de mensajes para desacoplar la comunicación entre módulos. Se encarga de enroutar la información ya inicializada proveniente del Application Controller hacia las colas de trabajo donde hay workers suscritos. Los resultados se publican en un exchange de resultados y son recolectados para formar la respuesta final al cliente.
- Workers: Son los responsables de procesar y resolver las consultas del cliente.
    - Filter: Paquete que engloba los workers encargados de filtrar información en los datos del cliente. Se tienen 3 filtros específicos para resolver las queries que alcanzan al sistema. Estos son: Un filtro para las transacciones realizadas entre 2024 y 2025, un filtro para las transacciones realizadas entre las 6:00 y las 23:00 hrs, y un filtro para las transacciones cuyo monto es mayor o igual a 75
    - Map: este es un worker de transformación por registro cuya funcionalidad es derivar campos auxiliares para luego poder aplicar operaciones de reduce y top. Se encarga de mapear registros de transacciones según su año y mes, y también mapear los semetres del año en los que se realizaron las transacciones.
    - Reduce: Se encarga de agrupar entradas realizando dos tipos de operaciones de agregación, count y sum. Es necesario para agregar la cantidad de transacciones de cada local, el revenue generado por cada producto para los locales, la suma del monto total acumulado por cada local y la cantidad de comprar realizadas por los usuarios en cada local.
    - Join: Se encarga de unir los resultados de dos tablas diferentes según una key determinada. Para la resolución de las queries del sistema, se necesita un join que unifique la tabla de menu, con las transacciones para obtener el nombre del item que se compró en un cada transaccón, y un join que unifique el nombre del local donde se realizaron determinadas transacciones.
    - Top: Se encarga de ordenar resultados y obtener un top N de valores para resolver las queries de q_month_top_qty, q_month_top_rev y q_top3_birthdays. Para estos 3 casos, necesitamos buscar, el item de menú más vendido en cada mes para cada local, el item de menú que más gananancias le generó a cada local y los 3 clientes que más compras hayan hecho en cada local.

1.5. Vista Física

Diagrama de Robustez

Se presentan los diagramas de robustez que modelan el funcionamiento de un sistema distribuido orientado al procesamiento de consultas. Para ello, se elaboró en primer lugar un diagrama general, donde se ilustran todos los workers y componentes principales con nombres genéricos, y posteriormente se construyeron los diagramas específicos para cada query, en los cuales se refleja la secuencialidad de las operaciones y el uso particular de los workers.

Los elementos principales son:

- Cliente: representa al usuario que interactúa con el sistema, enviando lotes (batches) de datos para su procesamiento.
- Gateway: se encarga de redirigir los datos y conexiones de un cliente hacia uno de los servidores activos disponibles.
- Application Controller: recibe los batches de datos provenientes del cliente, los envía al middleware para su procesamiento y finalmente retorna el resultado obtenido.
- Colas (Q): funcionan como puntos de intercambio asincrónico donde se encolan las tareas antes de ser procesadas por los distintos controllers.
- Controllers: son responsables de consumir tareas desde sus respectivas colas y procesar los datos de manera aislada. Cada uno cumple una función específica (filter, map, reduce, joins o top), y en algunos casos cuentan con almacenamiento propio para mantener contexto adicional de ejecuciones previas.
- Resultados (Results Q): cola final desde donde el sistema obtiene y devuelve al cliente los resultados procesados.

En los diagramas correspondientes a cada query se observa cómo los datos recorren diferentes etapas de procesamiento en función de los requisitos particulares de la consulta. De esta forma, se logra un diseño modular y distribuido, que facilita la escalabilidad y el manejo concurrente de múltiples peticiones.

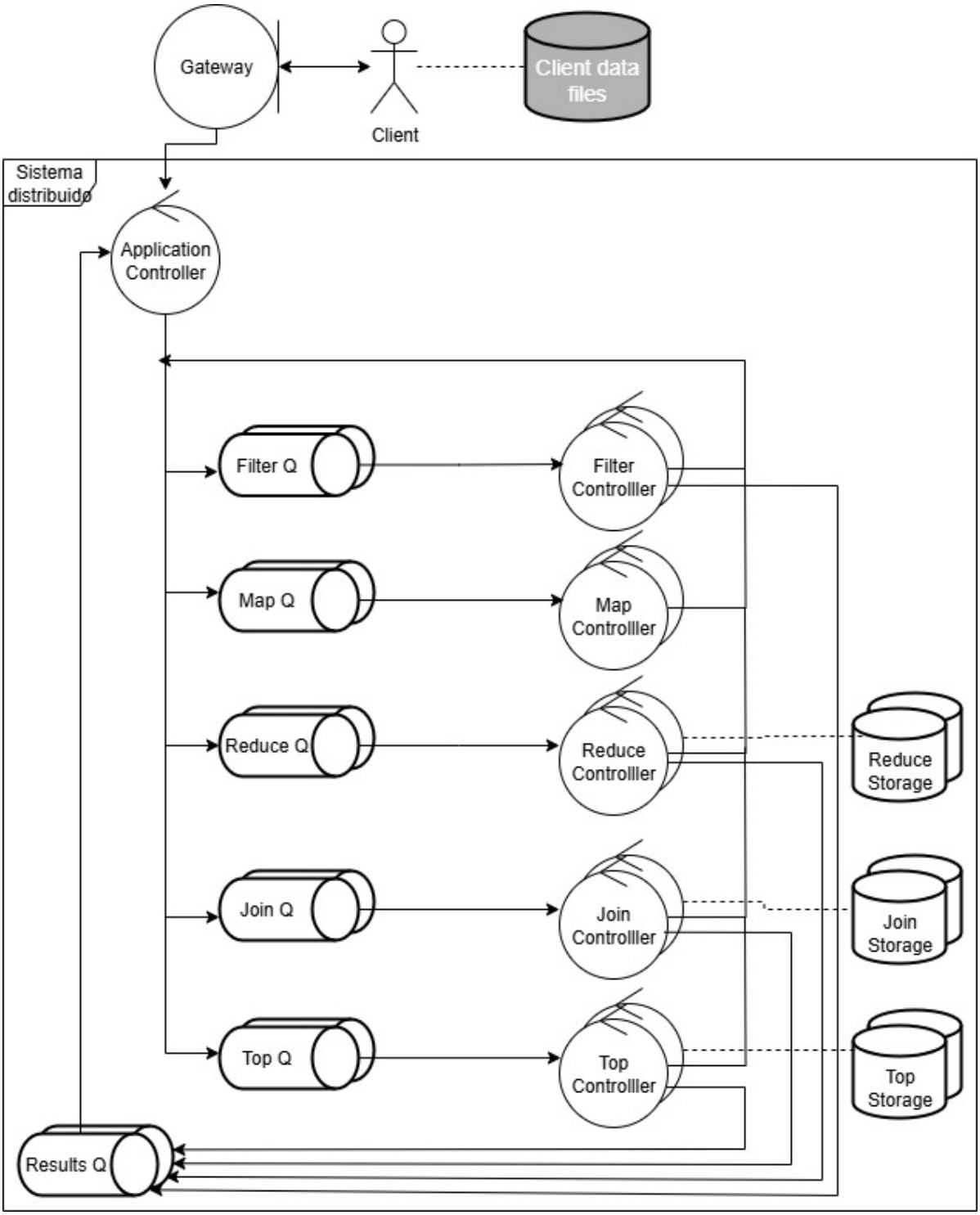


Figura 10: Diagrama de Robustez general.

En el diagrama de la Query 1 se observa un flujo orientado principalmente a la aplicación de filtros. El cliente envía los lotes de datos al Gateway, que los redirige al Application Controller. A partir de allí, los datos se encolan y son procesados de forma secuencial por

diferentes workers: el Inicializador de Filtros, el Filtro de Años y el Filtro de Horas. Cada uno de estos componentes va depurando la información según los parámetros establecidos. Finalmente, los resultados obtenidos se depositan en la cola de resultados (Results Q), desde donde se devuelven al cliente. Este query muestra un esquema sencillo y lineal, enfocado únicamente en operaciones de filtrado.

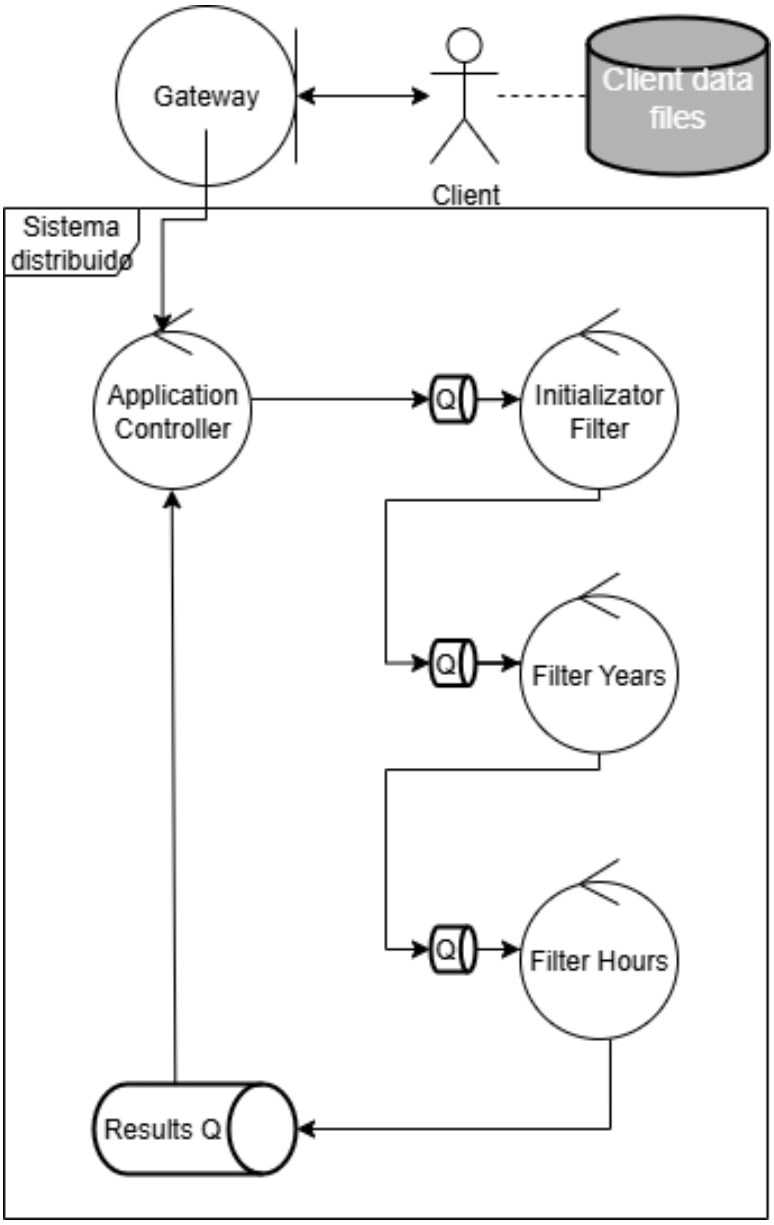


Figura 11: Diagrama de Robustez Query 1.

En la Query 2 el flujo se complejiza respecto de la anterior, ya que combina filtros, uniones y operaciones de agregación. Luego de la etapa inicial de filtros (Inicializador y Años), los datos pasan por procesos de unión con los nombres de los ítems del menú, seguido de un mapeo por año y mes. Posteriormente, se aplican reducciones para obtener

la cantidad agregada de transacciones y la suma de subtotales. Además, se incluyen procesos de ordenamiento que permiten identificar los ítems más vendidos y las tiendas con mayor generación de ingresos. Todo este procesamiento culmina con la recolección de resultados en la cola final, listos para ser consumidos por el cliente.

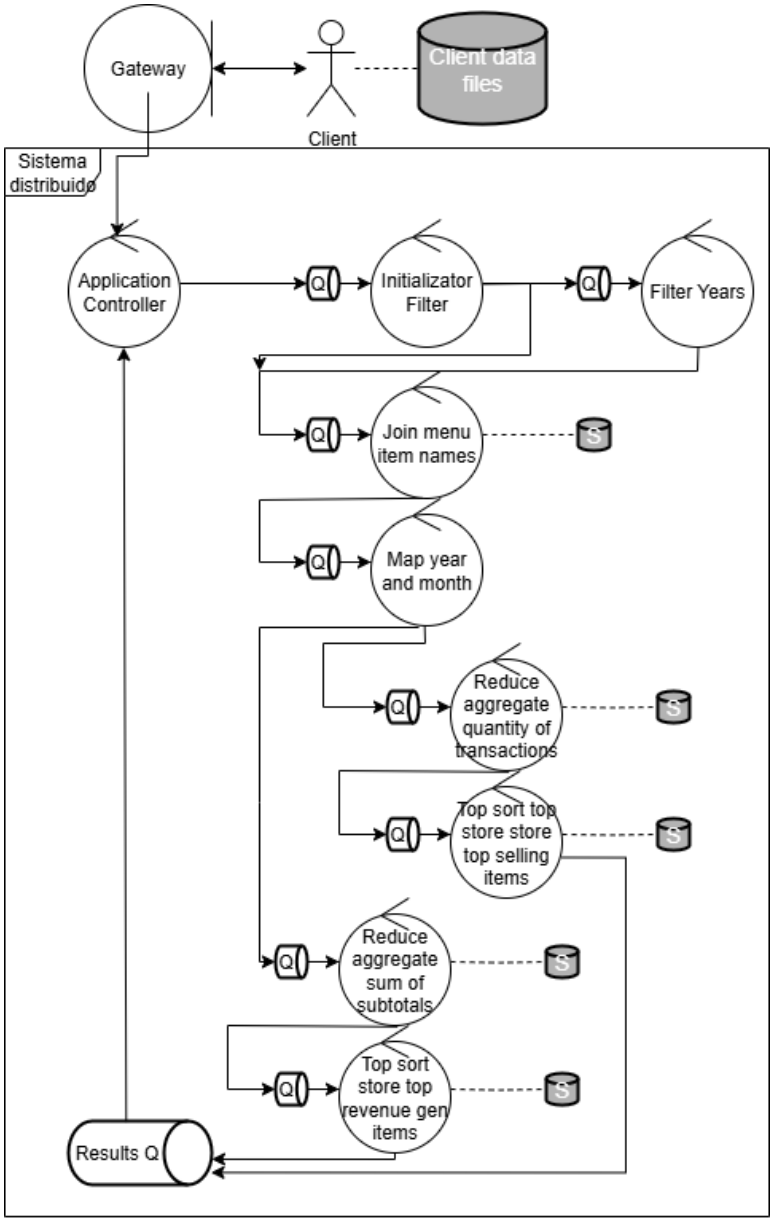


Figura 12: Diagrama de Robustez Query 2.

La Query 3 incorpora una combinación de filtros, uniones y reducciones aplicadas a los pagos. Tras pasar por los filtros iniciales y de años, los datos son procesados en paralelo en varias etapas: unión de nombres de tiendas, filtrado por horas, mapeo por semestre y reducción de valores de pago agregados. Cada una de estas transformaciones contribuye a organizar los datos según diferentes perspectivas temporales y comercia-

les. Los resultados parciales se integran en la cola final, ofreciendo al cliente una vista consolidada y ordenada de la información solicitada.

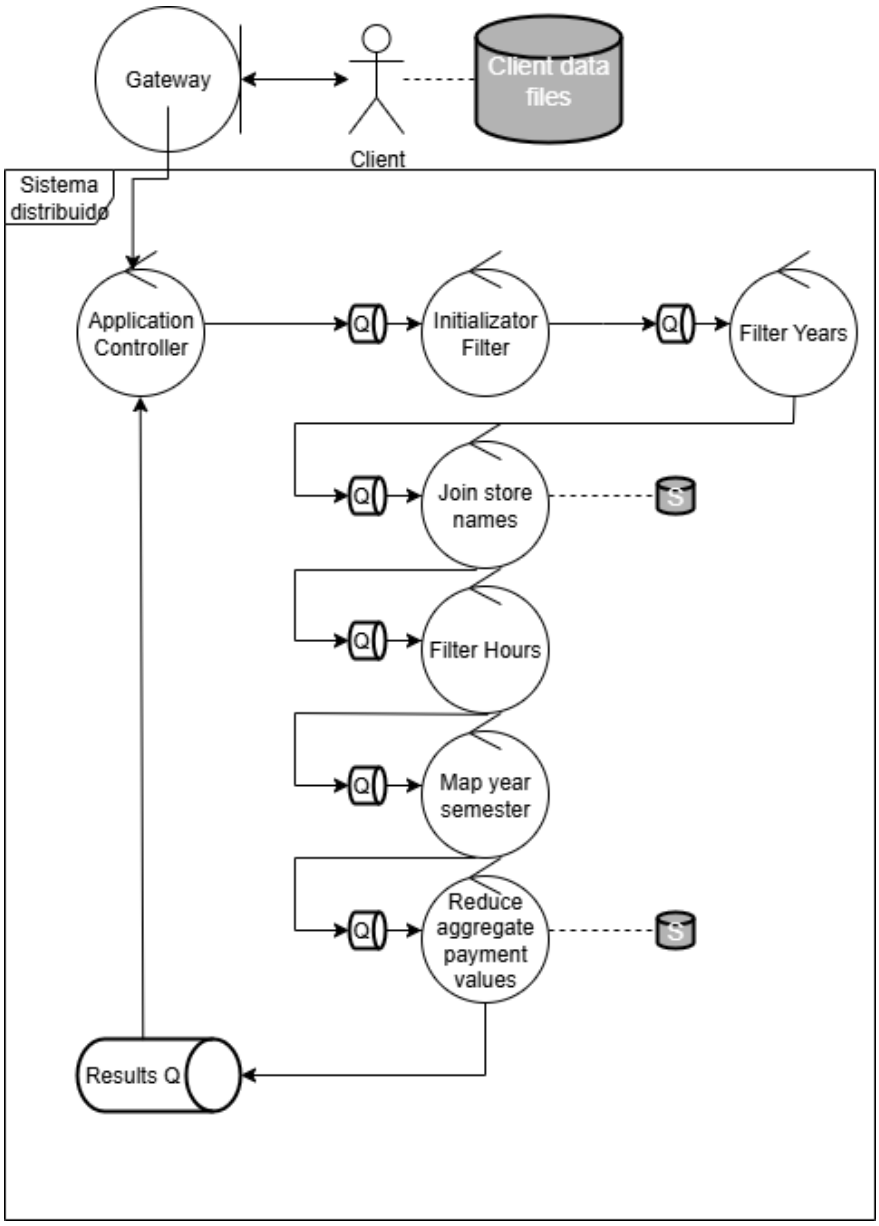


Figura 13: Diagrama de Robustez Query 3.

La Query 4 está orientada a consultas sobre el comportamiento de los usuarios. Luego del filtrado inicial y por años, los datos son sometidos a uniones con nombres de tiendas, reducciones que agregan compras de usuarios y un proceso de ordenamiento para identificar a los clientes con mayor volumen de compras. Finalmente, se incorpora información adicional mediante la unión con las fechas de cumpleaños de los usuarios. Este flujo permite obtener resultados enriquecidos que combinan hábitos de compra y datos demográficos, los cuales se devuelven a través de la cola de resultados.

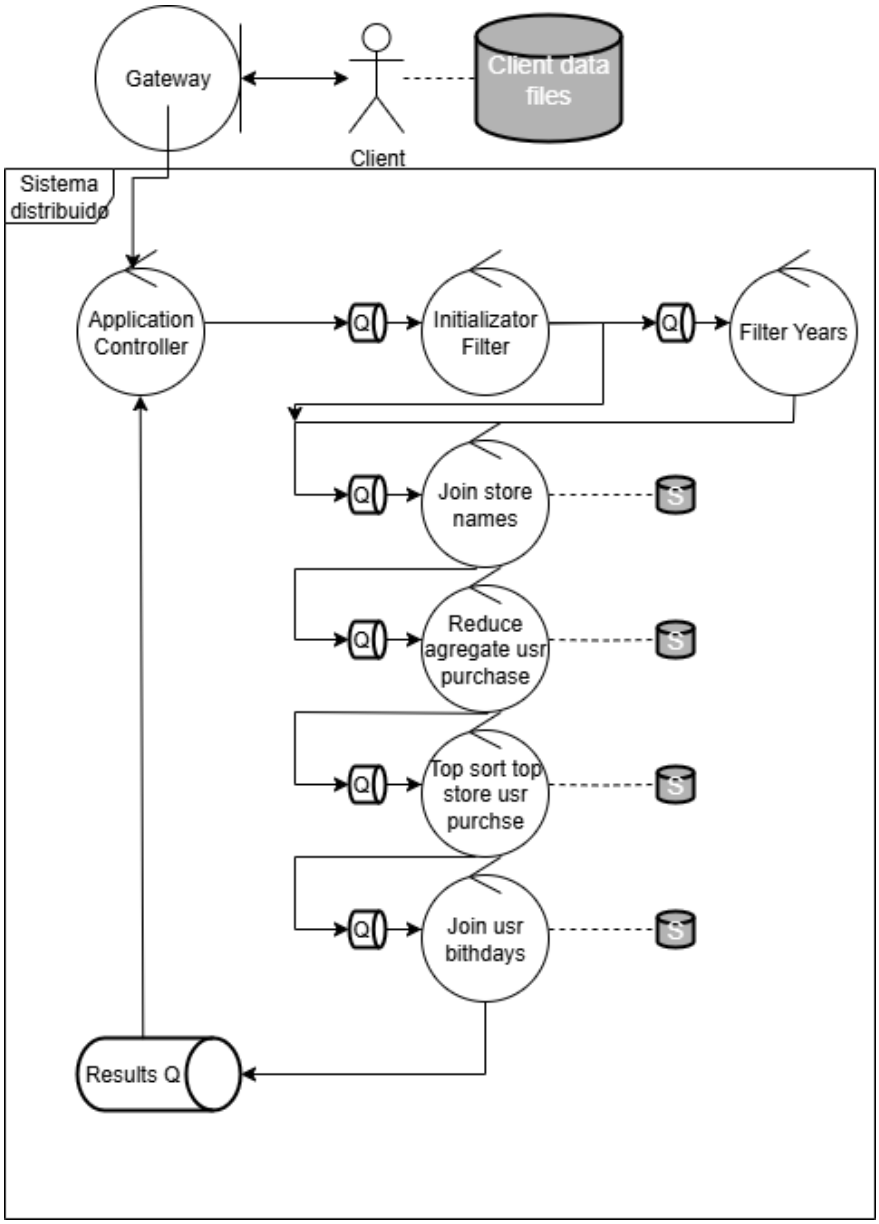


Figura 14: Diagrama de Robustez Query 4.

Diagrama de Despliegue

El diagrama de despliegue muestra la distribución física de los componentes de un sistema en un entorno de ejecución. Describe cómo se distribuyen los nodos y cómo se conectan entre sí.

Para nuestro sistema, representamos los siguientes nodos:

- Client: Envía mediante ClientProtocolLibrary los datasets al ApplicationController.
- Gateway: Gestiona la conexión de los clientes al sistema.

- ApplicationController: Mappea los datasets recibidos y los envía mediante RabbitMQ a los workers correspondientes.
- Middleware: Instancia de RabbitMQ
- Workers: Cada worker realiza una acción correspondiente para procesar la request y luego enviarla. Esta acción puede ser MAP, FILTER, TOP, REDUCE o JOIN.

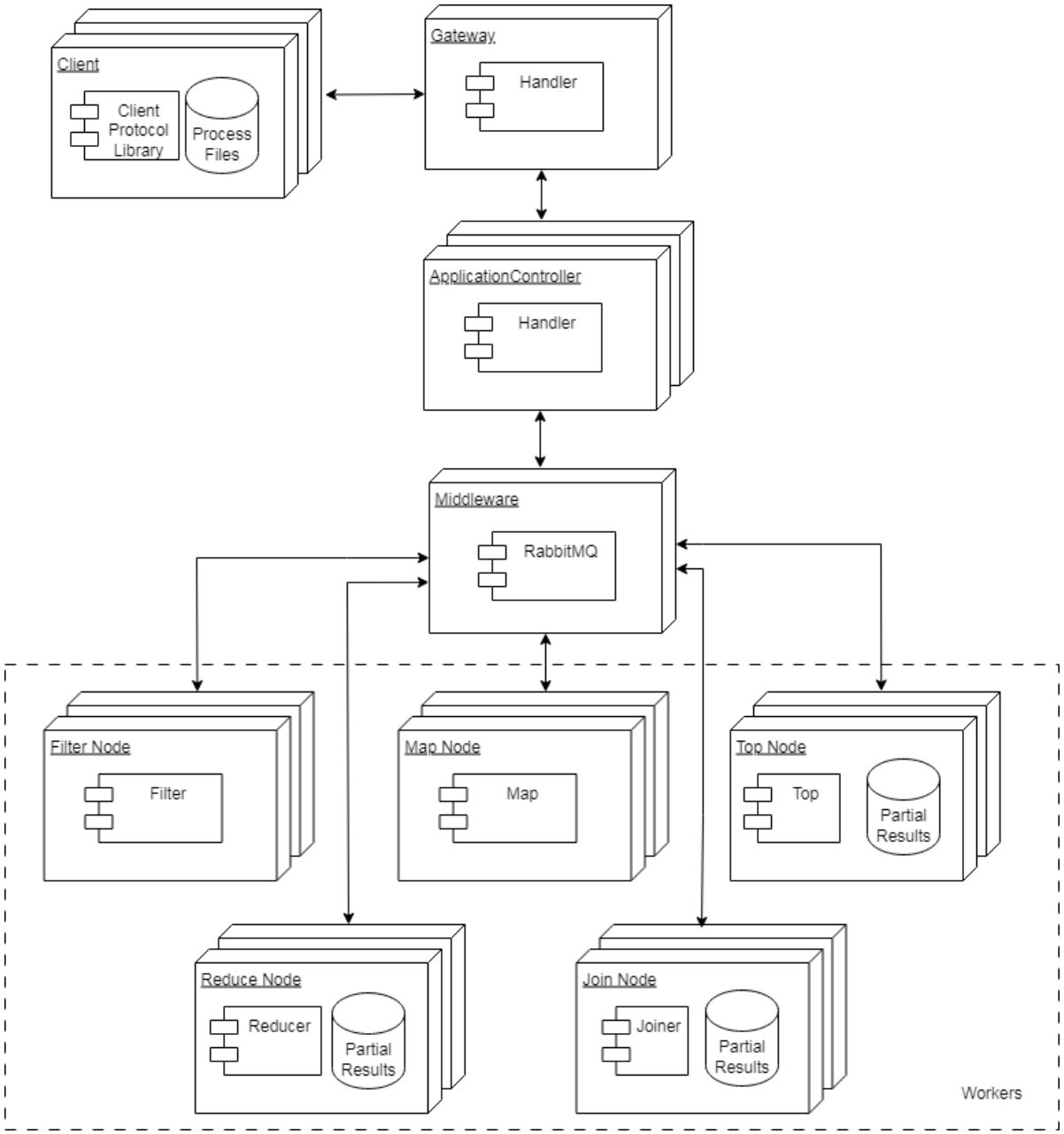


Figura 15: Diagrama de Despliegue.