# **Entendiendo sobre colas de mensajes con #RabbitMQ**

¿Qué son las colas de mensajes?

Es posible que alguna vez te hayas conseguido con el nombre *message queue* o su acrónimo *MQ* navegando por la red, si volvemos a los conceptos básicos de la programación de pilas y colas sabremos que es un concepto bastante sencillo donde los emisores producen mensajes y para que estos lleguen a su destinatario deben ser entregados a un intercambiador que los colocará en la cola del respectivo destinatario, finalmente el destinatario puede ir progresivamente desencolando y procesando los mensajes o dejar que el intercambiador se los haga llegar, esto por medio de diferentes tipos de rutas.

Por lo tanto podemos ver el concepto de las colas de mensajes como un intermediario entre los emisores y los destinatarios, o si lo orientamos a un escenario más real, entre clientes y servidores o publicadores y consumidores.

Ventajas

Desde luego estarás esperando que te demos argumentos que te hagan considerar implementar este enfoque de intercambio, veamos algunas:

* **Redundancia** – Si tu aplicación falla mientras está procesando alguna petición no te debes preocupar de que esta se pierda para siempre ya que esta estrategia le permite a la cola persistir el mensaje hasta que el mismo sea procesado por completo.
* **Naturaleza asíncrona** – Dependiendo de la manera en que funcione tu sistema puedes necesitar que los mensajes se vayan acumulando para procesarlos en lotes, la cola irá manteniendo tus mensajes para cuando decidas (o programes) su procesamiento.
* **Garantía de entrega y ordenamiento** – Se garantiza que el orden en el que llegan los mensajes será el orden en el que sean procesados, de igual manera un emisor y/o consumidor puede estar seguro de que este ha sido recibido y se procesará una única vez mediante la implementación de intercambio de banderas de reconocimiento.
* **Disponibilidad** – Si parte de tu arquitectura falla, los mensajes no se perderán ya que estos seguirán en la cola hasta que se les indique lo contrario, al mismo tiempo la cola podrá seguir recibiendo mensajes para el procesamiento posterior a la recuperación del sistema.
* **Elasticidad** – En situaciones donde tu sistema pudiera llegar al tope de su capacidad de recepción de peticiones y volverse incapaz de responder por un anormal flujo de mensajes, el hecho de tener una cola o *buffer* de peticiones permitirá balancear, filtrar y normalizar el flujo, lo cual evitará que tu sistema sea inundado de peticiones que no pueda responder causando perdida de los mismos o incluso el colapso del sistema.
* **Desacoplamiento** – El hecho de tener una capa intermedia de comunicación entre procesos nos da la flexibilidad en la definición de arquitectura de cada uno de ellos de manera separada, mientras cada uno se mantenga alineado a los mismos requerimientos de interfaz que representa la cola de mensajes no abran mayores problemas de compatibilidad ni mucho que cambiar de lado y lado.
* **Escalabilidad** – Por la misma ventaja de desacoplamiento podemos escalar fácilmente nuestro sistema, solo debemos agregar más unidades de procesamiento y el sistema de colas se encargará de balancear la carga entre ellos.

RabbitMQ

Luego de conocer un poco sobre lo que son las colas de mensajes podemos introducir la solución que utilizaremos aquí, RabbitMQ. Este programa nos ofrece las ventajas antes mencionadas de las colas de mensajes además de ofrece también otras características que lo hacen más flexible aún:

* Soporta múltiples protocolos, muchas soluciones solo manejan uno solo como AMQP.
* Dispone de librerías en casi todos los lenguajes de programación.
* Rastreo de mal comportamiento en las colas.
* *Clusterización* de varios servidores de colas para evitar que el sistema de encolamiento falle.
* Múltiples tipos de enrutamiento, incluso puedes crear tu propia ruta.
* Soporta *plugins* para extender su comportamiento.
* Amigable interfaz gráfica.

Existen algunas alternativas a RabbitMQ muy interesantes en el ámbito *cloud* o *SaaS* (software como servicio) como IronMQ y Amazon SQS; siempre promovemos el uso de herramientas de código abierto y por ello optamos por RabbitMQ, la solución que consideramos más robusta de manejo de colas de mensajes.