

Sistema de eventos bancarios con Kafka → RN

ALUMNO: NICOLAS CARDINAUX

Link GitHub Repo:

<https://github.com/NicolasCardinaux/ProyectosProgMovil.git>

IMÁGENES AL FINAL DEL PDF.

01

Requerimientos

Objetivo: simular el ciclo de vida de una **transacción bancaria** (iniciar, reservar fondos, antifraude, commit/reversa, notificación) publicando/hearing eventos en Kafka y **mostrarlos en tiempo real** en una app **React Native** (vía un **gateway WebSocket**).

02

Servicios

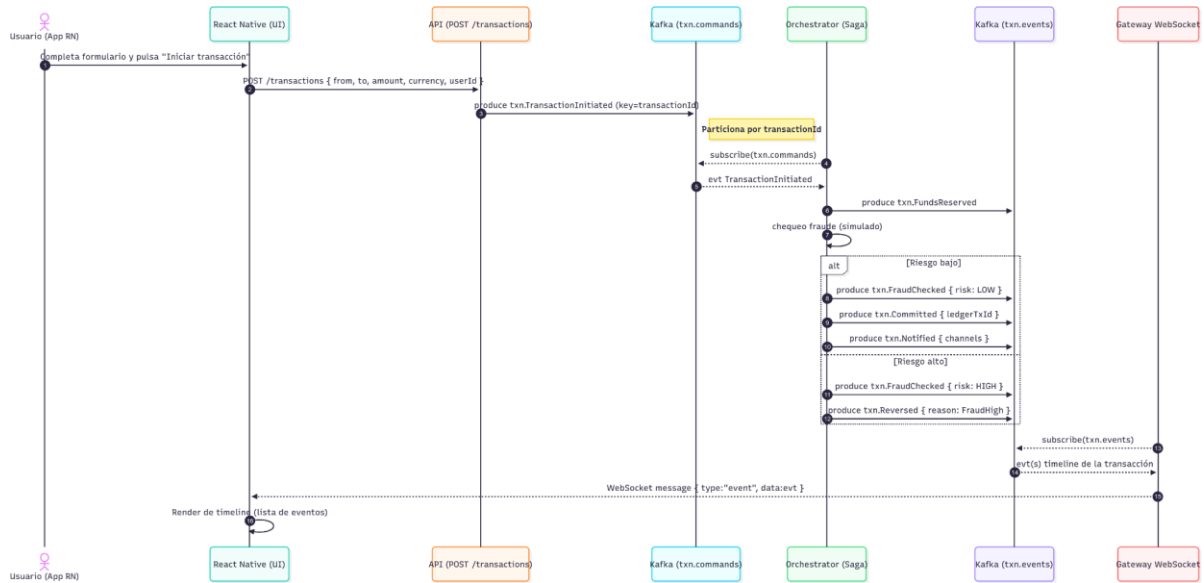
- **API** (`api.js`): recibe `POST /transactions` y publica `txn.commands`.
- **Orchestrator** (`orchestrator.js`): consume `txn.commands`, emite `txn.events` (FundsReserved, FraudChecked, Committed/Reversed, Notified) y maneja `txn.dlq` en errores.
- **Gateway WS** (`gateway.js`): consume `txn.events` y los reenvía por WebSocket a la app móvil (permite suscripción por `userId/transactionId`).
- **App RN** (`rn-txn`): inicia transacciones y visualiza el **timeline** en vivo.

Tópicos Kafka: `txn.commands`, `txn.events`, `txn.dlq`

Clave de partición: `transactionId` (garantiza orden por transacción).

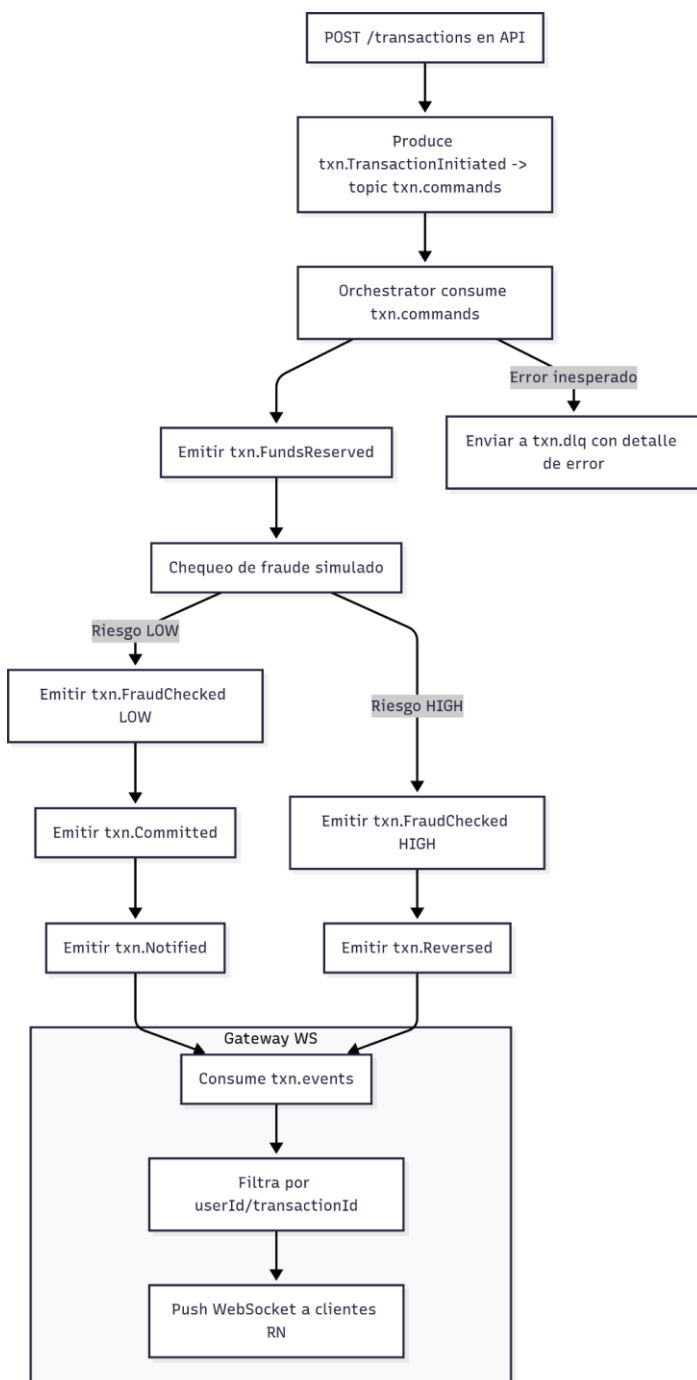
03

Diagrama de secuencia



04

Diagrama de Flujo



05

Contratos del evento

```
type EventEnvelope<T> = {  
  id: string;           // uuid v4  
  type: string;         // ej: "txn.FundsReserved"  
  version: number;      // 1  
  ts: number;           // epoch ms  
  transactionId: string; // partición  
  userId: string;  
  payload: T;  
  correlationId?: string;  
};
```

Ejemplos de `payload`:

- `TransactionInitiated: { fromAccount, toAccount, amount, currency, userId }`
- `FundsReserved: { ok: true, holdId, amount }`
- `FraudChecked: { risk: 'LOW' | 'HIGH' }`
- `Committed: { ledgerTxId }`
- `Reversed: { reason }`
- `Notified: { channels: string[] }`

06

Infra local

Docker

07

Backend

Nodejs - NestJS

08

App

React Native

09

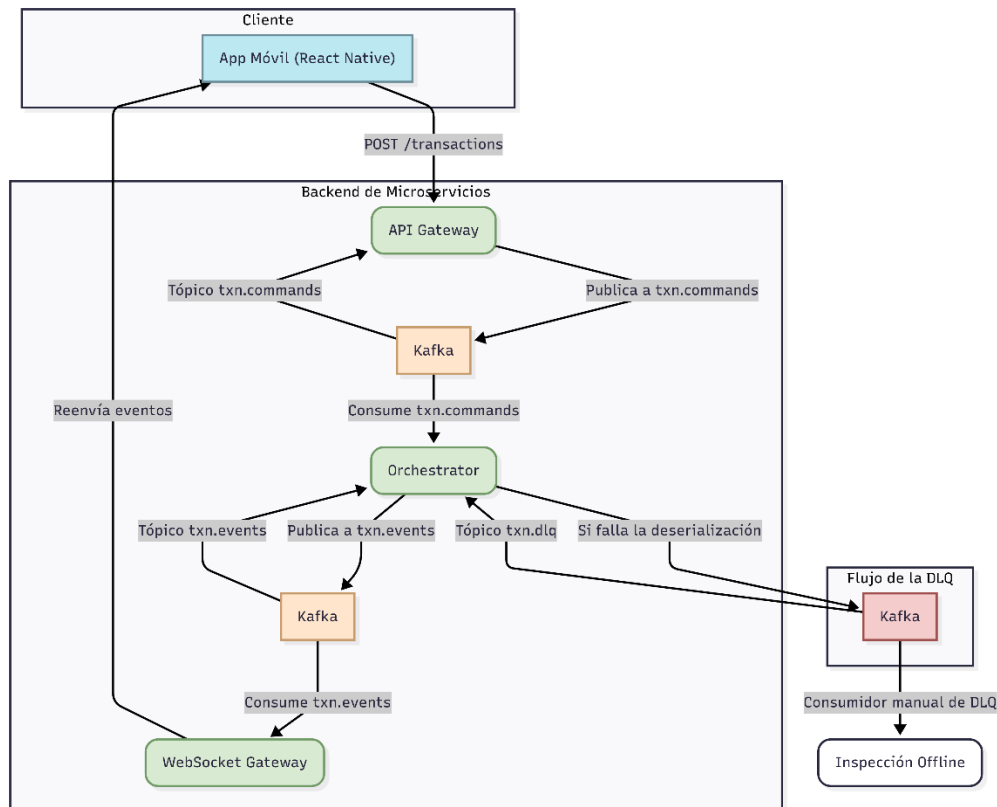
Qué se está practicando (buenas prácticas reales)

- **Orden por transacción:** usar `transactionId` como key \Rightarrow orden fuerte en `txn.events`.
- **Saga/Orquestación:** pasos con *retry simple*, *fraude simulado* y *rollback* (Reversed) si hace falta.
- **Contratos de evento:** un **envelope** estable evita dolores de versionado.
- **Aislamiento móvil:** RN no habla con Kafka; consume vía **WebSocket Gateway** (puede aplicar auth/JWT, rate-limit, wss, etc.).
- **Observabilidad:** podrías añadir un `/metrics` al gateway y contadores por tipo de evento.
- **DLQ:** errores “no recuperables” \rightarrow `txn.dlq` (inspección offline).

Licenciatura en Sistemas de Información Arq. Programación Móvil - 2025

- **Idempotencia** (siguiente paso): guarda *processed event ids* para no ejecutar dos veces efectos con side-effects (debito/credito).
- **Outbox pattern** (siguiente paso): si persistes estados en DB, utiliza Outbox para publicar eventos de forma transaccional.

Arquitectura:



Docker Compose levantado:

```

✓ backend-api Built
✓ backend-gateway Built
✓ backend-orchestrator Built
✓ Network backend_default Created
✓ Container zookeeper Started
✓ Container kafka Started
✓ Container backend_orchestrator Started
✓ Container backend_gateway Started
✓ Container backend_api Started
  
```

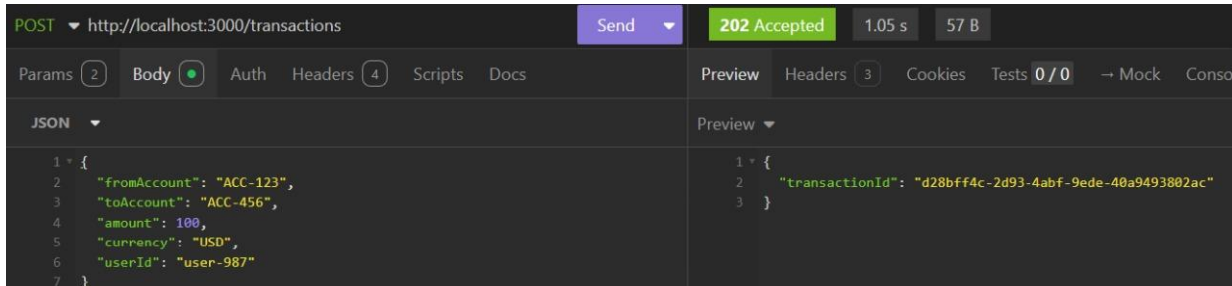
```

PS C:\Users\Nicolás Cardinaux\Desktop\Proyectos Prog. Movil\Tp4\backend> docker-compose up
time="2025-09-21T10:23:01-03:00" level=warning msg="C:\\Users\\Nicolás Cardinaux\\Desktop\\Pr
lease remove it to avoid potential confusion"
[+] Running 5/5
✓ Container zookeeper Running
✓ Container kafka Running
✓ Container backend_gateway Running
✓ Container backend_orchestrator Running
✓ Container backend_api Running
Attaching to backend_api, backend_gateway, backend_orchestrator, kafka, zookeeper
  
```

Licenciatura en Sistemas de Información

Arq. Programación Móvil - 2025

Kafka funcionando:



POST http://localhost:3000/transactions **202 Accepted** 1.05 s 57 B

Params (2) Body Auth Headers (4) Scripts Docs

JSON

```

1 {
2   "fromAccount": "ACC-123",
3   "toAccount": "ACC-456",
4   "amount": 100,
5   "currency": "USD",
6   "userId": "user-987"
7 }
  
```

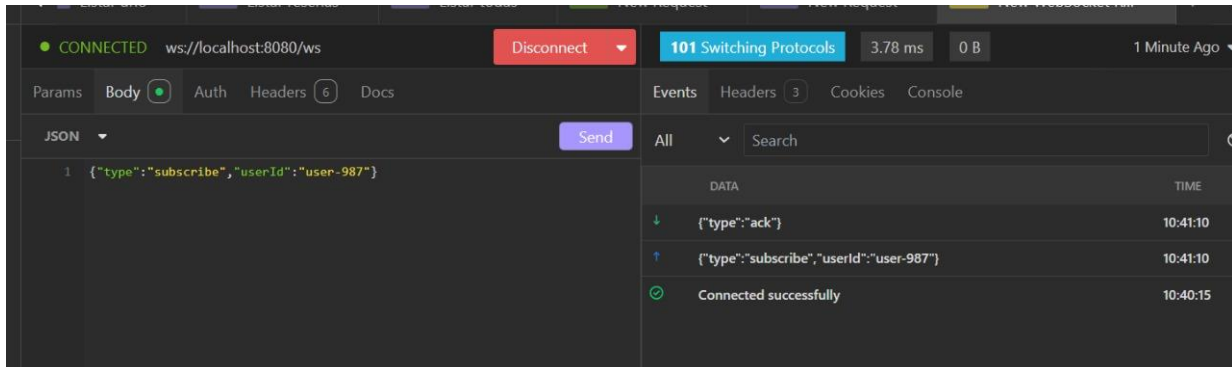
Preview Headers (3) Cookies Tests 0/0 → Mock Console

Preview

```

1 {
2   "transactionId": "d28bff4c-2d93-4abf-9ede-40a9493802ac"
3 }
  
```

Conexión WebSocket:



CONNECTED ws://localhost:8080/ws **101 Switching Protocols** 3.78 ms 0 B 1 Minute Ago

Params Body Auth Headers (6) Docs

JSON

```

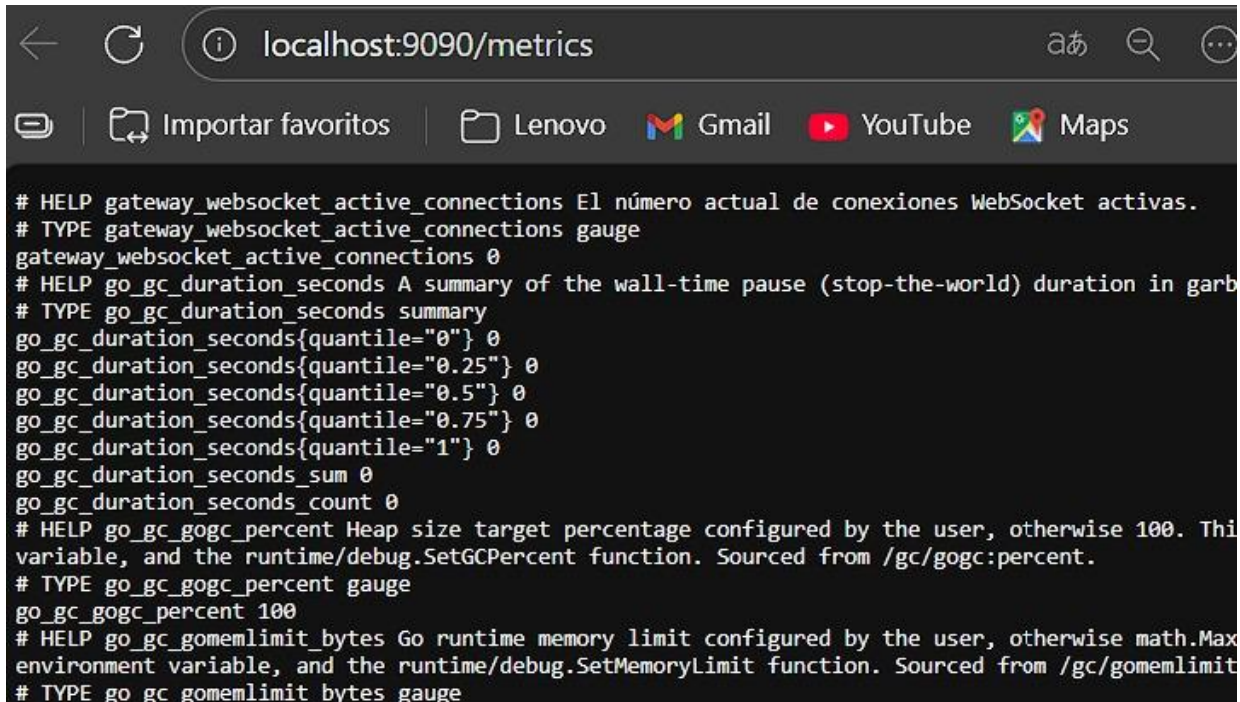
1 { "type": "subscribe", "userId": "user-987" }
  
```

Events Headers (3) Cookies Console

DATA TIME

- ↓ {"type": "ack"} 10:41:10
- ↑ {"type": "subscribe", "userId": "user-987"} 10:41:10
- ✓ Connected successfully 10:40:15

Métricas:



localhost:9090/metrics

Importar favoritos | Lenovo | Gmail | YouTube | Maps

```

# HELP gateway_websocket_active_connections El número actual de conexiones WebSocket activas.
# TYPE gateway_websocket_active_connections gauge
gateway_websocket_active_connections 0
# HELP go_gc_duration_seconds A summary of the wall-time pause (stop-the-world) duration in garbage collection events.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_gc_gogc_percent Heap size target percentage configured by the user, otherwise 100. This variable, and the runtime/debug.SetGCPercent function. Sourced from /gc/gogc:percent.
# TYPE go_gc_gogc_percent gauge
go_gc_gogc_percent 100
# HELP go_gc_gomemlimit_bytes Go runtime memory limit configured by the user, otherwise math.MaxInt64. This variable, and the runtime/debug.SetMemoryLimit function. Sourced from /gc/gomemlimit:bytes.
# TYPE go_gc_gomemlimit_bytes gauge
  
```

Licenciatura en Sistemas de Información Arq. Programación Móvil - 2025

Funcionamiento de la APP:

