Desafio Golden - Mini-ERP de Pedidos

Contexto

Você vai construir um mini-ERP de pedidos para uma loja fictícia. O sistema deve cadastrar clientes e produtos, criar pedidos com itens, calcular totais, consultar CEP externamente para enriquecer endereço e (opcional) consultar câmbio para exibir o valor do pedido em USD. Processos recorrentes devem rodar via tarefas agendadas.

Requisitos Funcionais (obrigatórios)

1. **Clientes**
   - CRUD completo.
   - Cada cliente possui: `id`, `nome`, `email` (único), `cpf` (único), `endereco` (logradouro, número, complemento, bairro, cidade, uf, cep).
   - Ao criar/atualizar um cliente com **CEP**, o sistema deve **enriquecer** automaticamente o endereço consultando a API ViaCEP (Feign). Se `logradouro/bairro/cidade/uf` não forem enviados, devem ser preenchidos a partir do CEP.
2. **Produtos**
   - CRUD completo.
   - Campos: `id`, `sku` , `nome`, `precoBruto`, `estoque`, `estoqueMinimo` , `ativo`.
3. **Pedidos**
   - Criar pedido para um cliente com lista de itens `{produtoId, quantidade, descontoOpcional}`.
   - Calcular totais: `subtotal`, `descontos`, `total` (Gere confiabilidade nas casas decimais e use apenas duas).
   - Status do pedido: `CREATED`, `PAID`, `CANCELLED`, `LATE`.
   - Ao criar o pedido, **baixar estoque** dos produtos envolvidos (se faltar, rejeitar com 422).
   - Endpoint para **pagar** pedido (muda status para `PAID`).
   - Endpoint para **cancelar** pedido (devolve estoque se ainda não pago).
4. **Listagens e busca**
   - Paginação, ordenação e filtros básicos (por exemplo: produtos ativos, pedidos por status, clientes por nome ou email).
5. **API REST**
   - Seguir boas práticas REST, retornar HTTP status adequados, DTOs de entrada/saída, validação Bean Validation.
6. **Persistência**
   - Spring Data JPA + Postgres.
   - Mapeamentos, constraints e índices adequados.
7. **Tarefas agendadas**

    - (A) **Marcar pedidos atrasados**: a cada hora, qualquer pedido `CREATED` com `dataCriacao + 48h` no passado vira `LATE`.
    - (B) **Reabastecimento**: diariamente às 03:00, registrar em log (ou tabela) os produtos com `estoque < estoqueMinimo`.
8. **Integração externa (Feign)**
    - **ViaCEP** para resolver CEP: `GET https://viacep.com.br/ws/{cep}/json/`
        - Mapear campos úteis (`logradouro`, `bairro`, `localidade`, `uf`).
        - Tratar CEP inválido (erro 400) e erro de rede (retry simples com backoff ou ao menos logs claros).
9. **Containerização**
    - Dockerfile do app.
    - `docker-compose.yml` com Postgres e a aplicação subindo e conversando entre si.

Requisitos Opcionais (bonus points)

- **Câmbio via Feign** (ex.: exchangerate.host): manter cache da cotação BRL→USD por 1h e expor `/orders/{id}/usd-total`.
- **Migrações de banco** com Liquibase
- **Observabilidade**: logs estruturados, métricas simples (quantidade de pedidos criados por hora).
- **Segurança**: autenticação simples (JWT)

Tarefas

- **Código completo** visando as boas praticas de desenvolvimento e arquitetura
- **Testes unitários** com cobertura de 60% da logica de negócio (classes com final Impl.java)
- **Execução simples** apenas com o comando docker compose up
- README.md completo com a arquitetura escolhida e os motivos dessa escolha.