.UBAfiuba
FACULTAD DE INGENIERÍA

Sistemas Distribuidos 1 TP N° 1 - reentrega: Bike Rides Analyzer

1° Cuatrimestre, 2023

Apellido y Nombre	Email
Sabatino, Gonzalo	gsabatino@fi.uba.ar

Índice

1. Scope del Trabajo

Se solicita un sistema distribuido que analice los registros de viajes realizados con bicicletas de la red pública provista por grandes ciudades. Los registros se ingresan progresivamente, al recibirse cada ciudad.

Se debe obtener:

- La duración promedio de viajes que iniciaron en días con precipitaciones >30mm.
- Los nombres de estaciones que al menos duplicaron la cantidad de viajes iniciados en ellas entre 2016 y el 2017.
- Los nombres de estaciones de Montreal para la que el promedio de los ciclistas recorren más de 6km en llegar a ellas.

2. Flujo de datos

En la Figura 1 se muestra el DAG propuesto para el sistema, mostrando cómo es el flujo de los tres distintos tipos de datos (station, weather, trip) desde que arriban hasta que, según sean o no filtrados previamente, llegan al resultado de cada query.

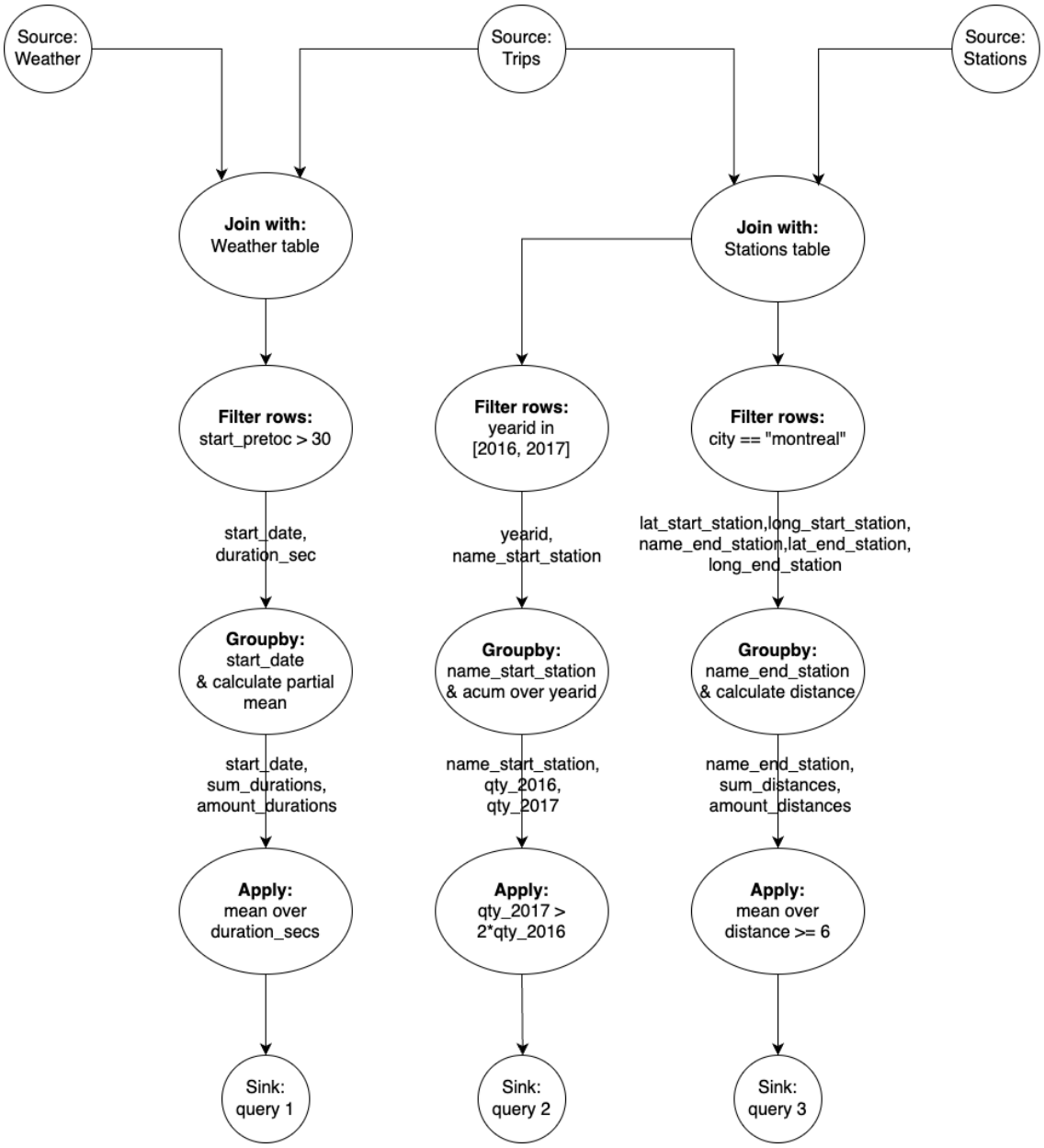

Figura 1: DAG mostrando el flujo de datos

A partir de entender las tres fuentes de datos, se propone el siguiente esquema de pasaje de datos:

1. Envío de datos estáticos (stations y weather) para las tres ciudades, los cuales serán almacenados en memoria en el nodo apropiado.
2. Envío de los trips para las tres ciudades.
3. Consulta por parte del cliente por la obtención de los resultados.

3. Escenarios: Casos de uso

En la presente sección se muestran los casos de uso del sistema. Como se aprecia en la Figura 2, los casos de uso del cliente son: postear datos estáticos, postear datos de viajes y pedirle a servidor el resultados de las tres queries.

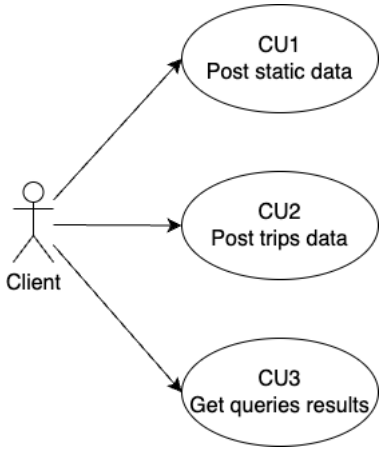

Figura 2: Casos de uso del cliente

Se aprecia entonces que el sistema cuenta de dos etapas: un procesamiento de la información que le brindó el cliente, y la entrega final de resultados una vez que se procesaron todos los datos.

A continuación, se desarrolla el resto de las vistas que especifican distintos aspectos de la arquitectura diseñada.

4. Desarrollo de diagramas: Vistas

4.1. Vista física

En esta sección se desarrollan los diagramas relativos a los componentes físicos del sistema, entendiendo cómo se despliegan y cómo se relacionan los distintos componentes entre sí.

Diagramas de Robustez

En la Figura 3 se muestra un diagrama. Luego, (Figura 4) se utiliza este esquema general para ejemplificar lo que ocurre con la combinación de datos estáticos de tipo weather y los trips.

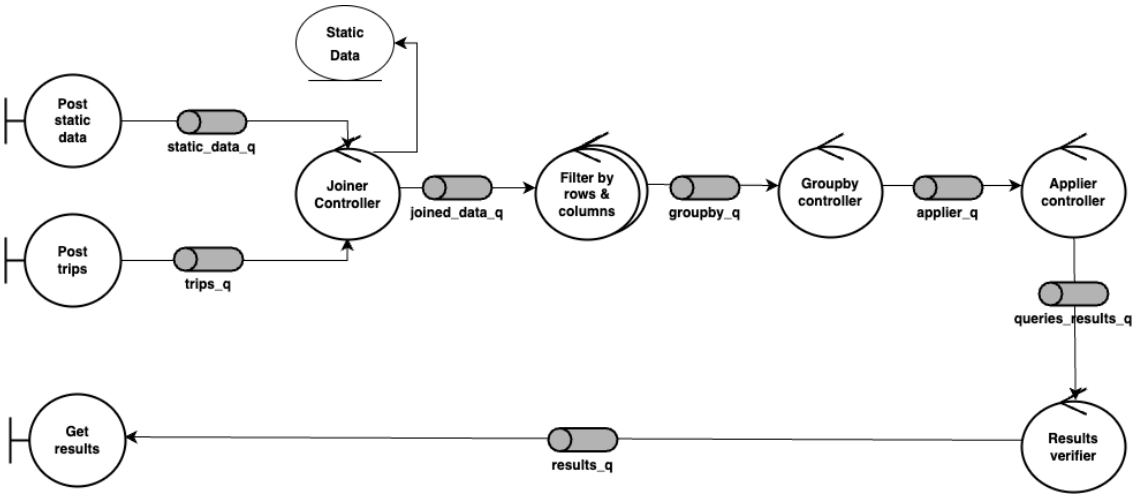

Figura 3: Diagrama de robustez - arquitectura general

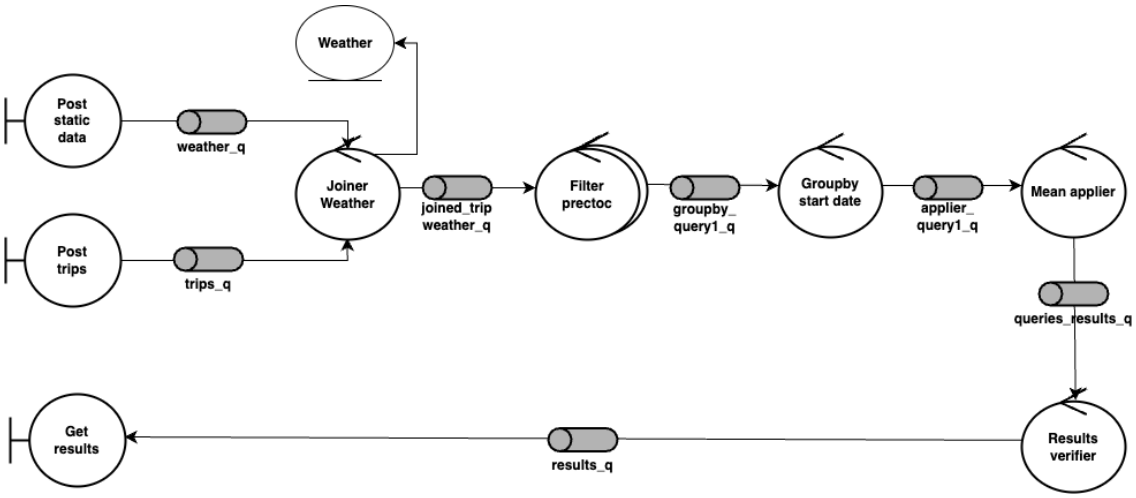

Figura 4: Diagrama de robustez - Caso particular: weather data

En las siguientes figuras (5, 6 y 7) se desarrolla en varias partes cómo se despliegan el middleware distribuido para manejar los últimos paquetes de cada tipo, llamado *EOFManager*.

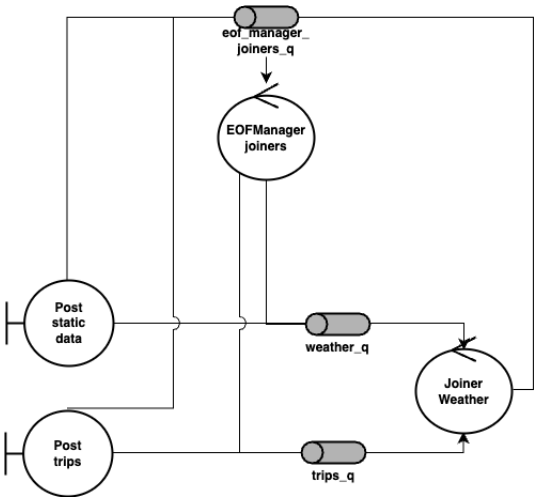

Figura 5: Diagrama de robustez - eof manager (1)

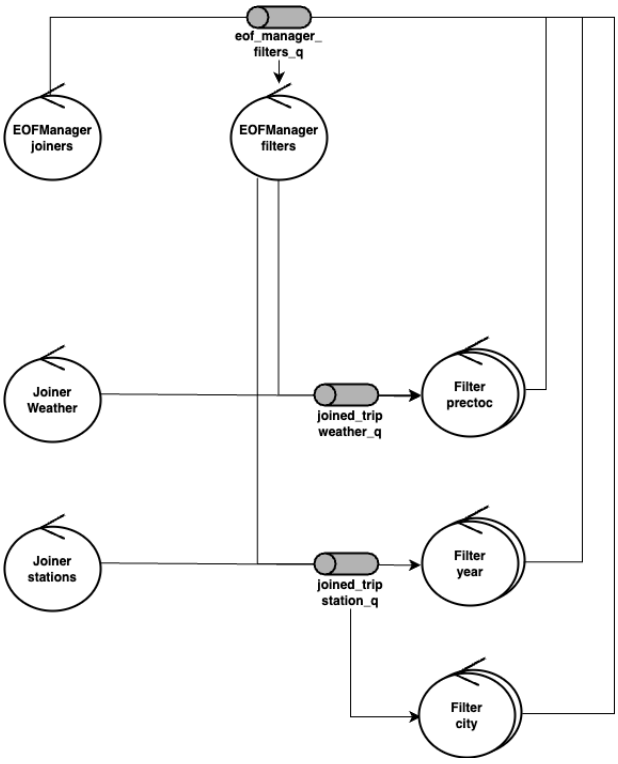

Figura 6: Diagrama de robustez - eof manager (2)

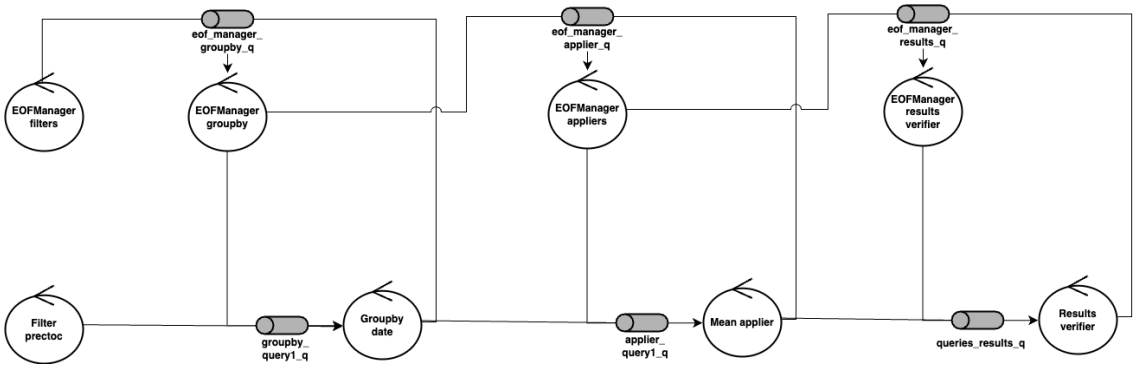

Figura 7: Diagrama de robustez - eof manager (3)

Diagramas de Despliegue

Una vez entendido la arquitectura general, examinando también ciertos casos particulares, se procede a describir cómo se despliegan los distintos componentes. En este caso, el componente que no aparece en el resto de los diagramas es el broker de rabbit, que tiene interacción con todos los componentes del servidor.

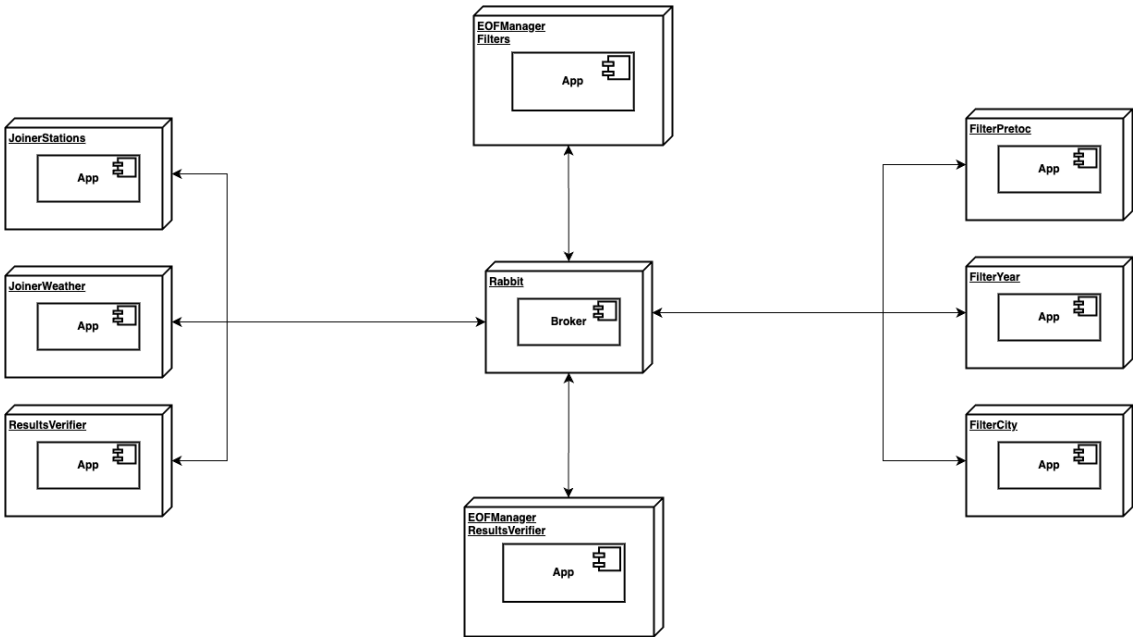

Figura 8: Diagrama de Despliegue (1)

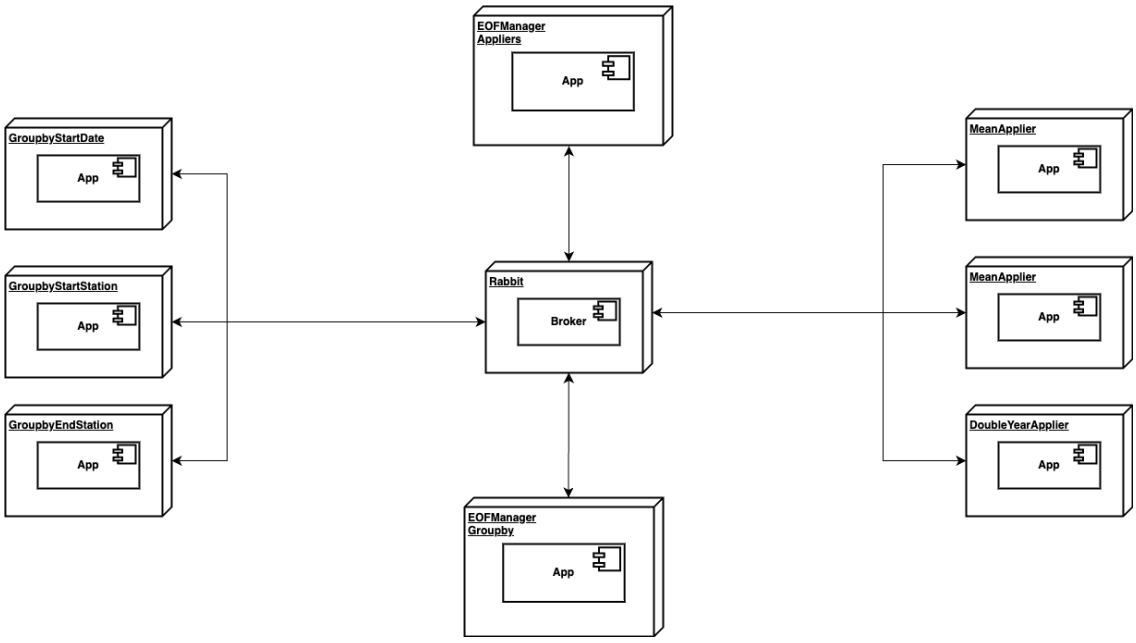

Figura 9: Diagrama de Despliegue (2)

4.2. Vista de Procesos

Diagramas de Actividades

En la Figura 10 se aprecia el primer diagrama de actividades, mostrando la actividad del cliente enviando datos estáticos para que sea procesado por los diversos nodos hasta llegar a almacenarse en memoria. A su vez, se muestra la actividad de consulta de resultados por parte del cliente, intentando conectarse con un nuevo endpoint hasta que la conexión sea exitosa (indicando que se procesaron todos los resultados).

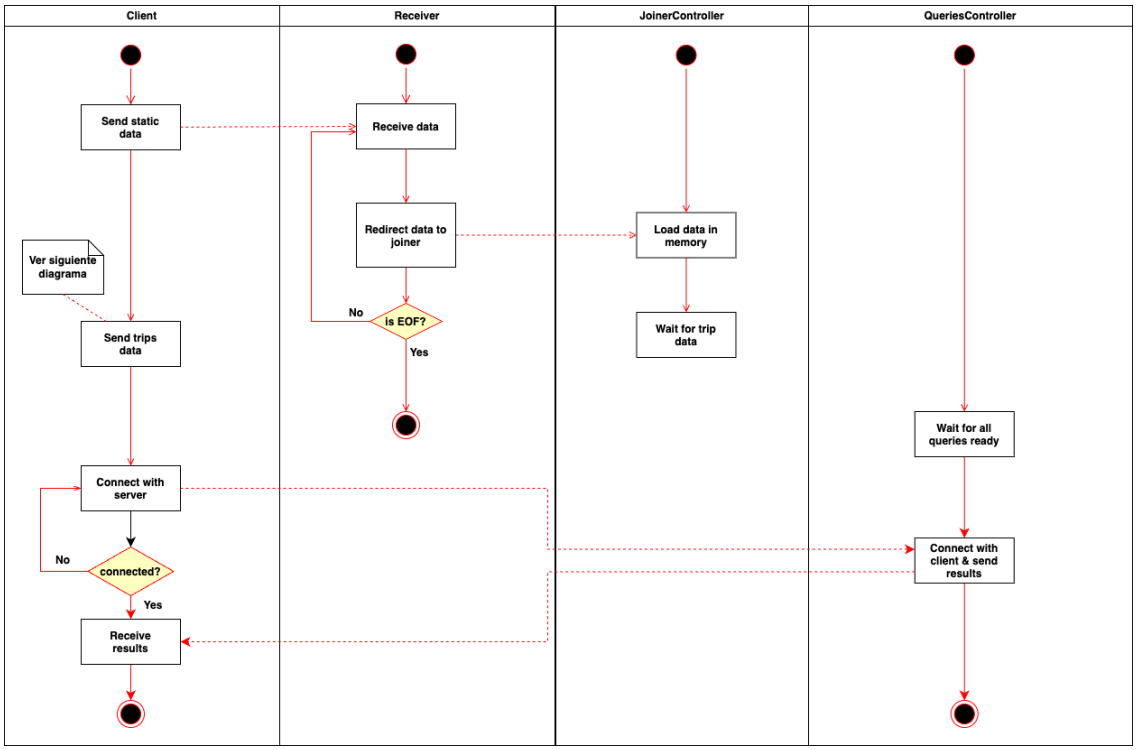

Figura 10: Diagrama de actividades - envío de datos estáticos y consulta de resultados

En las Figuras 11 y 12 se observa la actividad del cliente enviando viajes de forma iterativa, y cómo es procesado por los diversos nodos hasta poder obtenerse el resultado final.

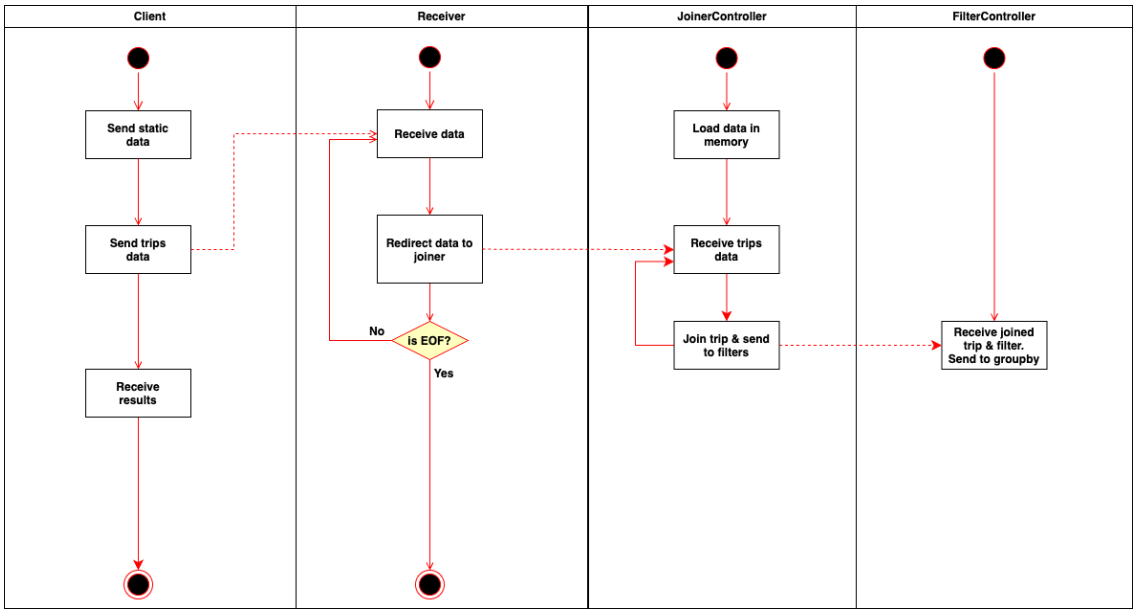

Figura 11: Diagrama de actividades - envío de viajes (1)

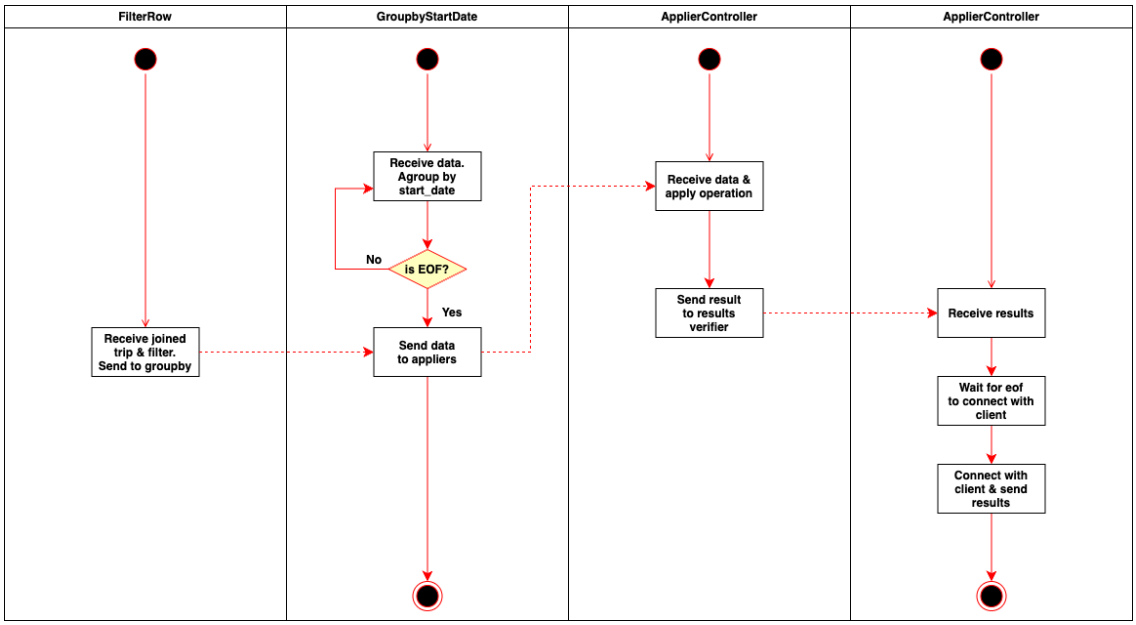

Figura 12: Diagrama de actividades - envío de viajes (2)

Cabe destacar que no se muestra la finalización de muchos componentes en estos diagramas, para no alargar la cantidad de los mismos. Se implementa graceful quit que al recibir una señal SIGTERM, se destruyen los recursos apropiados y cada nodo puede finalizar.

Diagramas de secuencia

En esta subsección se expanden las actividades de los diagramas anteriores para mostrar secuencias de mensajes. Entre ellos, se aprecia el caso particular de lo que ocurre cuando llega el último mensaje de cada tipo de dato (tanto estático como de viajes).

En la Figura 13 se muestra la secuencia de envío de datos estáticos, particularmente haciendo foco en la situación de envío de un último paquete de tipo *station*.

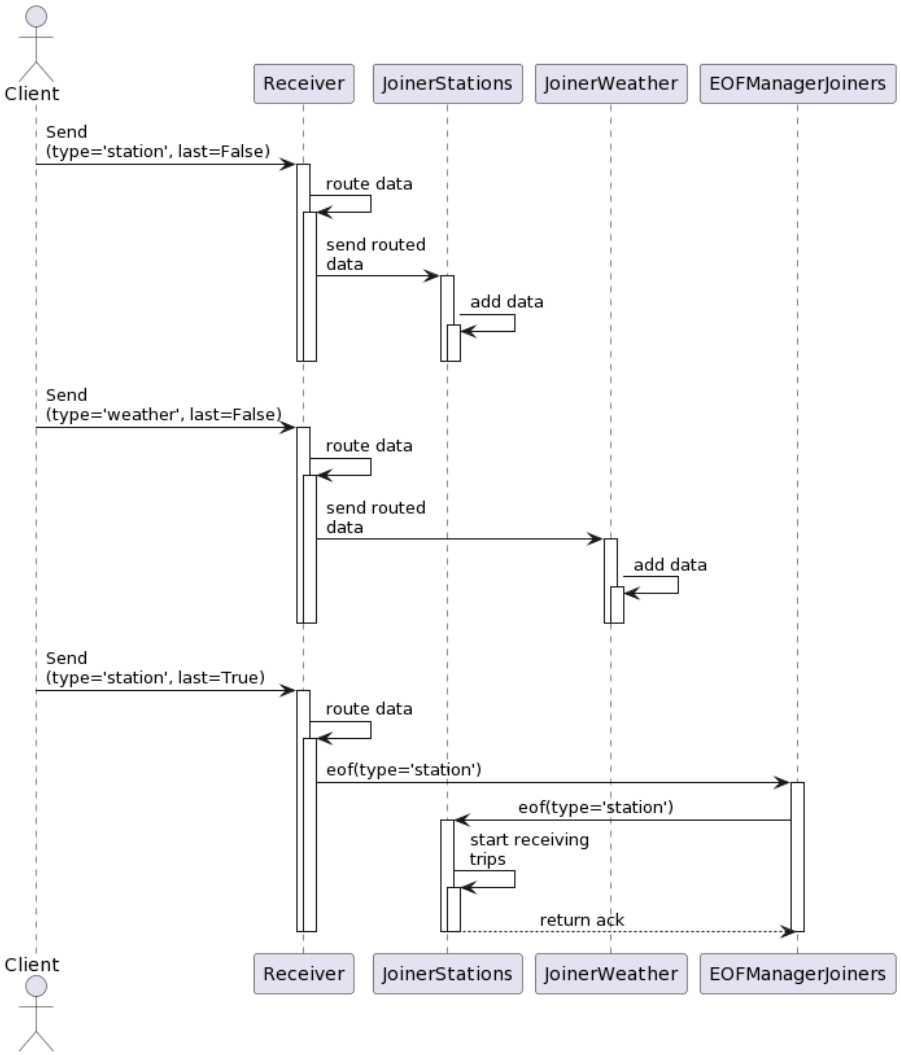

Figura 13: Diagrama de secuencia - Envío de datos estáticos

De una forma similar al diagrama anterior, se muestra lo que ocurre para el envío de un viaje (que no es el último) en la Figura 14.

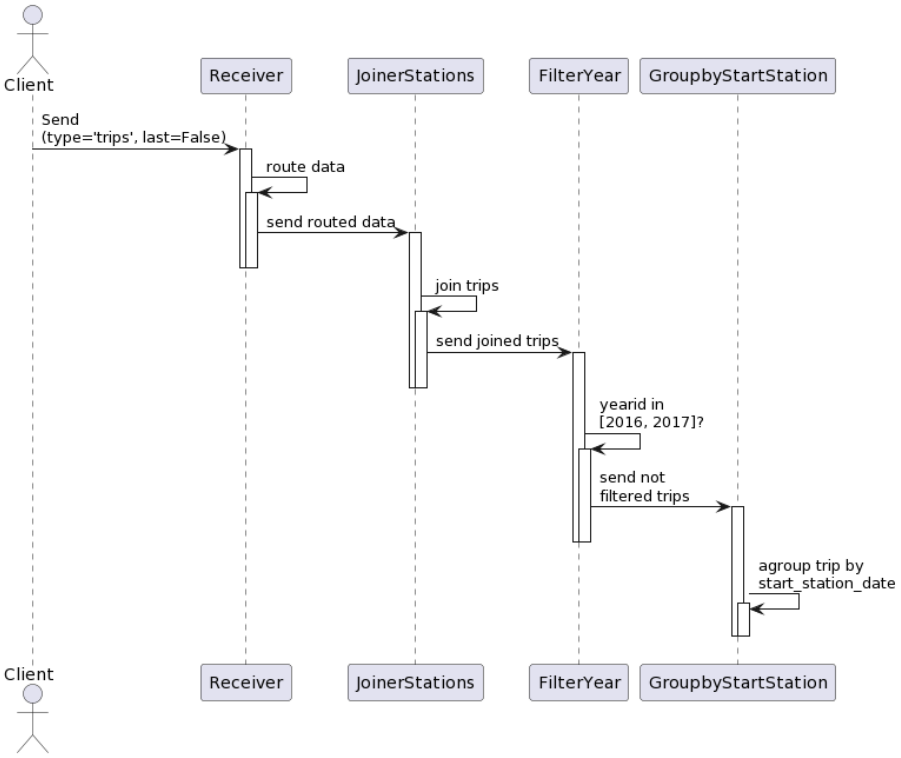

Figura 14: Diagrama de secuencia - Envío de viajes

A continuación, se muestran diagramas partidos (puesto que la longitud era muy grande para hacerlo en uno solo) mostrando lo que ocurre cuando se envía un último viaje. Se aprecia la secuencia de mensajes hasta que el cliente recibe los resultados.

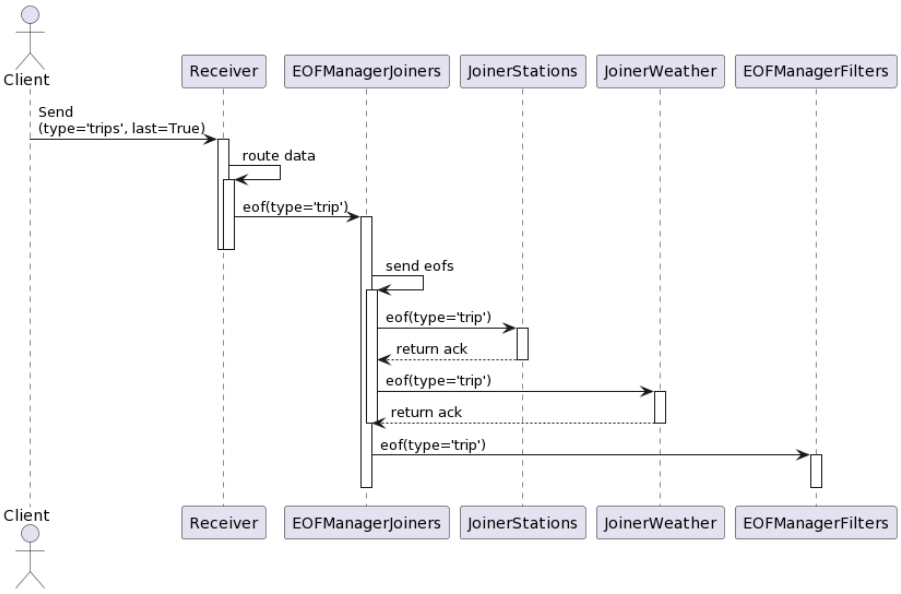

Figura 15: Diagrama de secuencia - Procesamiento de último viaje (1)

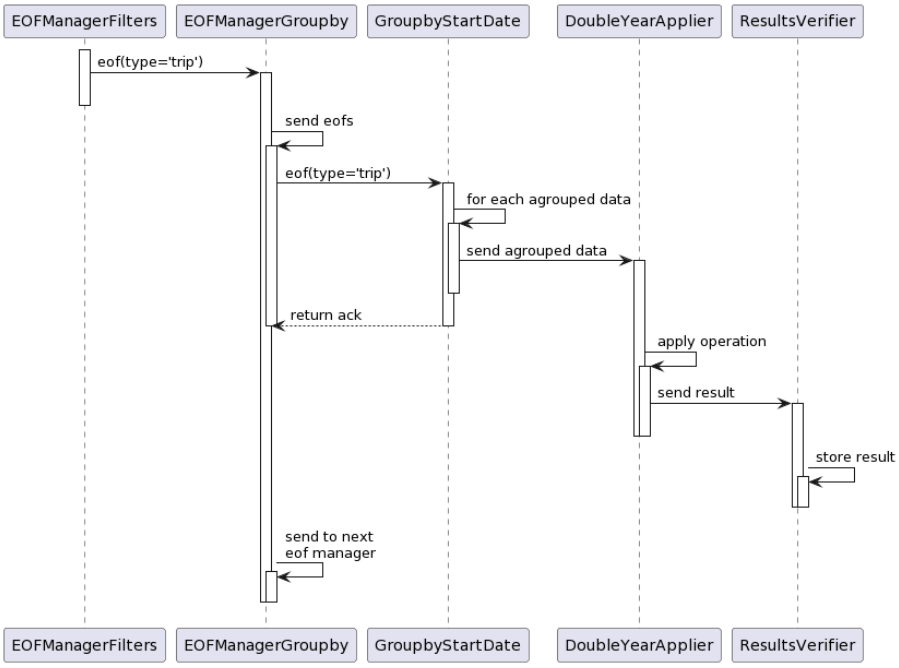

Figura 16: Diagrama de secuencia - Procesamiento de último viaje (2)

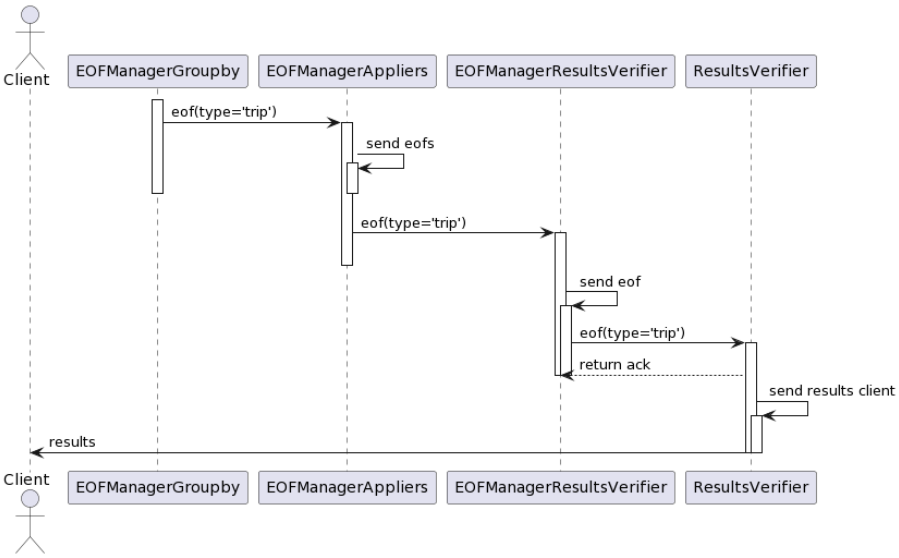

Figura 17: Diagrama de secuencia - Procesamiento de último viaje (3)

4.3. Vista de Desarrollo

Diagrama de Paquetes

Se muestran dos diagramas de paquetes para mostrar los artefactos que componen al sistema.

En la Figura 18 se muestra un diagrama general del sistema. En la Figura 19 se hace foco en el paquete *workers* que indica la clase que procesan datos en el sistema.

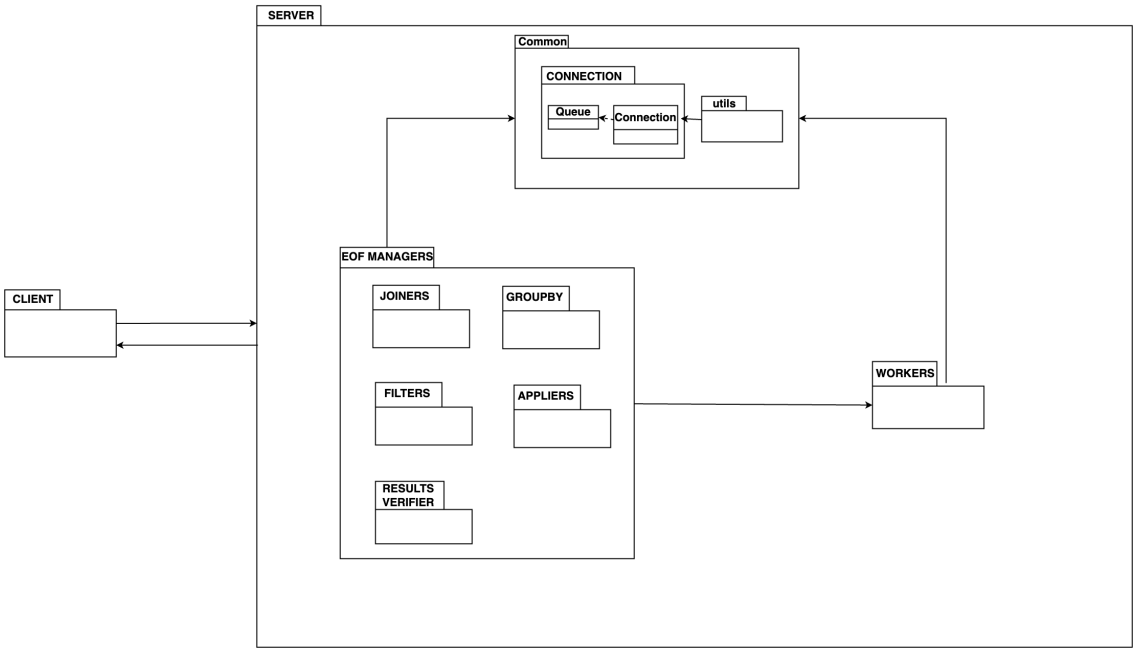

Figura 18: Diagrama de Paquetes - interacción general del sistema

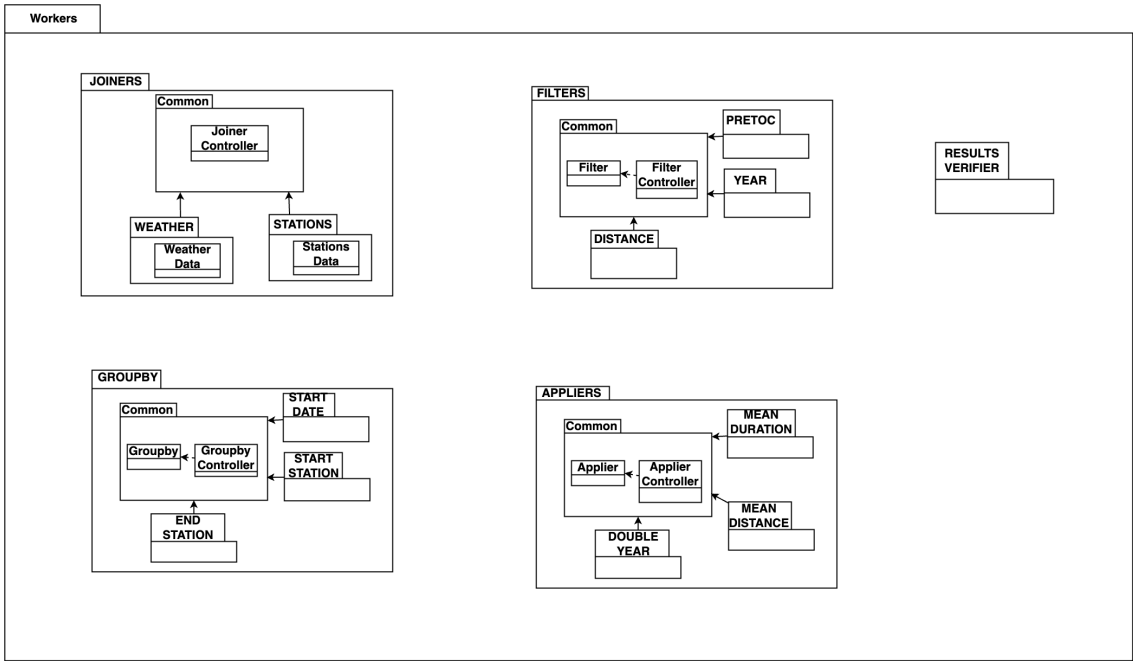

Figura 19: Diagrama de Paquetes - workers

5. Issues pendientes

En esta sección se desarrollan los *"known issues"* que, por cuestiones de tiempo, no se pudieron abordar.

Código

1. Utilizar logging en lugar de prints. Puesto que se decidió eliminar los loggings que arroja rabbit, si se utiliza la librería logging, no hay manera de eliminarlos. Si bien es una mejor práctica permitir la elección de un logging_level para visualizar a gusto los mensajes que uno elija, la estructura utilizando con prints igualmente sigue el estándar del trabajo anterior.
2. Mejor documentación de las clases y funciones. Se decidió incluir comentarios en aquellas funciones que tengan una mayor complejidad o una importancia mayor que el resto, pero una mejor práctica es documentar todas las funciones de forma correcta.
3. En ciertos componentes se tiene código repetido (como en los *EOFManagers*).
4. Se cuenta con algunos ifs/else anidados.

Diagramas

1. No cuento con una vista lógica desarrollando, por ejemplo, diagramas de clases.
2. Algunos diagramas no completan toda la lógica (por ejemplo, no se muestra un graceful quit). Esto porque la cantidad de componentes crecían mucho y decidí suprimir algunos casos borde.
3. No conseguí que el tamaño de la fuente del documento y de los diagramas sea el mismo puesto que para latex algunas imagenes necesité agrandarlas más para que se vean mejor, y las fuentes sufrieron variaciones a lo diseñado.