ACME/SA - Relatório Técnico

API distribuída de pedidos e estoque

1. Visão Geral

A plataforma simula o ecossistema ACME/SA com múltiplas filiais. Cada instância FastAPI mantém seu próprio banco SQLite (WAL + chaves estrangeiras) e expõe endpoints para clientes, produtos, usuários, pedidos e estoque.

Segurança: endpoints sem JWT retornam 401, tokens inválidos 401/403 e somente tokens válidos permitem criar pedidos ou manipular estoque.

2. Arquitetura

config.py controla variáveis (NODE_NAME, PEERS, REPLICATION_TOKEN, JWT_SECRET, ADMIN_*) para cada nó. database.py inicializa o schema completo. state.py encapsula regras, locks assíncronos e transações. sync.py mantém filas de replicação. security.py trata hash de senha (bcrypt) e emissão/validação de JWT.

Cada entidade (clientes, produtos, usuários, pedidos, estoque) possui rotas REST. As réplicas compartilham snapshots completos através do endpoint /replica/event, garantindo idempotência e reconstrução após panes.

3. Sincronização e Consistência

Pedidos e ajustes de estoque adquirem locks por produto e são executados dentro de uma única transação SQLite, mantendo consistência forte local. Cada movimentação incrementa ersao do estoque.

Eventos order_created carregam pedido + cliente + produtos para que o nó destino possa atualizar seu catálogo antes de gravar o pedido. stock_update inclui snapshot do estoque e do produto, permitindo aplicar somente versões mais novas.

4. Tolerância a Falhas

ReplicaSynchronizer mantém uma fila FIFO por peer. Se algum nó estiver indisponível, os eventos ficam em pending e o loop periódico (REPLICATION_RETRY_SECONDS) reenvia até receber HTTP 2xx.

Como os eventos são idempotentes (snapshots completos), processá-los novamente após uma queda não gera inconsistências. O endpoint /status exibe quantidades e o backlog por peer para auditoria operacional.

5. Segurança

Usuários são persistidos com senhas bcrypt e logam via /auth/login, recebendo JWT assinado (JWT_SECRET). Dependências FastAPI validam o token antes de qualquer operação crítica.

O tráfego entre réplicas é protegido por X-Replica-Token separado, evitando que um JWT de cliente seja usado para injetar eventos falsos. Perfis ■dmin e operador restringem rotas avançadas (ex.: cadastro de usuários/produtos).

6. Cenários Demonstrados

Concorrência: dois pedidos simultâneos para o mesmo produto com saldo limitado → apenas um

Consistência eventual: pedido criado no nó A altera imediatamente o seu estoque, enquanto o nó B reflete a mudança assim que processa stock_update.

Falha simulada: desligar um nó, continuar operando no outro e observar /status.pending; ao religar, a fila é drenada automaticamente.

Os nós comunicam-se por eventos HTTP assíncronos para garantir convergência eventual. Toda operação crítica é autenticada via JWT e auditada com metadados de origem/versão.

7. Passos de Execução

1) Criar o ambiente virtual, instalar dependências e definir variáveis (NODE_NAME, PEERS, REPLICATION_TOKEN, JWT_SECRET, ADMIN_*) para cada nó.

2) Subir as instâncias via uvicorn app.main:app --host 0.0.0.0 --port <porta>.

3) Autenticar (/auth/login), cadastrar clientes/produtos, criar usuários adicionais e processar pedidos. Monitorar /status para validar replicação e saúde das réplicas.