

RabbitMQ

Sergio García Sánchez Miguel Emilio Ruiz Nieto

9 de diciembre de 2021

Contenidos

- 1 Introducción
- 2 RabbitMQ
- 3 Ejemplos prácticos
- 4 Conclusiones
- 5 Bibliografía

- Los servicios web no tienen la capacidad de gestionar las peticiones que le llegan en un mismo momento

- Los servicios web no tienen la capacidad de gestionar las peticiones que le llegan en un mismo momento
- Ejemplos:
 - Web de venta de entradas ante un evento importante
 - Servicio de videojuego online con alta demanda de usuarios

- Los servicios web no tienen la capacidad de gestionar las peticiones que le llegan en un mismo momento
- Ejemplos:
 - Web de venta de entradas ante un evento importante
 - Servicio de videojuego online con alta demanda de usuarios
- Como consecuencia:
 - Caída del servicio
 - Pérdida económica
 - Pérdida de reputación

Por tanto

Es necesario procesar las peticiones “poco a poco”

Por tanto

Es necesario procesar las peticiones “poco a poco”

Y para ello

Hay que implementar **una cola de mensajes**

Introducción. Cola de mensajes

- Comunicación asíncrona *service-to-service* usado en arquitecturas serverless y microservicios
- Permite desacoplar procesos con mucha carga de trabajo o almacenar trabajo *en batch*

Cola de mensajes



- Mejor rendimiento
- Mayor fiabilidad
- Escalabilidad granular
- Desacoplamiento simplificado

- STOMP

- El protocolo más sencillo
- Implementado sobre HTTP
- Basado en intercambio de *frames*
- La infraestructura de queues, topics y exchanges quedan del lado del cliente

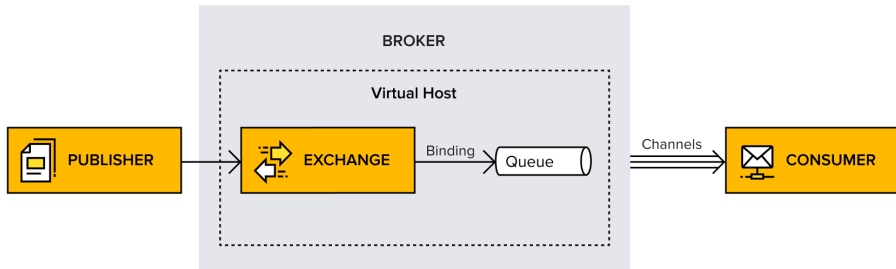
- MQTT

- Más ligero que STOMP
- Construido sobre TCP/IP
- Orientado a arquitecturas IoT
- Esquema *publisher-suscriber* síncrono

- AMQP

- Comunicación asíncrona *publisher-suscriber* mediante **broker**
- Permite el almacenamiento de mensajes
- Proporciona balanceo de carga

Colas de mensajes. AMQP



- Implementación del broker AMQP en Erlang
- Ofrece soporte para HTTP, STOMP y MQTT
- Interfaz web para manejar el broker y los componentes asociados
- Amplio soporte en múltiples lenguajes de programación

- El primer componente que recibe el mensaje en AMQP
- Toma el mensaje y lo redirige a una o más colas
- Las colas se enlazan a los Exchanges mediante *bindings*
- Se puede añadir un parámetro opcional **routing_key**

Tipo	Descripción
Direct Exchange	Envía el mensaje directamente a la cola basado en el routing_key (amq.direct)
Fanout Exchange	Envía el mensaje a todas las colas enlazadas (amq.fanout)
Topic Exchange	Envía el mensaje a las colas suscritas al <i>topic</i> (amq.topic)
Headers Exchange	Envía el mensaje mirando las cabeceras en lugar del routing_key (amq.headers)

- Se usan en los topic exchanges
- Lista de palabras separadas por puntos
- Ejemplos:
 - “health.sports.football”
 - “#.sports”
 - “sports.*”

Dead Lettering

- Ciertos mensajes pueden no ser consumidos por los consumers
- RabbitMQ ofrece mecanismos para gestionar el *Dead lettering*
 - Rechazo por TTL excedido
 - Rechazo por exceso de longitud de la cola
 - Rechazo por parte del propio consumer (`basic.reject`)
- Los mensajes son introducidos en un *Dead letter exchange* (DLX)

RabbitMQ vs Apache Kafka

RabbitMQ

Broker de mensajería AMQP

Utiliza protocolos de mensajería

Se pueden perder mensajes

Enfocado a comunicación entre arquitecturas microservicios

Apache Kafka

Plataforma de procesamiento de flujo de eventos

Utiliza modelo editor/suscriptor

No se pueden perder mensajes

Enfocado a Big Data

- AMQP - Uso de los exchanges y Dead Lettering
- STOMP over WebSocket

- Es necesario utilizar un mecanismo de cola de mensajes si la aplicación requiere de procesar un número grande de peticiones
- RabbitMQ ofrece soporte para diferentes lenguajes de programación dentro de su implementación

- RabbitMQ Documentation
<https://www.rabbitmq.com/documentation.html>
- RabbitMQ Essentials
- Learn RabbitMQ: Asynchronous Messaging with Java and Spring