ТЕМА 13

Міжсервісна взаємодія

Синхронна: REST, gRPC (оглядово) · Асинхронна: RabbitMQ, Redis

Бекенд-розробка, сервіси та мікросервіси

Комп'ютерна інженерія · 3 курс · Лекція 13 з 18

ПЛАН ЛЕКЦІЇ

7 питань сьогоднішньої теми

01

Синхронна vs асинхронна: коли яка

02

REST на практиці: таймаути й retry

03

gRPC: огляд альтернативи REST

04

Каскадні збої та Circuit Breaker

05

Черги повідомлень: Publish/Subscribe

06

RabbitMQ на практиці

07

Redis: Pub/Sub та черги задач

Коли яка потрібна

Критерій	Синхронна (REST/gRPC)	Асинхронна (черга)
Очікування відповіді	Так, негайно	Ні, з затримкою
Доступність обох сторін	Обидві одночасно	Отримувач може бути offline
Типовий сценарій	"Дані про товар зараз"	"Подія, обробити коли зручно"
Складність	Нижча	Вища (потрібна інфраструктура)

Таймаути: не чекати вічно

```
$ch = curl_init('http://products-service/api/products/42');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_TIMEOUT, 3);           // макс. 3с на всю операцію
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 1);    // макс. 1с на з'єднання

$response = curl_exec($ch);

if (curl_errno($ch) === CURLE_OPERATION_TIMEDOUT) {
    error_log('Products service timeout');
}
```

Без таймауту запит до недоступного сервісу "зависає" на невизначений час.

Повторні спроби: експоненційний backoff

```
function callWithRetry(callable $request, int $maxAttempts = 3) {
    $attempt = 0;
    while (true) {
        try {
            return $request();
        } catch (RequestException $e) {
            $attempt++;
            if ($attempt >= $maxAttempts) throw $e;
            sleep(2 ** $attempt);   // 2, 4, 8 секунд
        }
    }
}
```

404 повторювати безглуздо. Мережеві таймаути й 500/503 — варто.

REST vs gRPC

Критерій	REST	gRPC
Формат даних	JSON (текстовий)	Protocol Buffers (бінарний)
Протокол	HTTP/1.1	HTTP/2
Швидкість	Нижча	Вища
Читабельність	Висока	Низька
Сфера застосування	Публічні API	Внутрішня комунікація

Контракт сервісу: файл .proto

```
service ProductService {
    rpc GetProduct (ProductRequest) returns (ProductResponse);
}

message ProductRequest {
    int32 id = 1;
}

message ProductResponse {
    int32 id = 1;
    string name = 2;
    float price = 3;
}
```

Підтримка gRPC у PHP слабша, ніж у Go/Java — REST залишається основним вибором для PHP.

Каскадний збій

Products стає повільним → Orders вичерпує потоки/з'єднання, чекаючи на нього → Users, що викликає Orders, теж стає повільним. Проблема поширюється "хвилею" по всій системі.

Три стани Circuit Breaker

Closed

замкнений

Запити проходять як завжди; підраховуються помилки

Open

розімкнений

Запити одразу відхиляються без звернення до мережі

Half-Open

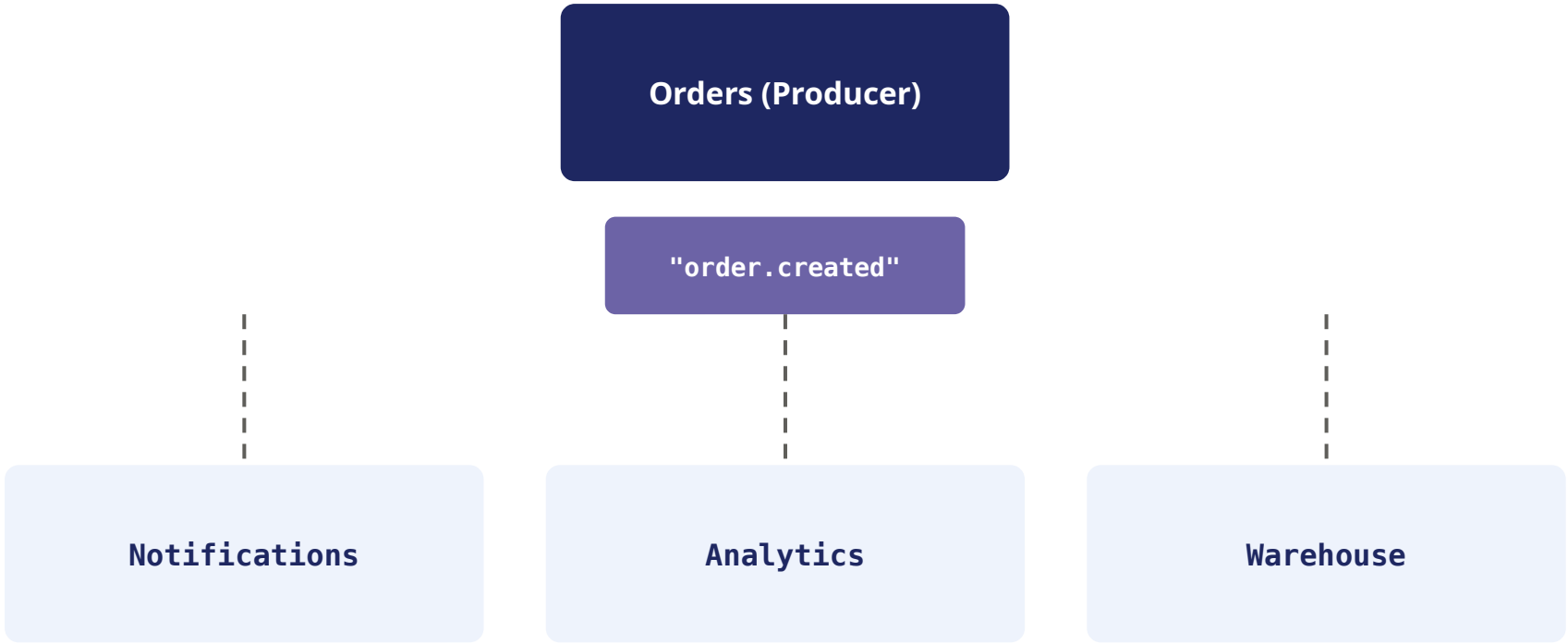
напіввідкритий

Пропускається обмежена кількість пробних запитів

Основні поняття

Термін	Значення
`Producer`	Сервіс, що надсилає повідомлення в чергу
`Consumer`	Сервіс, що отримує й обробляє повідомлення
`Queue`	Впорядкований список повідомлень, що очікують обробки
`Exchange`	Маршрутизує повідомлення від видавця до черги (RabbitMQ)

Publish/Subscribe: один до багатьох



Orders публікує подію один раз — не знаючи (і не турбуючись), скільки й яких підписників її отримають.

Публікація повідомлення (Producer)

```
use PhpAmqpLib\Connection\AMQPStreamConnection;
use PhpAmqpLib\Message\AMQPMessage;

$connection = new AMQPStreamConnection('localhost', 5672, 'guest', 'guest');
$channel = $connection->channel();

$channel->queue_declare('order_created', false, true, false, false);

$data = json_encode(['orderId' => 501, 'userId' => 42]);
$message = new AMQPMessage($data, ['delivery_mode' => 2]);

$channel->basic_publish($message, '', 'order_created');

$channel->close();
$connection->close();
```

Отримання повідомлення (Consumer)

```
$channel->queue_declare('order_created', false, true, false, false);

$callback = function (AMQPMessage $msg) {
    $data = json_decode($msg->getBody(), true);
    echo "Обробка замовлення #" . $data['orderId'];

    $msg->ack();   // підтвердити — повідомлення видаляється з черги
};

$channel->basic_consume('order_created', '', false, false, false, false, $callback);

while ($channel->is_consuming()) {
    $channel->wait();
}
```

Redis Pub/Sub

```php
// Видавець
$redis->publish('order_created', json_encode(['orderId' => 501]));

// Підписник (в окремому процесі)
$redis->subscribe(['order_created'], function ($redis, $channel, $message) {
    $data = json_decode($message, true);
    echo "Отримано замовлення #" . $data['orderId'];
});
```

Fire-and-forget: якщо жоден підписник не підключений у момент публікації — повідомлення втрачається.

RabbitMQ vs Redis

Критерій	RabbitMQ	Redis
Гарантія доставки	Висока (ack, диск)	Нижча (Pub/Sub) / базова (списки)
Складність налаштування	Вища	Нижча — часто вже є для кешу
Типовий сценарій	Критичні події (оплати)	Некритичні задачі (сповіщення)

Підсумок теми 13

- Кожен виклик до іншого сервісу потребує таймауту — мережа ненадійна
- Retry має сенс для тимчасових збоїв (5xx, timeout), а не для 404
- gRPC швидший за REST, але REST залишається основним вибором для PHP
- Circuit Breaker запобігає каскадним збоям, "здаючись швидко"
- RabbitMQ гарантує доставку через ack(); Redis Pub/Sub — fire-and-forget
- Обирайте чергу за критичністю: RabbitMQ для оплат, Redis для сповіщень