.UBAfiuba
FACULTAD DE INGENIERÍA

Bike Riders Analyzer

Middleware y Coordinación de Procesos

[75.74] Sistemas Distribuidos I
1C 2023

Apellido y Nombre	Padrón	Email
Fernandez Caruso Santiago Pablo	105267	sfernandezc@fi.uba.ar

Índice

1. Introducción

El presente informe reune la documentacion y distintas vistas de arquitectura del trabajo practico. El objetivo del mismo es implementar un sistema multicomputing que sea capaz de procesar un dataset de viajes en bicicleta en diferentes ciudades del mundo, logrando obtener ciertas metricas de dichos viajes.

2. Vista de Escenarios

2.1. Casos de Uso

En primer lugar se planteo el diagrama de casos de usos que intenta representar el problema a resolver y como el usuarion interactua con el sistema.

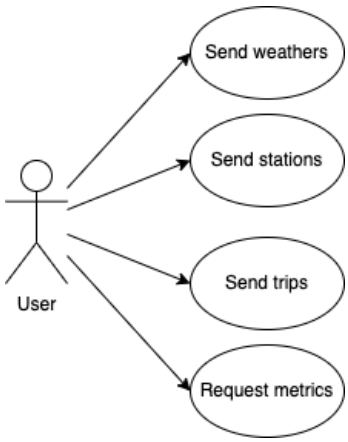

Figura 1: Diagrama de Casos de Uso

3. Vista Logica

La vista logica busca describir la funcionalidad del sistema a traves de diferentes modelos, en este caso presentamos un DAG (Direct Acyclic Graphs).

3.1. DAG

Se presenta el grafo asiclico dirigido el cual se utilizó para modelar las consultas que debe poder responder el sistema.

Figura 2: Direct Acyclic Graphs

4. Vista de Procesos

La vista de procesos busca representar los diferentes procesos que participan en un flujo y como interactuan entre si. Se consideran factores como la sincronizacion y comunicacion entre procesos.

4.1. Diagrama de Actividad

A continuacion se puede ver el diagrama de actividades correspondiente a la consulta numero 2: *Obtener los nombres de las estaciones que al menos duplicaron la cantida de viajes iniciados en ellas entre 2016 y 2017.*

Las lineas punteadas representan la comunicacion mediante las colas de RabbitMQ. El diagrama omite ciertos nodos para facilitar la lectura.

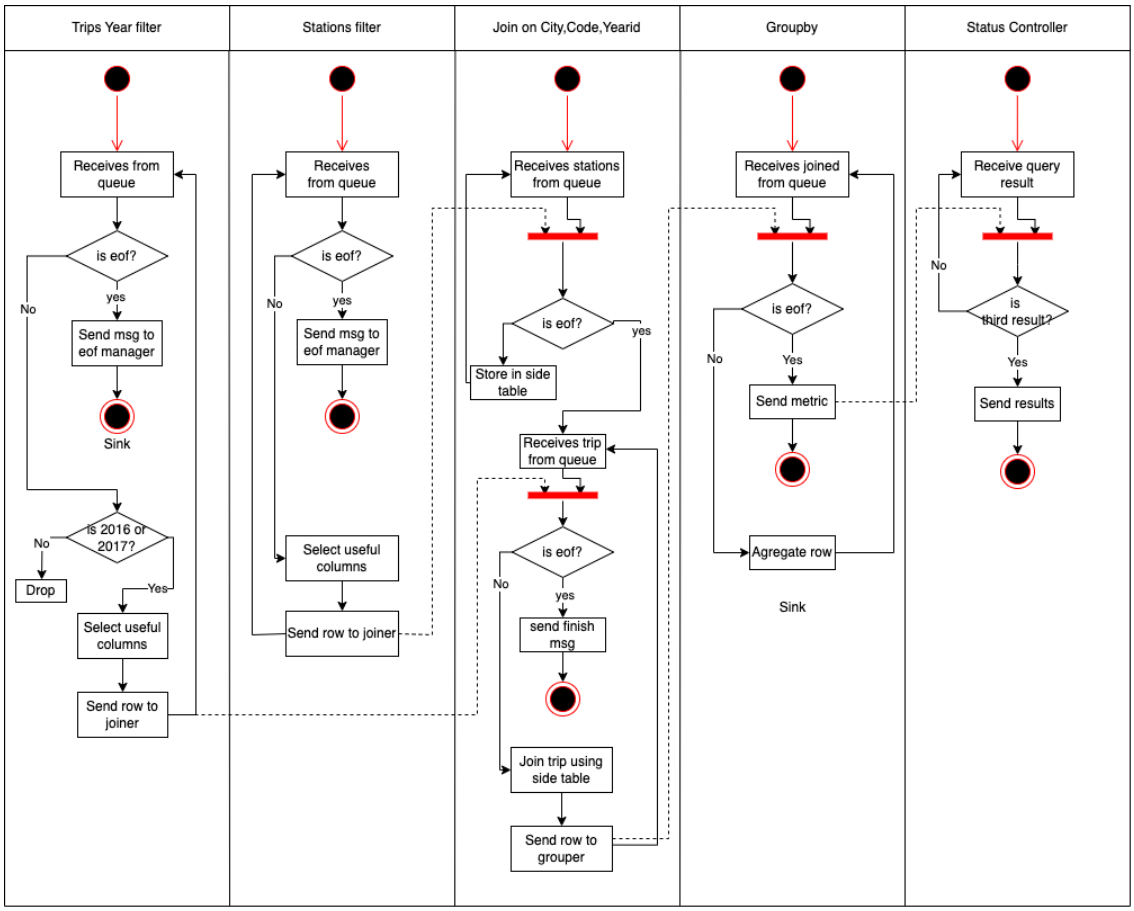

Figura 3: Diagrama de Actividad para consulta 2

Las demas consultas se resuelven de forma muy similar. Los mensajes de EOF se tratan utilizando un componente llamado *Eof Manager* el cual veremos mas detallado en los diagramas de secuencia.

4.2. Diagramas de Secuencia

En primer lugar se muestra un diagrama de secuencia de como es el proceso desde que el cliente envia los datos hasta que llegan al joiner.

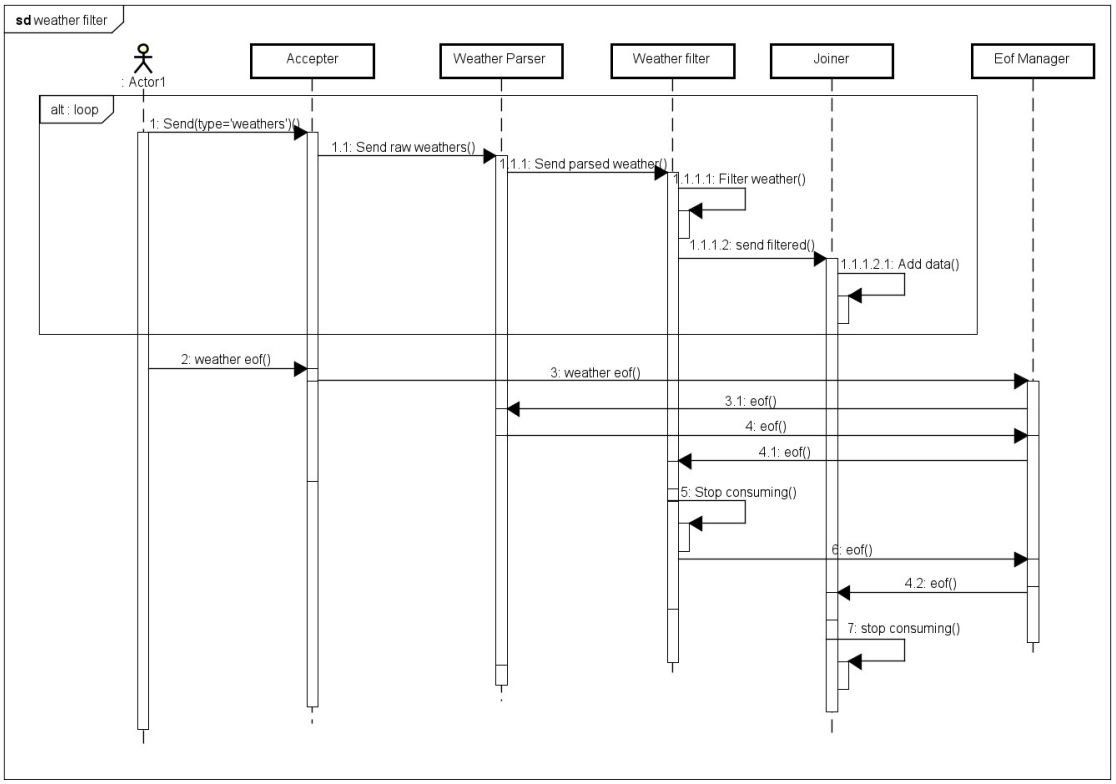

Figura 4: Secuencia filtrado de weathers

Se puede ver como todos los mensajes de EOF pasan primero por el Eof Manager antes de ir al componente correspondiente. Por otro lado tenemos el filtrado de trips, que es muy similar al anterior.

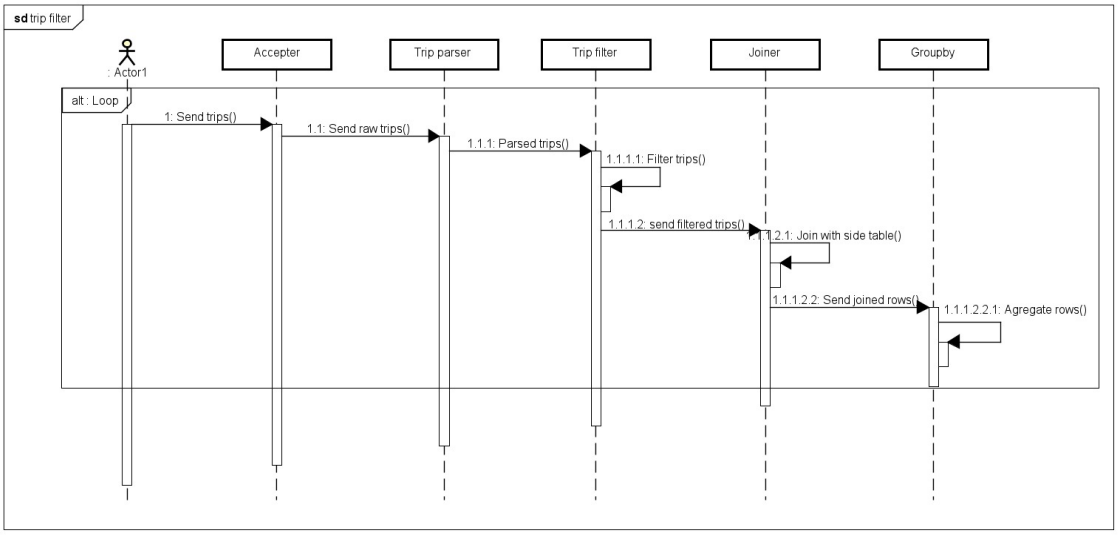

Figura 5: Secuencia filtrado de trips

Por ultimo un diagrama del funcionamiento del Eof Manager mas a detalle. En este caso tenemos un componente de parser, dos de filtrado y un joiner, por lo tanto el parser le envia datos a ambos filters, estos procesan los viajes y los envian al joiner.

Una vez que al Trip Parser le llega un EOF, deja de consumir mensajes y envia el EOF al Eof Manager, el Eof manager sabe que solo hay un Trip Parser por lo tanto sabe que ya puede enviar los EOF a la siguiente etapa. El Eof Manager tambien conoce que hay dos filters por lo cual encola dos mensajes de EOF en dicha cola. Cuando los filter procesan el EOF dejan de escuchar y envian EOF al Eof Manager.

Como el Eof Manager sabe que hay dos trip filter, espera por dos mensajes de EOF antes de enviar el EOF al joiner.

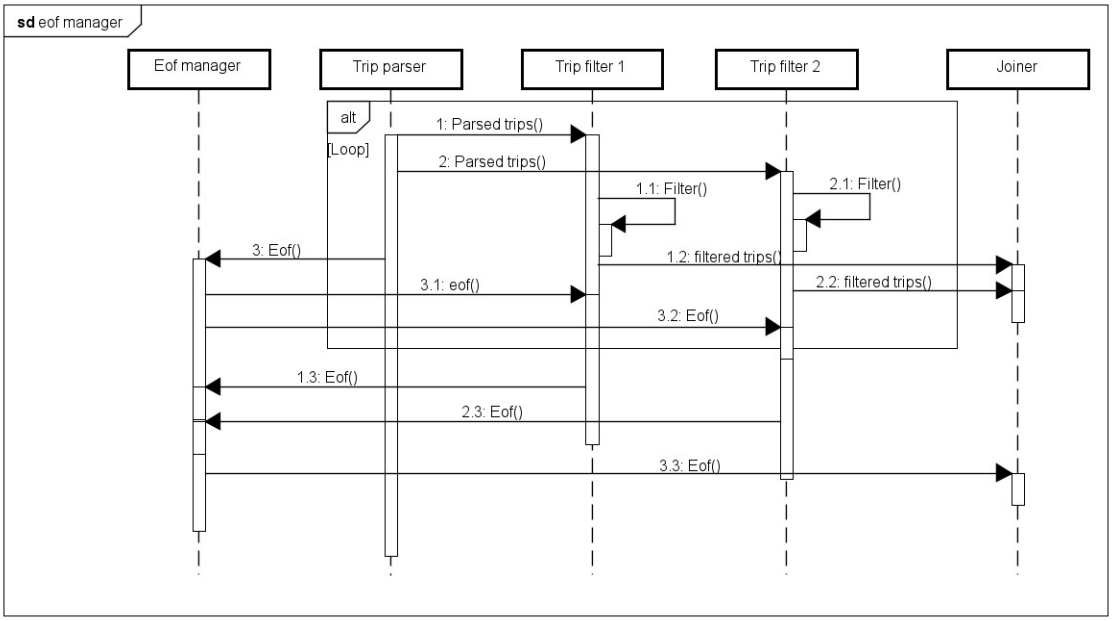

Figura 6: Secuencia funcioanmiento de Eof Manager

5. Vista Fisica

La vista física busca describir la estructura física del sistema, su distribución y como los componentes del sistema se comunican entre si. Para ello se utiliza un diagrama de robustez y un diagrama de despliegue.

5.1. Diagrama de Robustez

En el diagrama podemos ver todos los nodos del sistema y como se comunican entre si. La regla general es tomar mensajes de una cola, procesarlo, y enviarlo a otro nodo mediante otra cola.

En el diagrama también se puede ver cuales nodos son escalables y cuales no, buscaremos hacer nodos "genéricos"que se puedan parametrizar mediante variables de entorno, los nodos que se pueden escalar son los filters, los joiners, el calculador de distancia, el modificador de fechas y el parser. Los groupers son únicos ya que deben ir realizando una agregación cada vez que reciben un dato para poder calcular una métrica. También son únicos el accepter, el status controller y el Eof Manager.

Por otro lado, todos los nodos tienen la capacidad de hacer "select", esto es seleccionar solo algunas columnas antes de mandar el mensaje al próximo nodo.

Todas las colas que se ven son colas de RabbitMQ, algunas de estas colas son work queues y otras publisher/subscribe queues.

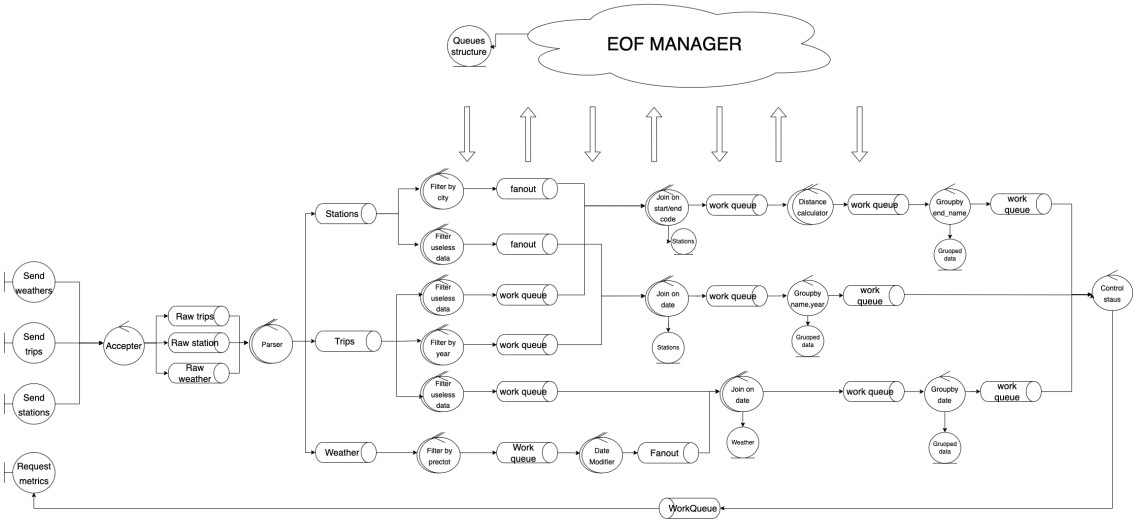

Figura 7: Diagrama de Robustez

5.2. Diagrama de Despliegue

Para no repetir información que ya fue provista en el diagrama de robustez, el diagrama se despliegue se simplificó en uno mas genérico que busca mostrar los diferentes tipos de nodos y con cuales se comunican.

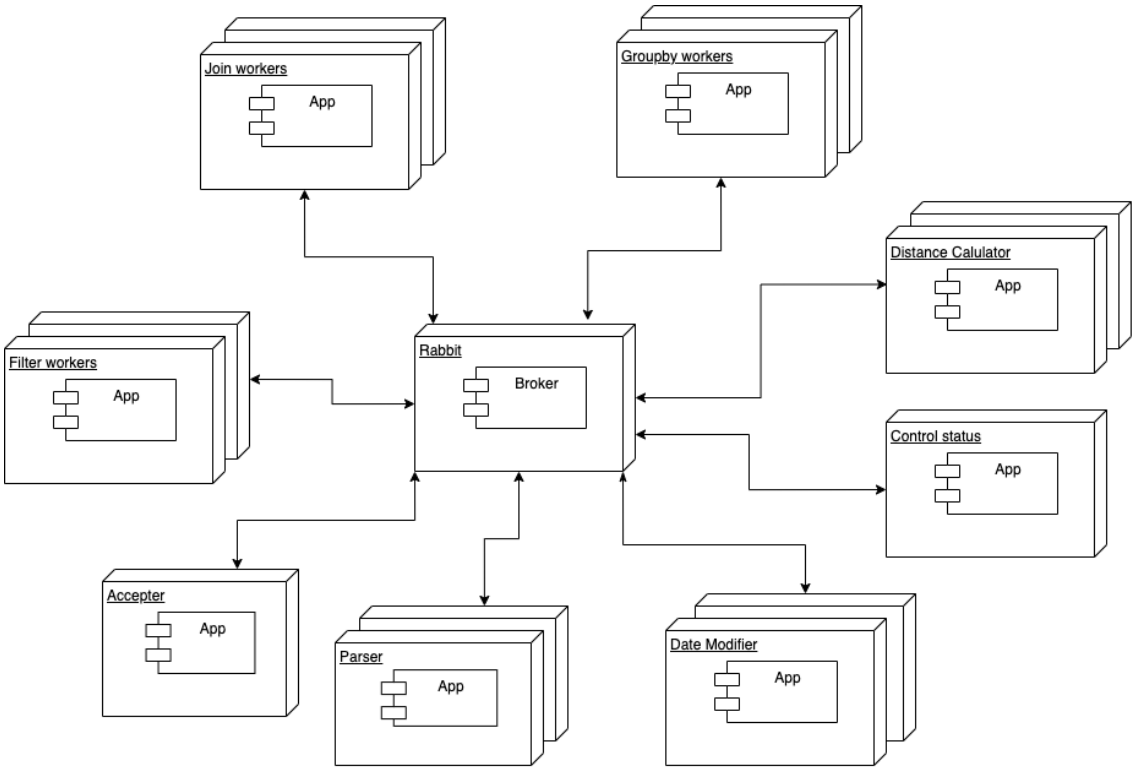

Figura 8: Diagrama de despliegue del sistema

6. Vista de Desarrollo

6.1. Diagrama de Paquetes

Para dicho diagrama se modelaron los diferentes módulos del sistema y como se relacionan unos con otros. Cada servicio es un paquete diferente que de despliega en un nodo diferente pero todos dependen del paquete commons que contiene la logica del middleware de comunicacion.

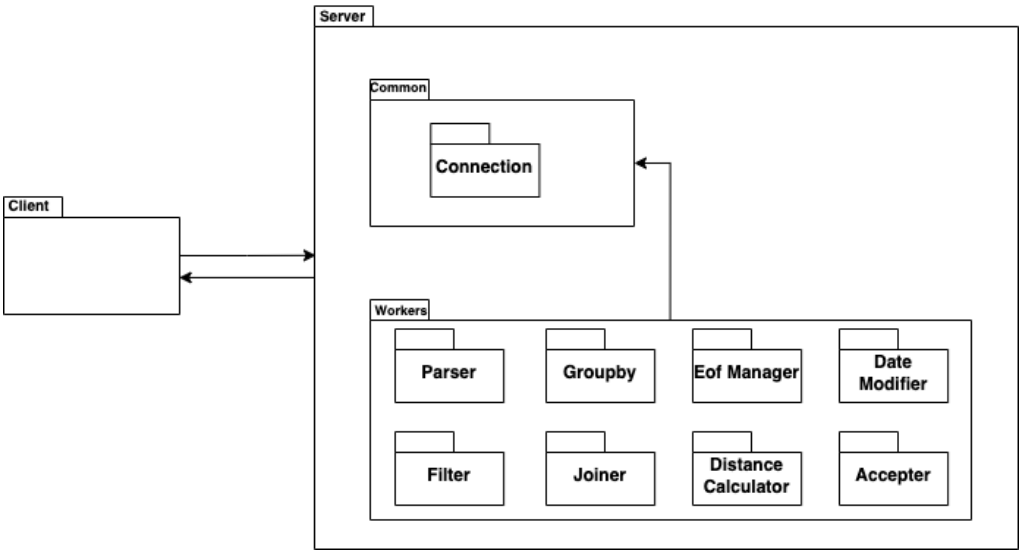

Figura 9: Diagrama de paquetes