Eventos duplicados ou fora de ordem

Proteja a transição, além da entrega

Pré-requisitos

Conheça webhooks assinados, estados de pagamento e transações. O laboratório usa Node.js 24, sem dependências externas. Eventos internos fictícios já passaram pela verificação de origem e pelo mapeamento para a tentativa local.

Distinga três situações [1]

Uma nova entrega do mesmo evento repete seu identificador. Dois eventos distintos também podem comunicar o mesmo efeito de negócio. Por fim, uma notificação antiga pode chegar depois de uma mais recente. Trate a identidade da entrega e a transição de negócio como verificações independentes.

Defina as transições permitidas

No exercício, iniciado informa o começo do pagamento e não muda uma confirmação existente. Aprovado confirma uma tentativa pendente uma vez. Estornos e disputas exigem estados e regras explícitos em um modelo mais amplo. Se houver dúvida sobre o estado financeiro, consulte o objeto autorizado no provedor e aplique a política local.

Dê atomicidade ao processamento

Em produção, grave o identificador processado e o efeito de negócio na mesma transação, com chave única para a entrega e condição de estado na escrita. Se o processamento falhar antes da confirmação da transação, a nova entrega ainda pode ser tratada. Use uma outbox para solicitar efeitos externos, como avisos, sem depender do tempo da chamada remota.

Escolha a chave pelo significado

O escopo do ID deve refletir origem e contexto da entrega. A restrição de negócio pode vincular uma confirmação à tentativa local. Uma mesma reserva pode ter várias tentativas. Confira se a tentativa corresponde à reserva antes de alterar o agendamento.

Eventos duplicados ou fora de ordem

Exercício resolvido • Conserve a confirmação

Enunciado e dados de partida

T45 começa pendente. Receba E8 de T46, E2 aprovado, E2 novamente, E1 iniciado e E3 aprovado, nesta ordem. Determine o último estado e conte quantas confirmações foram aplicadas.

1. Salve eventos.mjs

```
import assert from 'node:assert/strict';

const vistos = new Set();
let confirmacoes = 0;
let estado = 'pendente';

function consumir(evento) {
  if (evento.tentativa !== 'T45') return 'outra tentativa';
  if (vistos.has(evento.id)) return 'duplicado';
  vistos.add(evento.id);

  if (evento.tipo === 'aprovado' && estado === 'pendente') {
    estado = 'confirmado';
    confirmacoes++;
    return 'confirmado';
  }
  return 'sem transicao';
}

const evento = (id, tipo, tentativa = 'T45') => ({ id, tipo, tentativa });
assert.equal(consumir(evento('E8', 'aprovado', 'T46')), 'outra tentativa');
assert.equal(consumir(evento('E2', 'aprovado')), 'confirmado');
assert.equal(consumir(evento('E2', 'aprovado')), 'duplicado');
assert.equal(consumir(evento('E1', 'iniciado')), 'sem transicao');
assert.equal(consumir(evento('E3', 'aprovado')), 'sem transicao');
assert.equal(estado, 'confirmado');
assert.equal(confirmacoes, 1);
console.log('Estado confirmado; uma confirmacao');
```

2. Execute node eventos.mjs

A saída esperada é ‘Estado confirmado; uma confirmacao’. A função e o conjunto em memória demonstram decisões sequenciais. A execução não simula falha de processo, persistência nem entregas concorrentes.

Eventos duplicados ou fora de ordem

Confira cada entrega e o estado resultante

Percorra os eventos da esquerda para a direita

E8 é descartado porque pertence a T46. E2 muda a tentativa T45 para confirmado e incrementa a contagem. A repetição de E2 retorna duplicado. E1 informa um passo anterior e preserva o estado. E3 tem ID novo, mas a confirmação já foi aplicada, então não produz outra transição.

Caso de erro • Confie apenas no ID da entrega

Retire a condição estado === 'pendente' da decisão de aprovação. E3 passa pela deduplicação de entregas e confirma novamente. A expectativa para E3 falha. Restaure o predicado para proteger o efeito de negócio.

Variação • Mude a ordem

Entregue E1 antes de E2. O primeiro permanece sem transição, e o segundo confirma. Em ambos os percursos, confirme o mesmo estado final e a contagem 1. A ordem produziria resultados diferentes se iniciado rebaixasse um pagamento aprovado para pendente.

Critérios para conferir

Identifique qual verificação bloqueia cada efeito. O evento da outra tentativa não deve ser registrado como visto por T45. A duplicata não reexecuta o tratamento. Um evento novo não repete uma confirmação já concluída. Para testar em produção controlada, verifique também reinício, concorrência e conciliação com o provedor.

Referências primárias

[1] Stripe • Entrega de eventos, ordem e duplicatas

Eventos duplicados ou fora de ordem

Aplique cada evento conforme o estado atual

Separe entrega de efeito

Um identificador de evento já registrado protege contra reentrega. Eventos distintos ainda podem tentar a mesma transição. Use restrição única e transação para registrar o processamento e aplicar o estado. Ao receber um evento que parece antigo, confira o estado atual do provedor quando necessário.

Exercício adicional

O sistema processa a confirmação e depois recebe um evento anterior ainda pendente. O que deve acontecer?

Resolução

Registre o evento pendente para diagnóstico, preserve a confirmação e confira que nenhum efeito de confirmação ocorre pela segunda vez. Teste também a ordem inversa e a reentrega do mesmo identificador.

Ao assistir ao vídeo

No trecho sobre ordenação e duplicidade, compare deduplicação de entrega com a proteção da regra de negócio.

Vídeo complementar

Stripe Developers - Building resilient webhook systems at scale (em inglês)