Асинхронна передача даних в мікросервісних архітектурах за допомогою черг повідомлень

Стасько Тарас

Науковий керівник: Глибовець А.М.
Доктор технічних наук, Декан
Факультету Інформатики

Актуальність теми

Синхронна комунікація в мікросервісах призводить до тісного зв’язку між сервісами, каскадних збоїв, складностей масштабування.

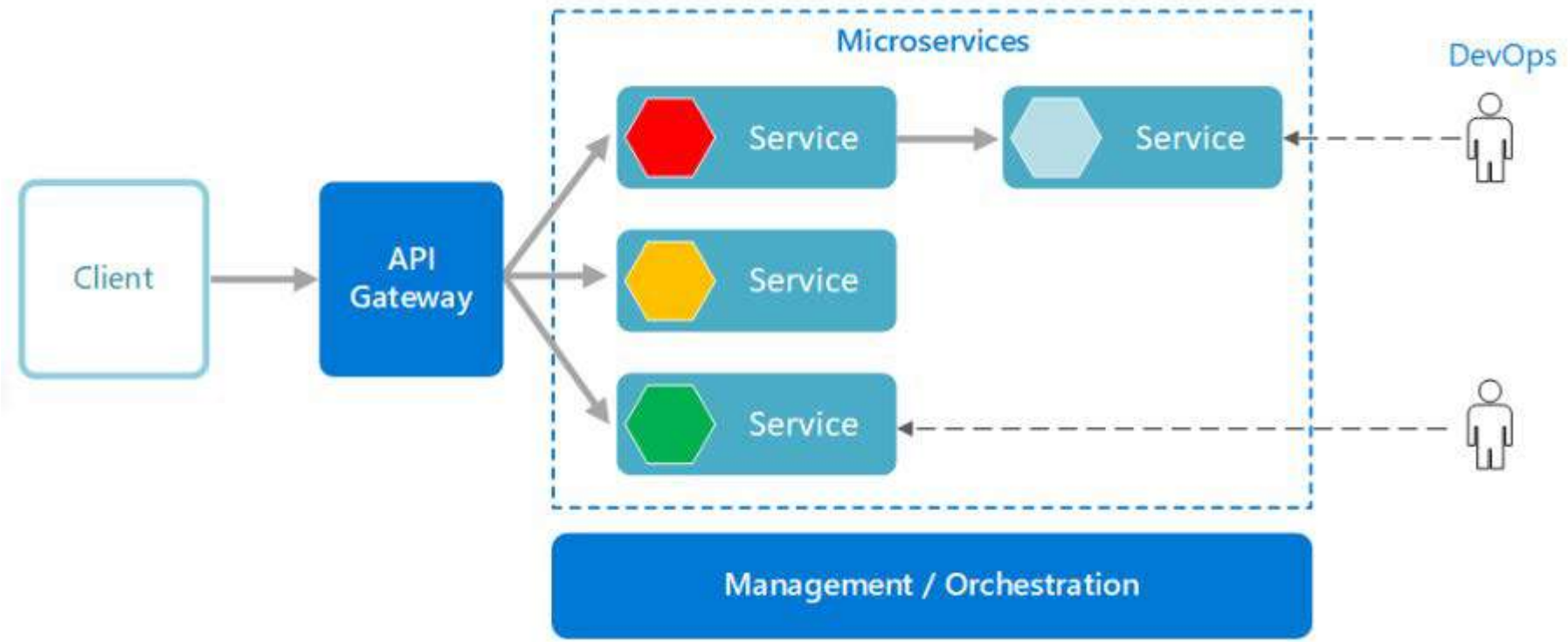
Асинхронна передача даних через черги повідомлень допомагає вирішити ці проблеми

Мета та завдання дослідження

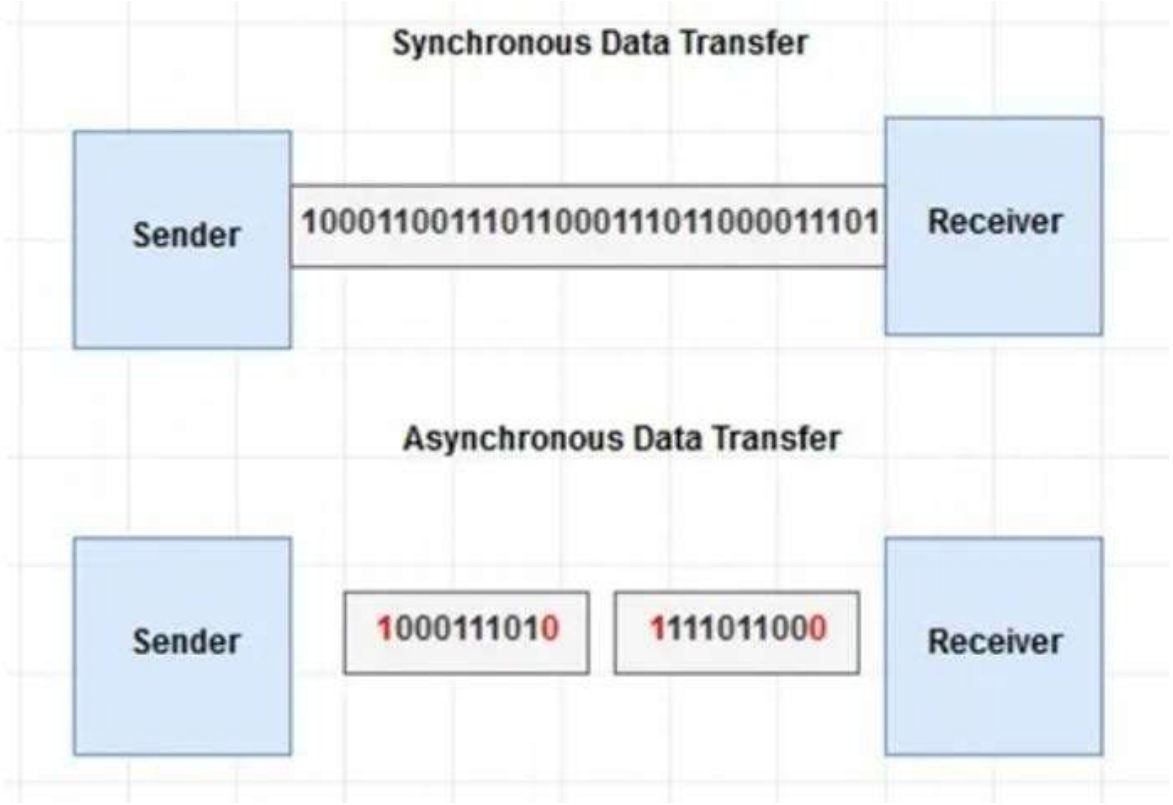
- Дослідити принципи асинхронної передачі даних у мікросервісах та визначити оптимальні підходи.

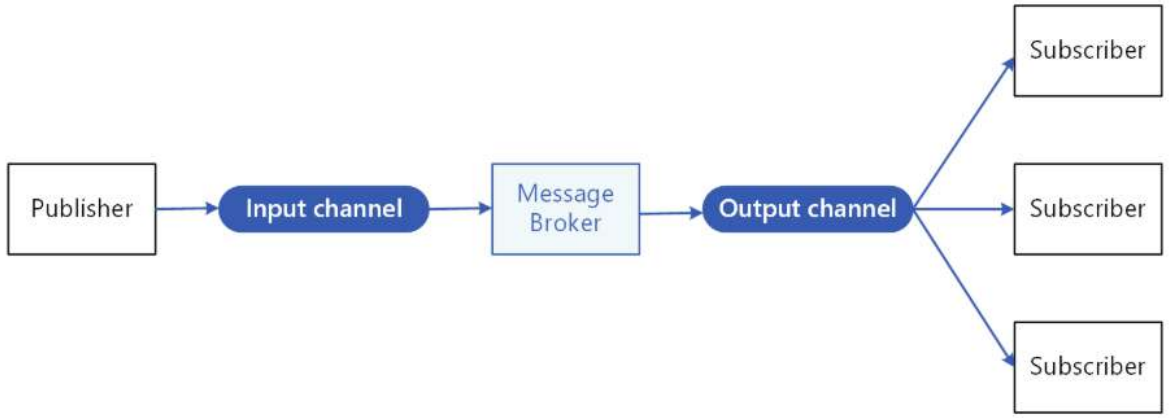
- Завдання:
  - Аналіз проблем синхронної комунікації.
  - Дослідження асинхронних патернів (Pub/Sub, Saga, CQRS).
  - Порівняння брокерів повідомлень.
  - Оптимізація продуктивності та безпеки.

Мікросервісна архітектура

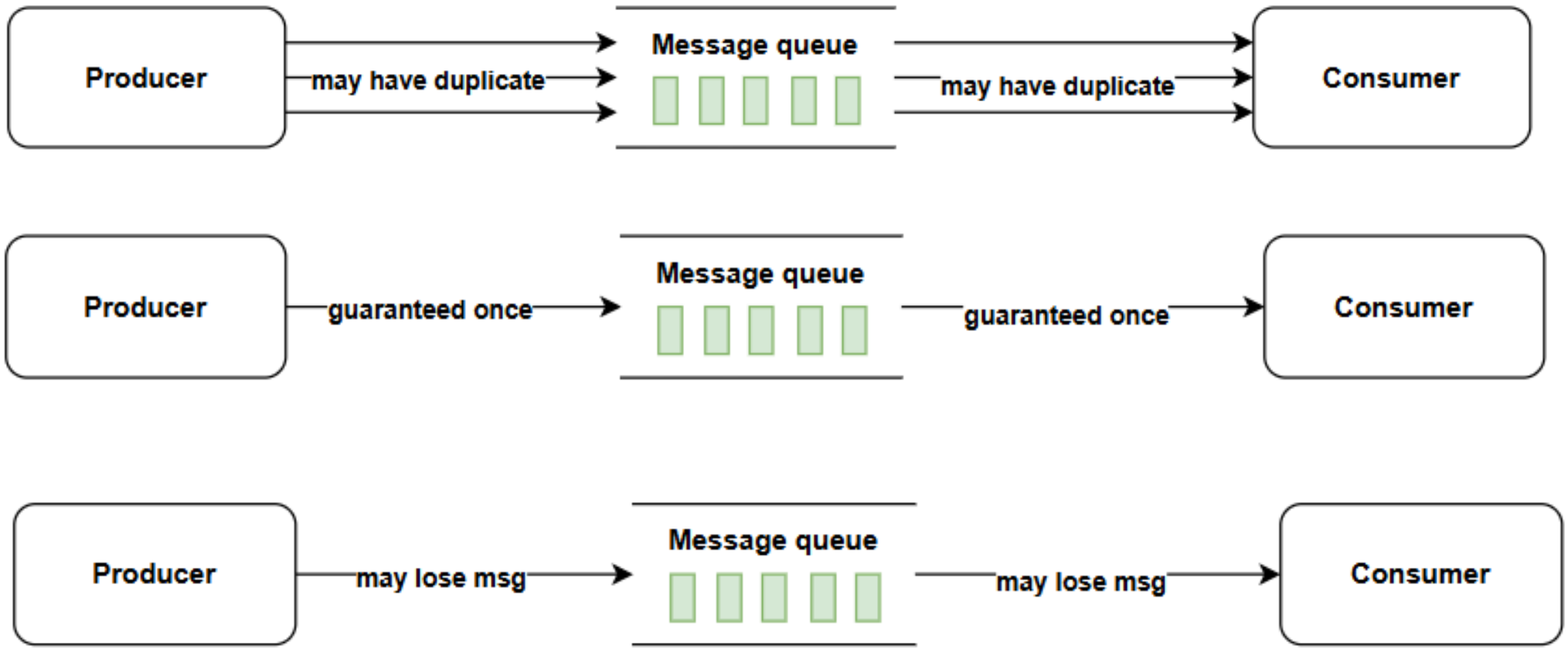


Асинхронна передача даних





Черги повідомлень



Практична реалізація

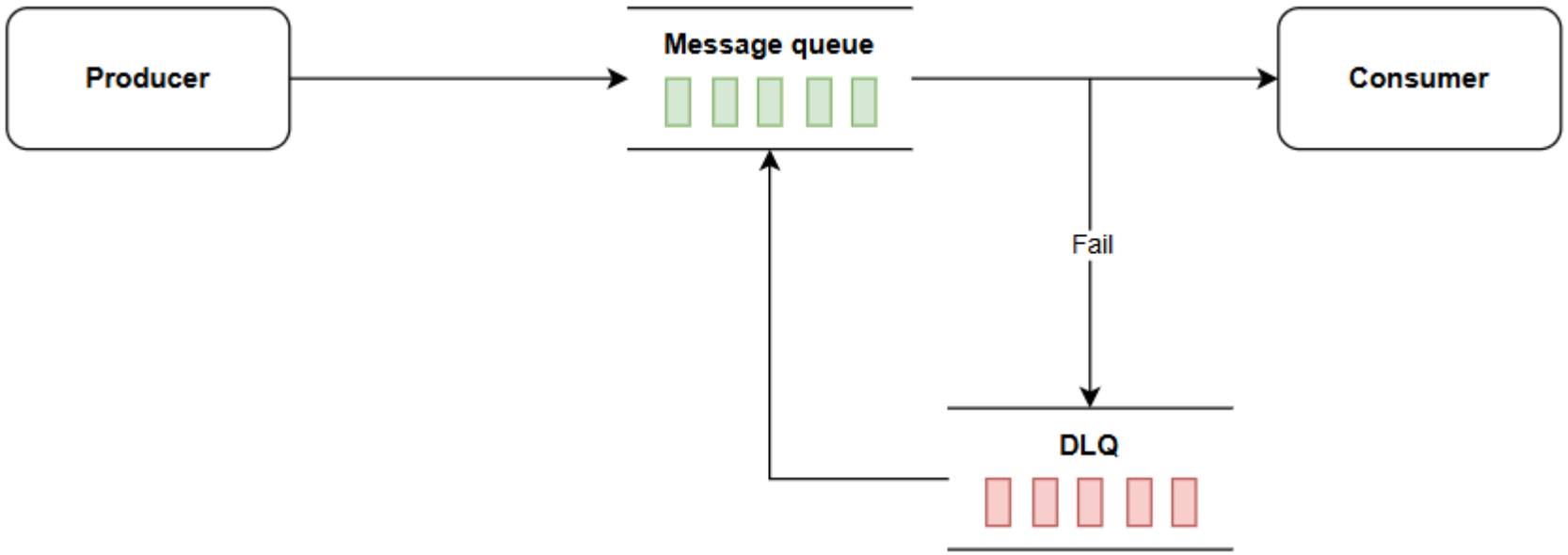
Мета:
Реалізувати обробку повідомлень через чергу з підтримкою повторних спроб і черги мертвих листів (DLQ).

Компоненти:

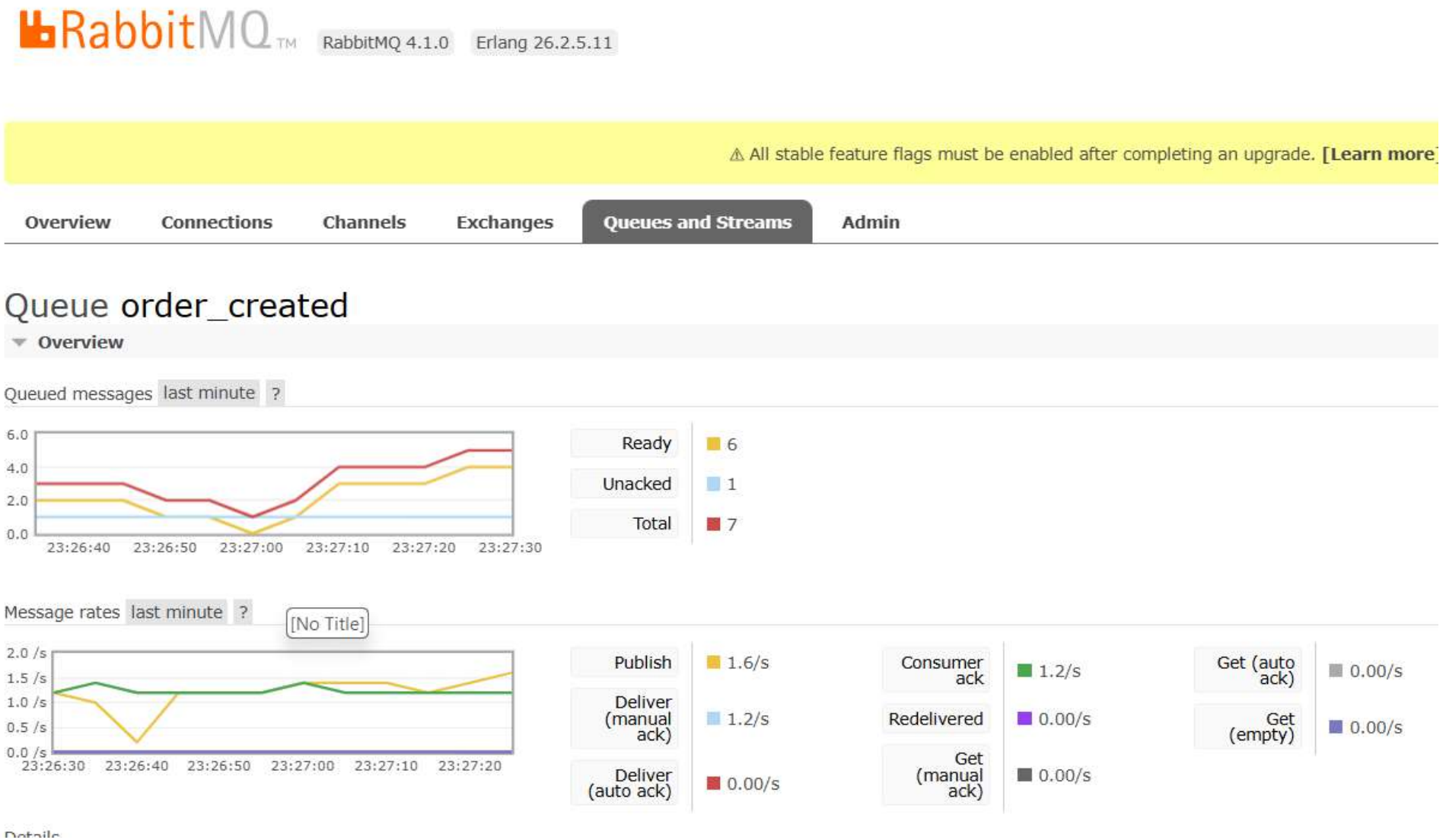
- **FastAPI Producer**: генерує та надсилає замовлення в RabbitMQ.
- **RabbitMQ Broker**: обробляє черги, маршрутизацію та повторні спроби.
- **Async Consumer**: читає повідомлення з черги, обробляє з можливими помилками.
- **Dead-Letter Queue (DLQ)**: отримує повідомлення, які не вдалося обробити після кількох спроб.

Практична реалізація

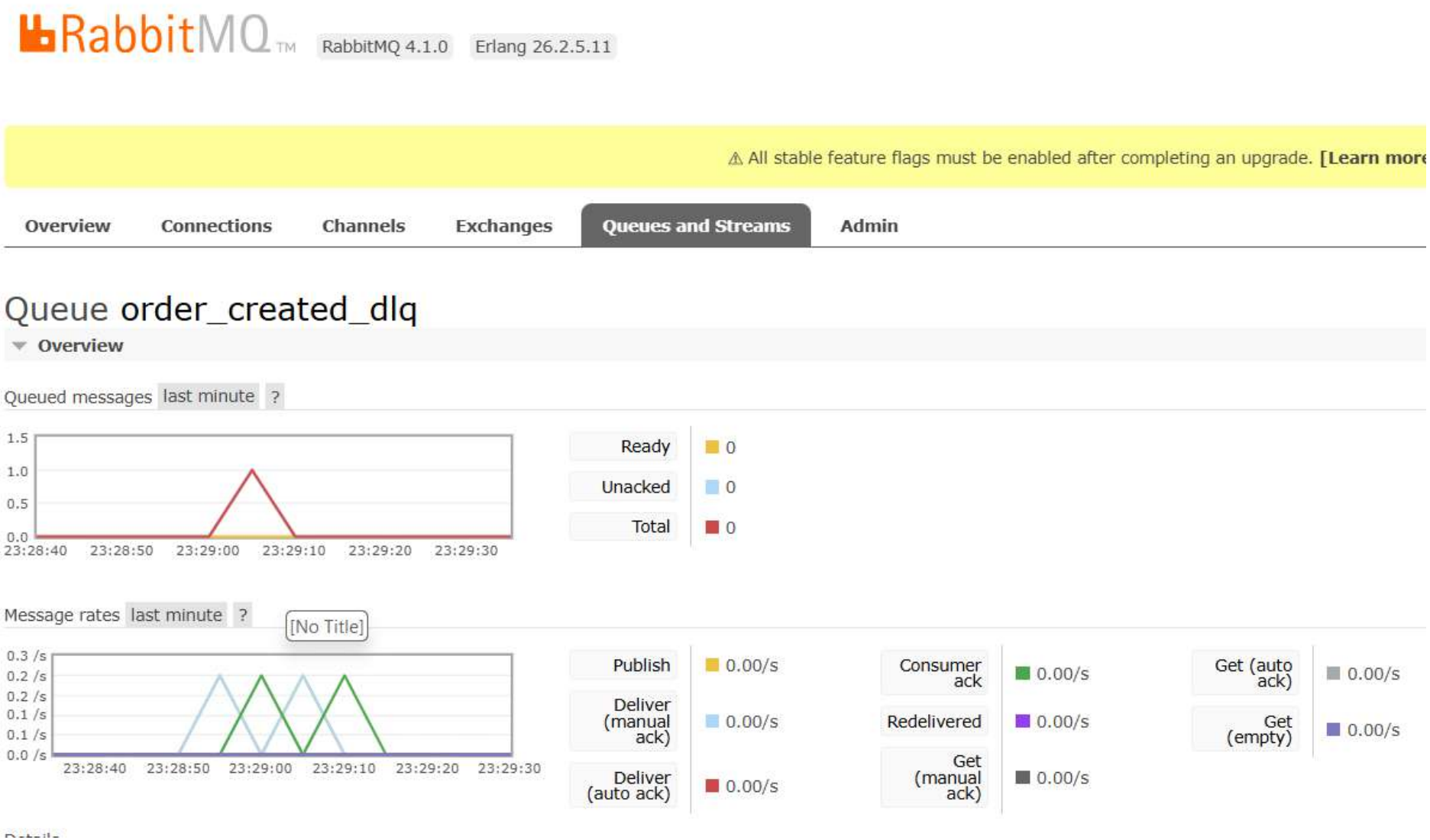
Схема взаємодії:



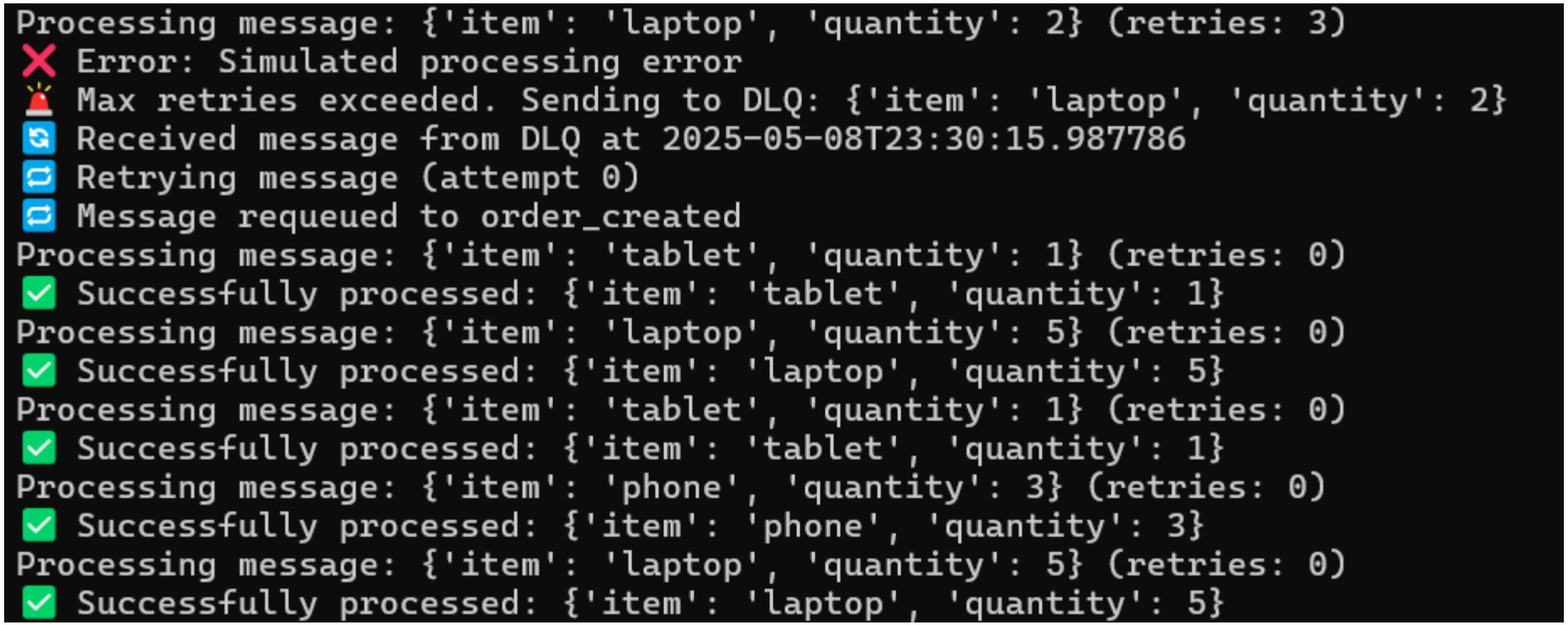
Робота системи



Робота системи



Робота системи



Приклади коду

- Генерація замовлень

```python
@app.on_event("startup")
async def startup_event():
    asyncio.create_task(create_and_send_orders())
```

- Відправка в чергу

```python
await channel.default_exchange.publish(
    aio_pika.Message(
        body=message_body,
        delivery_mode=aio_pika.DeliveryMode.PERSISTENT
    ),
    routing_key="order_created"
)
```

Приклади коду

- Логіка Retry:

```
if retries > MAX_RETRIES:
    await send_to_dlq(message.body)
else:
    await retry_message(message.body, retries)
```

- Налаштування DLX у RabbitMQ:

```
arguments = {
    "x-dead-letter-exchange": "dlx",
    "x-dead-letter-routing-key": "dlq"
}
```

Використані джерела

- https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices
- https://pediaa.com/what-is-the-difference-between-synchronous-and-asynchronous-data-transfer/
- https://learn.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber
- https://blog.bytebytego.com/p/at-most-once-at-least-once-exactly

Дякую за увагу!