Máquina de estados de pagamento

Modele o ciclo financeiro da tentativa

Pré-requisitos

Conheça tentativas de pagamento, eventos assíncronos e atualizações condicionais. Use Node.js 24. O modelo do laboratório cobre uma tentativa do sinal com devolução integral e eventos fictícios já verificados.

Separe o estado financeiro do agendamento

Uma tentativa paga pode corresponder a uma reserva expirada. Pagamentos responde pelo estado do sinal. Reservas aplica a política de confirmação e disponibilidade da vaga. Registre o identificador de cada tentativa para que um evento tardio não altere uma tentativa posterior.

Traduza os estados do provedor [1]

O PaymentIntent da Stripe tem estados próprios, como requires_action, processing e succeeded. No modelo simplificado, a espera de ação ou de processamento mantém a tentativa pendente. Um sucesso validado permite marcá-la paga. Uma falha recuperável do método de pagamento pode permitir nova tentativa no provedor. Só emita o evento interno recusar quando a política local considerar aquela tentativa encerrada.

Modele devolução como outra etapa [2]

Solicitar uma devolução não comprova que ela foi concluída. O modelo marca devolucao_pendente e espera o resultado. Uma devolução integral concluída encerra a tentativa em devolvido. Quando houver falha definitiva dessa devolução, retorne o resumo para pago e preserve o registro da falha. Para devoluções parciais ou simultâneas, mantenha registros e valores acumulados próprios.

Defina entradas e efeitos

A tabela indica o próximo estado por estado atual e evento validado. No serviço, confira valor, moeda e identidade da tentativa e da devolução. Persista o estado novo e os efeitos locais juntos. Um evento tardio sem transição prevista pode exigir conciliação, registro ou encaminhamento para revisão.

Máquina de estados de pagamento

Exercício resolvido • Verifique a tabela de transições

Enunciado

A tentativa começa pendente. Aplique aprovação, tente uma recusa tardia e solicite devolução. Verifique duas alternativas independentes a partir da devolução pendente: falha ou confirmação. Depois confira uma aprovação tardia sobre a tentativa já devolvida.

1. Salve estados.mjs

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

const transicoes = {
  pendente: { aprovar: 'pago', recusar: 'recusado' },
  pago: { solicitarDevolucao: 'devolucao_pendente' },
  devolucao_pendente: {
    confirmarDevolucao: 'devolvido',
    falharDevolucao: 'pago',
  },
  recusado: {},
  devolvido: {},
};

function aplicar(estado, evento) {
  const opcoes = transicoes[estado];
  if (!opcoes || !Object.hasOwn(opcoes, evento)) return estado;
  return opcoes[evento];
}

let estado = aplicar('pendente', 'aprovar');
assert.equal(estado, 'pago');
assert.equal(aplicar(estado, 'recusar'), 'pago');
estado = aplicar(estado, 'solicitarDevolucao');
assert.equal(estado, 'devolucao_pendente');
assert.equal(aplicar(estado, 'falharDevolucao'), 'pago');
estado = aplicar(estado, 'confirmarDevolucao');
assert.equal(estado, 'devolvido');
assert.equal(aplicar(estado, 'aprovar'), 'devolvido');
assert.equal(aplicar('pendente', 'recusar'), 'recusado');
console.log('Transicoes conferidas');
```

2. Execute node estados.mjs

O Node.js fornece assert. A saída esperada é ‘Transicoes conferidas’. A função consulta a tabela e conserva o estado quando a combinação está ausente. Ela não persiste dados nem chama o provedor.

Máquina de estados de pagamento

Confira o que deve mudar em cada evento

Percorra os dois ramos

Pendente com aprovação vira pago. Recusa tardia mantém pago. Solicitar devolução abre devolucao_pendente. Partindo desse estado, falha devolve pago e confirmação leva a devolvido. No script, a asserção de falha não altera a variável estado para que a confirmação parta da mesma condição inicial.

Caso de erro • Trate solicitação como confirmação

Troque a saída de solicitarDevolucao para devolvido. A asserção que espera devolucao_pendente falha. Isso evita que a interface mostre um reembolso concluído cedo demais. Em produção, associe a resposta ao ID da devolução corrente para distinguir eventos de operações financeiras diferentes.

Variação • Um estado intermediário do provedor

Considere um callback que informa processing. Mantenha a tentativa pendente e exiba o status em termos da jornada. Só avance para pago ao receber uma evidência de aprovação que satisfaça o contrato. Se o pagamento ficar pendente por muito tempo, agende uma verificação do estado oficial.

Critérios para conferir

A tabela recusa regressões após pagamento e devolução. As duas alternativas partem do estado devolucao_pendente. Recusa de uma tentativa pendente gera recusado. O laboratório testa apenas a lógica de transição. Conciliação, assinatura e concorrência precisam de testes de integração.

Referências primárias

[1] Stripe • Ciclo de vida de PaymentIntent

[2] Stripe • Devoluções e estados associados

Máquina de estados de pagamento

Modele o pagamento separado do horário

Defina transições entre dois registros

Uma tentativa de pagamento pode concluir depois que a reserva expira. Registre o fato financeiro e avalie em transação se a reserva correspondente ainda pode ser confirmada. Uma reserva expirada exige reconciliação conforme a política do negócio.

Exercício adicional

Uma tentativa ficou em processamento, a reserva expirou e o pagamento foi concluído depois. Qual é o resultado?

Resolução

Registre o pagamento concluído com a referência do provedor e mantenha a reserva expirada. Encaminhe o valor à reconciliação prevista e preserve outras reservas que já ocupem o horário. Repita o evento de conclusão no teste e verifique que a tarefa de reconciliação aparece uma única vez.

Ao assistir ao vídeo

Identifique os estados intermediários antes de um pagamento ser concluído.

Vídeo complementar

Stripe Developers - Payment Intent lifecycle management (em inglês)