Universidad de Las Américas

Facultad de Ingenierías y Ciencias Agropecuarias

Ingeniería de Software

Informe de laboratorio

1. DATOS DEL ALUMNO:

Juan David Ramírez

2. TEMA DE LA PRÁCTICA:

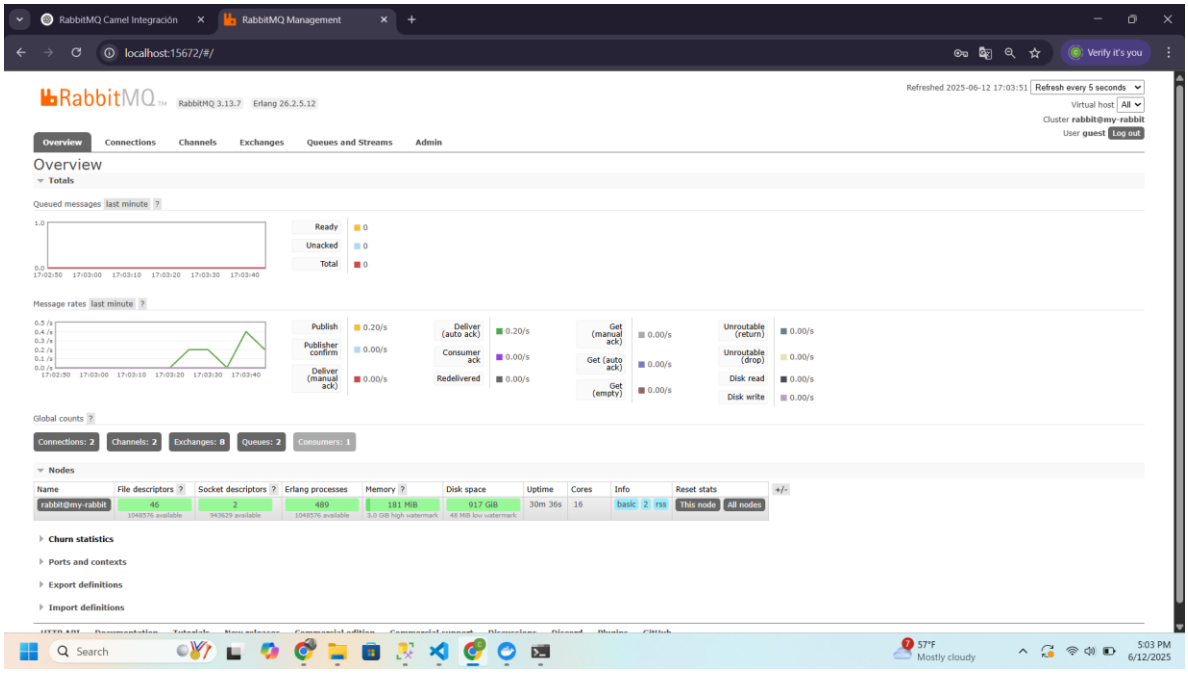
Taller – Configuración de RabbitMQ e Integración con Apache Camel

3. OBJETIVO DE LA PRÁCTICA

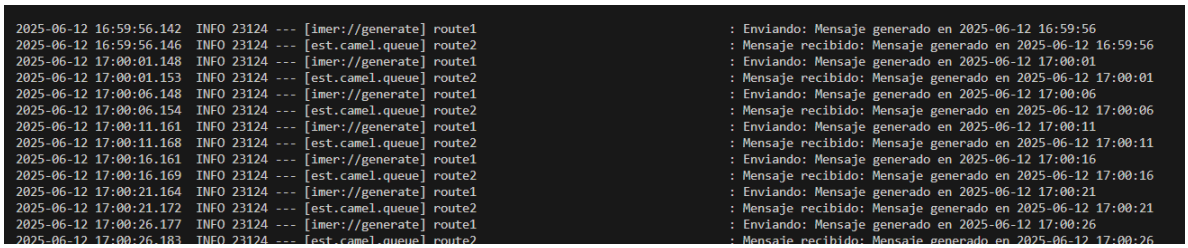
- Aplicar el patrón de mensajería asincrónica para demostrar el desacoplamiento entre productores y consumidores.
- Configurar un broker de mensajería RabbitMQ y conectar productores y consumidores de mensajes usando Apache Camel.

4. DESARROLLO DE LA PRÁCTICA Y RESULTADOS

i. Consola de RabbitMQ mostrando la cola



ii. Consola de logs de Camel (productor y consumidor funcionando)



iii. Documento breve explicando:

1. Qué patrón de integración se aplicó.

En este taller se aplicó el patrón de integración conocido como mensajería asincrónica, específicamente usando colas de mensajes. Este patrón es ampliamente utilizado cuando se busca que diferentes sistemas o componentes de software puedan comunicarse entre sí de manera desacoplada, sin necesidad de que estén activos o disponibles al mismo tiempo. Utilizando RabbitMQ como broker de mensajes, los datos enviados por un productor se almacenan temporalmente en una cola, hasta que algún consumidor esté listo para procesarlos, permitiendo así una integración más flexible y tolerante a fallos entre servicios.

2. Cómo se logró el desacoplamiento productorconsumidor.

El desacoplamiento entre productor y consumidor se logró a través del uso de RabbitMQ como intermediario de mensajes y Apache Camel como framework de integración. En lugar de que el productor y el consumidor se comuniquen directamente, el productor simplemente envía mensajes a una cola definida en RabbitMQ. Por su parte, el consumidor únicamente necesita estar suscrito a esa cola para recibir los mensajes. Así, el productor no necesita saber cuándo, cómo ni cuántos consumidores existen, ni siquiera si están activos en ese momento; y el consumidor tampoco depende de la lógica interna del productor. Este mecanismo elimina dependencias directas y permite que ambos lados evolucionen o escalen de manera independiente.

3. Ventajas que observaron durante la práctica.

Una de las principales ventajas observadas fue que la arquitectura basada en colas de mensajes facilita la separación de responsabilidades y simplifica la gestión de los flujos de datos. Además de esa manera el sistema es más tolerante a errores, ya que los mensajes se mantienen en la cola hasta ser procesados, incluso si el consumidor está temporalmente inactivo. También se hizo evidente la facilidad para escalar el sistema: es posible añadir más consumidores o productores sin realizar grandes cambios en la infraestructura.