Гарантована доставка повідомлень в мікросервісній архітектурі

Магістерська робота

Виконав: Смакула Роман
Керівник магістерської роботи: доктор техн. наук, доцент Глибовець А. М.

Постановка задачі

- Дослідити способи гарантування доставки повідомлень у мікросервісній архітектурі
- Реалізувати шаблон Transactional Outbox на мові програмування Kotlin та порівняти його швидкодію з аналогами

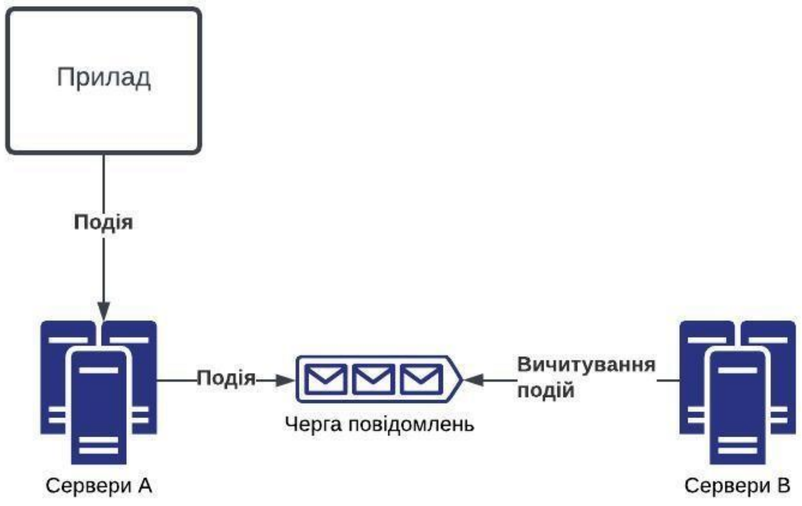
Опис предметної області

Прикладна проблема

Існує система IoT, в якій прилад надсилає сервісу А інформацію про події, які у нього виникають.

Відповідальність сервісу А обмежується комунікацією з приладом та збереженням поточного стану приладу.

Події обробляє та зберігає сервіс В.



Опис предметної області

Проблематика

- Сервіс А та В можуть втратити живлення в будь-який момент часу
- Зв’язок між сервісами може зникати
- Швидкість надсилання подій сервісом А інша від швидкості оброблення подій сервісом В
- Може існувати декілька екземплярів сервісу А та В; дублювання подій неприпустиме

Архітектурні шаблони

Аналіз рішень

- Transactional outbox
- 2PC
- Listen to yourself
- Event sourcing

Види реплікації черги повідомлень

Аналіз рішень

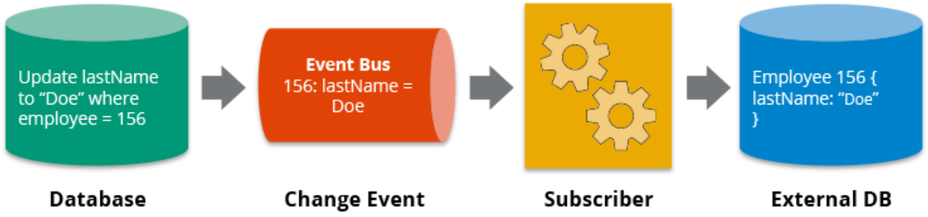
- Master/Slave
  - Classic
  - Quorum Queue
- Спільна файлова система

CDC для отримання змін даних

Аналіз рішень

Способи пошуку різниці між даними

- Використання допоміжних колонок
- Пошук різниці між таблицями
- Використання тригерів
- Використання лугу транзакцій



Реалізації CDC

Аналіз рішень

- Debezium
- Eventuate tram

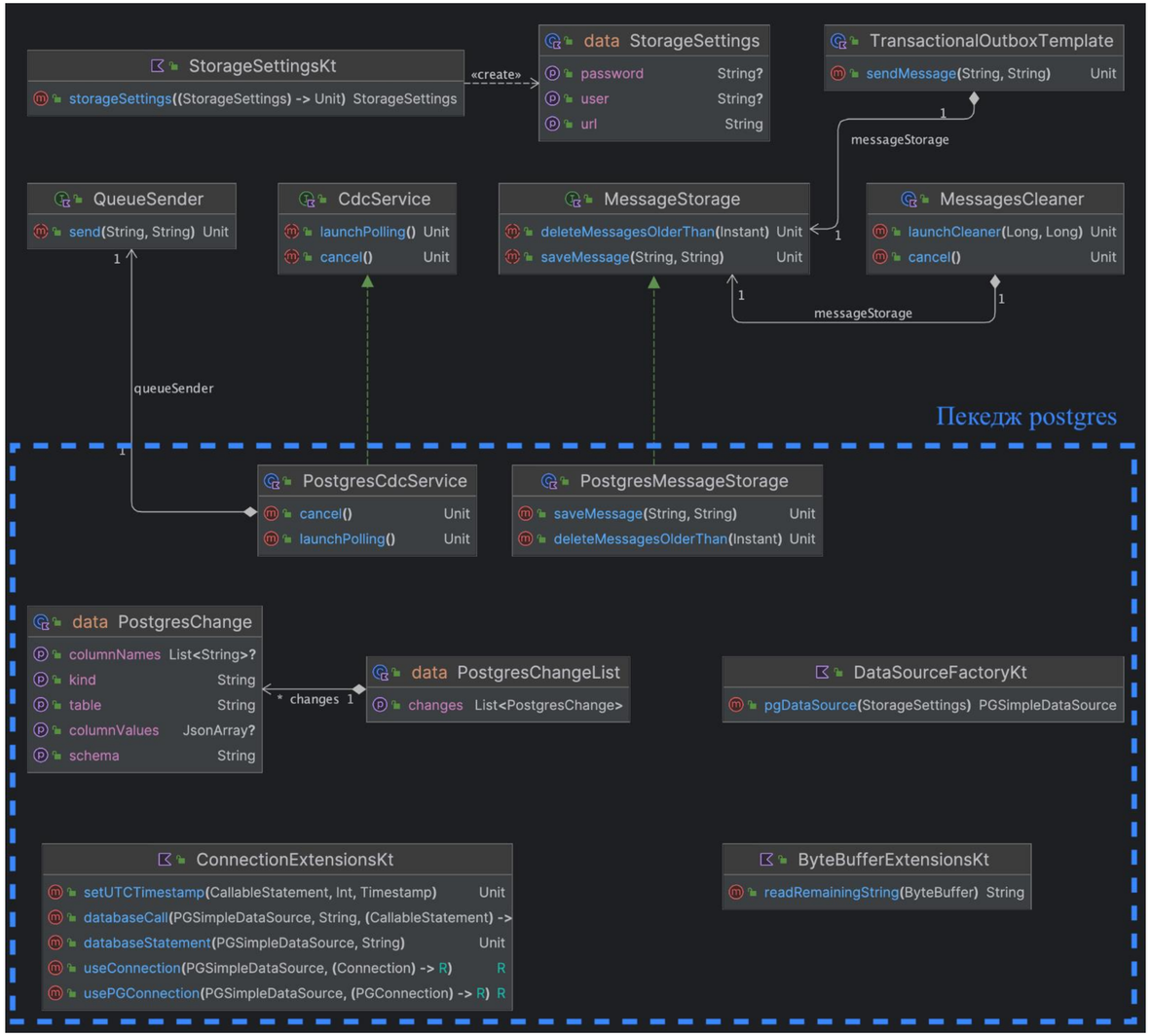
Реалізація Transactional Outbox

Бібліотека

- Мова програмування: Kotlin
- Спроектована виключно для реалізації шаблону TranactionalOutbox, на відміну від Debezium та Eventuate tram
- Підтримка вичитування логу транзакцій Postgres
- Легко інтегрується з Spring Boot

Бібліотека

Структура проєкту



Дослідження швидкодії запропонованого рішення

Компоненти експерименту

- Емулятор приладу
- Сервіси A та B написані з використанням фреймворку Spring Boot
- База даних PostgreSQL
- Кластер трьох черг RabbitMQ з використанням Quorum Queue

Результати дослідження

Заміри швидкості надсилання повідомлення з приладу на сервіс B

- Debezium: 547.57 ms
- Eventuate tram: 274.09 ms
- Власна бібліотека: 67.69 ms

Результати

- Проаналізовано архітектурні шаблони Transactional Outbox, 2PC, Listen to Yourself, Event sourcing
- Досліджено методи резервування черги повідомлень
- Розроблено бібліотеку на мові програмування Kotlin, яка надає інструменти для реалізації шаблону Transactional Outbox
- Оцінено швидкодію розробленої бібліотеки та її аналогів Debezium, Eventuate tram

Висновки

Використання шаблонів Transaction Outbox, 2PC, Listen to Yourself, Event Sourcing з чергою, яка зарезервована за допомогою механізму Quorom Queue, не гарантує доставки повідомлень, проте зменшить ймовірність можливості втрати повідомлення.

Запропоновану реалізацію Transactional Outbox можна використовувати у застосунках, які написані на Kotlin або на мові, що компілюється в Java bytecode. Розроблену бібліотеку адаптовано під використання разом з фреймворком Spring Boot.

Дякую за увагу!