Informe Técnico Punto 2 “TryRabbitMQ”

Simulación de Sistema de Alertas se seguridad con RabbitMQ

Jhonatan Steven Baron Suarez

SIstemas Distribuidos
(Ing Martín Chiquillo)

Ingeniería de Sistemas y Computación
Facultad de Ingeniería
UPTC-TUNJA
2025

Índice

1. Objetivo

El objetivo de este ejercicio es modelar un sistema de notificaciones de seguridad que clasifique eventos en tres niveles de alerta: informativa, advertencia y crítica. Se usó la herramienta TryRabbitMQ para simular una arquitectura de mensajería basada en RabbitMQ, implementando:

- Un exchange de tipo direct
- Tres colas enlazadas a diferentes niveles de alerta
- Enrutamiento de mensajes mediante routing keys
- Publicación de mensajes y recepción mediante consumers

2. Configuración técnica

Exchange

- Nombre: alerts.direct
- Tipo: direct
- Propósito: Enviar los mensajes solamente a las colas cuyos bindings coinciden exactamente con la routing key del mensaje. Esto permite un control fino del flujo de datos hacia las colas correctas.

Colas configuradas

Se crearon tres colas que representan los diferentes niveles de severidad de las alertas de seguridad. Cada una está diseñada para recibir solo los mensajes con la routing key correspondiente.

Nombre de la cola	función
queue_info	Cola para alertas de tipo informativo
queue_warning	Cola para alertas de advertencia (warning)
queue_critical	Cola para alertas críticas o severas

Bindings creados

Cada cola se enlazó al exchange alerts.direct usando la siguiente configuración de routing keys:

Nombre del binding	Exchange	Cola vinculada	Routing Key
binding_info	alerts.direct	queue_info	info
binding_warning	alerts.direct	queue_warning	warning
binding_critical	alerts.direct	queue_critical	critical

Mensajes publicados (producers)

Se enviaron tres mensajes con el Publisher de TryRabbitMQ, cada uno con una clave de enrutamiento diferente:

Routing Key	Payload del mensaje	Propósito
info	Alerta de información	Evento informativo sin urgencia
warning	Posible actividad sospechosa	Evento que puede requerir revisión
critical	Fuga de datos detectada	Evento urgente que requiere acción

Cada mensaje fue correctamente enrutado hacia la cola correspondiente.

3. Resultados observados

Después de añadir un consumer por cada cola, se observó lo siguiente:

Routing Key	Cola receptora	Mensaje recibido	Observación
info	queue_info	Alerta de información	Recibido correctamente solo en info
warning	queue_warning	Posible actividad	Recibido correctamente solo en

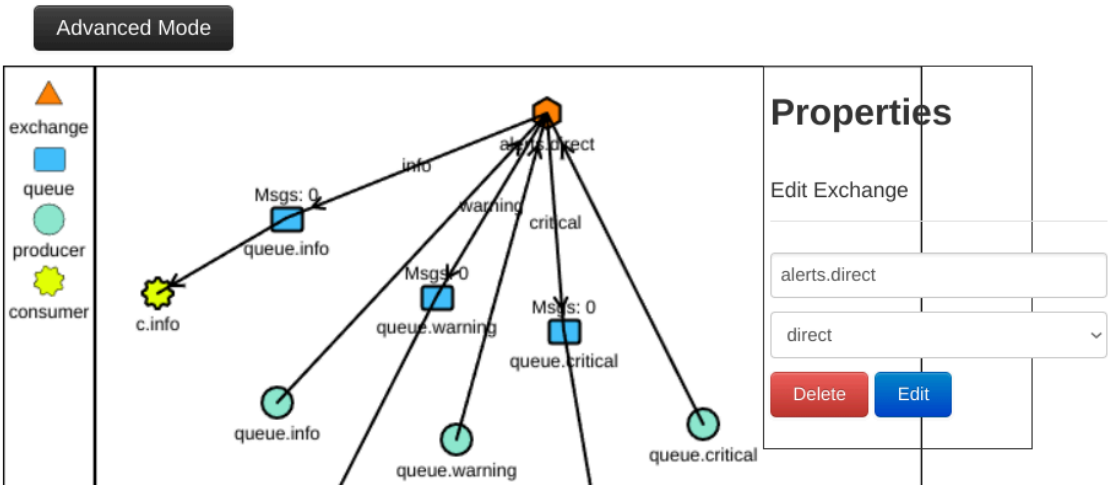
		sospechosa	warning
critical	queue_critical	Fuga de datos detectada	Recibido correctamente solo en critical

Este resultado valida el funcionamiento del exchange direct, asegurando un enrutamiento exacto por clave.

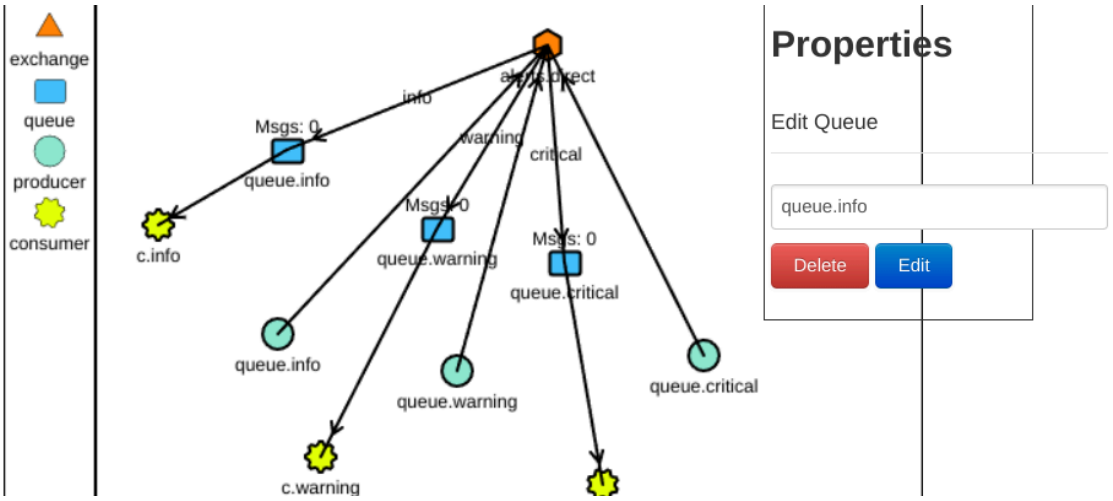
4. Evidencia visual

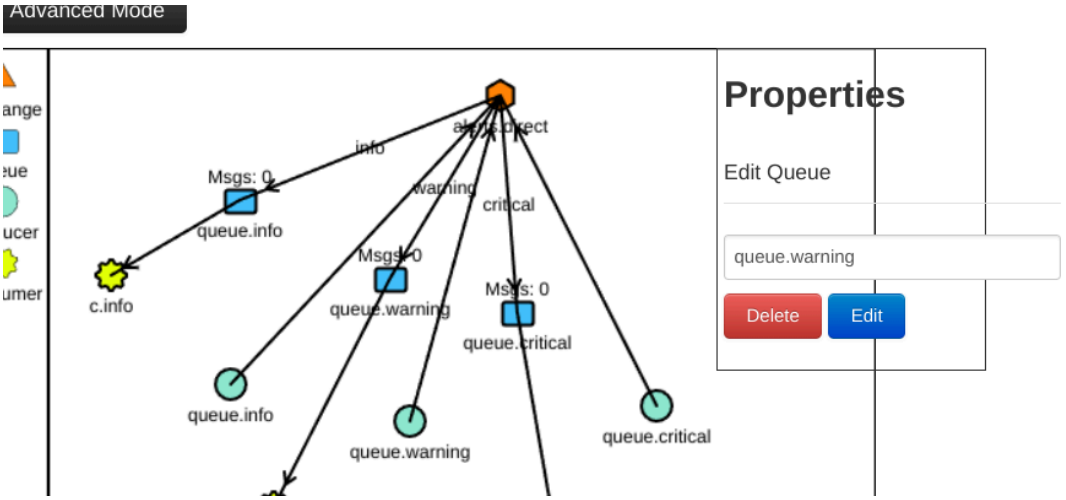
Se tomaron capturas de pantalla de las siguientes secciones en TryRabbitMQ (no se incluyen aquí por texto, pero deben estar en el informe final):

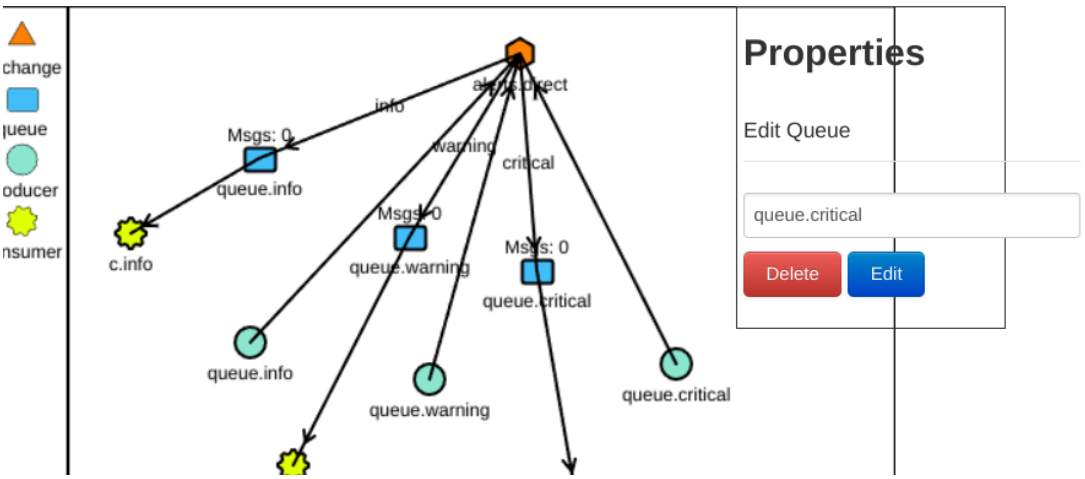
- Configuración del exchange alerts.direct



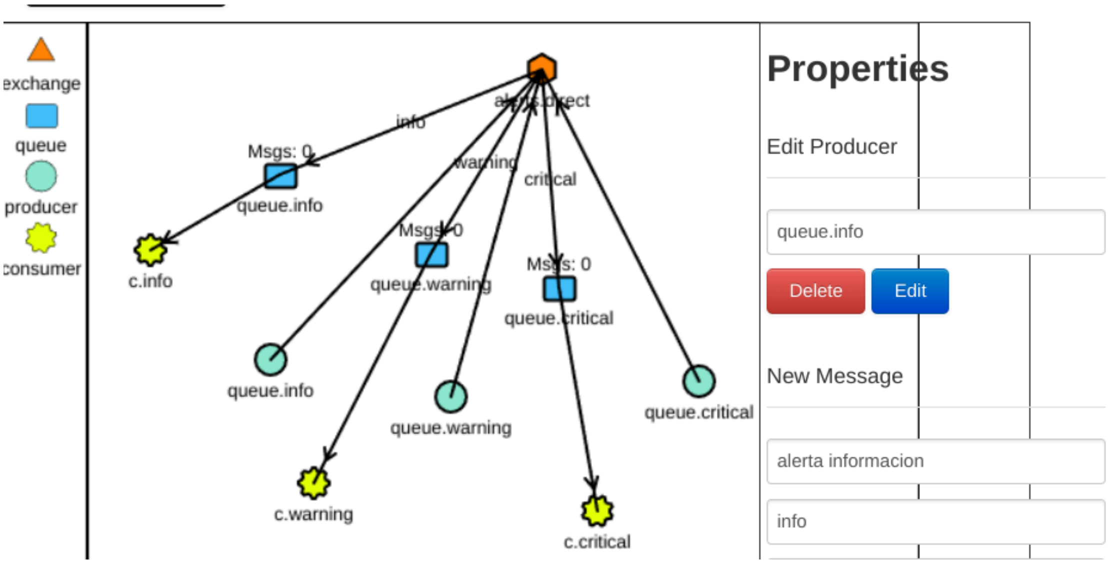
- Creación y binding de las colas

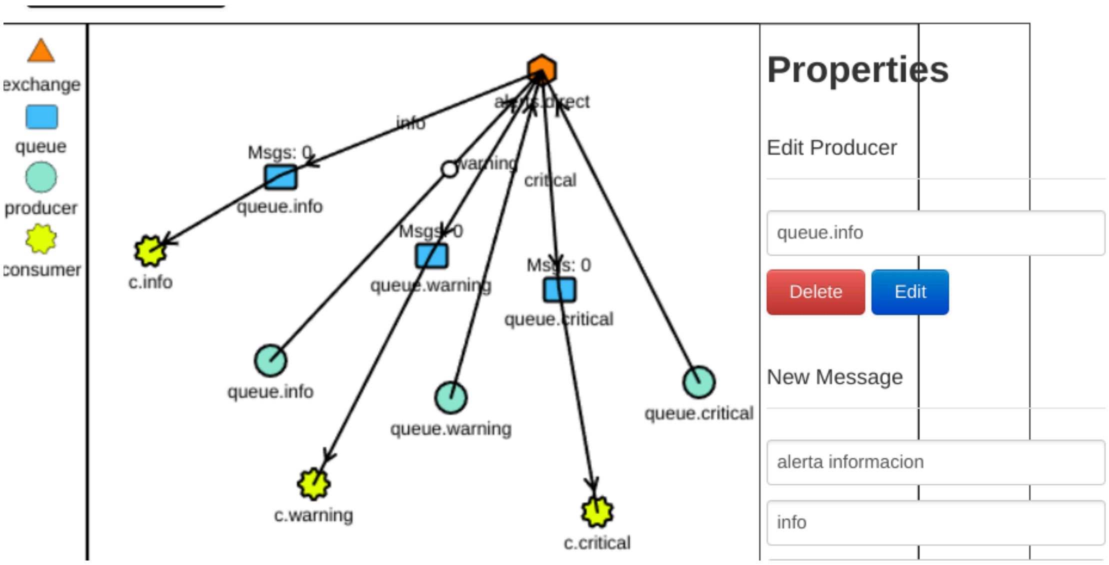


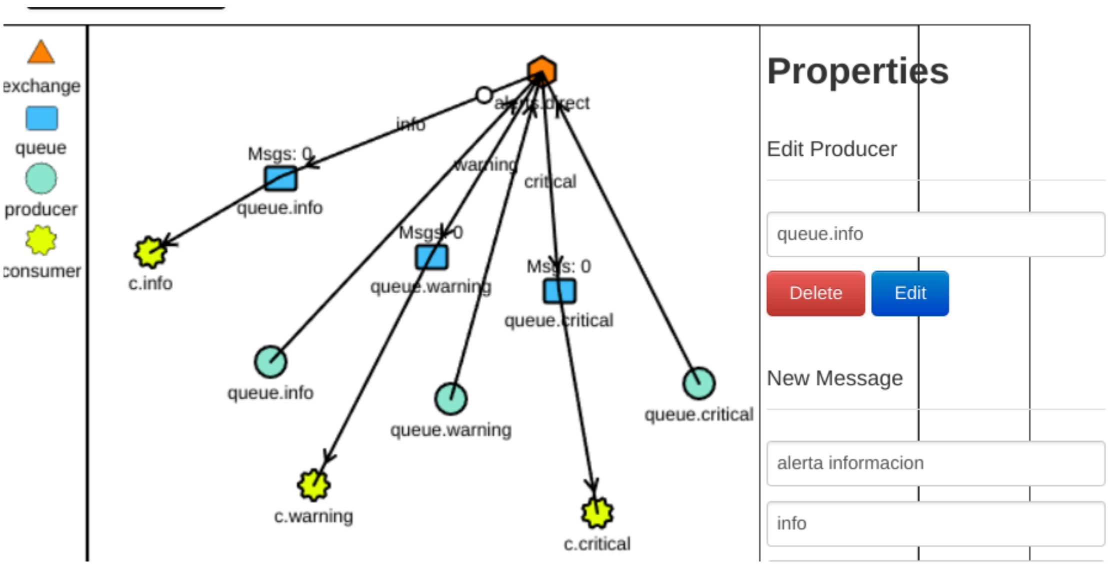


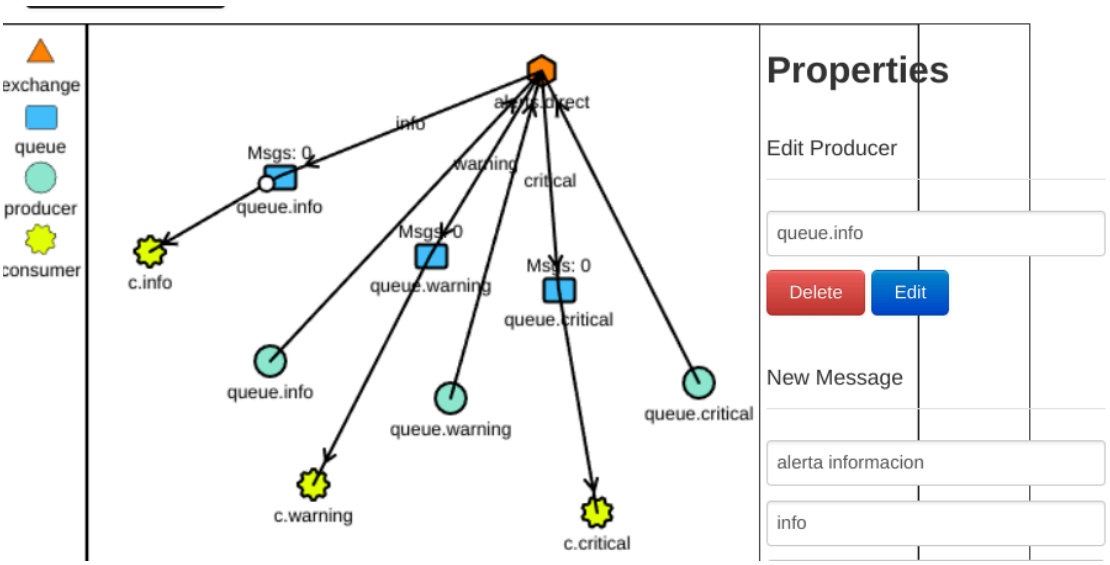


- Publisher enviando mensajes con distintas routing keys

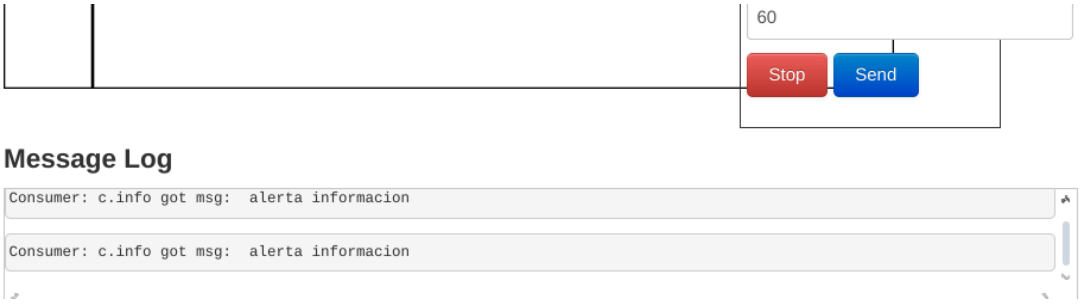








- Panel del consumer mostrando recepción del mensaje en la cola correspondiente



5. Conclusiones

- El exchange de tipo direct es ideal para sistemas en los que se requiere un ruteo exacto y segmentado por categorías.

- RabbitMQ permite desacoplar emisores y receptores, mejorando la escalabilidad del sistema.

- Esta arquitectura puede escalar fácilmente, agregando más colas con nuevas claves si aparecen más niveles de alerta.

- Es fundamental documentar los bindings para evitar errores de enrutamiento que podrían implicar pérdida de información crítica.