Informe técnico

Luego de realizar una aplicación de chat con dos diferentes arquitecturas distribuidas basadas en microservicios

* RPC utilizando grpc de Google
* Mensajes asíncronos con RabbitMQ

**Complejidad**: La programación en grpc requiere mayor conocimiento para el desarrollo de la aplicación, como es el uso de Protocol Buffer para la serialización de mensajes. En cambio, utilizando RabbitMQ el desarrollo se vuelve mas sencillo.

**Sincronismo**: La aplicación creada con RPC es síncrona, dado que un cliente necesita que el servidor lleve a cabo el envió del mensaje, por esta misma razón es necesario que cliente y servidor estén encendidos, de lo contario la aplicación falla. La comunicación por paso de mensajes es asíncrona, lo cual es muy beneficioso para una aplicación de chat, permite que los mensajes sean enviados al encenderse cliente o servidor, puesto que los mensajes son almacenados en el gestor de colas.

**Protocolo**: ambas arquitecturas soportan diferentes protocolos de transporte de información, para esta aplicación en particular no tendremos inconvenientes con ninguna.

**Persistencia**: RabbitMQ proporciona persistencia para mensajes que aún no han sido enviados, lo cual es una ventaja, ya que permite que un cliente se desconecte y reciba sus mensajes al momento de conectarse nuevamente. Por otro lado, en RPC no se pueden persistir los mensajes si alguna de las partes falla, para resolver el problema es necesario utilizar un intermediario.

**Recomendación**

Considerando la naturaleza de una aplicación de chat, la utilización de mensajes asíncronos con RabbitMQ es la mejor opción. Haciendo uso de este gestor de colas se puede lograr una gran escalabilidad, lo cual es muy importante para este tipo de aplicaciones que deben soportar una gran cantidad de dispositivos. Otro punto importante señalar es la complejidad del uso de ambas tecnologías, en el cual RabbitMQ es el ganador.