Universidad Técnica Federico Santa María
Campus Santiago San Joaquín
Sistemas Distribuidos
Semestre 2019-2

Informe Tarea 2

Gustavo Barrios 201573510-7 gustavo.barrios@sansano.usm.cl
Johany Herrera 201573535-2 johany.herrera@sansano.usm.cl

1. gRPC

gRPC es un framework open source de RPC (Remote Procedure Call). Entre los distintos escenarios de uso, es frecuentemente utilizado dentro de sistemas distribuidos para la conexión de nodos (p.e. computadores y dispositivos móviles) a los diversos servicios ofrecidos por el backend.

Para contextualizar, un RPC es una comunicación entre procesos (IPC) que permite a un programa ejecutar en otro espacio de direcciones (típicamente en otra máquina) sin que el programador codifique explícitamente los detalles para esta interacción remota. Esta técnica utiliza una arquitectura de cliente-servidor y para llevarla a cabo se deben utilizar Stubs, los cuales se encargan de enmascarar toda discrepancia entre el cliente y servidor así como también de entregar información acerca de los métodos que se pueden utilizar.

De forma similar, gRPC se basa en esta idea, definiendo servicios que explícitamente establecen que procedimientos pueden ser llamados remotamente junto con sus parámetros y retornos. En especifico, gRCP posee una arquitectura Cliente-Servidor. En el lado del servidor, se implementan los procedimientos del Stub y se ejecuta el gRPC server para manejar las llamadas remotas de parte de los clientes. Y en el lado del cliente, se tiene acceso al Stub que provee los mismos métodos implementados en el servidor. Cabe mencionar que el método utilizado por gRPC para denifir estos Stubs es el Protocol Buffers, un mecanismo open source fabricado por Google para serializar datos estructurados.

Teniendo en consideración lo anterior, la arquitectura que se implementó en la tarea fue una de cliente-servidor. En el lado del cliente, se obtiene el servicio de envío y recepción de mensajes, así como también el de obtención de mensajes y clientes (conectados), trabajo que es realizado por medio de los stubs, y se ejecuta el código para realizar los RPC. En el lado del servidor, se implementan los métodos que son usados por el cliente y se ejecuta el servidor para esperar llamadas de los clientes.

2. RabbitMQ

Por otro lado, RabbitMQ es un software open source de “message broker”, el cual entra en la categori a de Middleware de mensajería y que tiene como principal objetivo servir como intermediario para la comunicación entre un productor y un consumidor. Message broker es un programa intermedio que sirve sirve como un mecanismo mediador entre dos aplicaciones, para minimizar el grado de conocimiento mutuo que deben tener estas aplicaciones para intercambiar mensajes, lo cual permite desacoplar efectivamente su desacoplamiento.

RabbitMQ implementa protocolos de mensajerías, que se enfocan en la comunicación de mensajes asíncronos, persistentes y con garantía de entrega, por medio de colas que llevan a cabo el traspaso de

información. Mas específicamente, la arquitectura de este software se basa en que existe dos aplicaciones, uno llamada productor, el cual crea los mensajes y los envía a un intermediario (las colas de mensajes), y otro llamado consumidor, el cual se conecta a estas colas y se suscriben a los mensajes que llegan a estas.

Otra de las características de RabbitMQ, es que, si bien los mensajes son enviados del productor a una cola, esto no se realiza de forma directa, sino que son realizados por medio de un componente llamado "exchange", el cual es el encargado de recibir los mensajes enviados al broker y depositarlos en las colas adecuadas, según una cierta llave de enrutamiento. Así, en caso de querer enviar un mensaje a mas de una cola, no se envié a cada una de estas, sino que el exchange sea el encargado de esto. Además, producto del almacenamiento persistente, no es necesario que el productor y el consumidor estén activos simultáneamente, ya que los mensajes son guardados en las colas hasta que estos sean entregados.

En resumen, RabbitMQ es un software que se enfoca en la comunicación de mensajes de forma asíncrona por medio de colas, cuyo flujo de comunicación se basa en que un productor publique mensajes al exchange, el cual es el encargado de enviar los mensajes a sus respectivas colas, para que los consumidores puedan suscribirse a dichas colas y procesar estos mensajes.

3. Comparación

Como se ve reflejado en las secciones anteriores, gRPC y RabbitMQ tienen grandes diferencias entre ellas, tanto en su enfoque como en su implementación.

Una de las características más importantes en gRPC, es que para llevar a cabo su implementación, es necesario conocer exactamente los parámetros y el lenguaje de programación ocupado en ambos nodos, de modo que puede traer consigo diversos problemas en el traspaso de mensajes. A diferencia de RabbitMQ, pues se basa en una arquitectura "message broker", la cual actúa como un intermediario entre los nodos, minimizando así el conocimiento mutuo necesario para su implementación y permitiendo de esta forma una mayor libertad en el traspaso de mensajes.

Además, se logra observar que RabbitMQ se enfoca principalmente en un sistema de mensajería, por esta razón entrega muchas facilidades al momento de desarrollar un sistema de este índole. A diferencia de gRPC, el cual tiene un enfoque mas general gracias a que debe implementar los RPC, causando que sus principios de diseño tengan una perspectiva alineada a una mayor flexibilidad de utilización, pero con mayores restricciones y sin tantos beneficios.

Por lo anterior, es posible concluir que la mejor tecnología para desarrollas el sistema pedido en esta tarea, es RabbitMQ, producto de la facilidad y los beneficios que entrega esta tecnología al momento de desarrollar un sistema de mensajería, el cual esta por encima de los beneficios que entrega gRPC.