Taller – Configuración de RabbitMQ e Integración con Apache Camel

Objetivo:

- 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**.

Tipo de Actividad: Individual

1. **Entregables del Taller**
   - Link a repositorio que contenga: https://github.com/matiasscs1/deberProgreso3Integracion.git
     a. Código fuente completo del proyecto (camel-rabbit-lab).
     b. Capturas de pantalla de:
        i. Consola de RabbitMQ mostrando la cola.
        ii. Consola de logs de Camel (productor y consumidor funcionando).
        iii. Documento breve (máx. 2 páginas) explicando

1. Patrón de Integración Utilizado

En este taller se implementó el patrón de mensajería asincrónica, utilizando RabbitMQ como intermediario entre los módulos productores y consumidores. Este patrón resulta esencial en arquitecturas desacopladas, ya que permite que los distintos componentes del sistema se comuniquen sin depender directamente unos de otros. El productor envía mensajes sin necesidad de saber quién los recibirá o si el receptor está disponible, ya que estos mensajes se almacenan temporalmente en una cola (en este caso, en RabbitMQ) hasta que el consumidor esté listo para procesarlos.

---

2. Separación entre Productor y Consumidor

La independencia entre productor y consumidor se logró mediante la implementación de RabbitMQ como sistema de mensajería. Apache Camel fue la herramienta utilizada para definir las rutas que gestionan tanto el envío como la recepción de los mensajes, utilizando conectores específicos de RabbitMQ.

- **Productor:** Se configuró una ruta en Camel que, mediante un temporizador, genera un mensaje cada 5 segundos y lo envía a la cola `test.camel.queue`.
- **Consumidor:** Se definió otra ruta que permanece a la espera de mensajes en la misma cola, procesándolos tan pronto como llegan.

Esta arquitectura permite que ambos componentes funcionen de forma autónoma, lo cual facilita el mantenimiento, mejora la escalabilidad del sistema y aporta mayor tolerancia ante fallos.

---

3. Beneficios Identificados

A lo largo del ejercicio se identificaron varias ventajas del uso conjunto de RabbitMQ y Apache Camel:

- **Integración sencilla:** Con poca configuración, Camel permite conectar distintos sistemas mediante flujos de mensajes.
- **Capacidad de escalar:** Al estar desacoplados, es posible aumentar la capacidad del productor o del consumidor de manera independiente.
- **Alta disponibilidad:** En caso de que el consumidor no esté activo, los mensajes permanecen en la cola hasta que puedan ser atendidos.
- **Monitoreo eficiente:** La interfaz de administración de RabbitMQ ofrece una visión clara y en tiempo real del flujo de mensajes.
- **Desarrollo ágil:** La combinación de Spring Boot con Camel proporciona una base sólida y rápida para construir aplicaciones de mensajería.