Sistemas Distribuidos

Steam Analysis

TP Tolerancia a fallas

Grupo: **1**

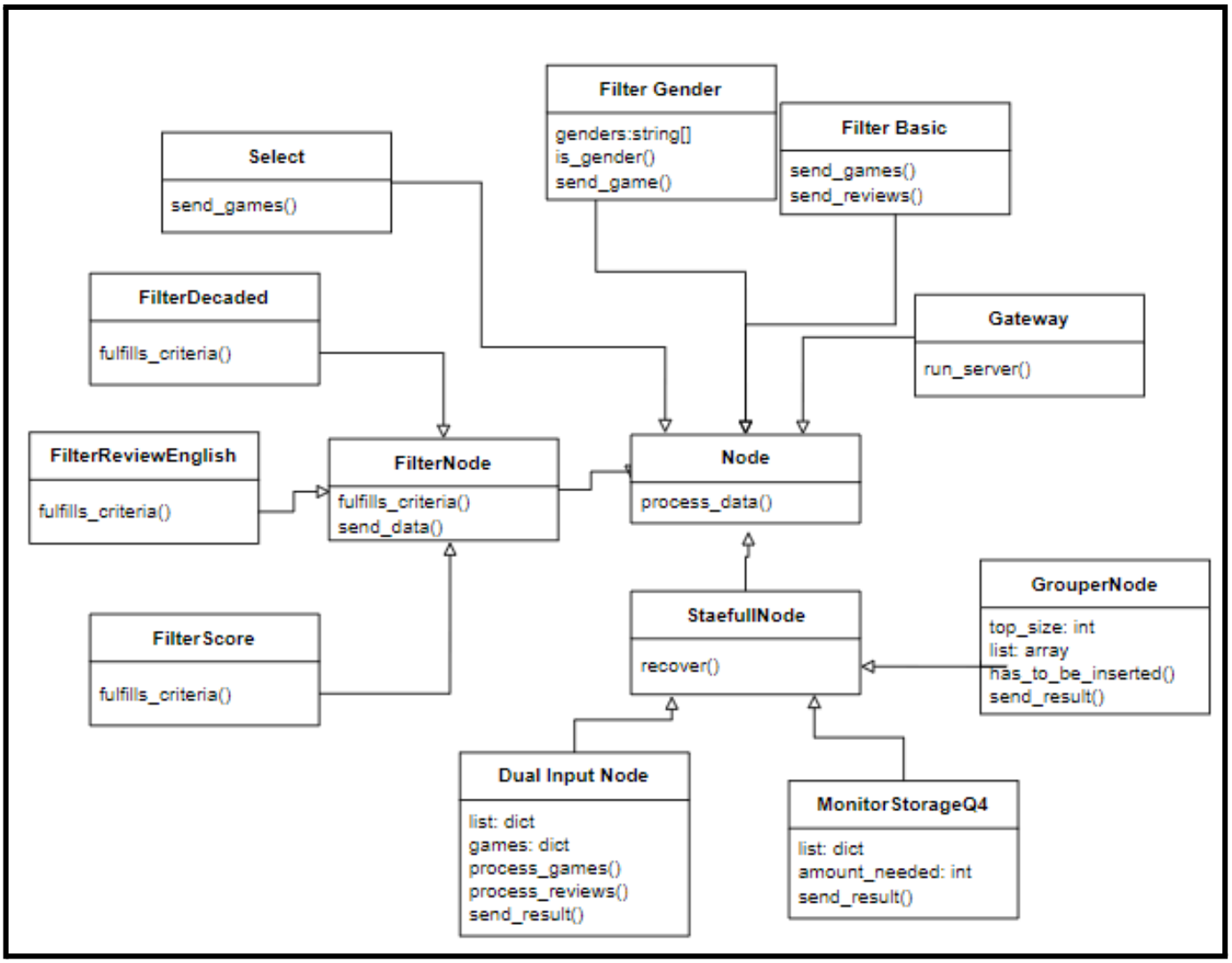
Integrantes:

Nombre Completo	**Padrón**
Abraham Osco	*102256*
Franco Primerano	*106004*
Kevin Ariel Gadacz	*104531*

1. Modelo de Vistas de arquitectura 4+1:

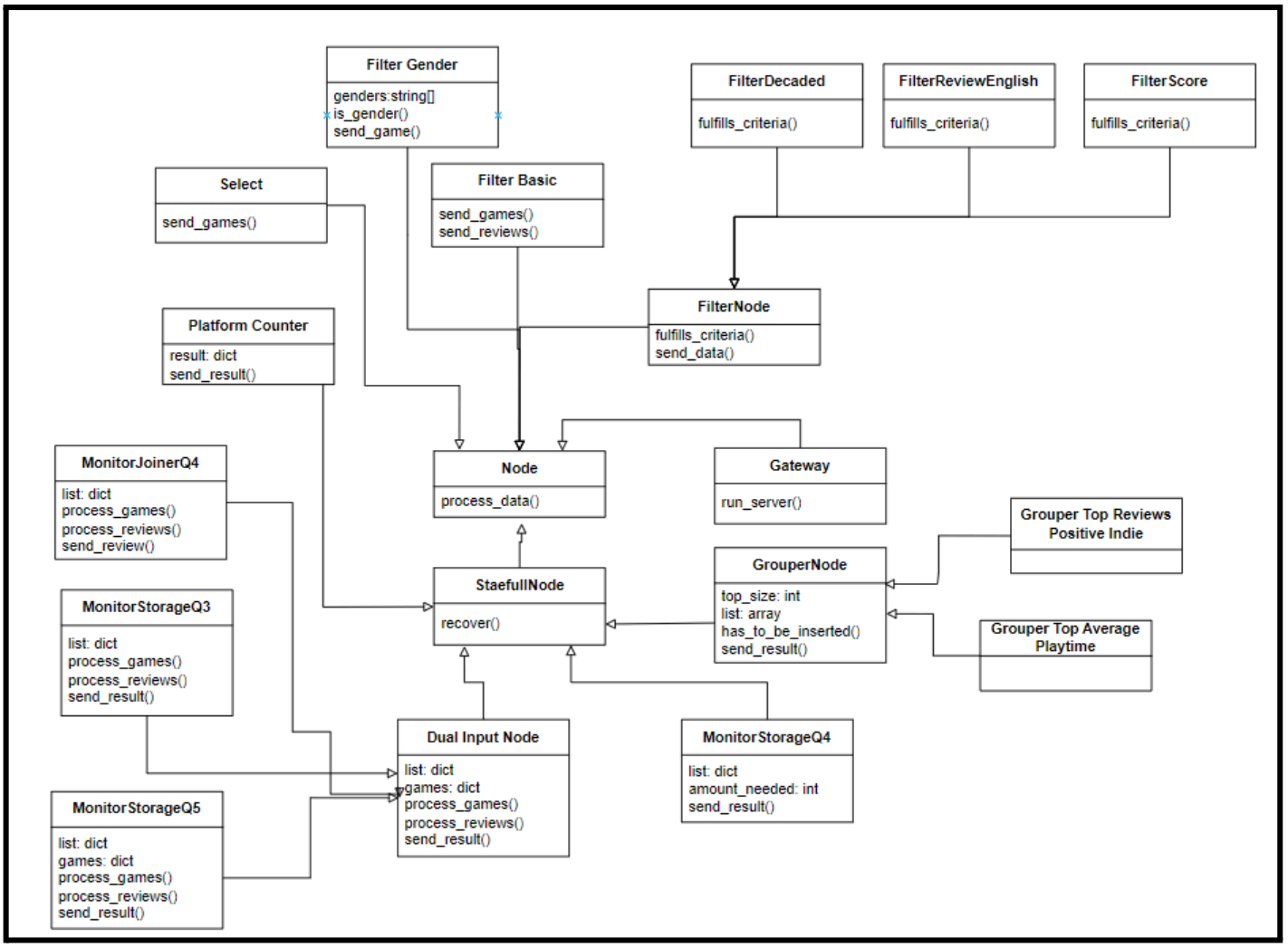
1. Vista Lógica (Vista Estática):

Vista de abstracciones



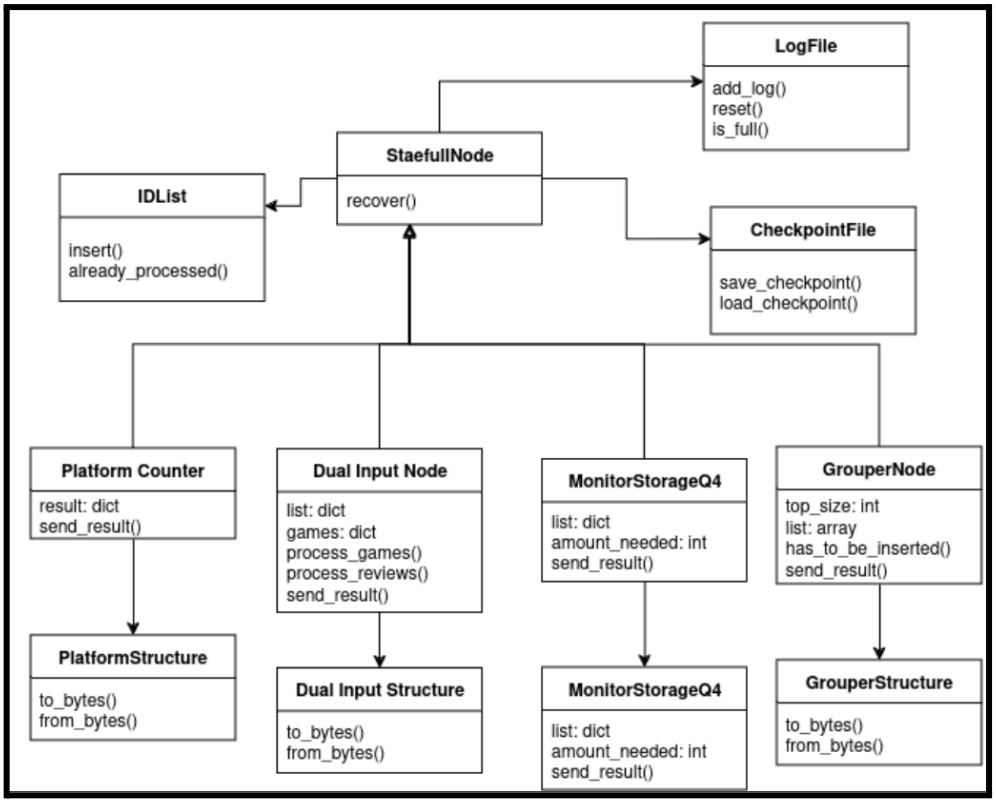
Todos los nodos del sistema que deban utilizar rabbitmq heredan de algún modo de la clase **Node**, que brinda las funcionalidades en común que deben tener. Tiene más métodos que los mostrados en el diagrama.

Herencia de Controllers con la clase **madre Node.**



Todos los nodos stateful(salvo el gateway) heredarán de stateful Node, que posee funcionalidades de tolerancia extras. El gateway tiene necesidades especiales que son implementadas por separado. Los Nodos DualInputNode son aquellos que reciben mensajes de dos colas: games y reviews. Estos tienen un trato diferencial según el tipo de DTO que reciban.

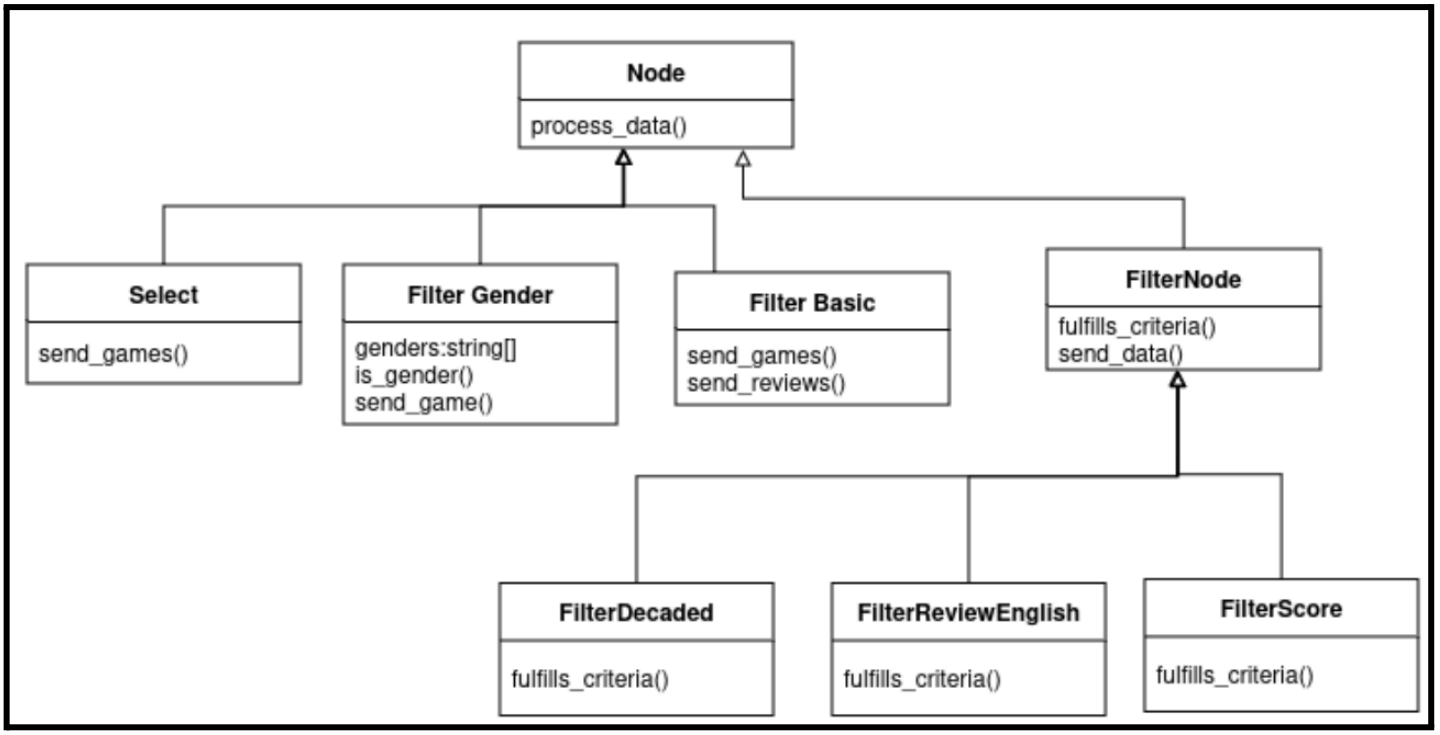
Nodos stateful



Los nodos stateful tienen relaciones con clases para manejar su tolerancia. En primera instancia cada **clase Stateful:**

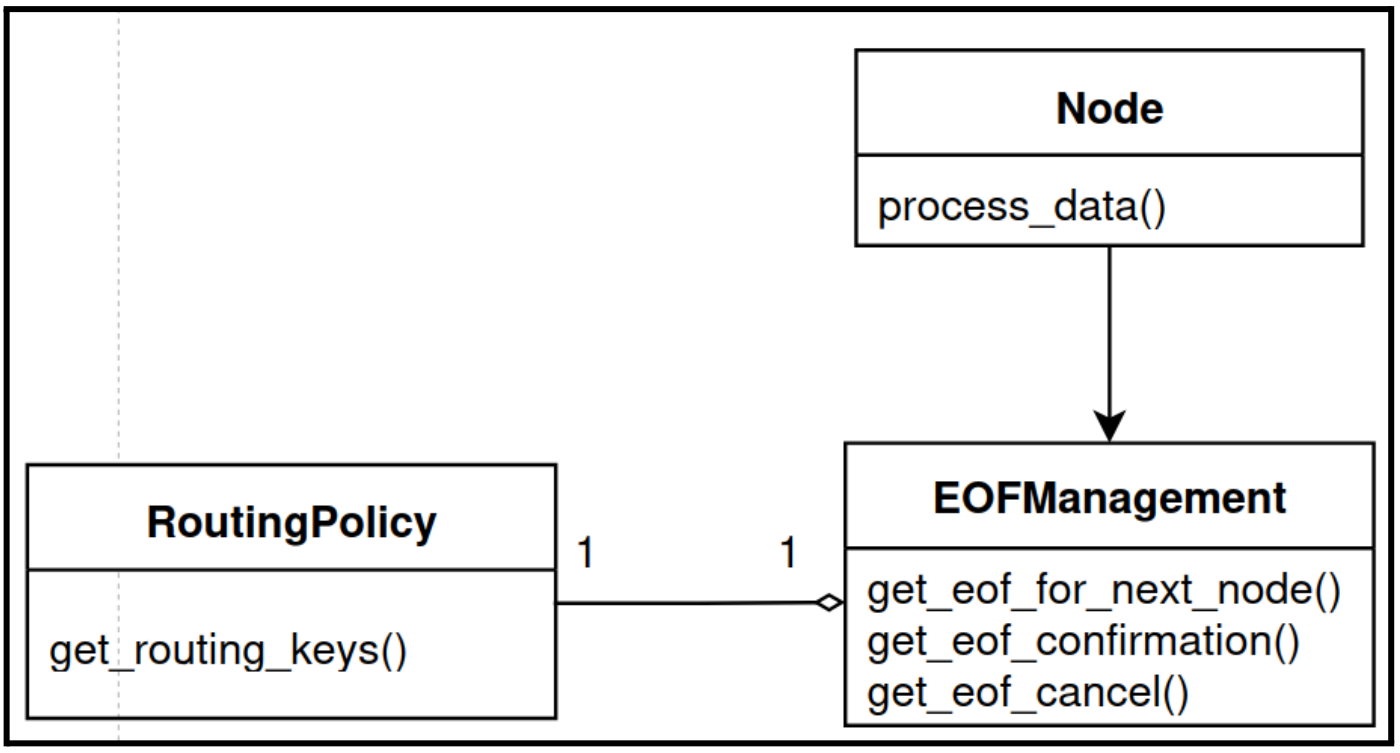
1) Tienen una **clase estructura asociada** tiene en su estado la información que permanece en memoria (y debe ser persistida), también es capaz de serializar o deserializar esa información.
2) Por otro lado poseen una clase que representa al archivo de log donde van escribiendo los DTO que van recibiendo (en bytes) a su archivo de log asociado, de modo que si se caen puedan reprocesarlos como si estuvieran viniendo de rabbit.
3) También se relacionan con una clase (**CheckpointFile**) que representa al archivo de checkpoint, con funcionalidades de guardado y restauración.
4) Por último tienen una o más listas de los últimos IDs procesados de modo que garantice la idempotencia al no procesar mensajes más de una vez.

Nodos stateless:



Los nodos stateless comparten mucha funcionalidad en común. Por ejemplo todos los selects utilizan una única clase, teniendo que solo configurar el origen y destino, y las columnas a remover. Algunos filtros son tan sencillos que basta implementar una función **“fulfills_criteria”** para saber que tratamiento hacerle al DTO.
Todos estos nodos no presentan ningún trato especial frente a caídas, ya que el reprocesamiento de mensajes será de duplicado por algún nodo stateful más adelante

Relacion de Node con **EOFManagement:**



Elección de líder:

Se desarrolló el algoritmo de ring para la elección de líder usando socket TCP.

1) Cada nodo médico, tiene 3 sockets TCP (Aceptador, Connect y el Peer) y un servidor UDP interno.
2) Cada nodo **se conecta** con su vecino cuyo id es el siguiente al propio.

EJ: Médico con id 500 se conecta al Médico con id 501 (socket Accept) y se genera un peer, donde se iniciará el protocolo.

Protocolo Completo:

1) Un Nodo médico antes de hacer un connect. Envía un paquete UDP “ping” al Servidor UDP que tiene el nodo médico al que quiere conectarse.
    a) Si recibe un “ping” como respuesta procederá a hacer el connect.
    b) Si ocurre un timeout o no puede resolver el DNS (socket.gaierror) no intentara conectarse y pasará a conectar al siguiente nodo médico.

2) El nodo médico al **conectarse** con éxito. Enviará un TokenDTO:
    a) Este será de tipo **Election** y tendrá un diccionario con el id y la ip numérica del Nodo. ej: {500:”172.25.125.100}
    b) El nodo siguiente agrega su id y su ip numérica quedando

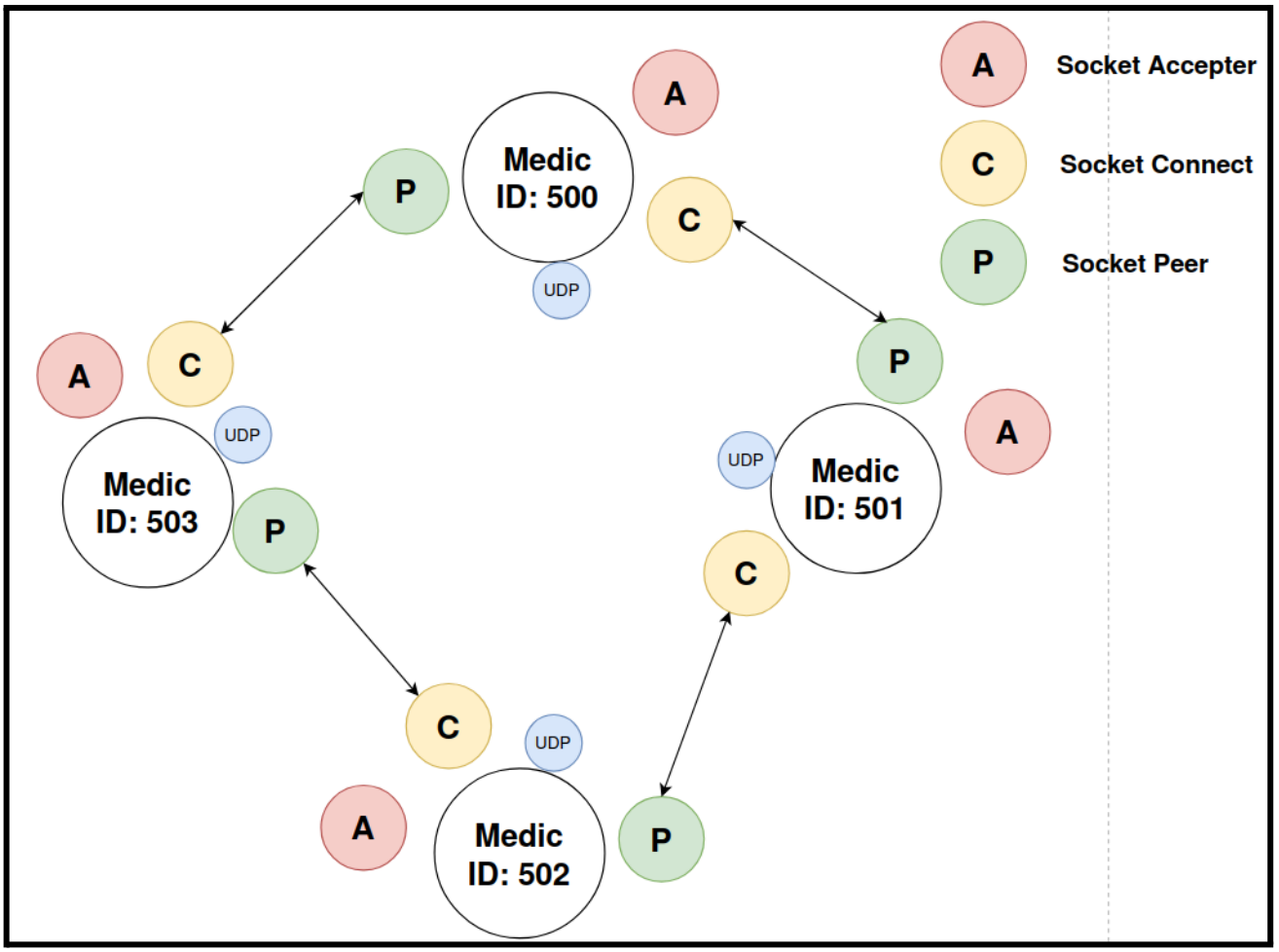
```
{
        500: “172.25.125.100”,
        501: “172.25.125.101”
}
```

    y así con el resto de los nodos.

3) Cada nodo al recibir un tokenDTO chequea **si su id esta en las claves**:
    a) **Con TokenDTO de tipo Election**: **Si esta su id** en el diccionario (lo que significa que el Token dio toda la vuelta), cambia el tipo de token a **Coordinator** setea el id maximo y la ip numérica a los campos del DTO, y en el diccionario deja solo su id e ip. Envía el tokenDTO a su siguiente.
    b) **Con TokenDTO de tipo Coordinator**: Si está su id esta en el diccionario no hace nada.

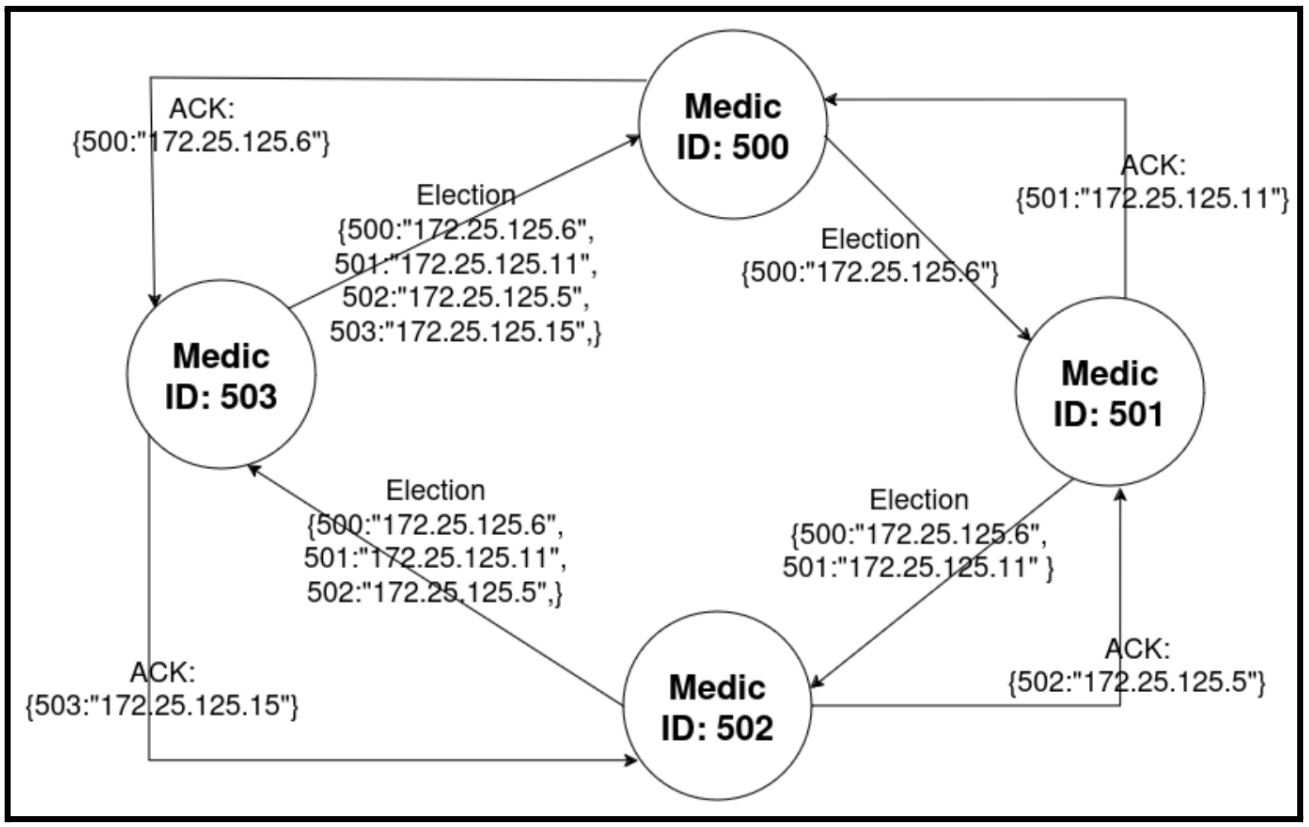
4) Luego de enviar el TokenDTO (Election o Coordinator) procederá a enviar un TokenDTO (**ACK**).
5) Cuando el médico detecta que su id esta en el tokenDTO (Coordinator) dejará de enviar el tokenDTO.

Estructura interna del ring de Nodos médicos.



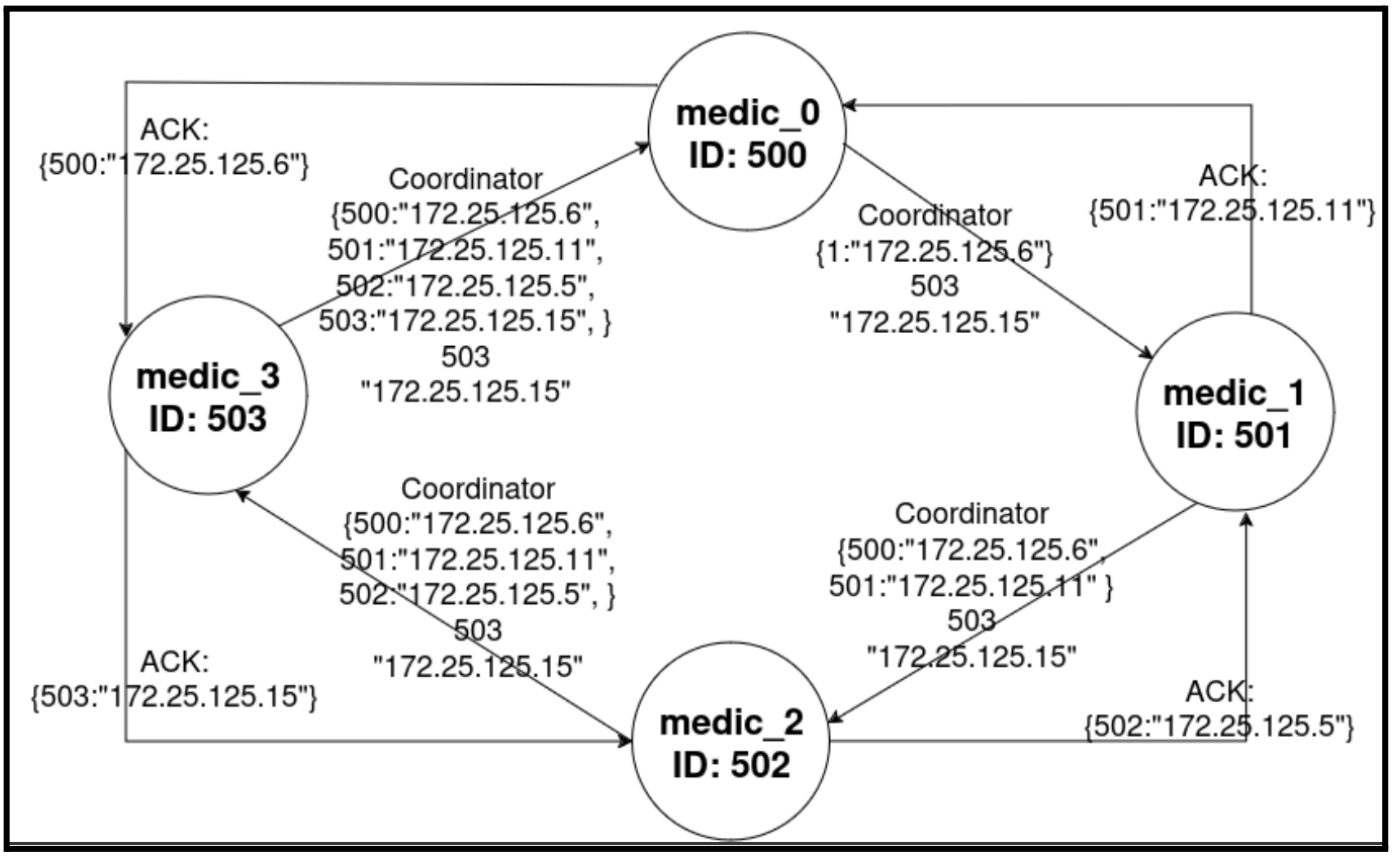
Cada Nodo médico contará con un socket Aceptador, Connect y un peer y un socket UDP.

Protocolo inicial para la búsqueda del líder envío del mensaje Election.



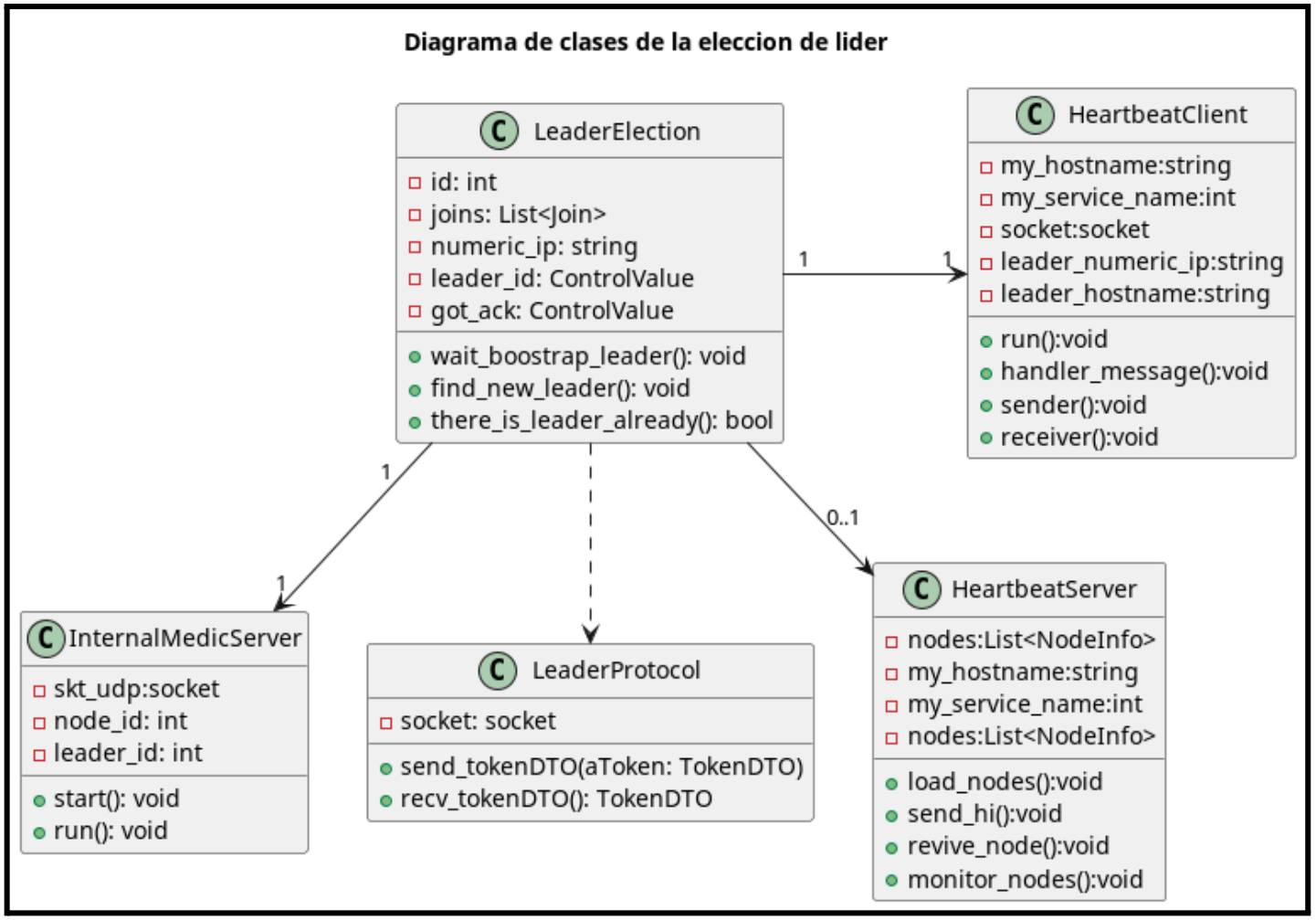
En este caso el médico con id= 500. Inicia la elección de líder envía un elección con su id y su ip numérica en un DTO (TokenDTO). Este TokenDTO irá circulando por todos los nodos y cada nodo e irá agregando su id y su ip numérica.

Protocolo del envío del mensaje Coordinator



Si el médico que inició la elección (en este caso Medic ID: 500 detecta que **su id esta en el TokenDTO**, ahora enviara el **tokenDTO** al próximo medico **con un Coordinator** junto con el id y la ip numérica del líder y su propio id/ip numérica.

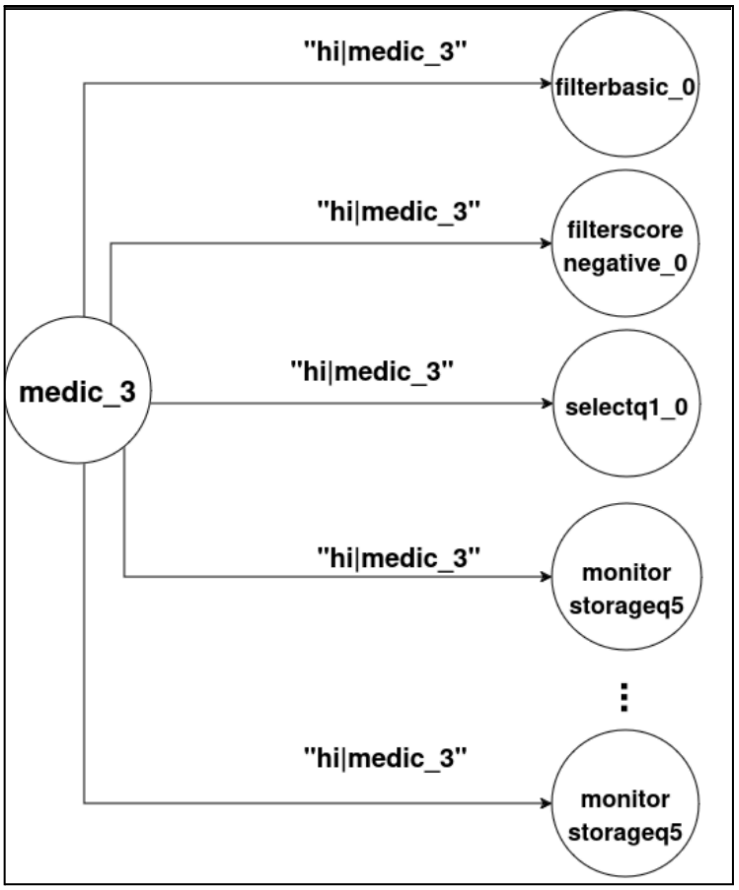
Diagrama de clases de las clases asociadas a la elección de líder.



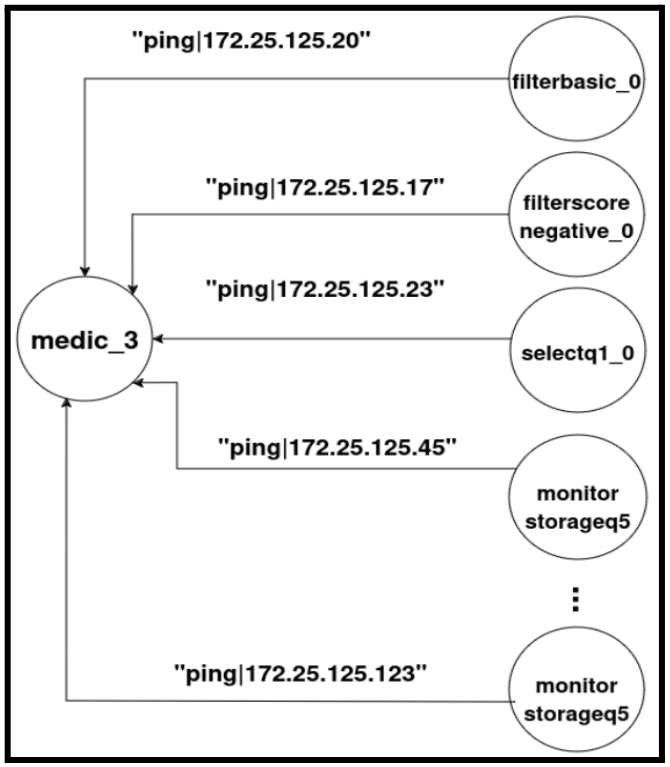
Leader Election (Que sería un **NodoMédico**) tiene una instancia de HearbeattClient. El médico líder tendrá un objeto del tipo **HearbeatServer** que lo usa para revivir/monitorear los demás nodos del sistema**. InternalMedicServer** tiene el socket UDP mencionado anteriormente.

Protocolo de Monitoreo de los nodos del sistema.

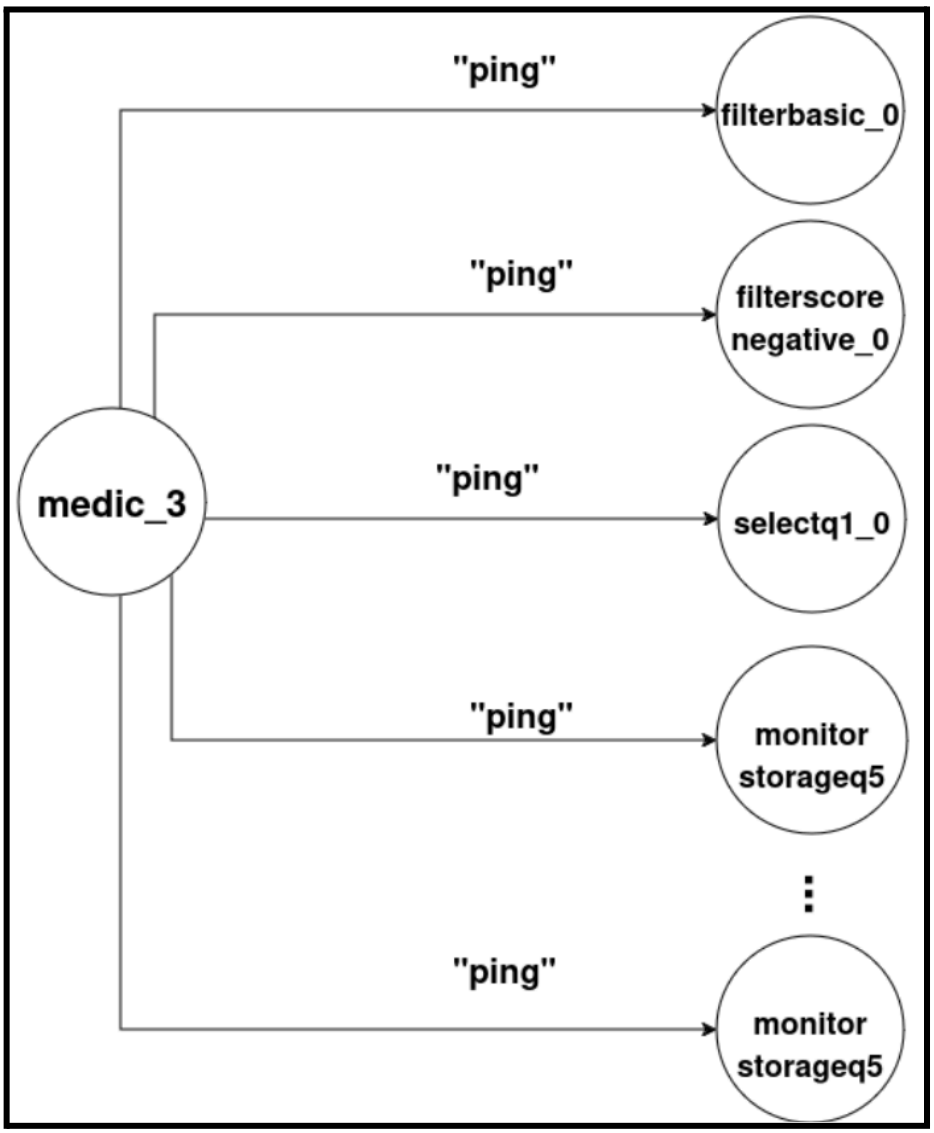
1) Cuando un nodo leader recién inicia. (Intenta revivir a todos los nodos del sistema, ya sea si esta vivo o no, para q pueda responder al primer mensaje rápido").
2) El primer mensaje que envía el leader es "**hi|{hostname}**" a todos los nodos del sistema (broadcast).



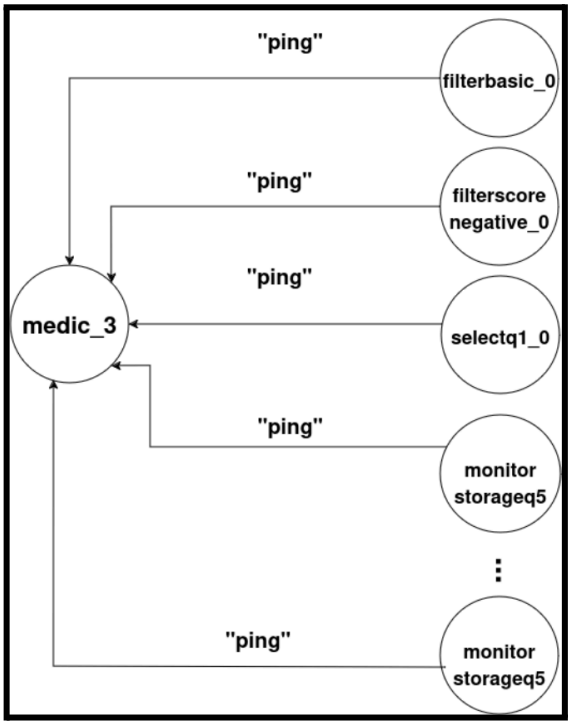
3) Los otros nodos del sistema reciben este mensaje especial se guarda la ip del leader y responde con un "ping" junto a su ip numérica. El leader almacena y asocia esas ip numéricas a cada uno respectivamente.



4) Luego el leader va a hacer un broadcast del mensaje “**ping**” cada **3s** para que cada nodo sepa que el leader sigue siendo el mismo.



5) L**os nodos del sistemas estarán enviando cada 0.45s un ping al lider.** SI el leader no recibe un “ping” **almenos cada 0.9s (en cada nodo)** considerara al **nodo muerto y lo revivirá**.



6) En caso de que un nodo está caído, se le enviará un ”hi|{leader_hostname}” y seguido de un “ping”.

Tolerancia a fallas mientras se busca un líder

Para demostrar la tolerancia a fallas mientras se busca un lider se necesita un sleep de 5s en los send del tokenDTO antes de enviar mensajes.
Se inicia 4 nodos médicos y antes de que envíen su primer mensaje Election se cae el **nodo medic_1 (ID: 501).**
Notar que cuando se cae el **medic_1** el **socket peer** del **medic_2 lanza excepción** y nunca podra recibir mensaje de medic_0 ni enviarle ACKs:

```
medic_0  | 2024-11-23 22:13:00 INFO     Sending [Connect]: Type: 0 {500: '172.25.125.2'} To: 20501 👆🔥
medic_1  | 2024-11-23 22:13:00 INFO     [501] Node: medic_2 is Alive! ✅
medic_1  | 2024-11-23 22:13:00 INFO     [501] Trying to connect to 502 🔄
medic_1  | 2024-11-23 22:13:00 INFO     [501] Node: medic_2 is Alive! ✅
medic_2  | 2024-11-23 22:13:00 INFO     There's a new socket peer! 🤝 ✅
medic_1  | 2024-11-23 22:13:00 INFO     Sending [Connect]: Type: 0 {501: '172.25.125.3'} To: 20502 👆🔥
medic_2  | 2024-11-23 22:13:00 INFO     [502] Node: medic_3 is Alive! ✅
medic_2  | 2024-11-23 22:13:00 INFO     [502] Trying to connect to 503 🔄
medic_2  | 2024-11-23 22:13:00 INFO     [502] Node: medic_3 is Alive! ✅
medic_3  | 2024-11-23 22:13:00 INFO     There's a new socket peer! 🤝 ✅
medic_2  | 2024-11-23 22:13:00 INFO     Sending [Connect]: Type: 0 {502: '172.25.125.4'} To: 20503 👆🔥
medic_3  | 2024-11-23 22:13:01 INFO     [503] Node: medic_0 is Alive! ✅
medic_3  | 2024-11-23 22:13:01 INFO     [503] Trying to connect to 500 🔄
medic_3  | 2024-11-23 22:13:01 INFO     [503] Node: medic_0 is Alive! ✅
medic_0  | 2024-11-23 22:13:01 INFO     There's a new socket peer! 🤝 ✅
medic_3  | 2024-11-23 22:13:01 INFO     Sending [Connect]: Type: 0 {503: '172.25.125.5'} To: 20500 👆🔥
medic_2  | 2024-11-23 22:13:03 INFO     Error action: recv_number_1_byte | result: fail | 👆
medic_2  | 2024-11-23 22:13:03 INFO     [502] Current skt Peer is broken 👆 we will have a new! ✅
medic_1 exited with code 137
```

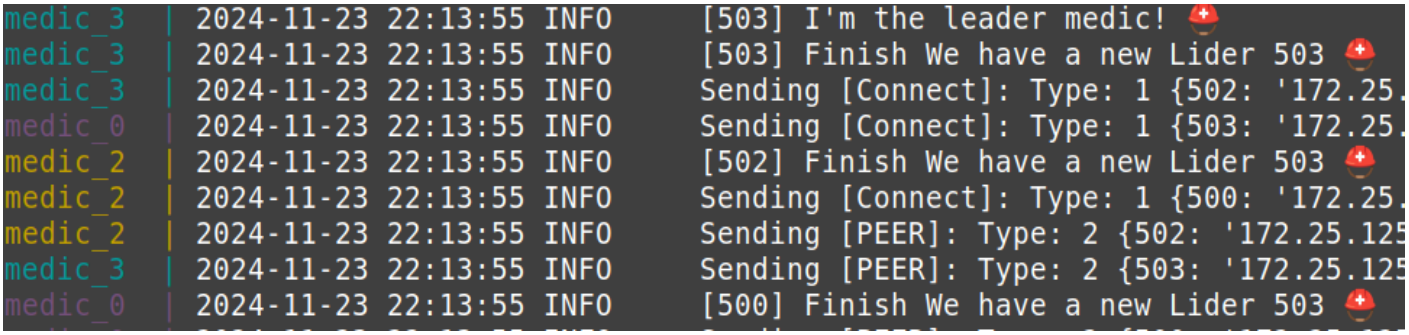
Acá vemos que **medic_0 detectó** un timeout del mensaje ELECTION {500: “172.25.125.2”}, luego chequea el estado de medic_1, detecta que **su siguiente esta conectado a un medico** caido, seteara el socket connect a None e interara conectarse con el vecino disponible. En este caso se conecta al **medic_2** y finalmente envía el mensaje:

```
medic_0  | 2024-11-23 22:13:25 INFO     [500] Connect Timeout to get a ack! from 501 Type: 0 {500: '172.25.125.2'}
 We try with the next! 🔕 501 👆
medic_0  | 2024-11-23 22:13:30 ERROR    [500] -> Node: [medic_1] Can't connect (Resolve DNS): [Errno -3] Try again
👆
medic_2  | 2024-11-23 22:13:30 INFO     There's a new socket peer! 🤝 ✅
medic_0  | 2024-11-23 22:13:30 INFO     [500] This Node medic_1 is dead, Let's skt connect to None
medic_0  | 2024-11-23 22:13:30 INFO     [500] Trying to connect to 502 🔄
medic_0  | 2024-11-23 22:13:30 INFO     [500] Node: medic_2 is Alive! ✅
medic_0  | 2024-11-23 22:13:30 INFO     Sending [Connect]: Type: 0 {500: '172.25.125.2'} To: 20502 👆🔥
```

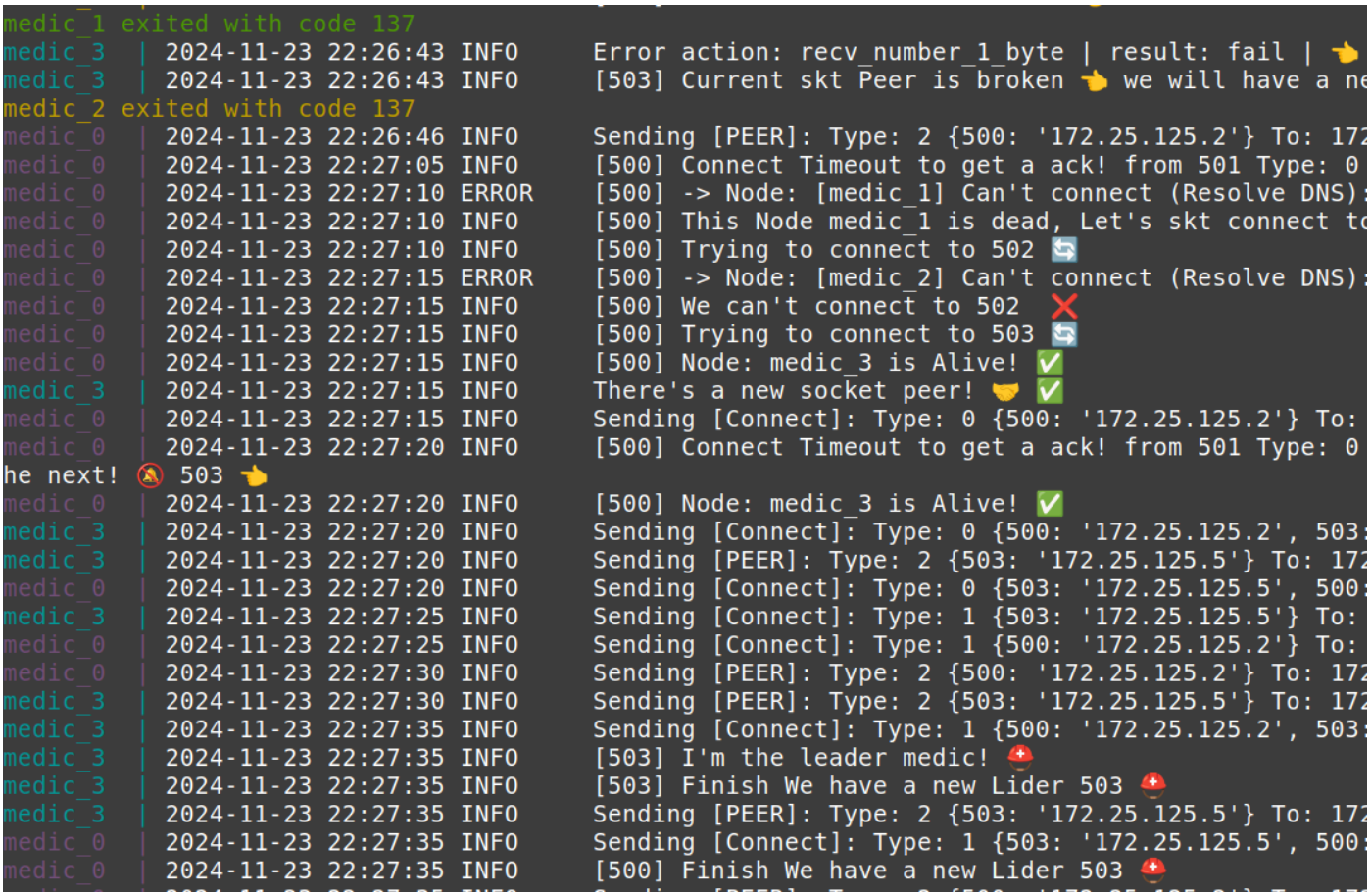
Para los proximos timeout, ya detectara que su siguiente esta conectado a un medico disponible e enviara el mensaje mucho mas rapido.

```
medic_0  | 2024-11-23 22:13:35 INFO     [500] Connect Timeout to get a ack! from 501 Type: 0 {503: '172.25.125.5',
500: '172.25.125.2'}  We try with the next! 🔕 502 👆
medic_2  | 2024-11-23 22:13:35 INFO     Sending [Connect]: Type: 0 {500: '172.25.125.2', 502: '172.25.125.4'} To: 2
0503 👆🔥
medic_0  | 2024-11-23 22:13:35 INFO     [500] Node: medic_2 is Alive! ✅
medic_0  | 2024-11-23 22:13:35 INFO     Sending [Connect]: Type: 0 {503: '172.25.125.5', 500: '172.25.125.2'} To: 2
0502 👆🔥
```

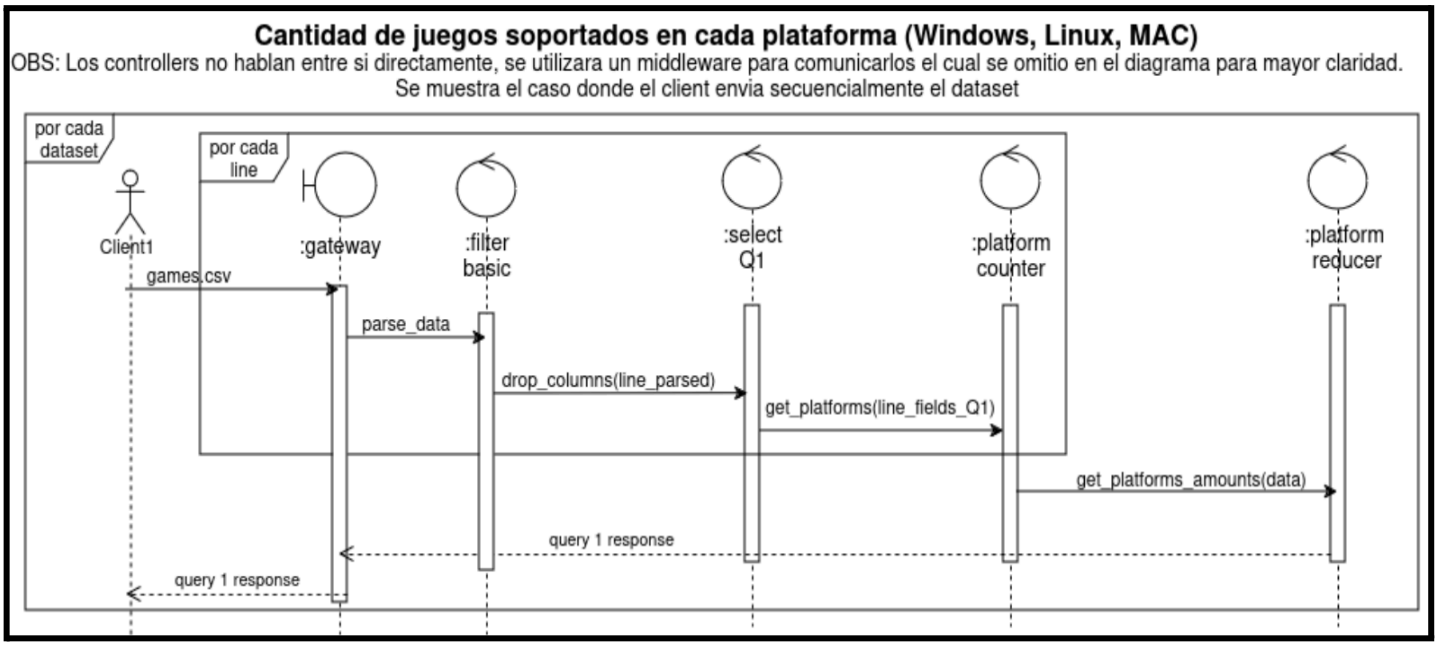
Finalmente todos detectaron el líder exitosamente:

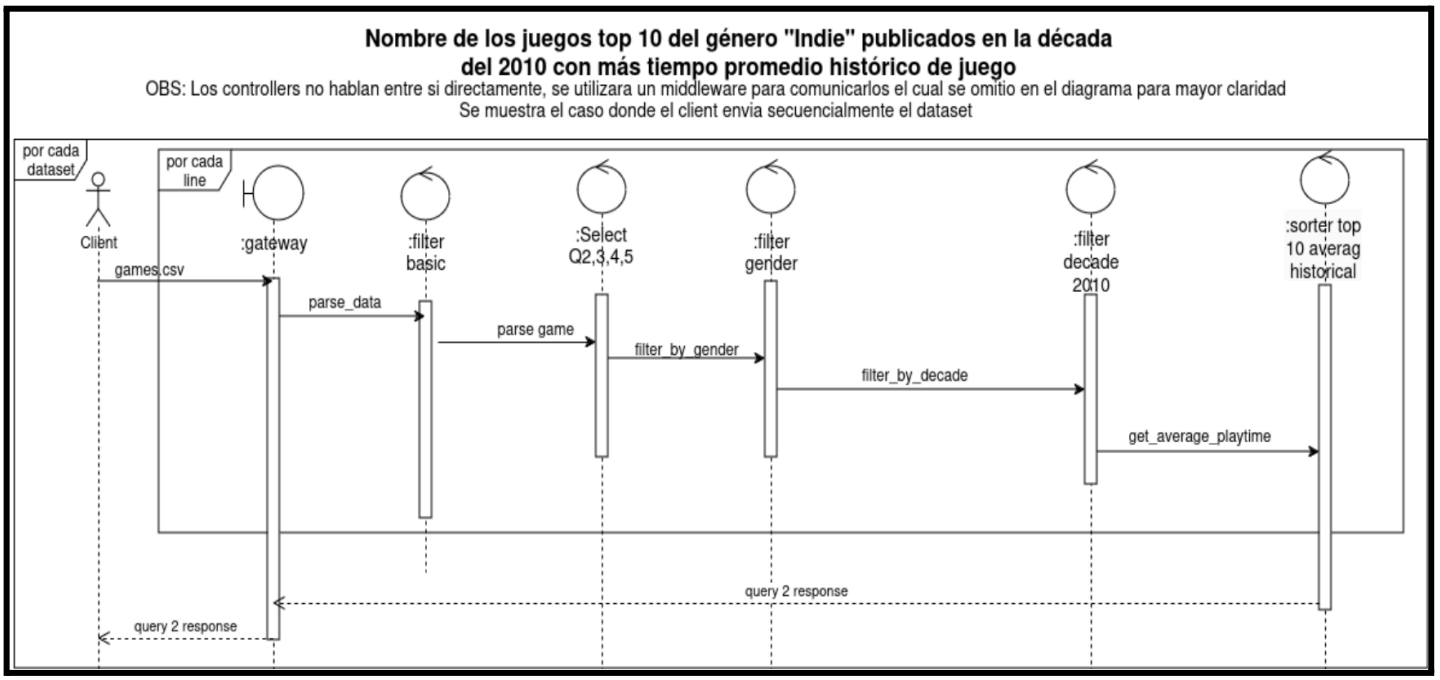


Con los medic_1 y medic_2 caídos, es tolerante a fallas y se logra definir un líder.



2. **Vista de procesos (Vista Dinámica):**

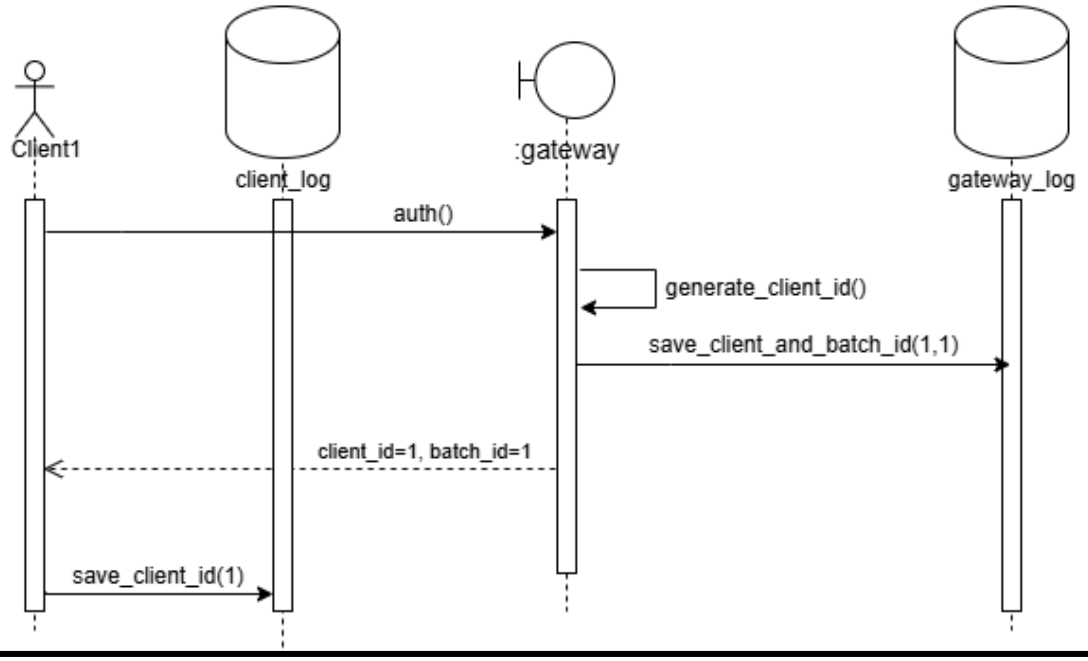




A continuación mostraremos cómo el cliente se recupera en caso de caerse para seguir desde el lugar donde había quedado. Se puede ver en la primera imagen como un cliente que corre por primera vez se comunica con el gateway el cual le asigna un identificador y el cliente lo guarda en su log. En la segunda imagen se puede ver un cliente que ya había corrido previamente pero sufrió una caída y al recuperarse lee su archivo de log para ver qué identificador le había asignado el gateway y así continuar con el procesamiento pendiente.

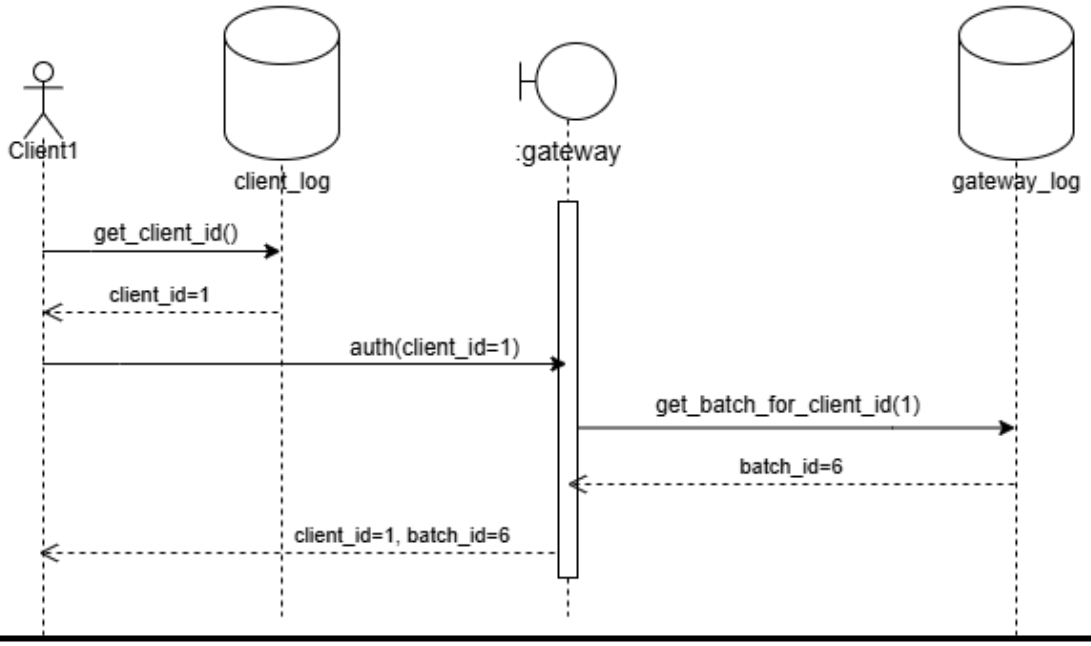
Autenticacion de un Cliente nuevo con el Gateway

Se puede ver como un cliente nuevo se intenta autenticar con el gateway el cual genera el client_id=1 para el mismo y batch_id=1 ya que aun no envio nada



Reautenticación de un cliente que ya existía con el Gateway

Se puede ver como un cliente con id 1 que ya existia y se habia caido se intenta autenticar con el gateway nuevamente el cual le dice hasta que batch id pudo procesar



Recuperación para los nodos stateless:

En caso de los nodos **stateless** como los **filter, select, etc** no tienen un mecanismo de recuperación y cuando estos fallen y **no hayan hecho el ACK** para el mensaje que procesaban en el instante de la falla, **volverán a reprocesar a partir de ese mensaje**.
Si un nodo reprocesa un mensaje que eventualmente ya enviaron a la siguiente cola no es un problema ya que luego en los nodos stateful se revisa los global counter procesados y en caso de ya tener procesado dicho mensaje se descarta

Recuperación para los nodos Stateful:

Cada nodo stateful tendrá:

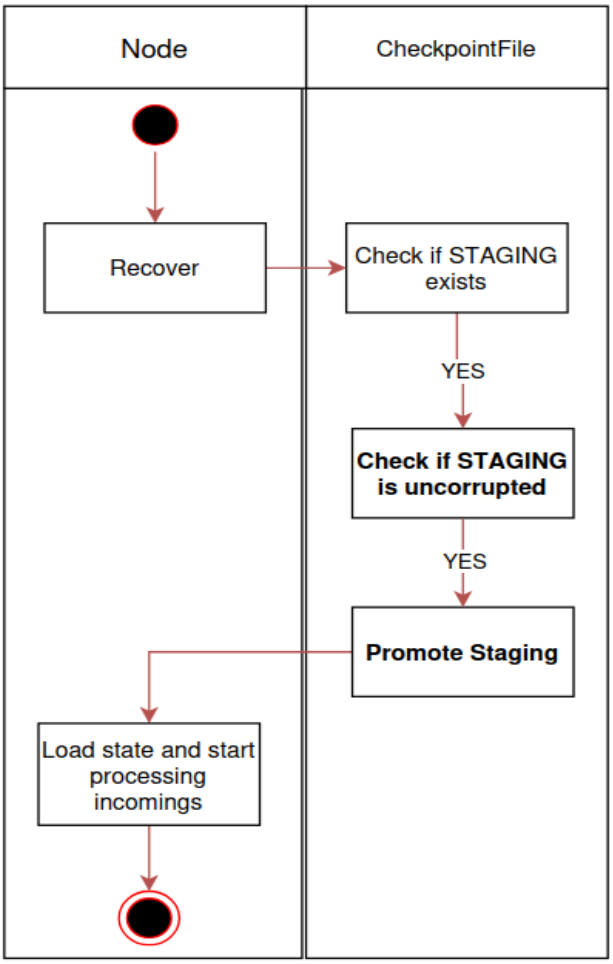
1) Un archivo binario asociado, donde escribirá el DTO que recibe en bytes. (**logs**), guarda hasta un máximo de 100 DTOS.
2) Un archivo binario asociado, que puede estar en estado **STAGING** o **PRODUCTION** (sufijo que se agrega al final **_PRD**, **_ STG**) ej: **monitor_PRD.** Este archivo almacena en bytes el **estado interno del nodo,** persistiendo así los resultados parciales.

Cuando un nodo **stateful falla (se cae)**, al momento de levantarse inicia su recuperación la cual consiste en:

1) En primer lugar el nodo chequea si tiene un **archivo STAGING no corrupto** (significa que el archivo asociado con el sufijo **_STG** contiene la marca "**UNCORRUPTED END**" al final del archivo).
2) En función de (1) se considerarán todos los casos posibles relacionados con la existencia o no del archivo staging (o si solo existe el archivo Production), así como su posible corrupción.

Caso 1:
Un nodo intenta recuperarse y existe su archivo **STAGING** el cual se encuentra NO CORRUPTO, entonces:

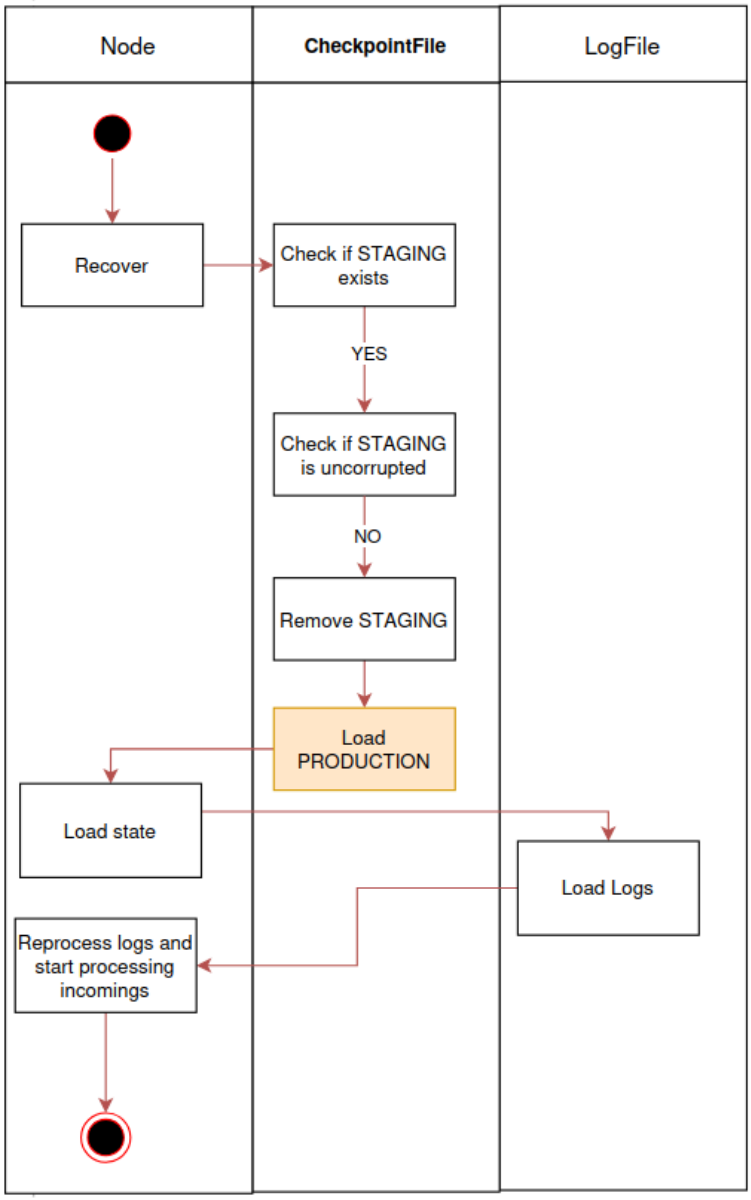
1) Se renombra el archivo STAGING reemplazando el sufijo _STG por _PRD.
2) Se lee el archivo prod y deserializando los bytes obteniendo el estado de la clase y lo cargamos.



Caso 2

Un nodo intenta recuperarse existe su archivo STAGING asociado pero esta corrupto :

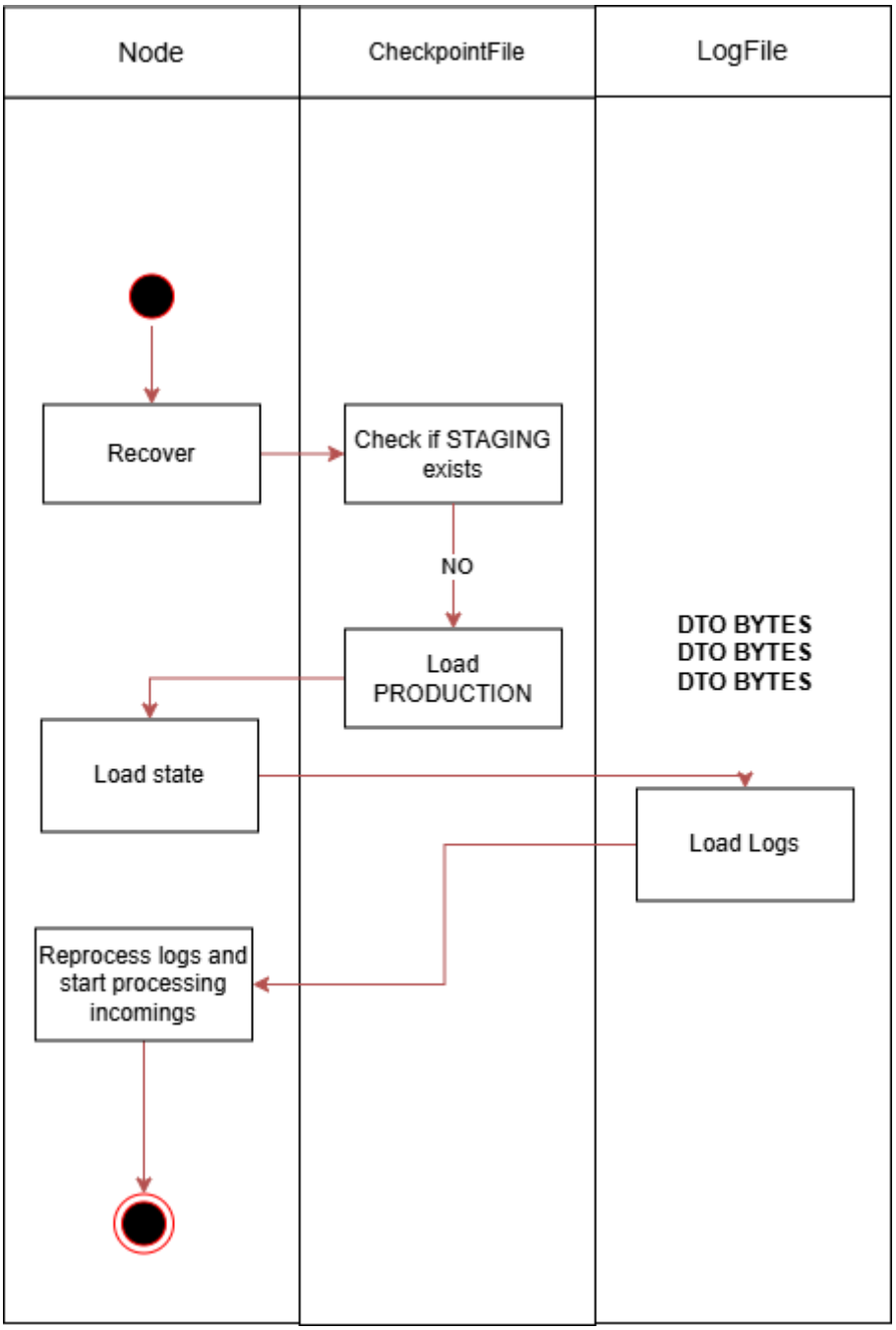
1) Esto puede deberse a que cuando se estaba grabando justo se cayó por lo que quedó con los datos por la mitad), **se elimina** el **archivo STAGING.**
2) Si existe el archivo de **PRODUCTION,** se lo utiliza el cual contiene el **estado de la clase valido hasta ese momento**, con esa información **se recupera** y continúa el procesamiento, usando el archivo de Log en caso de que exista.
3) Si no existe el archivo de PRODUCTION, se procesa el archivo de Log en caso de que exista.


Node
CheckpointFile
LogFile
Recover
Check if STAGING exists
YES
Check if STAGING is uncorrupted
NO
Remove STAGING
Load PRODUCTION
Load state
Load Logs
Reprocess logs and start processing incomings

Caso 3:

Un nodo intenta recuperarse cuando no existe su archivo STAGING**:**

1) Si existe su archivo PRODUCTION, se lo usa para recuperar el estado de la clase valido anterior y luego procesa el archivos de LOG en caso de que exista.
2) Si no existe su archivo Production, procesa el archivo de log en caso de que exista.



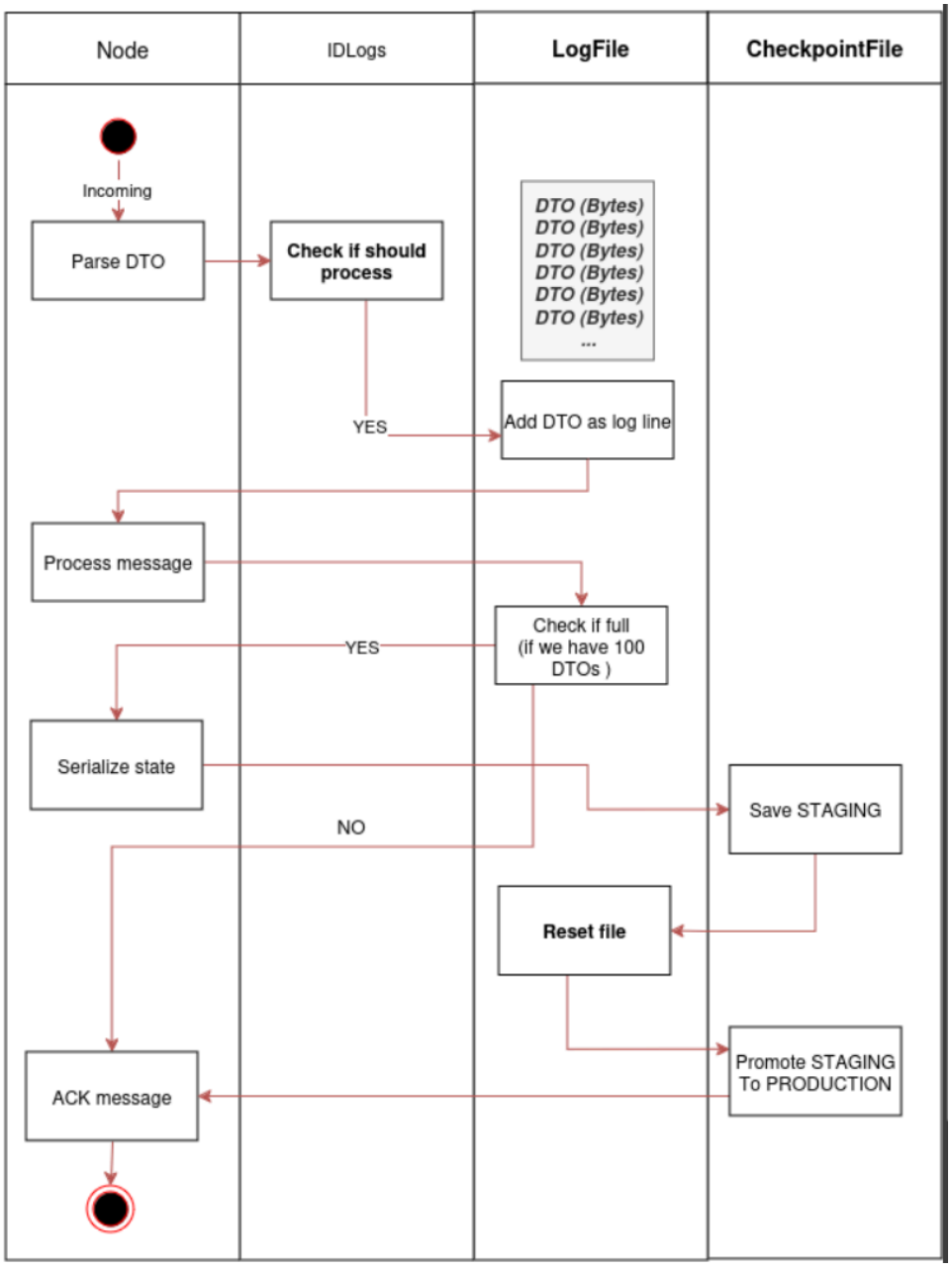
Caso 4:

Un nodo intenta recuperarse cuando no existe su STAGING y tampoco su PRODUCTION:

1) Este caso se podría dar cuando un nodo se cae y no llega a hacer su primer archivo STAGING (y por lo tanto su primer PRODUCTION).
2) Deberá volver a procesar todo nuevamente los DTOS en bytes en el archivo de log en caso de que exista .

Finalmente se deja un ejemplo de cómo sería todo el procesamiento para un nodo no escalable donde se puede ver lo siguiente:

1) Cuando un nodo stateful recibe un DTO antes de procesarlo chequea el **global counter del DTO** y chequea si está en su **id_list** (lista donde guarda los últimos 200 global counter procesados por el nodo).
    a) Si el global counter del DTO esta en **id_list** lo descarta (No procesamos duplicados).
    b) Si el global counter del DTO no está en id_list, lo procesamos.
2) Una vez el DTO procesado, se lo escribe en bytes al archivo de Log asociado.
3) Si el archivo de log alcanzó los 100 registros entonces se va a hacer un checkpoint se guarda el estado de la clase (en bytes) en un archivo STAGING cuyo sufijo es _STG.
4) Luego se proceda a eliminar el archivo de Logs.
5) Por ultimo renombramos el archivo STAGING cambiando el sufijo _STG por _PRD obteniendo el archivo PRODUCTION.

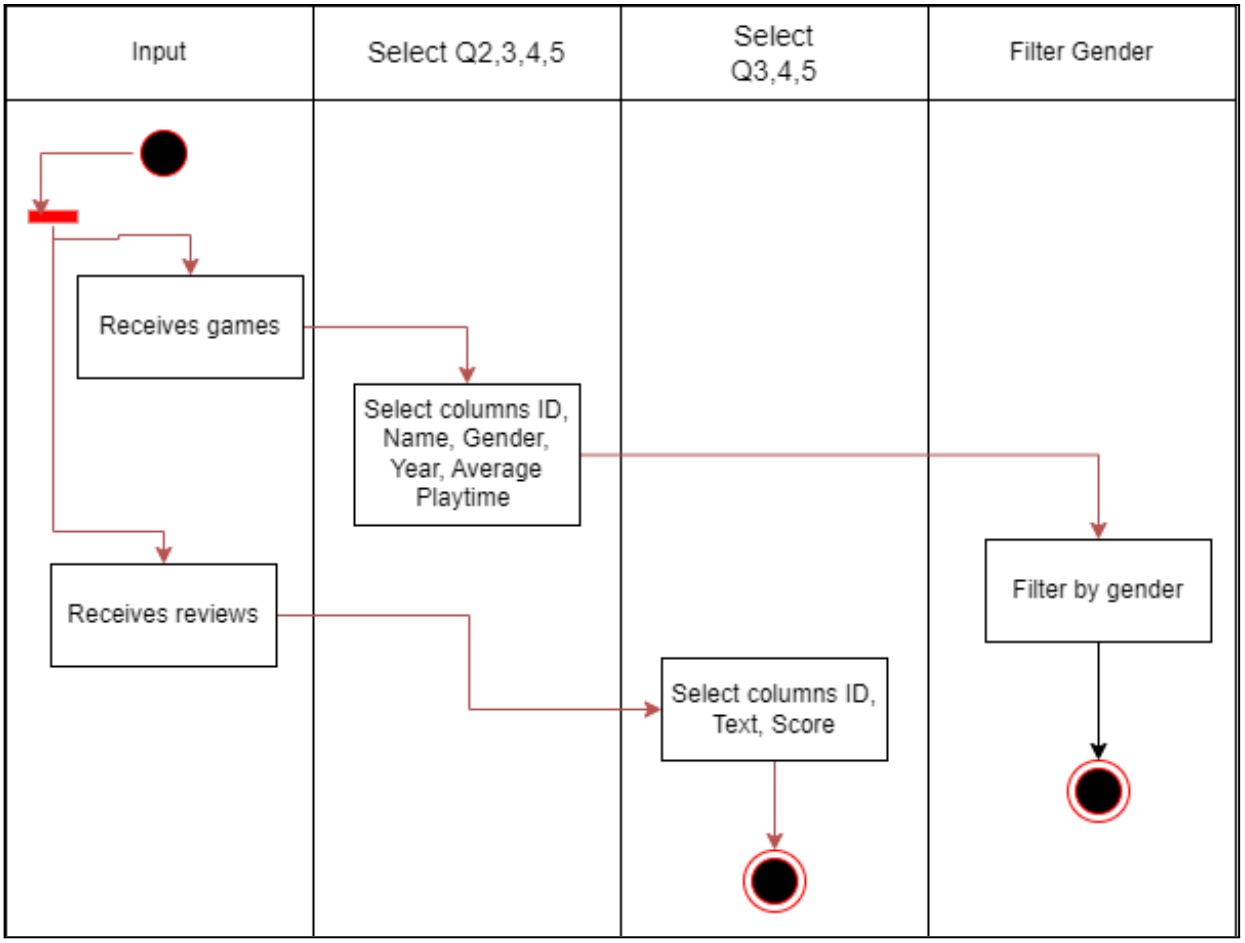


Nodos Stateful Escalables

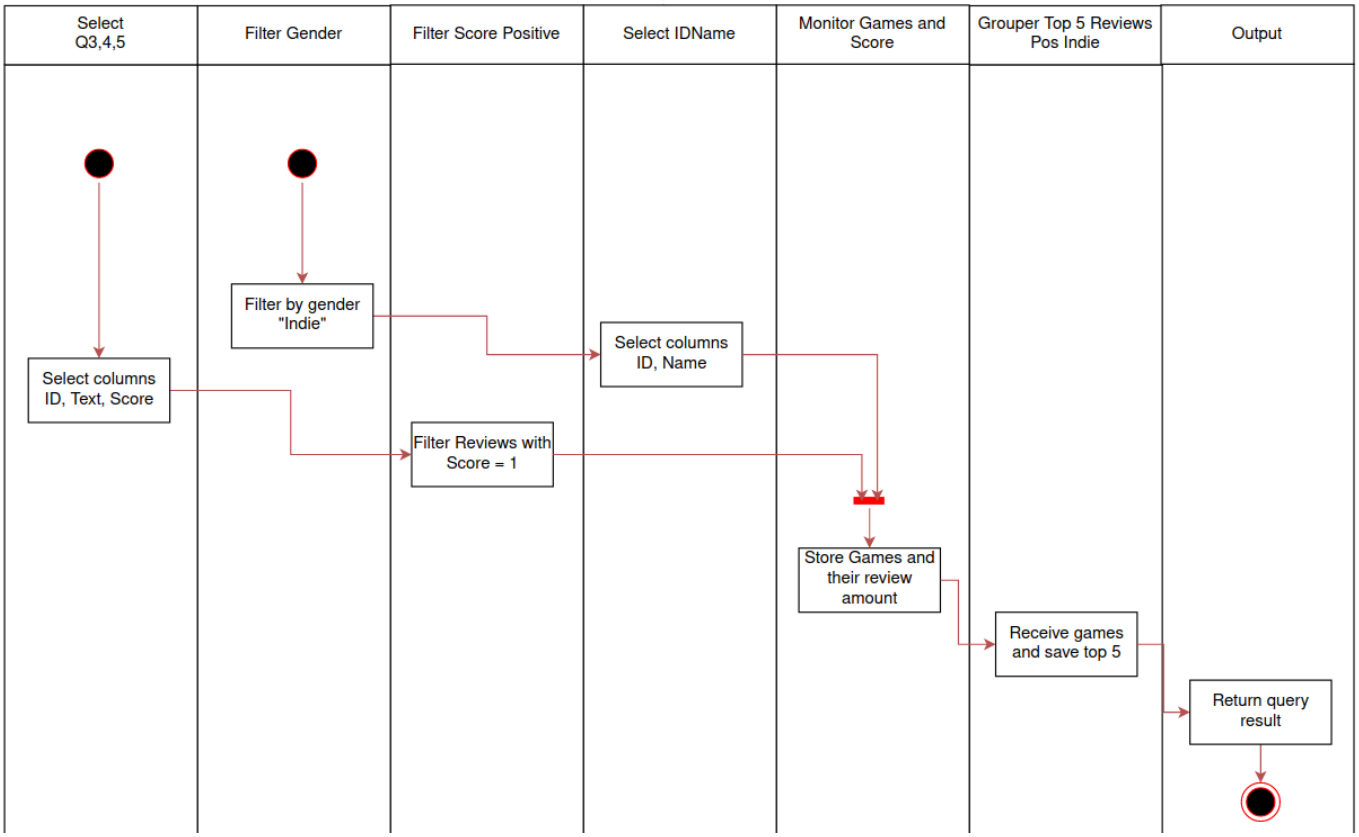
El único de este tipo de nodos en nuestro sistema es el platform counter. Para poder realizar todo el mecanismo antes descrito tenemos que garantizar que cada nodo procesa mensajes que no son procesados por otros nodos (de modo que se evite reprocesamiento en caso de caídas) y también se pueden deduplicar mensajes. Para

ello en este tipo de nodos se realiza una partición de datos. El nodo anterior debe saber la cantidad de nodos escalados y realizará la operación módulo con dicha cantidad, enviando el mensaje con la routing key resultante. Cada nodo escalado se bindea solo a una de dichas routing keys.

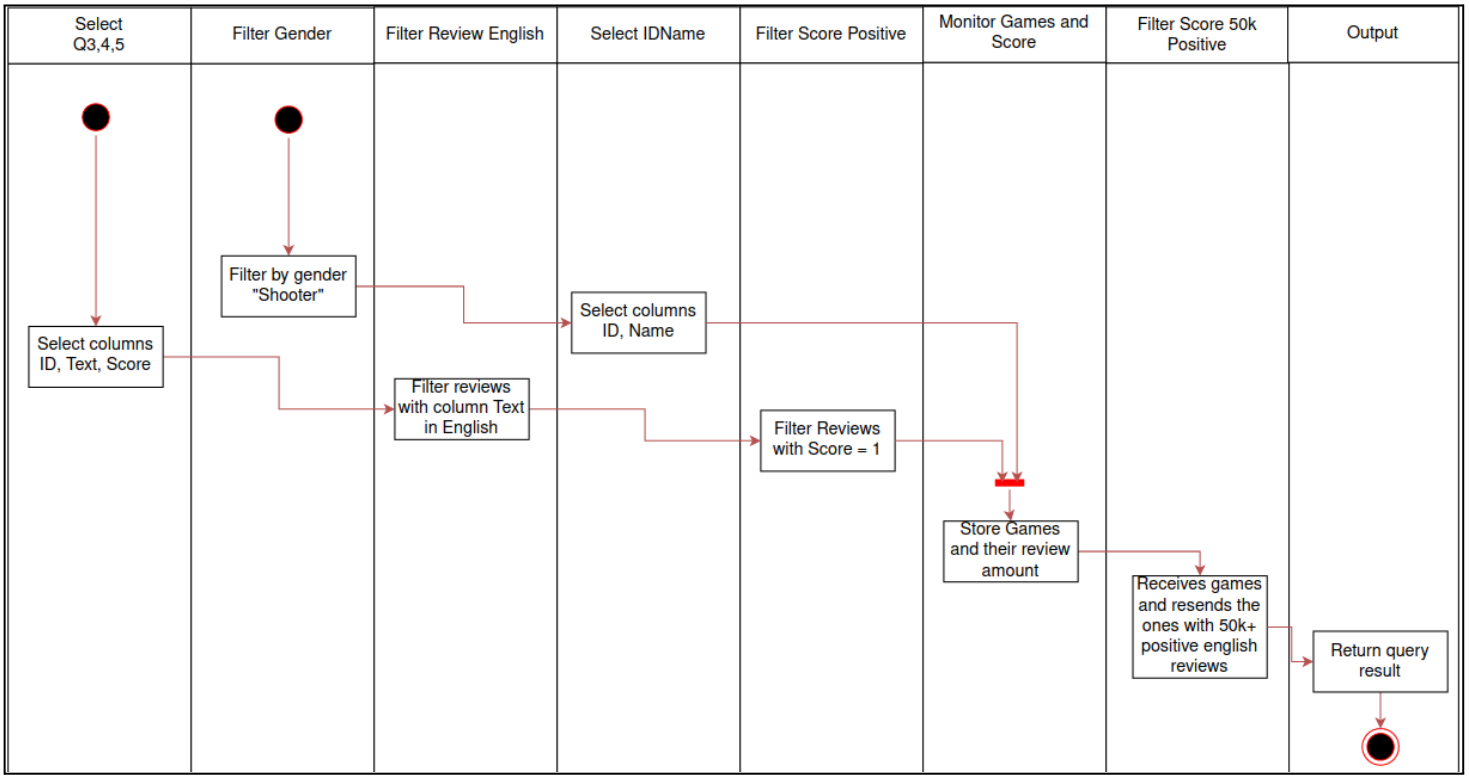
Diagrama Inicial que comparten las Q3, Q4 y Q5:



Q3:Nombre de los juegos top 5 del género "Indie" con más reseñas positivas
(Observar antes el diagrama inicial de la q3)


Select Q3,4,5
Filter Gender
Filter Score Positive
Select IDName
Monitor Games and Score
Grouper Top 5 Reviews Pos Indie
Output
Filter by gender "Indie"
Select columns ID, Text, Score
Filter Reviews with Score = 1
Select columns ID, Name
Store Games and their review amount
Receive games and save top 5
Return query result

Q4:Nombre de juegos del género "shooter" con más de 50.000 reseñas positivas en idioma inglés

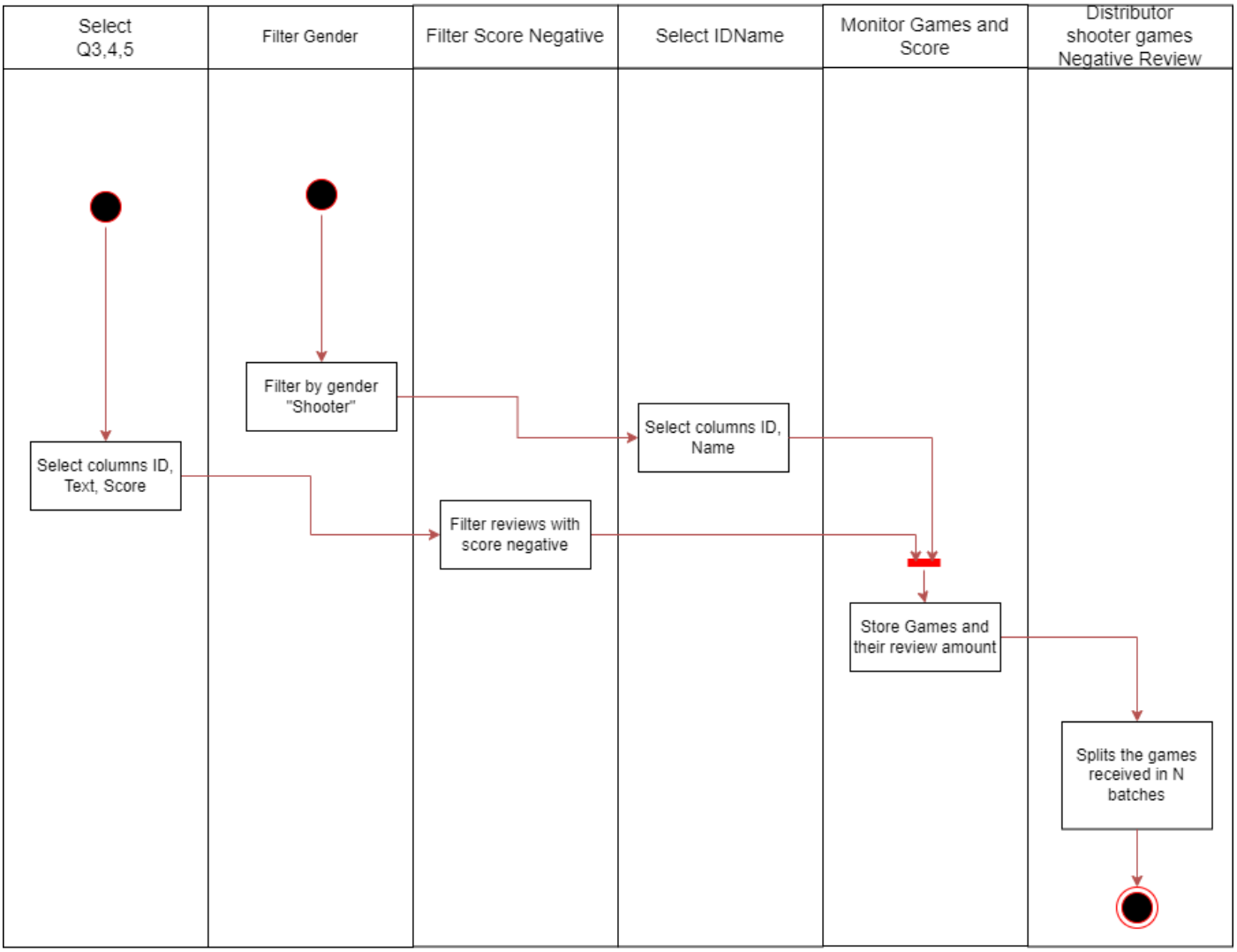


Lo que se intenta mostrar acá es cómo el sistema orquesta diversas fases de procesamiento en paralelo, lo que permite identificar interdependencias y potenciales puntos de optimización o mejora.
Por ejemplo, los filtros aplicados sobre géneros y reseñas de juegos destacan el enfoque específico en ciertos tipos de datos, lo que sugiere la posibilidad de mejorar la eficiencia ajustando estos filtros o introduciendo paralelismo en la evaluación de distintos juegos.

El sistema es también fácilmente adaptable, ya que se podrían agregar nuevos filtros o parsers sin perturbar significativamente la arquitectura. Además, la separación explícita de responsabilidades entre módulos garantiza que las modificaciones puedan hacerse en segmentos aislados, lo cual es fundamental para la escalabilidad y mantenimiento a largo plazo del sistema.

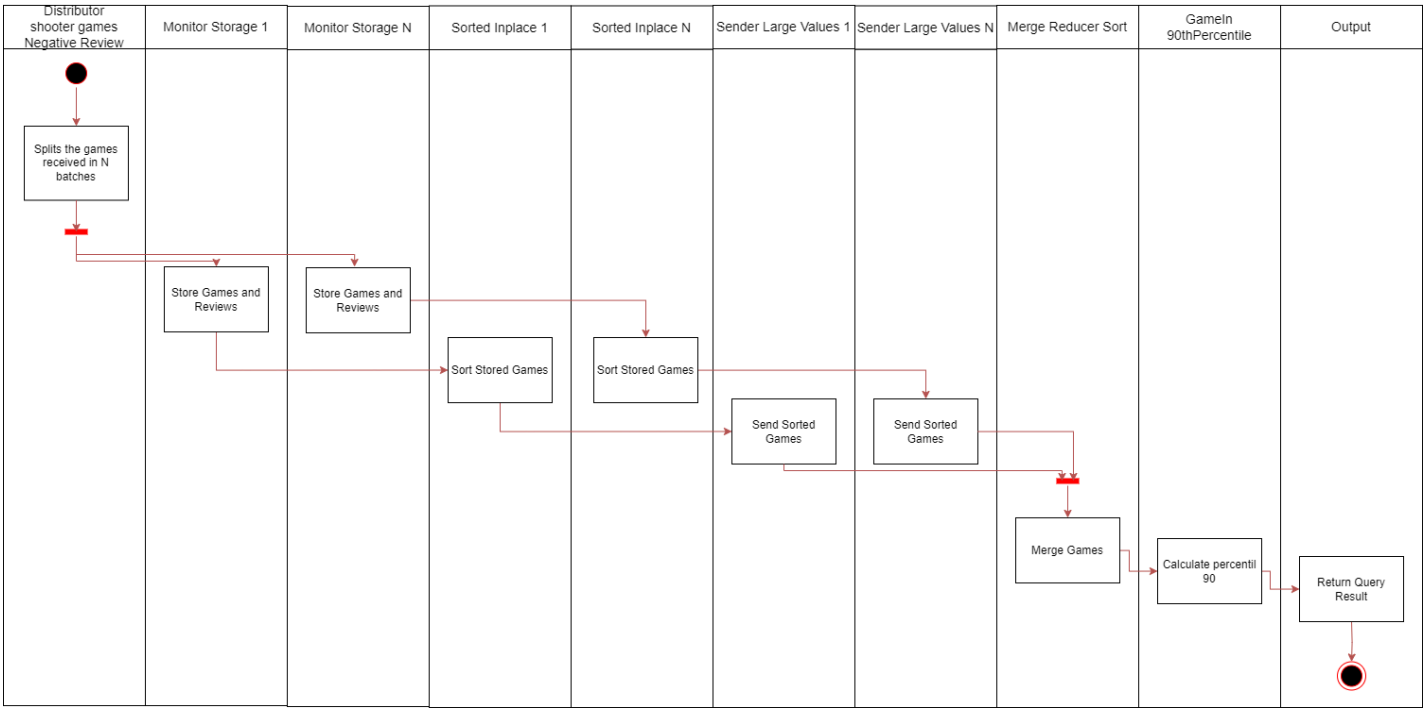
Q5:Nombre de juegos del género "shooter" dentro del percentil 90 en cantidad de reseñas negativas:

Q5-Parte1)



En esta primera parte mostramos el procesamiento general para la query 5 y como en determinado momento los juegos y las reviews convergen en un monitor que almacenará ambos datos.

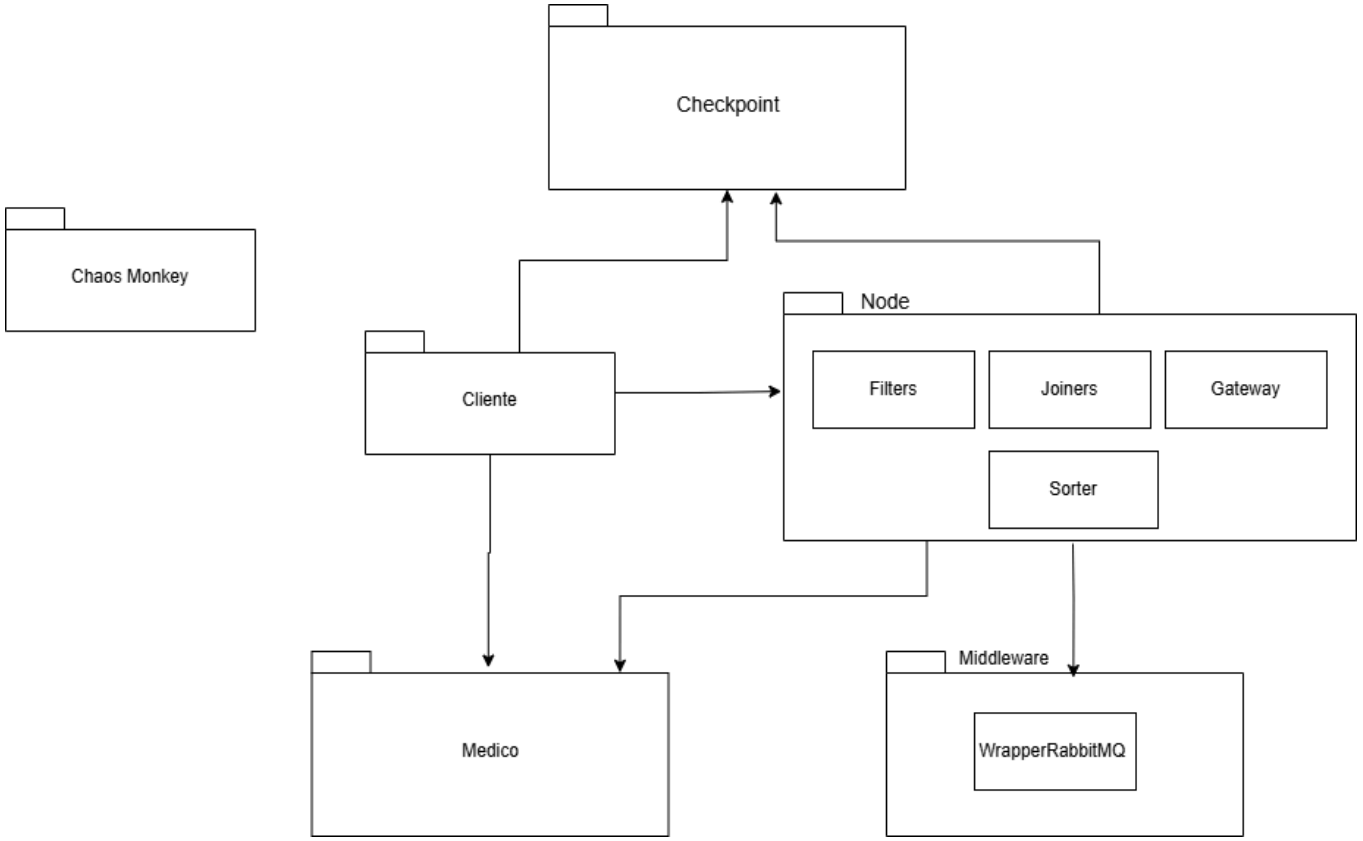
Q5-Parte2)



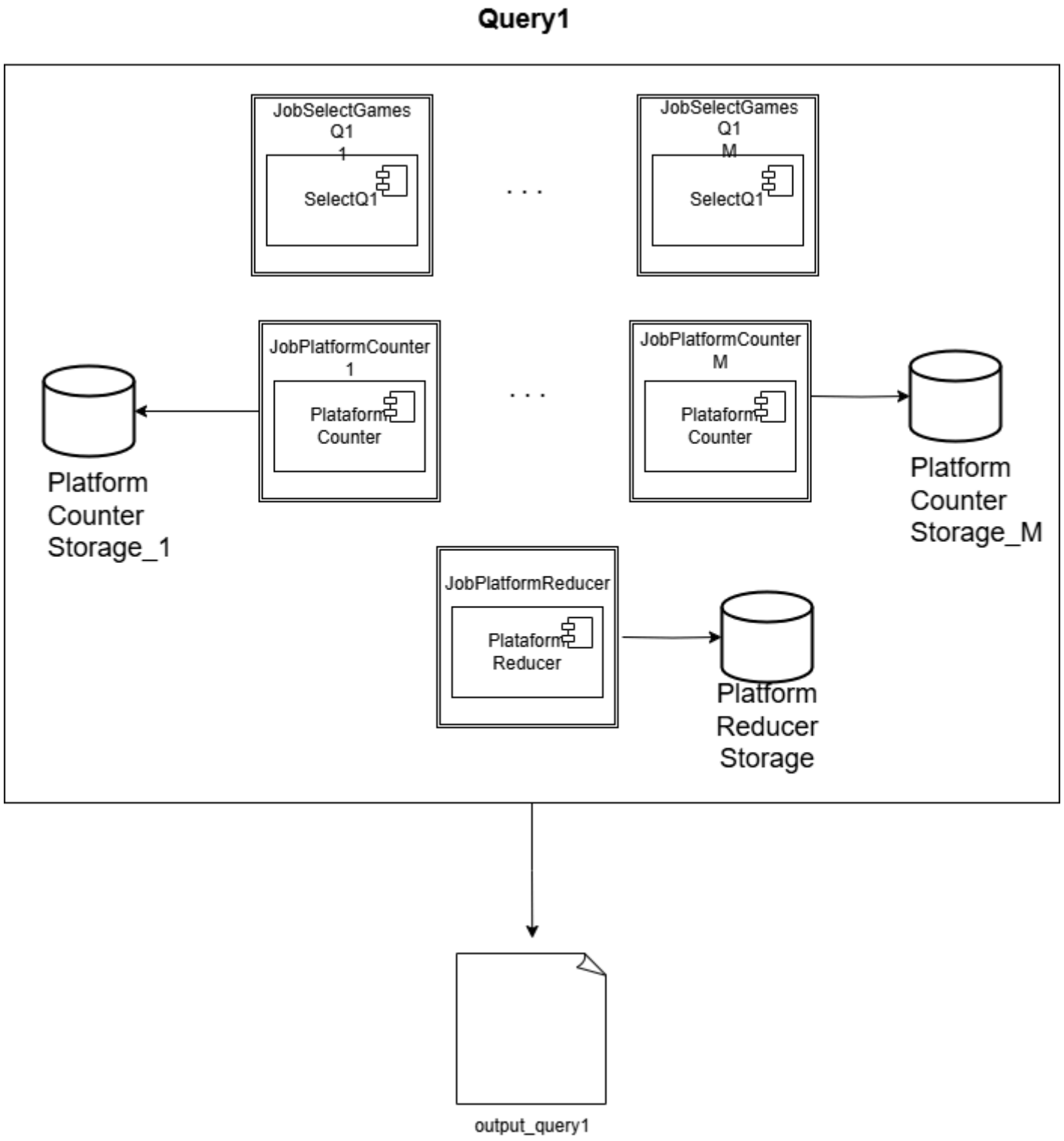
El diagrama intenta mostrar cómo un sistema distribuido o pipeline paralelo maneja el procesamiento de grandes volúmenes de datos relacionados con juegos shooter con reseñas negativas, destacando los pasos de partición, almacenamiento, ordenación, envío, merge, y cálculo

final del percentil. Se simplifica a 2 entidades “1” y “N” para indicar que tiene de 1 a N. A la hora de escalar se pueden agregar más cantidad de estas entidades.

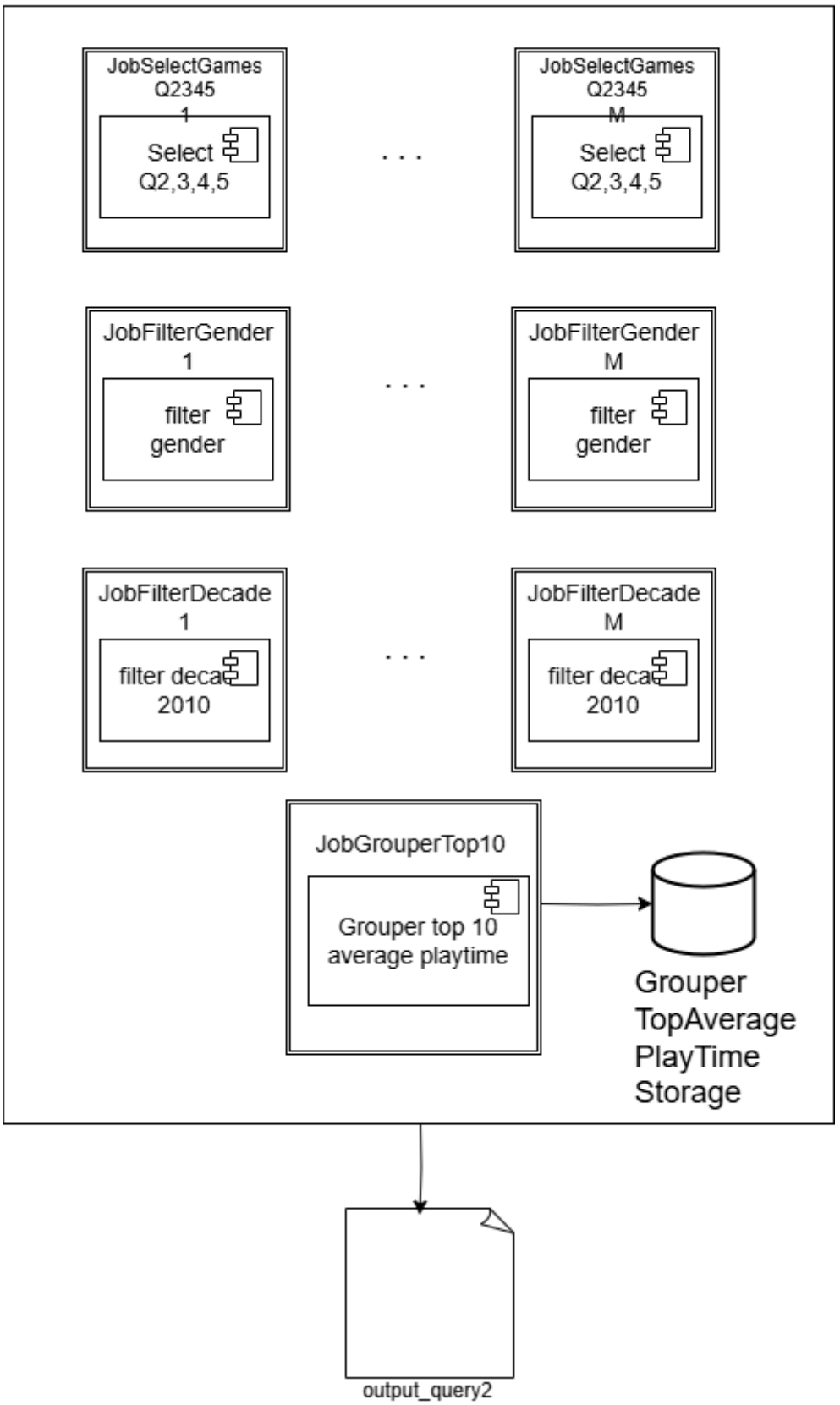
3. **Vista de desarrollo (Vista de componentes, Vista de implementación):**

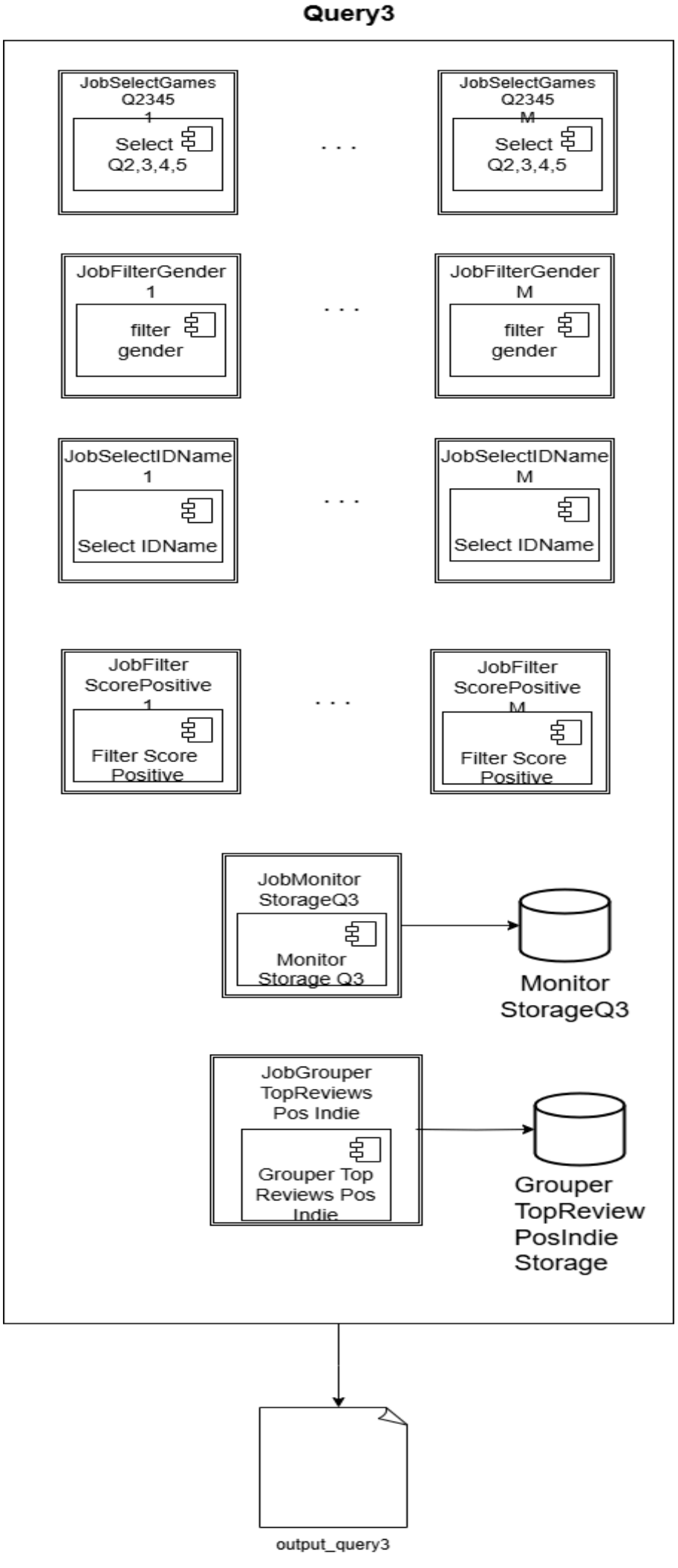


4. **Vista Física (Despliegue):**

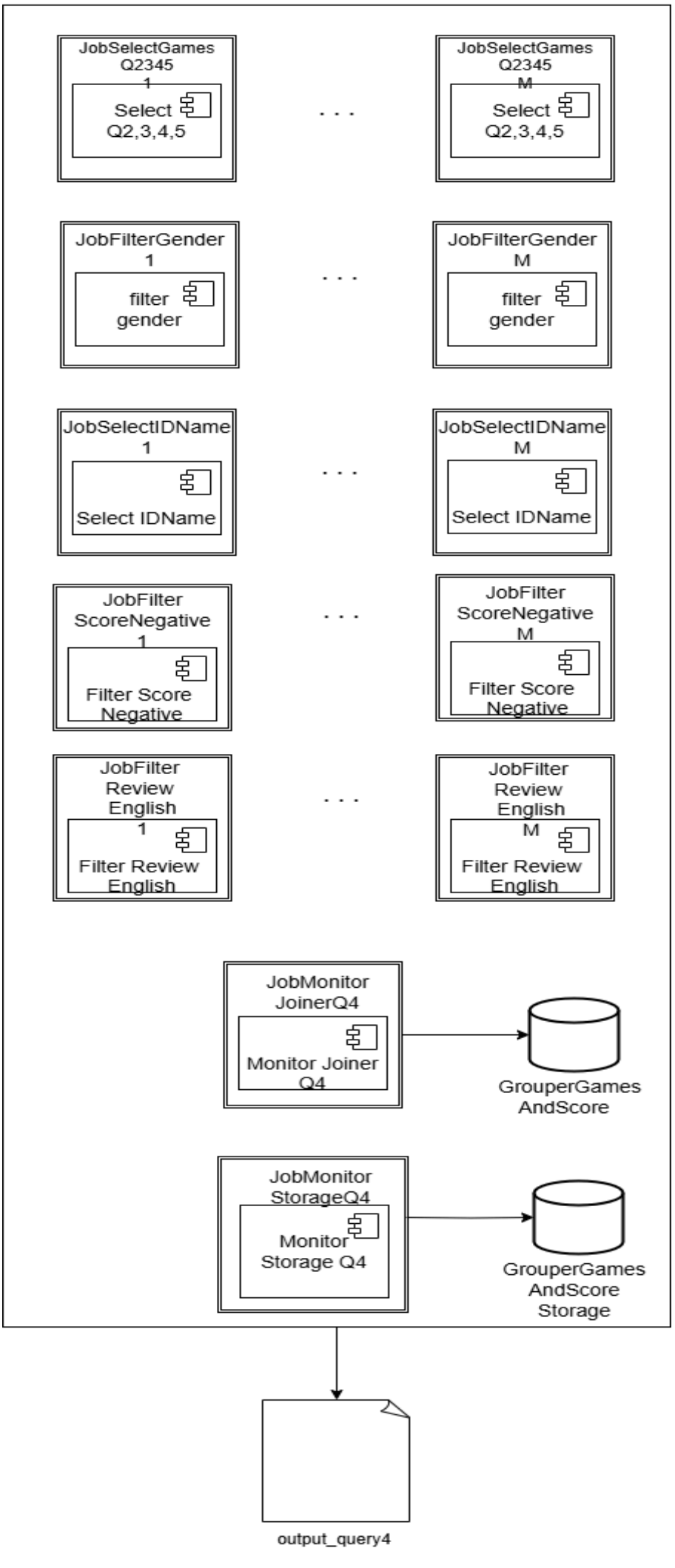


Query2

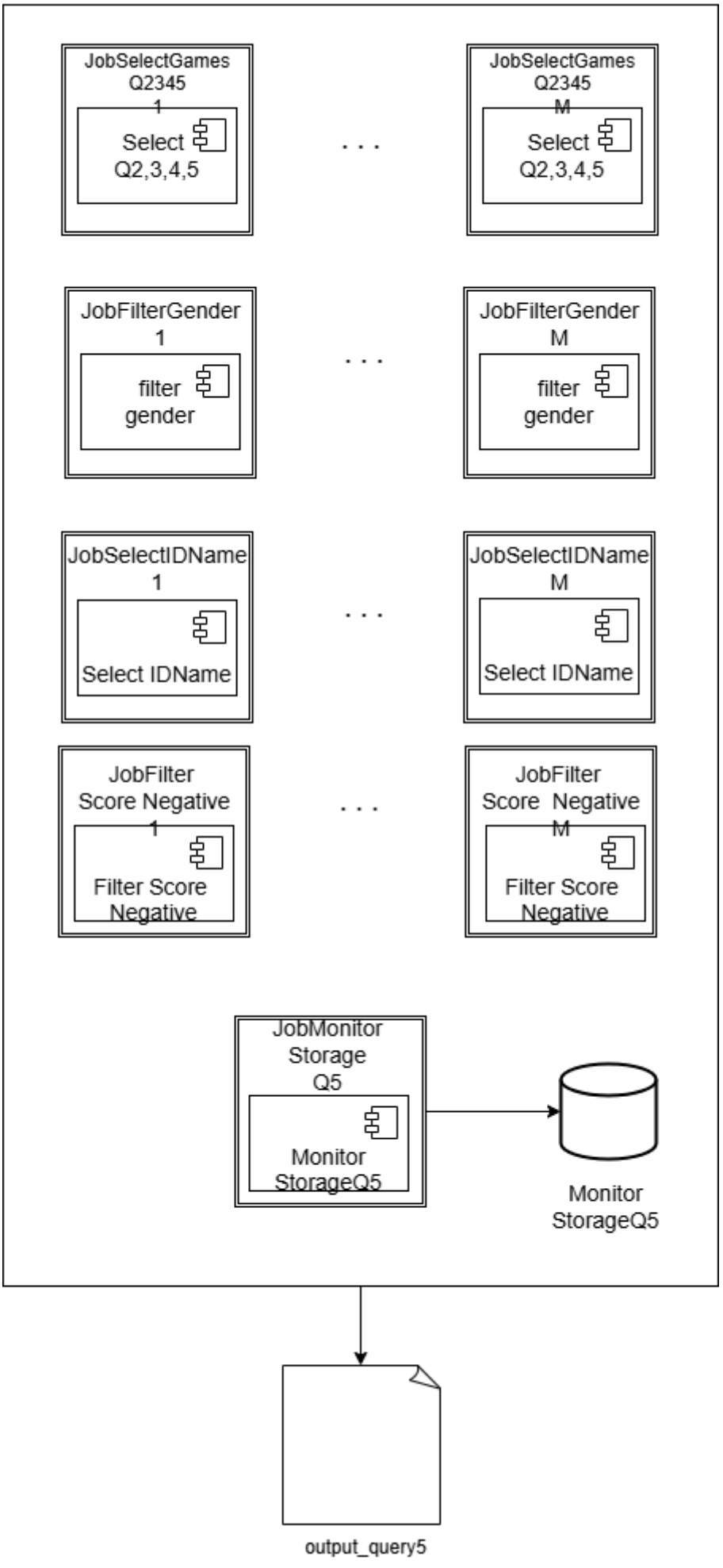

JobSelectGames Q2345 1
Select Q2,3,4,5
. . .
JobSelectGames Q2345 M
Select Q2,3,4,5
JobFilterGender 1
filter gender
. . .
JobFilterGender M
filter gender
JobFilterDecade 1
filter deca 2010
. . .
JobFilterDecade M
filter deca 2010
JobGrouperTop10
Grouper top 10 average playtime
Grouper TopAverage PlayTime Storage
output_query2


Query3
JobSelectGames Q2345 1
Select Q2,3,4,5
. . .
JobSelectGames Q2345 M
Select Q2,3,4,5
JobFilterGender 1
filter gender
. . .
JobFilterGender M
filter gender
JobSelectIDName 1
Select IDName
. . .
JobSelectIDName M
Select IDName
JobFilter ScorePositive 1
Filter Score Positive
. . .
JobFilter ScorePositive M
Filter Score Positive
JobMonitor StorageQ3
Monitor Storage Q3
Monitor StorageQ3
JobGrouper TopReviews Pos Indie
Grouper Top Reviews Pos Indie
Grouper TopReview PosIndie Storage
output_query3

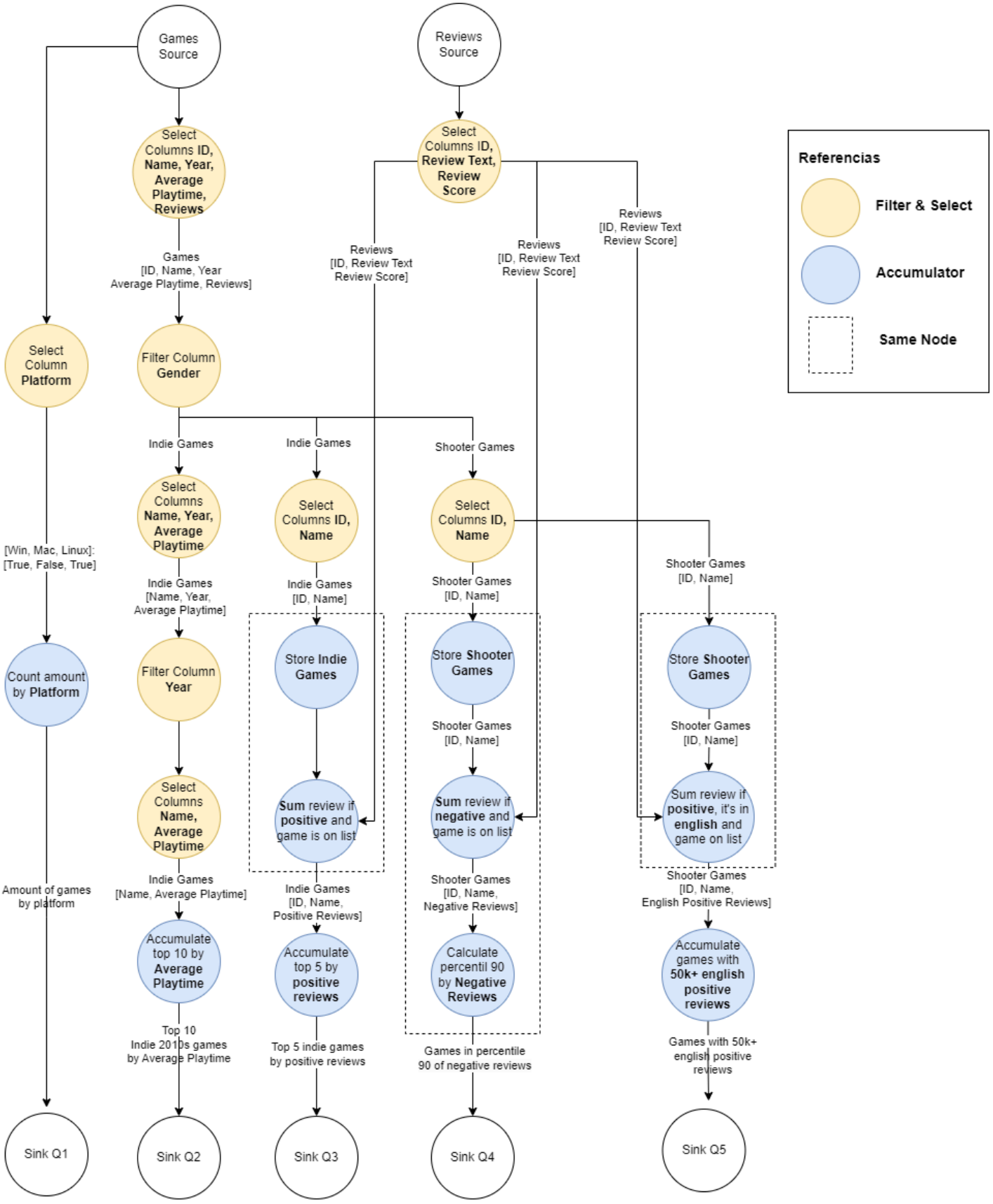
Query4


JobSelectGames Q2345 1
Select Q2,3,4,5
. . .
JobSelectGames Q2345 M
Select Q2,3,4,5
JobFilterGender 1
filter gender
. . .
JobFilterGender M
filter gender
JobSelectIDName 1
Select IDName
. . .
JobSelectIDName M
Select IDName
JobFilter ScoreNegative 1
Filter Score Negative
. . .
JobFilter ScoreNegative M
Filter Score Negative
JobFilter Review English 1
Filter Review English
. . .
JobFilter Review English M
Filter Review English
JobMonitor JoinerQ4
Monitor Joiner Q4
GrouperGames AndScore
JobMonitor StorageQ4
Monitor Storage Q4
GrouperGames AndScore Storage
output_query4

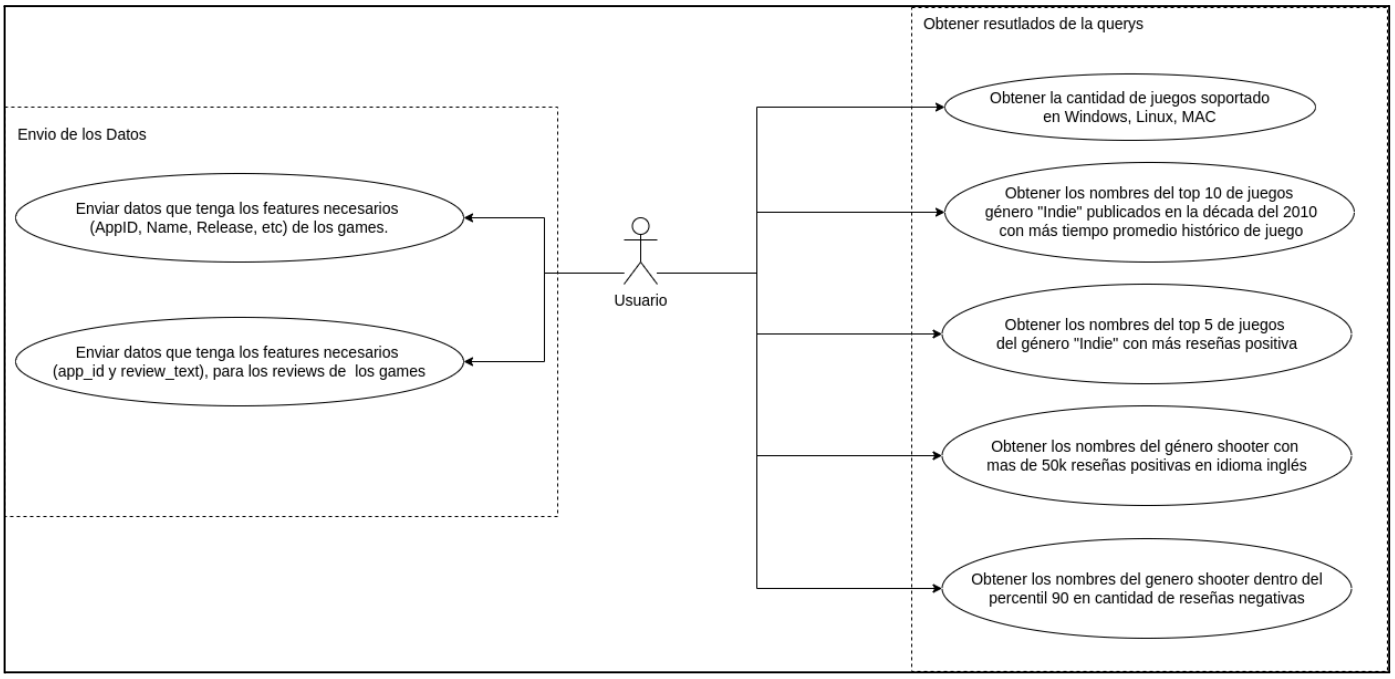
Query5


JobSelectGames Q2345 1
Select Q2,3,4,5
. . .
JobSelectGames Q2345 M
Select Q2,3,4,5
JobFilterGender 1
filter gender
. . .
JobFilterGender M
filter gender
JobSelectIDName 1
Select IDName
. . .
JobSelectIDName M
Select IDName
JobFilter Score Negative 1
Filter Score Negative
. . .
JobFilter Score Negative M
Filter Score Negative
JobMonitor Storage Q5
Monitor StorageQ5
Monitor StorageQ5
output_query5

5. DAG - Flujo de los datos



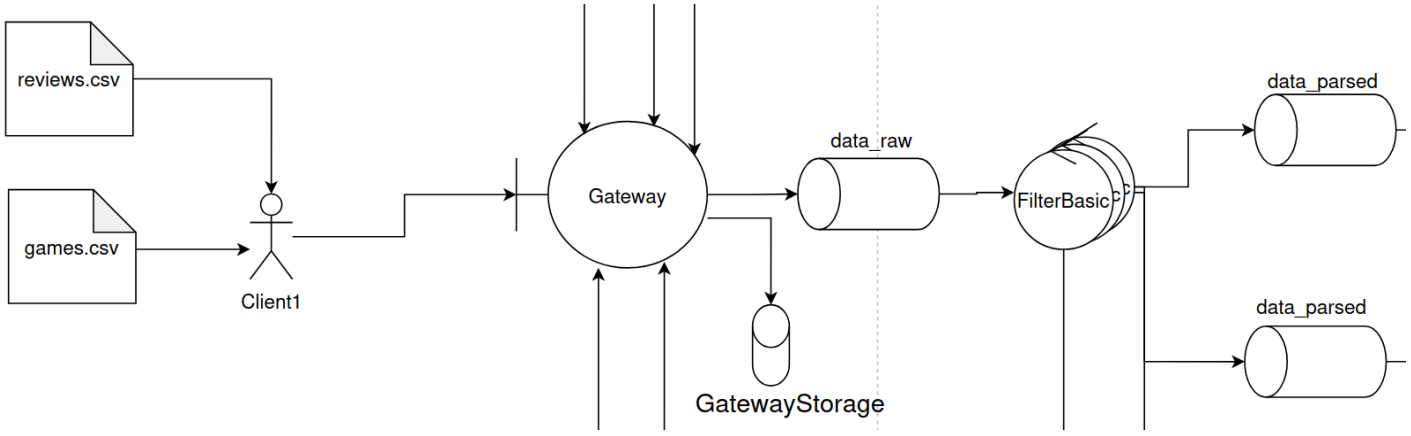
6. **Casos de uso (Escenarios):**



2. Diagrama de robustez:

Para definir los diagramas de robustez se priorizo el requisito no funcional **Escalabilidad** para lograrlo, los controllers se modelaron lo más granular posible, para poder detectar algún posible cuello de botella y escalarlo de forma particular. Para los diagramas de robustez se decidió hacer un diagrama unificado y un **diagrama de robustez por cada query y del Bootstrapping del sistema** comunicar más fácilmente el sistema distribuido:

Bootstrapping del sistema



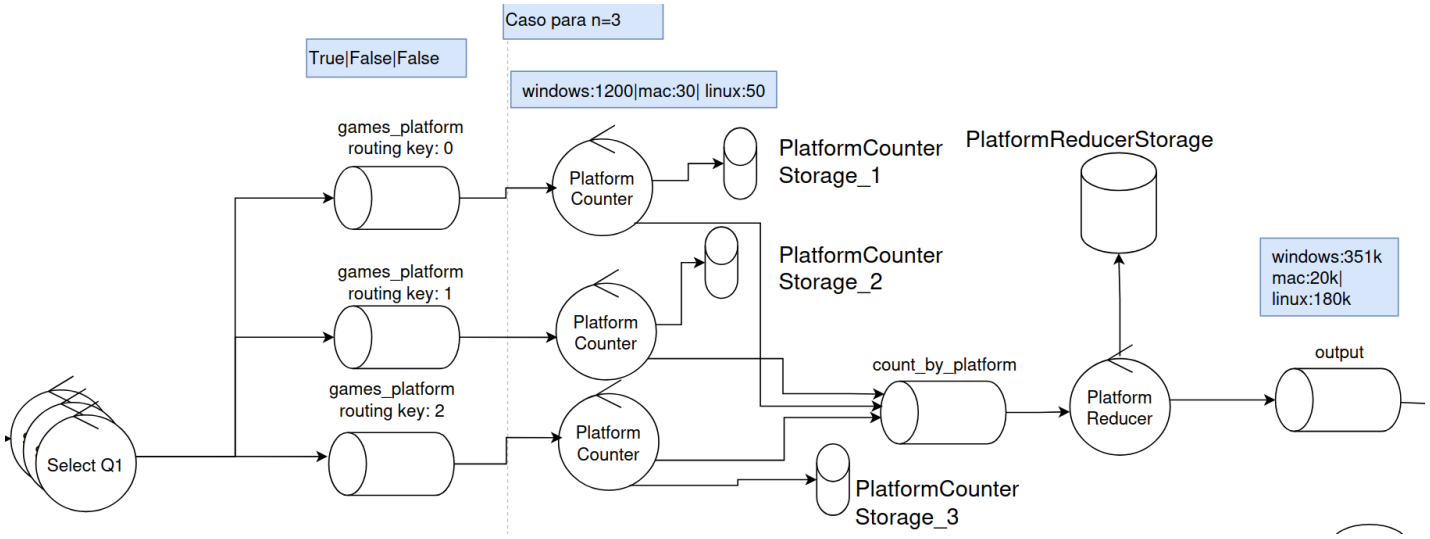
El cliente usando la consola, envía los datos de los csv (review.csv y games.csv) para poder obtener la respuestas de las querys. La puerta de entrada al sistema será el gateway

Diccionario de Elementos:

Diccionario de elementos		
Boundary	Gateway	Recibe DTOs, les agrega un global counter y los reenvía.
Actor	Client1	Cliente que quiere obtener el resultado de las consultas.
Entity	reviews.csv	Debe contener al menos datos como app_id, review_text y review_score.
	games.csv	Debe contener al menos datos como ,ID, Name, Gender, Year, Average Playtime de un game.
	GatewayStorage	Almacena los resultados eofs de los clientes, ultimo batche de los clientes y las respuestas a los clientes.
Controller	FilterBasic	Selecciona los campos basicos que se usaran en todas las queries, ya sea del tipo game o review.

Diagrama de robustez de la query1:

Query1) Cantidad de juegos soportados en cada plataforma (Windows, MAC, Linux).

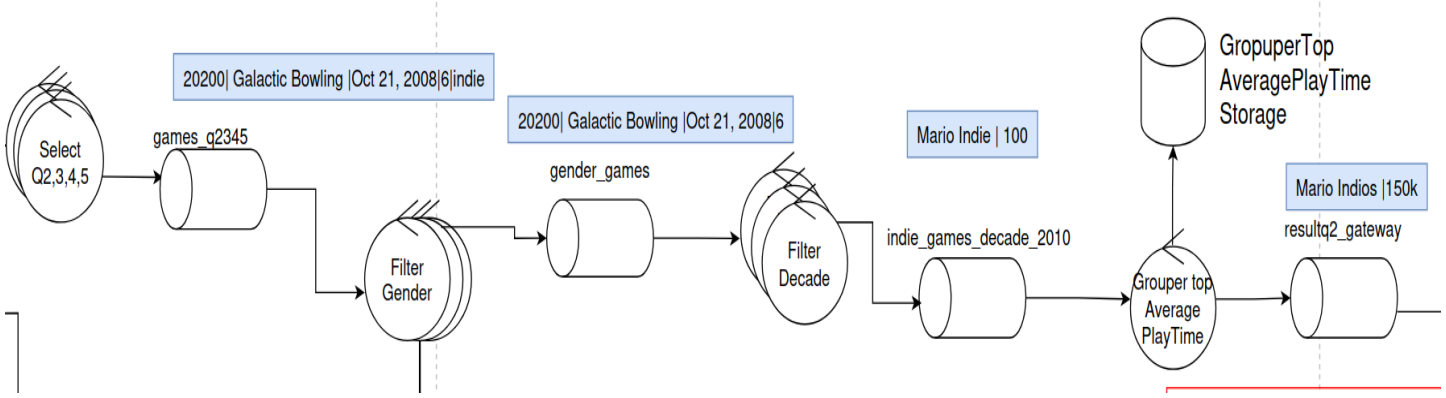


Tenemos **N** nodos platform counter que toman un mensaje “True | False | False” y lo mapean a “Windos:1, Mac:0, Linux:0” para luego enviarlo al nodo reducer.

Diccionario de elementos		
Entity	PlatformCounter Storage_n	Contiene la cantidad de juegos compatibles en windows, linux y mac
	Platform ReducerStorage	Contiene el total actual acumulado de la cantidad de juegos en windows, linux y mac
Controller	Select Q1	Recibe juegos, les quita todas las columnas excepto Windows, Linux y Mac

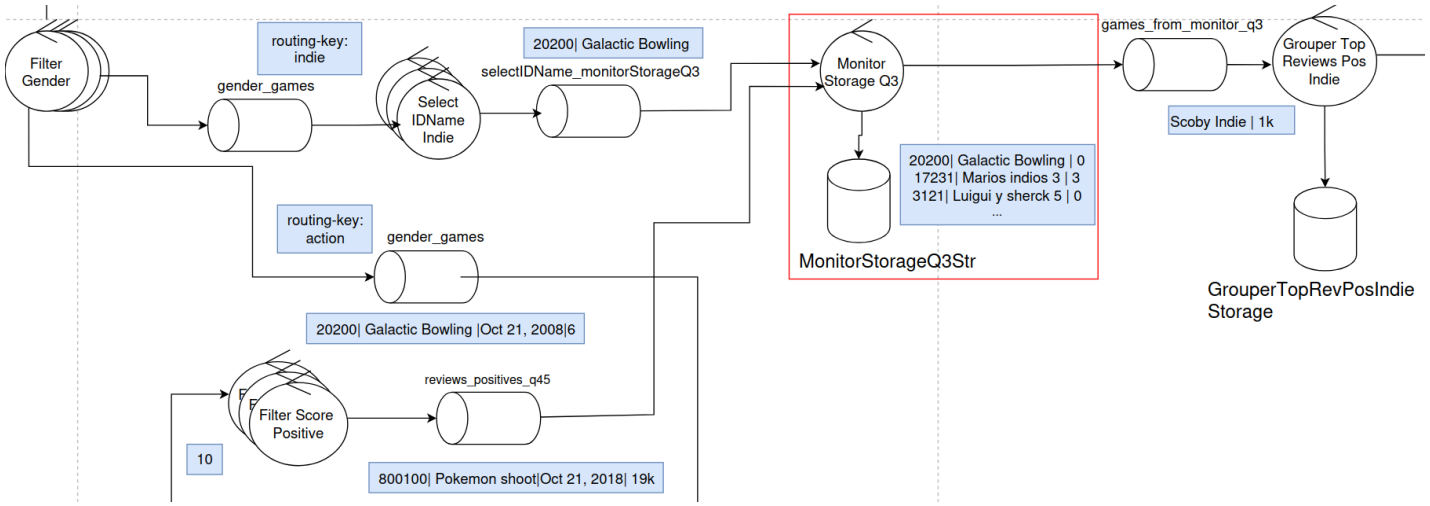
	Platform Counter	Recibe columnas del tipo Windows,Mac,Linux y va agrupandolas hasta recibir un EOF.
	Platform Reducer	Recibe listas de cantidad de juegos por plataforma agrupada por los distintos counter y las suma.

Query 2) Nombre de los juegos top10 del género "Indie" publicados en la década del 2010 con más tiempo promedio histórico de juego



Diccionario de elementos		
Entity	GrouperTop Average PlaytimeStorage	Almacena como máximo 10 juegos, aquellos con mayor average playtime.
Controller	Select Q2345	Recibe juegos, se queda solo con las columnas ID, Name, Gender, Year, Average Playtime.
	Filter Gender	Recibe juegos, según la columna gender los envía a un exchange con determinado routing key, dropea el campo gender.
	Filter Decade	Solo envía los juegos que sean de la década 2010, el resto los dropea. Además de los que envía dropea la columna Year y el ID.
	Grouper Top Average Playtime	Mientras recibe juegos, va comparando el avg playtime y va quedando en memoria con el top 10 con mayor avg playtime.

Query 3) Nombre de los juegos top 5 del género "Indie" con más reseñas positivas.



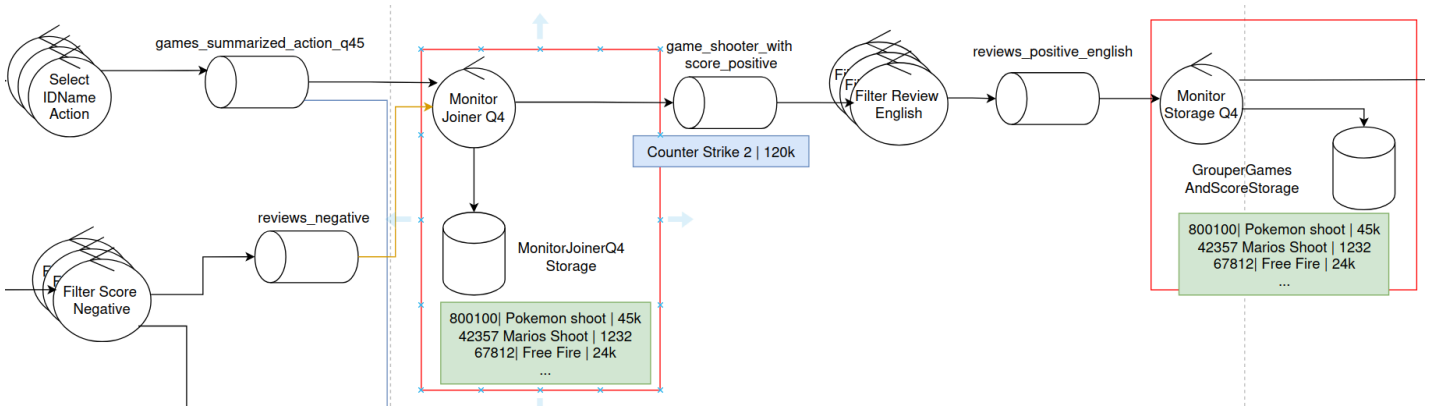
El controller **Monitor Games and Score** almacena todos los juegos que recibió.

1) Por cada uno crea un registro del estilo: **20200| Galactic Bowling|0.**

Aca el controller Monitor recibirá los id con reviews positivas, y por cada id el Monitor guardará un +1 para dicho juego para el cliente correspondiente:

Diccionario de elementos		
Entity	MonitorStorage Q3Str	Almacena los nombres de los juegos y una cantidad de reviews asociada.
	Grouper Top Reviews Pos	Almacena como máximo 5 juegos, aquellos con mayor cantidad de reviews.
Controller	Filter Gender	Recibe juegos, según la columna gender los envía a un exchange con determinado routing key, dropea el campo gender.
	Select IDName	Recibe games y selecciona únicamente a las columnas ID y Name.
	Filter Score Positive	Selecciona el ID y el score filtrando los score positivos (=1)
	Monitor StorageQ3	Inserta los games que le envían, y actualiza su cantidad de reviews según el id que le envían. Finalmente al recibir los EOFs envía los games con sus reviews al siguiente nodo
	Grouper Top Reviews Pos Indie	Mientras Recibe los game indie con una cantidad de review positivas, se va guardando el top 5 de juegos con mayor cantidad de reviews positivas.

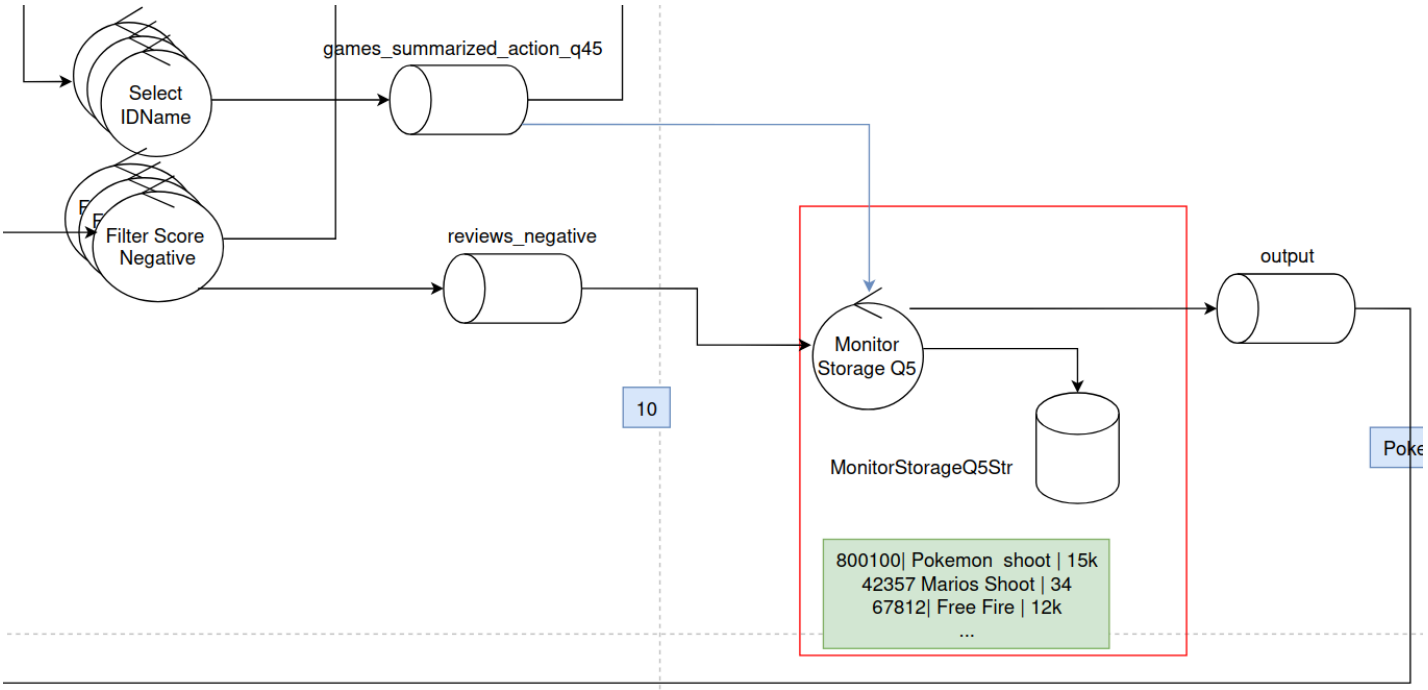
Query 4) Nombre de juegos del género "shooter" con más de 50.000 reseñas positivas en idioma inglés



Filtramos los juegos del tipo action (routing-key es action), y cuando el monitor joiner Q4 haya recibidos los games y los EOFs necesarios, empezará a reenviar las reviews al siguiente nodo. Ese nodo filtrará aquellas que sean en ingles y finalmente el **MonitorStorageQ4** almacenará la cantidad. Si alguno llegara a la cantidad pedida será enviado como respuesta.

Diccionario de elementos		
Entity	GrouperGamesAndScoreStorage	Contiene los nombres de los juegos
	MonitorJoinerQ4 Storage	Contiene los nombres de los juegos con sus reviews
Controller	Select IDName	Recibe games y selecciona únicamente a las columnas ID y Name.
	Filter Score Negative	Selecciona el ID y el score quedándose solo con los que tienen score negativo (<0)
	MonitorJoiner Q4	Inserta los games que le envían. Finalmente al recibir los EOFs empieza a procesar las reviews para dicho cliente y envía los games al siguiente filtro
	Filter Review English	Recibe una review y si está en inglés la envía por la queue destino, si no la descarta.
	MonitorStorageQ4	Recibe reviews de games y actualiza su cantidad de reviews según el id que le envían. En caso de llegar a 5000 reviews, envía dicho game como respuesta.

Query 5) Nombre de juegos del género "action" dentro del percentil 90 en cantidad de reseñas negativas



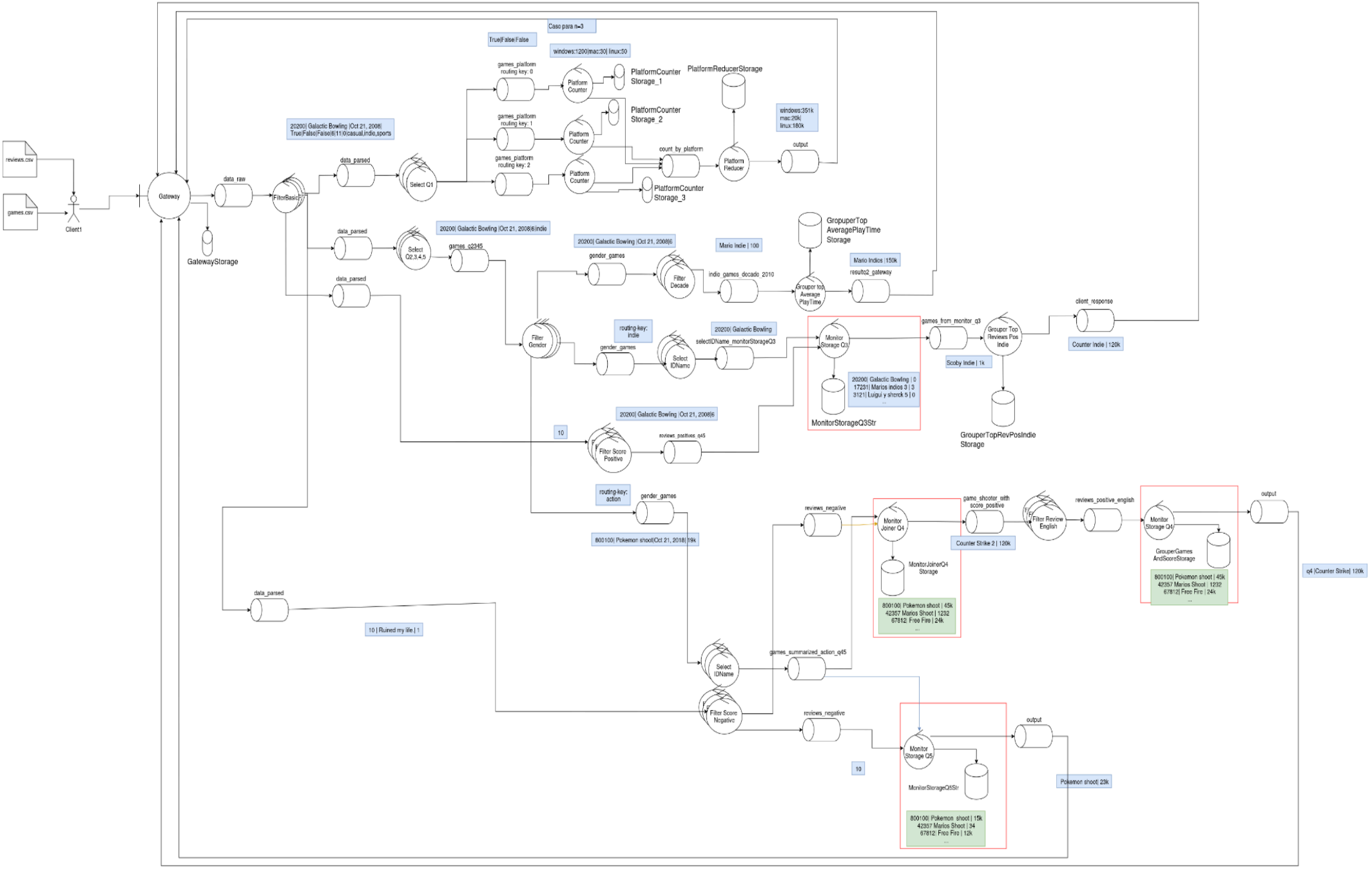
Diccionario de elementos		
Entity	MonitorStorage Q5Str	Contiene los nombres de los juegos action y una cantidad de reviews negativas asociada.
Controller	Select IDName	Recibe games y selecciona únicamente a las columnas ID y Name.
	Filter Score Negative	Selecciona el ID y el score quedándose solo con los que tienen score negativo (<0)
	Monitor Storage Q5	Inserta los games que le envían, y actualiza su cantidad de reviews según el id que le envían. Finalmente al recibir todos los EOF, calcula el percentil y lo envía como respuesta.

Lista de exchanges:

Producer	Nombre	Tipo	Keys	Consumer
Input	DataRaw	Direct	Default	Filter Basic
Filter Basic	DataParsed	Direct	games.q1;games.q2345;reviews.raw	SelectQ1, SelectQ2345, SelectQ345
SelectQ1	GamesPlatform	Direct	Default	Platform Counter
PlatformCounter	CountByPlatform	Direct	Default	Platform Reducer
SelectQ2345	GamesQ2345	Direct	Default	FilterByGender
FilterByGender	GamesIndieQ2	Direct	Indie	FilterDecade
FilterDecade	GamesIndieDecadeQ2	Direct	Default	GrouperTopAveragePlaytime
FilterByGender	GamesIndieQ3	Direct	Indie	Select IDName
SelectIDName_Q3	GamesIndieQ3Sumarized	Direct	Default	Monitor Storage Q3
MonitorStorageQ3	GamesIndieMonitor	Direct	Default	GrouperTopReviewsPos
FilterByGender	GamesActionQ45	Direct	Action	SelectIDName
SelectIDName_Q45	GamesActionQ45Sumarized	Fanout		MonitorJoinerQ4; MonitorStorageQ5
MonitorJoinerQ4	GamesActionPositives	Direct	Default	FilterReviewEnglish
FilterReviewEnglish	ReviewsEnglishPositives	Direct	Default	MonitorStorageQ4
Input	ReviewsRaw	Direct	Default	FilterScorePositive, FilterScoreNegative
FilterScoreNegative	ReviewsNegatives	Direct	Default	MonitorStorageQ5
FilterScorePositives	ReviewsPositives	Fanout		MonitorJoinerQ4; MonitorStorageQ3
MonitorStorageQ4,MonitorStorageQ5,PlatformReducer,GrouperTopAveragePlaytime,Grouper	ClientsResponse	Direct	Default	Output

TopReviewsPos				

Diagrama de robustez unificado:


reviews.csv
games.csv
Client1
Gateway
GatewayStorage
data_raw
FilterBasic
data_parsed
Select Q1
Caso para n=3
True|False|False
windows:1200|mac:30|linux:50
games_platform routing key: 0
games_platform routing key: 1
games_platform routing key: 2
Platform Counter
PlatformCounter Storage_1
PlatformCounter Storage_2
PlatformCounter Storage_3
PlatformReducerStorage
count_by_platform
Platform Reducer
windows:351k mac:20k| linux:180k
output
20200| Galactic Bowling |Oct 21, 2008| True|False|False|6|11|0|casual,indie,sports
Select Q2,3,4,5
games_q2345
20200| Galactic Bowling |Oct 21, 2008|6|indie
20200| Galactic Bowling |Oct 21, 2008|6
gender_games
Filter Decade
indie_games_decade_2010
Mario Indie | 100
GropuperTop AveragePlayTime Storage
Grouper top Average PlayTime
Mario Indies |150k
result2_gateway
Filter Gender
routing-key: indie
Select IDName
20200| Galactic Bowling
selectIDName_monitorStorageQ3
Monitor Storage Q3
20200| Galactic Bowling | 0
17231| Marios indios 3 | 3
3121| Luigui y sherck 5 | 0
...
MonitorStorageQ3Str
games_from_monitor_q3
Grouper Top Reviews Pos Indie
Scoby Indie | 1k
GrouperTopRevPosIndie Storage
client_response
Counter Indie | 120k
10
Filter Score Positive
reviews_positives_q45
routing-key: action
800100| Pokemon shoot|Oct 21, 2018| 19k
reviews_negative
Monitor Joiner Q4
MonitorJoinerQ4 Storage
800100| Pokemon shoot | 45k
42357 Marios Shoot | 1232
67812| Free Fire | 24k
...
game_shooter_with score_positive
Counter Strike 2 | 120k
Filter Review English
reviews_positive_english
Monitor Storage Q4
GrouperGames AndScoreStorage
800100| Pokemon shoot | 45k
42357 Marios Shoot | 1232
67812| Free Fire | 24k
...
output
q4 |Counter Strike| 120k
data_parsed
10 | Ruined my life.| 1
Select IDName
games_summarized_action_q45
Filter Score Negative
reviews_negative
Monitor Storage Q5
MonitorStorageQ5Str
800100| Pokemon shoot | 15k
42357 Marios Shoot | 34
67812| Free Fire | 12k
...
10
output
Pokemon shoot| 23k

Diagramas están embebidos aca:

https://app.diagrams.net/#G15YEBNOWZSgx7o8dx3Z4zthoyh2cpAFXL#%7B%22pageId%22%3A%22yFwRr3H4ifiSJDiVAae2%22%7D

Listado de tareas a ejecutar y división entre integrantes

https://docs.google.com/spreadsheets/d/1knQ4Veg2v8ogLjhzOWekipYF7Jqwo9umv_sJSXtyS4s/edit?gid=0#gid=0