FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Sistemas Distribuidos 1 (75.74)

Documento de Arquitectura: Steam Analyzer

Segundo Cuatrimestre de 2024

Grupo 15

Alumno	**Padrón**	**Mail**
Ignacio Ariel Loscocco	104002	iloscocco@fi.uba.ar
Theo Lijs	109472	tlijs@fi.uba.ar
Luciano Lorenzo	108951	llorenzo@fi.uba.ar

Índice

1. Introducción

En el siguiente documento se describe la arquitectura del sistema Steam Analyzer, cuyo objetivo es el procesamiento de datasets de juegos y reviews de forma distribuida para así responder cinco consultas distintas.

1.1. Hipótesis y Supuestos

Para realizar el sistema se tuvo en cuenta las siguientes

- El servicio de RabbitMQ no va a caerse en ningún momento.
- El procesamiento de los clientes se realiza de manera secuencial.
- No se tiene en cuenta ante la caída del cliente el guardado del estado parcial de los datos y que este puede seguir haciendo la consulta en donde se quedo.

2. Vista Escenarios

2.1. Casos de uso

El sistema recibe del usuario los datasets de juegos y reviews de la plataforma steam para luego responder con la siguiente información:

- Cantidad de juegos soportados en cada plataforma (Windows, Linux, MAC)
- Nombre de los juegos top 10 del género 'Indie' publicados en la década del 2010 con más tiempo promedio histórico de juego
- Nombre de los juegos top 5 del género 'Indie' con más reseñas positivas
- Nombre de juegos del género 'shooter' con más de 50.000 reseñas positivas en idioma inglés
- Nombre de juegos del género 'shooter' dentro del percentil 90 en cantidad de reseñas negativas

2.2. Requerimientos No Funcionales

- El sistema debe estar optimizado para entornos multicomputadoras
- Se debe soportar el incremento de los elementos de cómputo para escalar los volúmenes de información a procesar
- Se requiere del desarrollo de un Middleware para abstraer la comunicación basada en grupos.
- Se debe soportar una única ejecución del procesamiento y proveer graceful exit frente a señales SIGTERM.

3. Visión Lógica

3.1. Flujo de Datos

En la figura 1 se puede ver el DAG, en donde se muestran los datos de input y output en cada paso que hace el sistema para completar las queries. Por ejemplo 'Accumulator by platform' recibe como input count (cantidad parcial de juegos en x plataformas) y devuelve la cuenta acumulada de los juegos que hay en cada plataforma.

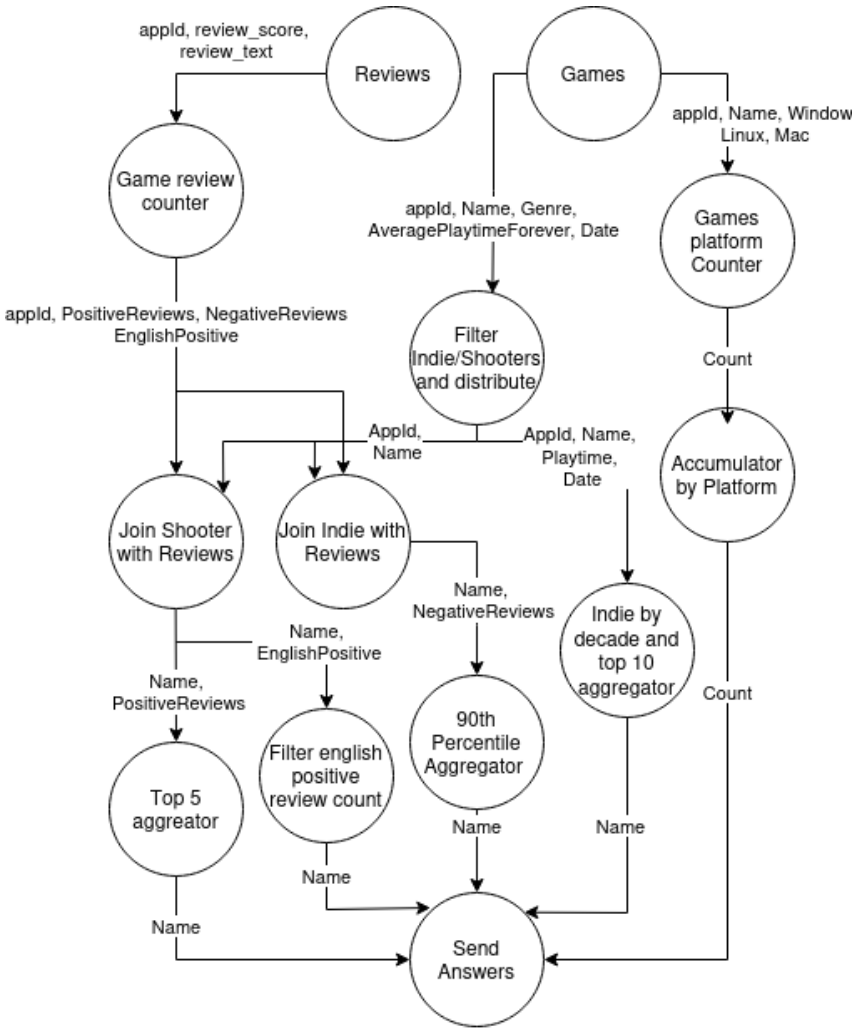

Figura 1: DAG

Cabe aclarar que Send Answers se realizaría en donde el servidor se comunica con el cliente.

4. Vista Física

4.1. Distribución de Componentes

A continuación, la figura 2 muestra la topología del sistema a nivel de vista física y las conexiones entre sus componentes físicos.

Se utiliza rabbitmq como message broker y por lo tanto middleware, que se encarga de toda la comunicación dentro del sistema entre los workers.

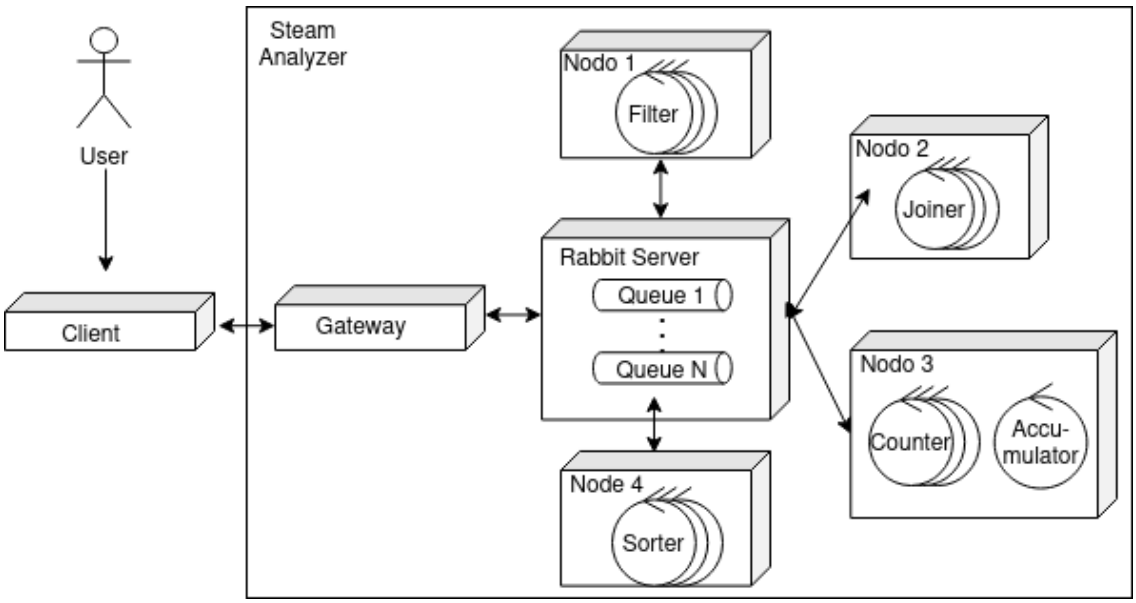

Figura 2: Diagrama de despliegue

En el nodo 3 se puede observar que puede estar mas de un worker en el mismo nodo. No es necesaria una distribución especifica de workers por dispositivo ya que la comunicación se realiza exclusivamente por el middleware.

4.2. Interacciones de Componentes

A continuación en la figura 3 se muestra la interacción entre los diferentes servicios del sistema con respecto a los juegos. Cabe aclarar que el client handler, el input controller, y el genre filter son los mismos para los dos diagramas. ambos diagramas suceden a la vez pero están separados para mayor legibilidad.

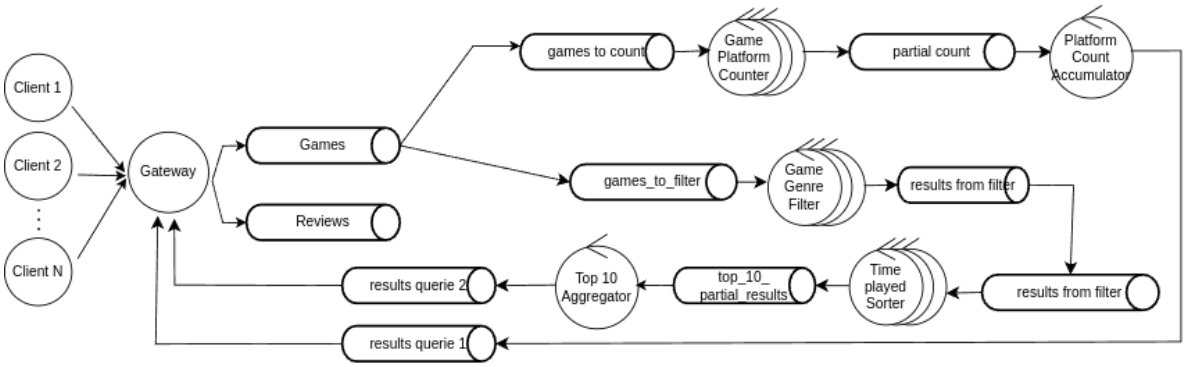

Figura 3: Diagrama Robustez sección juegos

Seguidamente, en el siguiente diagram (figura 4) se ve la sección de juegos-reviews. En esta se particionan los juegos y reviews por las IDs de los juegos para luego poder escalar la seccion de los joiners.

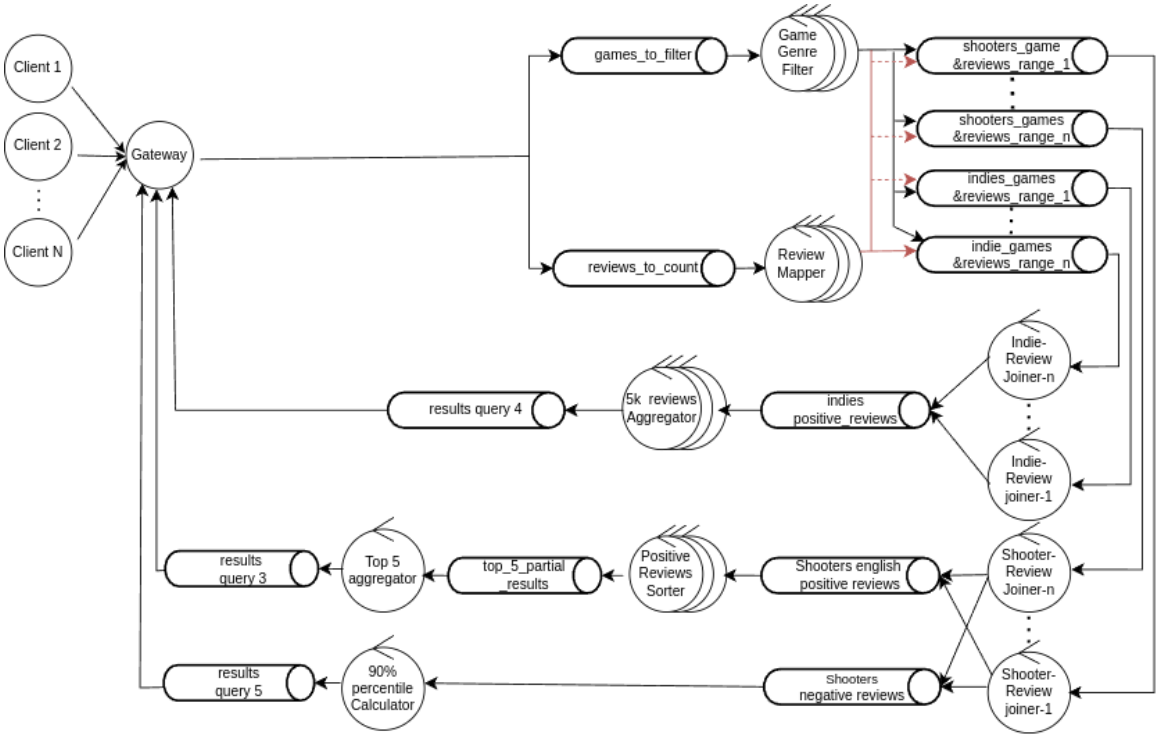

Figura 4: Diagrama Robustez sección juegos y reviews

Para el manejo de múltiples clientes, se le asigna a cada uno de ellos un sessionId, el cual es creado por el gateway y es enviado en todos los mensajes que pasan a través de las colas. Estos son enviados en los headers de los mensajes de rabbitmq.

4.3. Servicios del sistema

En esta sección se encuentran descritos los nodos pertenecientes al diagrama de robustez:

- Client Handler: Se encarga de recibir los datasets de juegos y reviews del cliente para luego pasarselos a los inputs controllers. También es el encargado de enviar las respuestas de las queries al ir recibiendolas.
- input Controller: Obtiene los datasets de juegos y clientes del Client Handler y los divide para mandárselos a las diferentes queues de entrada del sistema.
- Platform Counter: Recibe los juegos del input controller para contar las plataformas en las que esta disponible cada uno. Se comunica luego con el accumulator a través de una cola.
- Platform Accumulator: Suma la suma de los juegos enviando los resultados parciales de la querie 1 a través de la respectiva cola.
- Game Filter: Recibe juegos del input controller y filtra los que nos son de interes para el sistema (se queda con shooter e indies). A su vez se encarga de ir agrupando los juegos por rango de appIds. Una vez que va agrupando cierta cantidad de juegos los manda a traves de las queues del rango y genero correspondiente a travez de un topic exchange creado con rabbitmq. Cabe aclarar que a la cola de indie games le llegan los juegos indie con la información extra necesaria par realizar esta cola sin importar el rango.
- Review Accumulator: Recibe las reviews del input controller para unificar las diferentes reviews al juego especifico. Va unificando la cantidad de reviews positivas, negativas y postivas en ingles de cada juego para luego mandarlo a los Genre-review-joiner.
- genre-review-Joiner: Nodo que se encarga de emparejar los juegos que recibe de cierto genero con las reviews. Recibe los juegos del game filter y las reviews del review accumulator.
- Sorter: Ordena juegos o reviews por diferentes criterios según distintas variables de entorno. Decidimos juntar este servicio en uno ya que la lógica es simplemente ordenar. A su vez estos sorter tienen la opción de mandar x cantidad en el arreglo resultante de datos para poder conseguir los top x si es necesario.
- Percentile Calculator: Une los distintos arreglos ordenados provenientes del sorter para luego poder calcular el porcentil. Devuelve los elementos a partir de X percentil, siendo estos los resultados de la querie 5.
- Decade-filter: Filtra los juegos a partir de una epoca. Recibe los datos del game filter y los envia a un sorter para resolver la querie 2.
- Top x Aggregator: Recibe los tops x proveniente de los sorters para devolver el verdadero top x global de la entrada de datos.

5. Vista de Procesos

5.1. Comunicación Cliente-Servidor

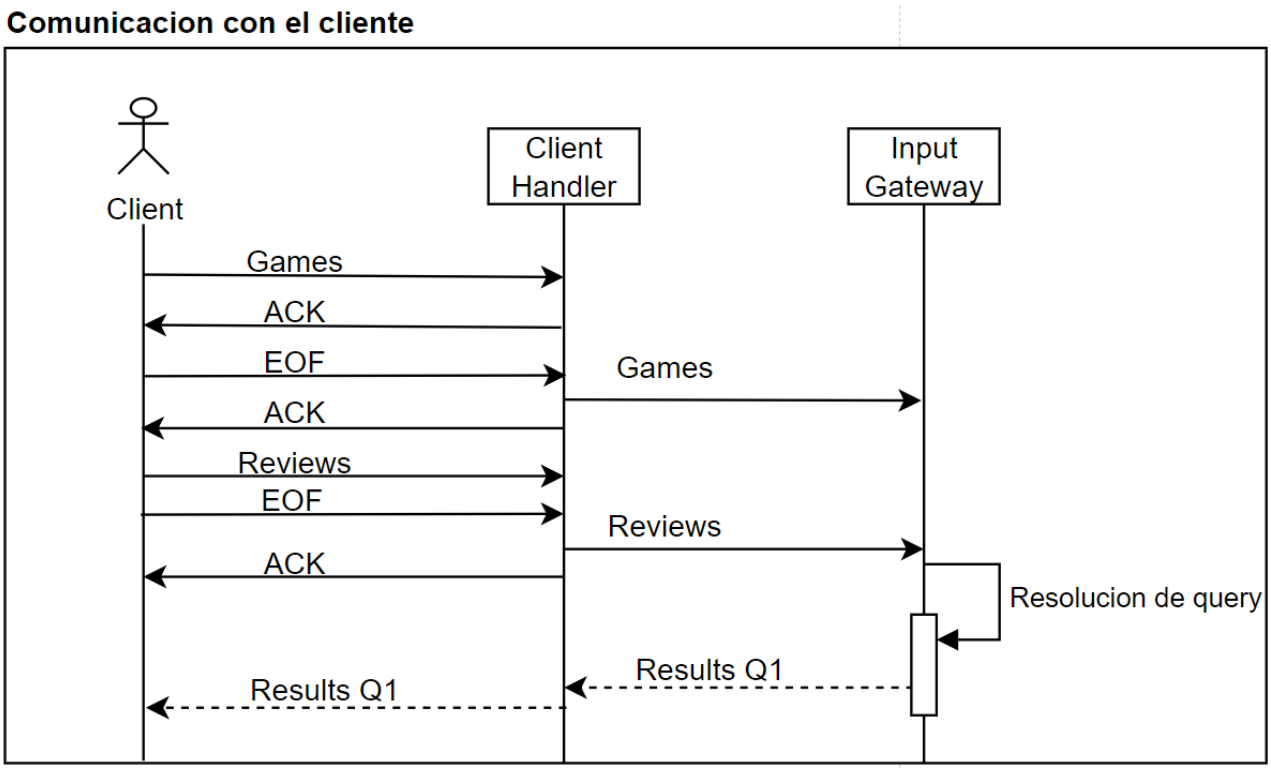

Figura 5: Secuencia Cliente-Servidor

En el siguiente diagrama de secuencia se puede observar el flujo de mensajes que se intercambian al realizar la comunicación entre el cliente y el sistema de consultas. Se observa el inicio de la comunicación y la recepción de los datasets por parte del servidor, luego se abstrae en este diagrama la resolución de la query y por último se puede ver la devolución de los resultados obtenidos.

5.2. Resolución de Query

En esta sección se puede observar un ejemplo de como el sistema se comunica internamente (a traves del middleware) para resolver la querie 3.

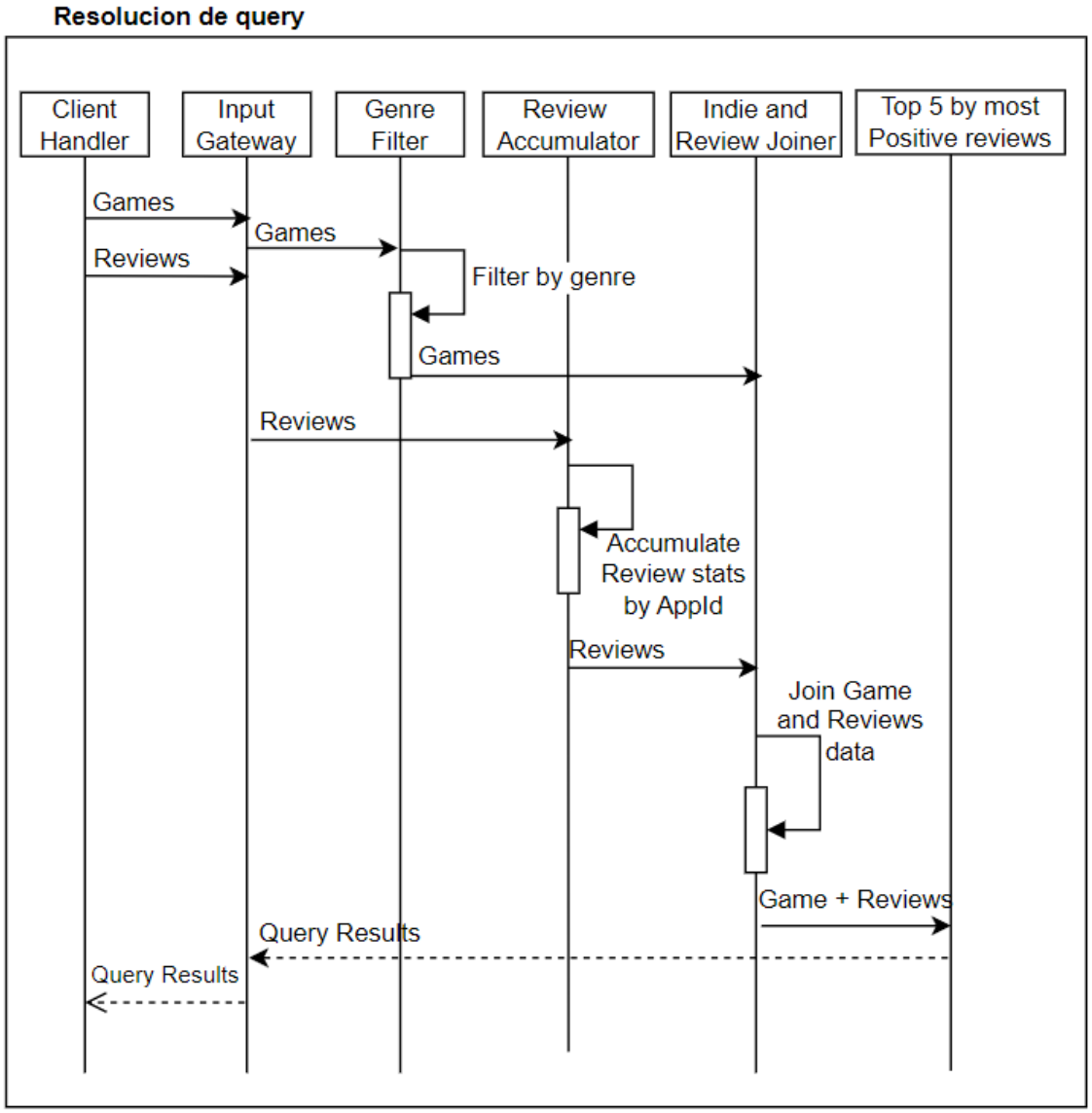

Figura 6: Secuencia de query 3

En el diagrama de secuencia (figura 6) se puede observar el flujo de los mensajes entre las distintas entidades del sistema, que contempla la resolución de todas las queries requeridas. Se puede ver que para el caso de la entidad Platform Accumulator, hace falta solamente el dataset de juegos para resolver la primera query, mientras que para las demás será necesario utilizar el dataset de reviews.

Cabe aclarar que las queries parecen ser secuenciales en el diagrama pero se pueden dar en cualquier orden ya que son independientes. Lo diagramamos todo ordenado para que sea mas claro.

Otra aclaración es que los resultados de las queries 1 y 4 son devueltas a medida que se devuelven

los resultados parciales para mostrar interactividad de parte de la aplicación del servidor

5.3. Comportamiento del sistema

En los siguientes diagramas de actividades podemos observar el comportamiento del sistema modelado. En cada diagrama se modela la resolución de una query distinta de forma que se puede apreciar en cada caso el flujo de trabajo del sistema y las actividades que se realizan en cada servicio.

En primer lugar en la figura 7 se puede ver la resolución de la query 2, que es independiente de reviews:

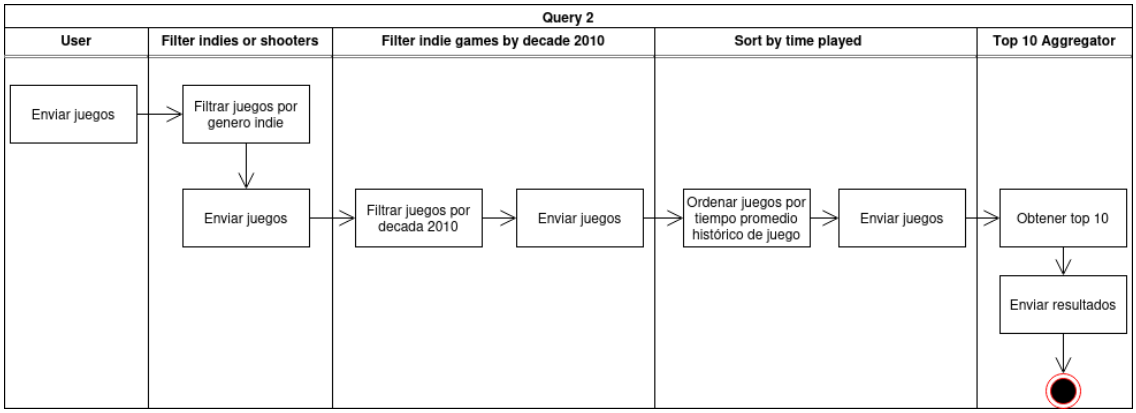

Figura 7: Diagrama de Actividades sobre Query 2

Luego por otro lado en el siguiente diagrama (figura 8) se incluyo la resolución de la query 4 para poder apreciar el join de reviews-game.

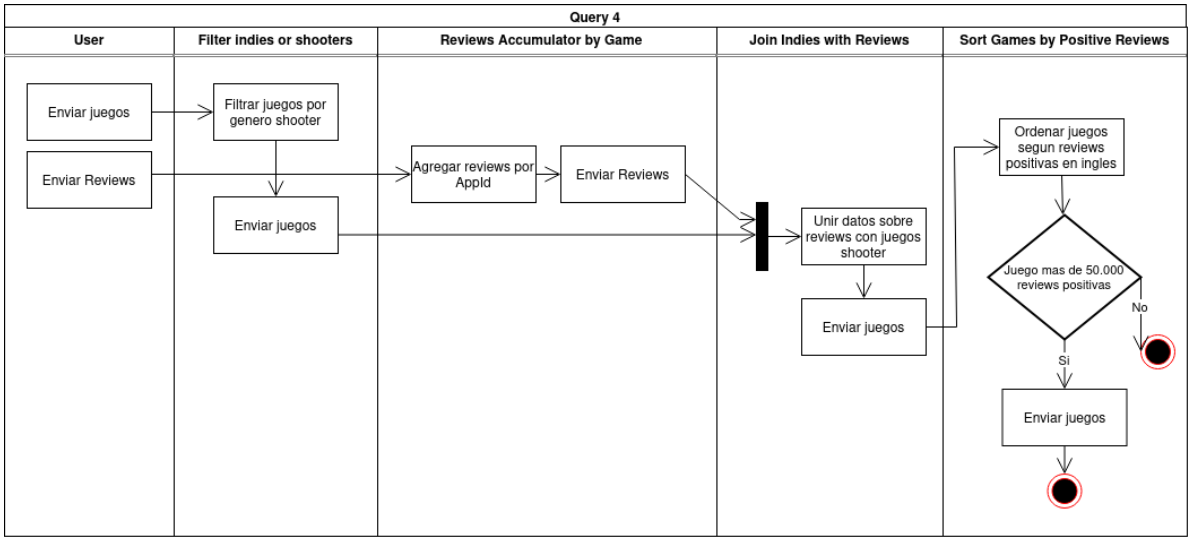

Figura 8: Diagrama de Actividades sobre Query 4

6. Vista de desarollo

6.1. División del Sistema

Por ultimo en esta sección se muestra como el sistema esta ordenado a nivel paquetes.

En la figura 9 se puede ver el diagrama de paquetes muestra la estructura interna del sistema y cliente.

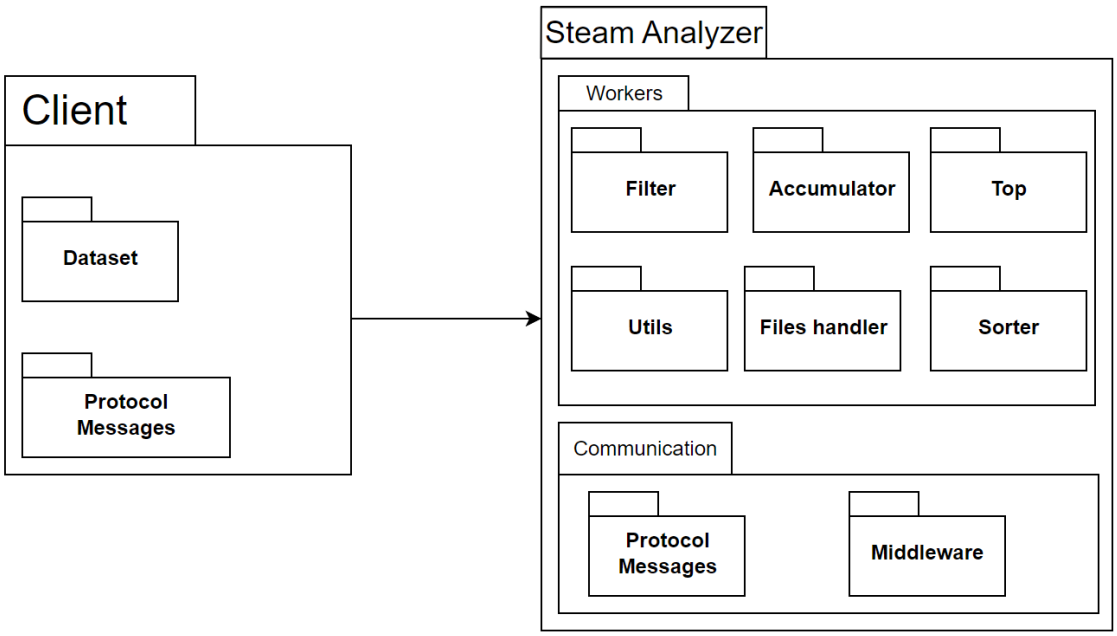

Figura 9: Diagrama de paquetes

El cliente posee dos paquetes para trabajar con la lectura, serialización y envío de datos de las reviews y los juegos.

Dentro del paquete System se encuentran dos secciones. En primer lugar la sección de los workers que trabajan con los datos recibidos de las reseñas y juegos, procesándolos de acuerdo a su función específica. Cada worker se especializa en una tarea particular que tiene sus clases especificas para trabajar con los datos, como por ejemplo percentile result que seria el tipo de mensaje con el que se comunica el resultado del percentile 90 en la query 5.

Por otro lado, adentro del sistema también se encuentra un paquete común de comunicación en donde irían los protocolos de mensajes comúnes entre los diferentes workers del sistema como sería mandar Juegos, reviews, etc.

Cabe aclarar que dentro de los paquetes habría mas paquetes para diferenciar lógicas entre sí. Por ejemplo en el paquete de 'protocol messages' estarían los paquetes distintos para cada tipo de mensaje.