CASO TÉCNICO DE PORTFÓLIO

Transactional Wallet

Consistência para operações que podem se repetir.

Backend transacional com precisão monetária, histórico de movimentações e recuperação de falhas no processamento assíncrono.

TypeScript · NestJS · Bun · MikroORM · PostgreSQL

SQS / LocalStack · Decimal.js

Um comando pode chegar mais de uma vez

01

Repetição

Retries e reentregas precisam preservar um único efeito no saldo.

02

Indisponibilidade

A confirmação da movimentação precisa sobreviver a falhas temporárias da fila.

03

Rastreabilidade

Reversões e divergências precisam manter referências e histórico para investigação.

Domínio estudado: BET, WIN, REFUND e ROLLBACK.

Regras, transação e processamento assíncrono

DOMÍNIO	**Money e regras da carteira** Valores, moeda, saldo e compensações.
APLICAÇÃO	**Processamento de comandos** Idempotência e coordenação da transação.
INFRAESTRUTURA	**PostgreSQL e SQS / LocalStack** Persistência, bloqueios, consumo e publicação.

Contrato de consistência: Wallet + Ledger + Outbox na mesma transação.
Inbox e resultado idempotente acompanham os efeitos aplicáveis.

Escolhas técnicas e seus custos

Precisão monetária	Decimal.js e valores em string; escopo de duas casas decimais.
Idempotência persistente	Provedor, chave e hash do payload distinguem repetição de conflito.
Concorrência por carteira	SELECT FOR UPDATE serializa alterações, com custo de contenção.
Inbox e Outbox	Retomada após falhas; reenvios exigem deduplicação no consumidor.

Entrega pelo menos uma vez. As garantias dependem de atomicidade e constraints.

Cenários de falha já exercitados

9 cenários aprovados

Registro de execução: 21/10/2026

- Fila indisponível e retry de publicação
- Mensagem duplicada e estorno fora de ordem
- Reversão repetida e detecção de divergência

VALIDAÇÃO INTEGRAL PENDENTE

O Docker estava inativo na execução anterior, conforme relato do autor.
A suíte completa precisa ser repetida com os serviços ativos e o banco preparado.

Da demonstração à reprodução completa

01 **Reproduzir o ambiente**

Ativar os serviços, conferir a conexão PostgreSQL e aplicar migrations.

02 **Reexecutar e registrar**

Rodar a suíte completa e associar versões, saída e commit ao resultado.

03 **Ampliar a validação**

Concorrência real, quedas entre etapas, retry, DLQ e observabilidade.

Aplicações possíveis: webhooks, pedidos e créditos internos.
Os padrões exigem adaptação às regras e ao ambiente de cada negócio.