# **Aporte Redis y Kafka**

# **Bootcan de Java Tech Girl Power NTTDATA**

# **Proyecto:**

# **Sistema Bancario basado en Microservicios**

# **Integrante:**

ALEXIS PESCORAN FERNANDEZ

LUZ MARIA MOGOLLON RIVEROS

YESENIA LIZET PECHE VERGARAY

ANAIS BUSTAMANTE TORRES

# **Microservicios desarrollados:**

Account-MS

Customer-MS

Transaction-Ms

Fecha de entrega: 29 de septiembre de 2025

Empresa NTTDATA

# **Introducción**

El sistema bancario desarrollado durante el **Bootcamp de Java** se basa en una **arquitectura de microservicios**, la cual se compone de tres servicios principales:

* **Customer-MS**
* **Account-MS**
* **Transaction-MS**

En este contexto, herramientas como **Redis** y **Kafka** juegan un papel fundamental para mejorar aspectos clave como **rendimiento, escalabilidad, alta disponibilidad** y **eficiencia en la comunicación entre servicios.**

Mientras **Redis** se enfoca en mejorar la velocidad de acceso a datos mediante almacenamiento en memoria, **Kafka** asegura la comunicación confiable y en tiempo real entre microservicios, logrando una integración sólida y flexible en sistemas distribuidos.

A continuación, se presentan las definiciones de cada tecnología, un caso de uso real y su aplicación hipotética en nuestro sistema bancario.

## **Redis**

### **Definición**

Redis es una **base de datos en memoria**, de **código abierto**, diseñada para almacenar datos de manera temporal o persistente. Es ampliamente utilizada como **caché**, sistema de mensajería y almacenamiento rápido para optimizar el rendimiento de aplicaciones que requieren respuestas inmediatas.

### **Caso de uso real**

En el ámbito bancario, Redis puede ser utilizado para:

* **Gestión de sesiones:** Almacenar temporalmente información de sesión de los clientes, como credenciales o datos de autenticación, reduciendo la carga en la base de datos principal.
* **Caché de información crítica:** Guardar datos como el saldo de una cuenta, de forma que las consultas sean más rápidas y frecuentes sin saturar el sistema.
* **Procesamiento temporal de operaciones:** Manejar colas de transacciones bancarias en escenarios de alta demanda, asegurando que las operaciones se procesen sin pérdidas ni duplicaciones.

## **Kafka**

### **Definición**

Kafka es una plataforma distribuida de mensajería y procesamiento de flujos de datosen tiempo real. Permite que diferentes microservicios se comuniquen de manera asíncrona mediante la publicación y suscripción a eventos. Además, ofrece alta disponibilidad, tolerancia a fallos y escalabilidad, características esenciales para sistemas con alto volumen de transacciones.

### **Caso de uso real**

En el sector financiero, Kafka es utilizado para:

* Procesar en tiempo real grandes volúmenes de eventos como depósitos, retiros y transferencias.
* Coordinar operaciones entre diferentes servicios, garantizando que las transacciones se manejen de forma confiable y sin bloqueos.
* Implementar auditorías y trazabilidad, ya que cada evento puede ser registrado y monitoreado, mejorando la seguridad y transparencia.

## **Caso hipotético de uso en nuestros microservicios**

A continuación, se describe cómo Redis y Kafka podrían integrarse en cada uno de los microservicios desarrollados:

### **1. Customer-MS**

* **Redis:**
  + Almacenar información de sesión de los clientes, como credenciales y preferencias personalizadas.
  + Mantener un caché con datos frecuentes, reduciendo la cantidad de consultas a la base de datos principal.
* **Kafka:**
  + Emitir eventos cuando un cliente realiza acciones importantes, como registrarse, iniciar sesión o modificar su información.
  + Permitir que otros microservicios, como *Account*-MS o Transaction-MS, se suscriban a estos eventos para actualizar datos en tiempo real.

### **2. Account-MS**

* **Redis:**
  + Cachear los saldos de las cuentas para que las consultas sean rápidas y no dependan directamente de la base de datos persistente.
  + Guardar información temporal durante procesos de validación, como una transferencia en proceso.
* **Kafka:**
  + Publicar eventos cuando una cuenta se actualiza, por ejemplo, tras un depósito, retiro o transferencia.
  + Permitir que otros microservicios mantengan sus datos sincronizados y consistentes.

### **3. Transaction-MS**

* **Redis:**
  + Almacenar el **estado de las transferencias** durante su procesamiento, evitando duplicaciones y problemas de concurrencia.
  + Mejorar la velocidad de validación y procesamiento en escenarios de alto tráfico.
* **Kafka:**
  + Gestionar la **comunicación entre microservicios** durante cada fase de la transferencia: solicitud, verificación y finalización.
  + Garantizar que todos los pasos se procesen correctamente y que los servicios involucrados estén sincronizados en tiempo real.

## **Beneficios de integrar Redis y Kafka**

El uso combinado de Redis y Kafka en una arquitectura basada en microservicios ofrece múltiples ventajas:

1. **Escalabilidad:**
   * Permiten manejar grandes volúmenes de datos y transacciones de forma eficiente, algo esencial para sistemas bancarios con alta demanda.
2. **Alta disponibilidad:**
   * Redis y Kafka están diseñados para entornos críticos, asegurando que los servicios estén siempre operativos y disponibles.
3. **Mejora del desempeño:**
   * Redis optimiza el rendimiento gracias a su almacenamiento en memoria.
   * Kafka permite procesar y distribuir datos en tiempo real, agilizando la comunicación entre microservicios.
   * Ejemplo: Redis responde rápidamente a una consulta de saldo, mientras Kafka notifica a otros microservicios cuando dicho saldo cambia.
4. **Desacoplamiento de servicios:**
   * Kafka permite la comunicación asíncrona, lo que reduce la dependencia directa entre microservicios y evita cuellos de botella.

## **Conclusión**

La integración de **Redis** y **Kafka** en nuestro sistema bancario basado en microservicios permite construir una **arquitectura moderna, escalable y eficiente**.

Redis se enfoca en optimizar el acceso a datos y mejorar la velocidad de respuesta mediante almacenamiento en memoria, mientras que Kafka asegura una comunicación confiable y en tiempo real entre servicios.

Esta combinación no solo mejora el rendimiento actual del sistema, sino que también **prepara la plataforma para crecer** y adaptarse a futuras necesidades, garantizando la **consistencia, confiabilidad y eficiencia** en la gestión de operaciones críticas como cuentas, movimientos y transferencias bancarias.