Falhas parciais

Preserve o resultado quando uma dependência falha

Pré-requisitos

Conheça transações, filas e requisições remotas. Use Python 3 com sqlite3. O laboratório simula em memória uma reserva e uma entrega de notificação. A falha acontece depois de o destinatário receber o evento e antes de o remetente receber a resposta.

Trate resultado desconhecido como um estado próprio [1]

Um timeout limita quanto tempo o remetente espera. A operação pode ter ocorrido no destino. Se a API de notificações expirou depois que a reserva foi confirmada, informe o estado conhecido da reserva e registre o envio como pendente. O sistema pode consultar ou repetir a entrega usando uma identidade estável.

Evite duas escritas independentes [2]

Gravar a reserva e só então publicar cria uma janela em que uma queda perde o evento. Publicar antes da gravação pode anunciar uma reserva que nunca existiu. A outbox armazena o pedido de publicação na mesma transação da reserva. Depois, outro passo lê as linhas pendentes e tenta entregar.

Projete para repetição

A entrega pode ocorrer sem que a marca de envio seja gravada. Uma nova tentativa usa o mesmo ID, permitindo ao destinatário reconhecer a repetição. Num serviço real, registre esse ID de forma durável e coordene a deduplicação com o efeito. Para um provedor externo, verifique o suporte a chaves de idempotência e o prazo em que valem.

Contenha a falha

Aplique tempo limite, espera crescente, variação aleatória e limite de tentativas conforme a dependência. Registre idade e volume de pendências. O teste de laboratório força uma falha e tenta novamente de imediato para observar o estado. Em produção, uma pane prolongada pede investigar ou encaminhar pendências para tratamento.

Falhas parciais

Exercício resolvido • Faça a entrega sobreviver à falha

Enunciado e dados de partida

Confirme R45 e gere E1. O destinatário recebe E1, mas o remetente observa timeout. Reenvie sem perder a reserva e sem aplicar duas vezes a mesma notificação simulada.

1. Salve a primeira parte de falhas.py

```
import sqlite3

db = sqlite3.connect(':memory:')
db.executescript("""
CREATE TABLE reservas (id TEXT PRIMARY KEY);
CREATE TABLE saida (
    evento TEXT PRIMARY KEY,
    reserva TEXT NOT NULL,
    enviado INTEGER NOT NULL DEFAULT 0
);
""")

with db:
    db.execute("INSERT INTO reservas VALUES ('R45')")
    db.execute("INSERT INTO saida VALUES ('E1', 'R45', 0)")

recebidos = set()

def entregar(evento, falhar_apos_receber):
    recebidos.add(evento)
    if falhar_apos_receber:
        raise TimeoutError('Resposta da notificacao perdida')

def publicar(falhar_apos_receber=False):
    pendentes = db.execute(
        'SELECT evento FROM saida WHERE enviado = 0'
    ).fetchall()
    for (evento,) in pendentes:
        entregar(evento, falhar_apos_receber)
        with db:
            db.execute('UPDATE saida SET enviado = 1 WHERE evento=?',
                       (evento,))
```

Falhas parciais

Verifique o estado antes e depois da recuperação

2. Acrescente ao arquivo

```
try:
    publicar(falhar_apos_receber=True)
except TimeoutError:
    pass
else:
    raise AssertionError('Falha esperada')

def enviados():
    return db.execute('SELECT enviado FROM saida').fetchone()[0]

assert enviados() == 0
assert db.execute('SELECT id FROM reservas').fetchall() == [('R45',)]
assert recebidos == {'E1'}

publicar()
assert enviados() == 1
assert recebidos == {'E1'}
print('Reserva preservada e evento conciliado')
```

3. Execute e explique os resultados

Rode python falhas.py. Na primeira entrega, E1 já está em recebidos. O timeout impede a marca de envio, então a linha continua pendente. A reserva R45 continua gravada. Na repetição, a inclusão no conjunto preserva apenas E1 e a marca passa a 1. O conjunto simula a deduplicação e se perde ao encerrar o processo.

Variação • Marque antes de entregar

Considere trocar a ordem para marcar enviado e depois chamar entregar. Se a chamada falhar antes de o destinatário receber o evento, a próxima leitura ignora E1. A reserva segue confirmada, mas a notificação desaparece. Preserve a marca após confirmação e mantenha a deduplicação no destinatário.

Critérios para conferir

Confira a reserva, a pendência e a presença de E1 no destinatário após o timeout. Depois confirme que a pendência termina, com a mesma chave de evento. O exercício tem um único publicador. Um sistema com vários publicadores precisa coordenar a seleção dos itens e lidar com a entrega repetida.

Referências primárias

[1] AWS Builders' Library • Tempos limite, tentativas e backoff

[2] AWS Prescriptive Guidance • Transactional outbox

Falhas parciais

Preserve a reserva quando uma dependência falhar

Defina quais passos podem esperar

Se o envio de e-mail falhar após o commit, a reserva pode continuar confirmada. Grave a intenção de envio na mesma transação e processe-a em segundo plano. Use retentativas limitadas e idempotência para lidar com uma resposta ambígua do provedor.

Exercício adicional

O serviço de e-mail caiu por dez minutos. Qual comportamento o teste deve observar?

Resolução

A reserva continua disponível no histórico. O pedido de envio permanece pendente e pode ser retomado quando o serviço voltar. Simule duas entregas do mesmo evento e confirme, com um provedor de teste, que há um único efeito observável.

Ao assistir ao vídeo

Compare o comportamento do circuito fechado com a resposta quando ele está aberto.

Vídeo complementar

Circuit Breaker para microsserviços - Dias de Dev