

## Taller - Implementación del patrón Pub/Sub con Apache Camel y RabbitMQ

Martín Vargas y Kevin Rosero

### Objetivo

Simular un sistema de notificaciones en el que:

- Un publicador emite alertas cada 5 segundos.
- Dos suscriptores (consumidores) reciben el mismo mensaje, de forma desacoplada.

Repositorio Github: <https://github.com/MartinVargas07/Taller-Patron-Pub-Sub-Vargas-Rosero.git>

Capturas de Imágenes:

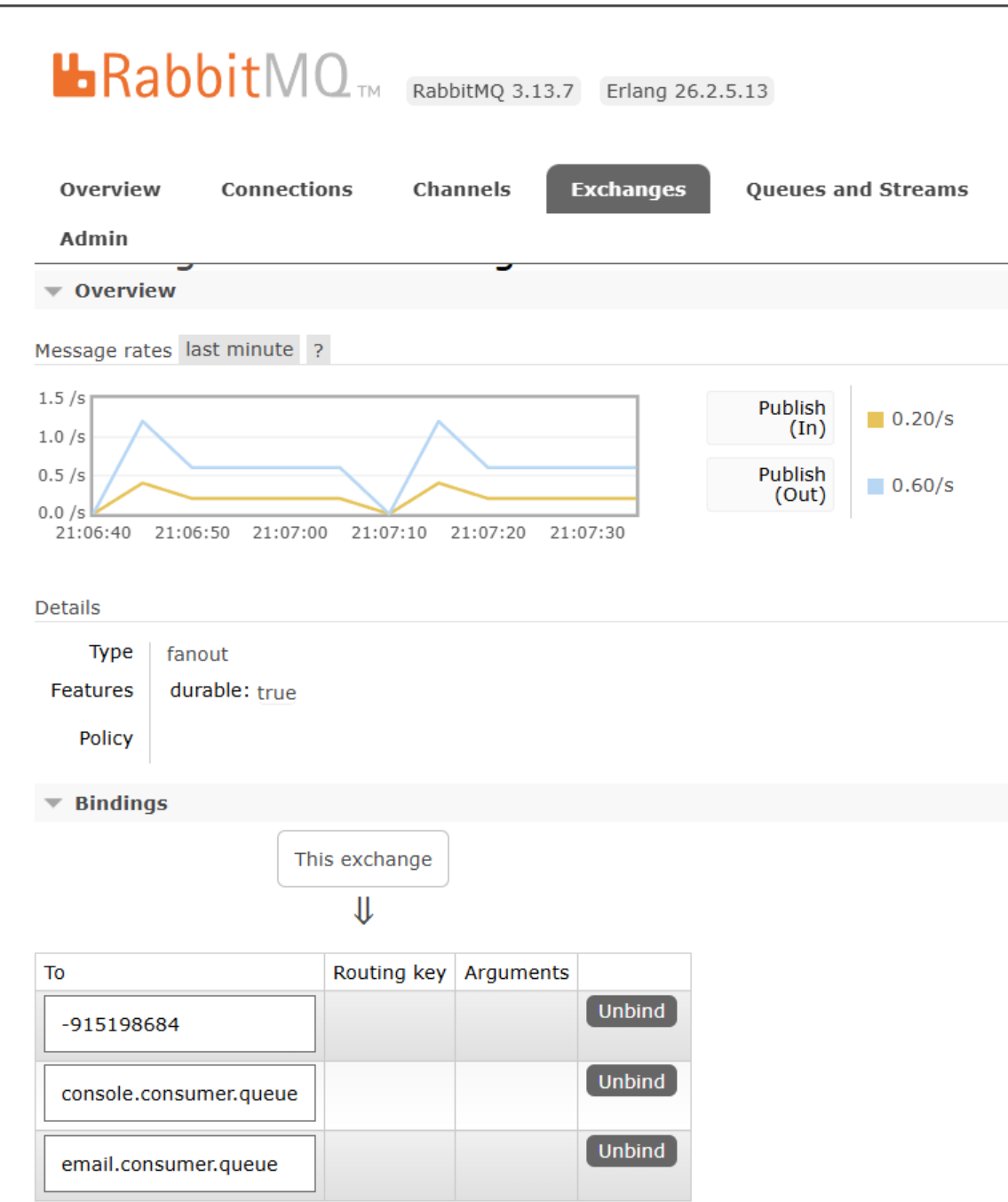
Mensajes recurrentes del Publicador y los Suscriptores 1 y 2

```
[INFO] Attaching agents: []

=====
:: Spring Boot ::
(v2.7.15)

2025-06-25 21:04:10.314 INFO 29748 --- [main] com.ejemplo.camel.App : Starting App using Java 21.0.6 on MARTIN-ROGSTRIX with PID 29748 (C:\Users\marti\OneDrive\Documentos\proyectos\camel-pubsub-lab\target\classes started by marti in C:\Users\marti\OneDrive\Documentos\proyectos\camel-pubsub-lab)
2025-06-25 21:04:10.315 INFO 29748 --- [main] com.ejemplo.camel.App : No active profile set, falling back to 1 default profile: "default"
2025-06-25 21:04:10.909 WARN 29748 --- [main] o.a.c.c.rabbitmq.RabbitMQComponent : The old syntax rabbitmq://hostname:port/exchangeName is deprecated. You should configure the hostname on the component or ConnectionFactory
2025-06-25 21:04:10.945 WARN 29748 --- [main] o.a.c.c.rabbitmq.RabbitMQComponent : The old syntax rabbitmq://hostname:port/exchangeName is deprecated. You should configure the hostname on the component or ConnectionFactory
2025-06-25 21:04:10.955 WARN 29748 --- [main] o.a.c.c.rabbitmq.RabbitMQComponent : The old syntax rabbitmq://hostname:port/exchangeName is deprecated. You should configure the hostname on the component or ConnectionFactory
2025-06-25 21:04:10.962 INFO 29748 --- [main] o.a.c.impl.engine.AbstractCamelContext : Apache Camel 3.20.2 (camel-1) is starting
2025-06-25 21:04:11.256 INFO 29748 --- [main] o.a.c.impl.engine.AbstractCamelContext : Routes startup (started:3)
2025-06-25 21:04:11.256 INFO 29748 --- [main] o.a.c.impl.engine.AbstractCamelContext : Started route1 (rabbitmq://localhost/alert-exchange)
2025-06-25 21:04:11.256 INFO 29748 --- [main] o.a.c.impl.engine.AbstractCamelContext : Started route2 (rabbitmq://localhost/alert-exchange)
2] Simulando envió por email: Alerta generada: 2025-06-25 21:04:12

2025-06-25 21:04:17.265 INFO 29748 --- [timer://alerta] route3 : Publicando mensaje: Alerta generada: 2025-06-25 21:04:17
2025-06-25 21:04:17.267 INFO 29748 --- [alert-exchange] route1 : [Consumer 2] Simulando envío por email: Alerta generada: 2025-06-25 21:04:17
2025-06-25 21:04:22.265 INFO 29748 --- [timer://alerta] route3 : Publicando mensaje: Alerta generada: 2025-06-25 21:04:22
2025-06-25 21:04:22.265 INFO 29748 --- [alert-exchange] route2 : [Consumer 1] Mensaje recibido: Alerta generada: 2025-06-25 21:04:22
2025-06-25 21:04:22.265 INFO 29748 --- [alert-exchange] route1 : [Consumer 2] Simulando envío por email: Alerta generada: 2025-06-25 21:04:22
2025-06-25 21:04:27.265 INFO 29748 --- [timer://alerta] route3 : Publicando mensaje: Alerta generada: 2025-06-25 21:04:27
2025-06-25 21:04:27.272 INFO 29748 --- [alert-exchange] route2 : [Consumer 1] Mensaje recibido: Alerta generada: 2025-06-25 21:04:27
2025-06-25 21:04:27.272 INFO 29748 --- [alert-exchange] route1 : [Consumer 2] Simulando envío por email: Alerta generada: 2025-06-25 21:04:27
2025-06-25 21:04:32.270 INFO 29748 --- [timer://alerta] route3 : Publicando mensaje: Alerta generada: 2025-06-25 21:04:32
2025-06-25 21:04:32.275 INFO 29748 --- [alert-exchange] route2 : [Consumer 1] Mensaje recibido: Alerta generada: 2025-06-25 21:04:32
2025-06-25 21:04:32.275 INFO 29748 --- [alert-exchange] route1 : [Consumer 2] Simulando envío por email: Alerta generada: 2025-06-25 21:04:32
2025-06-25 21:04:37.270 INFO 29748 --- [timer://alerta] route3 : Publicando mensaje: Alerta generada: 2025-06-25 21:04:37
2025-06-25 21:04:37.270 INFO 29748 --- [alert-exchange] route1 : [Consumer 2] Simulando envío por email: Alerta generada: 2025-06-25 21:04:37
2025-06-25 21:04:37.270 INFO 29748 --- [alert-exchange] route2 : [Consumer 1] Mensaje recibido: Alerta generada: 2025-06-25 21:04:37
2025-06-25 21:04:42.275 INFO 29748 --- [timer://alerta] route3 : Publicando mensaje: Alerta generada: 2025-06-25 21:04:42
2025-06-25 21:04:42.278 INFO 29748 --- [alert-exchange] route1 : [Consumer 2] Simulando envío por email: Alerta generada: 2025-06-25 21:04:42
2025-06-25 21:04:42.278 INFO 29748 --- [alert-exchange] route2 : [Consumer 1] Mensaje recibido: Alerta generada: 2025-06-25 21:04:42
2025-06-25 21:04:47.276 INFO 29748 --- [timer://alerta] route3 : Publicando mensaje: Alerta generada: 2025-06-25 21:04:47
2025-06-25 21:04:47.278 INFO 29748 --- [alert-exchange] route1 : [Consumer 2] Simulando envío por email: Alerta generada: 2025-06-25 21:04:47
2025-06-25 21:04:47.278 INFO 29748 --- [alert-exchange] route2 : [Consumer 1] Mensaje recibido: Alerta generada: 2025-06-25 21:04:47
```



Las dos colas: console.consumer.queue y email.consumer.queue.

Overview					Messages			Message rates				+/-
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack		
/	-915198684	classic	<span>D</span>	<span>running</span>	51	0	51	0.20/s				
/	console.consumer.queue	classic	<span>D</span>	<span>running</span>	0	0	0	0.20/s	0.20/s	0.00/s		
/	email.consumer.queue	classic	<span>D</span>	<span>running</span>	0	0	0	0.20/s	0.20/s	0.00/s		

## Queue console.consumer.queue

Overview

Queued messages [last minute](#) ?



Ready 0  
Unacked 0  
Total 0

Message rates [last minute](#) ?



Publish 0.20/s  
Consumer ack 0.00/s  
Get (auto ack) 0.00/s  
Deliver (manual ack) 0.00/s  
Redelivered 0.00/s  
Get (empty) 0.00/s  
Deliver (auto ack) 0.20/s  
Get (manual ack) 0.00/s

Details

Features	queue durable: true queue storage version: 1	State	running	Messages	0	Total	0	Ready	0	Unacked	0	In memory	0	Persistent	0	Transient, Paged Out	0
Policy		Consumers	1	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B
Operator policy		Consumer capacity	100%	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB
Effective policy definition																	

## Queue email.consumer.queue

Overview

Queued messages [last minute](#) ?



Ready 0  
Unacked 0  
Total 0

Message rates [last minute](#) ?



Publish 0.00/s  
Consumer ack 0.00/s  
Get (auto ack) 0.00/s  
Deliver (manual ack) 0.00/s  
Redelivered 0.00/s  
Get (empty) 0.00/s  
Deliver (auto ack) 0.00/s  
Get (manual ack) 0.00/s

Details

Features	queue durable: true queue storage version: 1	State	running	Messages	0	Total	0	Ready	0	Unacked	0	In memory	0	Persistent	0	Transient, Paged Out	0
Policy		Consumers	1	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B	Message body bytes	0 B
Operator policy		Consumer capacity	100%	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB	Process memory	67 KIB
Effective policy definition																	

## Queue -915198684

Overview

Queued messages [last minute](#) ?



Ready 76  
Unacked 0  
Total 76

Message rates [last minute](#) ?



Publish 0.20/s

Details

Features	queue durable: true queue storage version: 1	State	running	Messages	76	Total	76	Ready	76	Unacked	0	In memory	1	Persistent	0	Transient, Paged Out	75
Policy		Consumers	0	Message body bytes	2.7 KIB	Message body bytes	2.7 KIB	Message body bytes	2.7 KIB	Message body bytes	0 B	Message body bytes	36 B	Message body bytes	0 B	Message body bytes	2.6 KIB
Operator policy		Consumer capacity	0%	Process memory	193 KIB	Process memory	193 KIB	Process memory	193 KIB	Process memory	193 KIB	Process memory	193 KIB	Process memory	193 KIB	Process memory	193 KIB
Effective policy definition																	

Esa cola con un nombre generado automáticamente apareció porque, durante una ejecución anterior con errores, uno de los consumidores se conectó al exchange sin especificar un nombre de cola (?queue=...). En esos casos, RabbitMQ crea una cola temporal con un nombre único para ese consumidor. Ahora que el código está corregido, esa cola quedó huérfana y sin un consumidor activo, por lo que simplemente acumula los mensajes que recibe del publicador y puede ser eliminada de forma segura.

## *Preguntas a las respuestas:*

### **i. ¿Qué patrón de integración se aplicó?**

En esta práctica, implementamos el patrón de integración Publicar-Suscribir (Publish-Subscribe o Pub/Sub). La idea de este patrón es bastante directa: teníamos un componente, al que llamamos "publicador", que se encargaba de generar y enviar mensajes de alerta. Sin embargo, este publicador no enviaba las alertas directamente a quienes debían recibirlas. En lugar de eso, las publicaba en un canal central, que en nuestro caso fue un "exchange" en RabbitMQ. Por otro lado, teníamos dos "suscriptores" que estaban "escuchando" ese canal. Cada vez que el publicador enviaba una alerta, el canal se encargaba de distribuir una copia de ese mismo mensaje a cada uno de nuestros suscriptores.

### **ii. ¿Cómo se logró el desacoplamiento productor-consumidor?**

El desacoplamiento fue la clave de toda la práctica y lo logramos gracias a RabbitMQ, que actuó como un intermediario. El publicador (PublisherRoute) y los dos consumidores (Consumer1Route y Consumer2Route) nunca se comunicaron directamente entre sí.

El publicador solo conocía la dirección del "exchange" de RabbitMQ. Su única tarea era enviar los mensajes de alerta a ese punto central, sin preocuparse por quién o cuántos los recibirían.

Los consumidores, por su parte, solo conocían la dirección de ese mismo "exchange". Cada uno se conectaba de forma independiente para recibir los mensajes. No sabían nada sobre el publicador, ni siquiera sabían de la existencia del otro consumidor.

Esta separación, donde RabbitMQ actúa como un "cartero" que gestiona la distribución de los mensajes, es lo que garantiza que los componentes estén desacoplados. Pueden funcionar, modificarse o incluso fallar de manera independiente sin afectar a los demás.

### **iii. Ventajas que observamos durante la práctica**

Al desarrollar el taller, notamos varias ventajas prácticas de usar este patrón:

**Flexibilidad para crecer:** Fue evidente lo fácil que sería escalar el sistema. Si mañana necesitamos un tercer consumidor que guarde las alertas en una base de datos o envíe una notificación a un celular, solo tendríamos que crearlo y suscribirlo al mismo canal. No necesitaríamos tocar ni una línea de código del publicador original.

**Tareas diferentes con la misma información:** Vimos que, aunque el mensaje era el mismo para ambos suscriptores, cada uno podía hacer algo completamente diferente con él. Uno simplemente lo mostraba en la consola, mientras que el otro simulaba enviarlo por correo. Esto es muy útil en sistemas reales, donde un mismo evento puede disparar diferentes procesos de negocio.

**Agilidad en la comunicación:** El publicador enviaba la alerta y se olvidaba de ella, quedando libre para hacer otras cosas (en nuestro caso, esperar 5 segundos para generar

la siguiente). No tenía que esperar a que los consumidores confirmaran la recepción, lo que hace que la comunicación sea mucho más rápida y eficiente.