Pix Service - Code Assessment

Objetivo

Criar um microserviço de **carteira** com suporte a **Pix**, garantindo **consistência sob concorrência** e **idempotência**. Trate este projeto como produção, mesmo que simplificado.

Escopo & Diretrizes

- **Tempo**: Prazo de 5 dias após recebimento do mesmo.
- **Profissionalismo**: escreva como em produção (código limpo, testes, docs).
- **Assunções**: documente qualquer requisito ambíguo e o porquê da sua decisão.
- **Time tracking**: informe o tempo investido ao final.
- **Tecnologia**: **Java, Spring Boot**. Estruture em **Clean Architecture** (camadas bem separadas).

Requisitos Funcionais

- **Criar Conta/Carteira**: criação de carteira para um usuário.
- **Registrar Chave Pix**: vincular uma chave Pix única à carteira (e.g., email/telefone/EVP).
- **Consultar Saldo**: saldo atual da carteira.
- **Saldo Histórico**: saldo em um timestamp passado.
- **Depósito**: crédito na carteira (simula entrada de recursos).
- **Saque**: débito na carteira (valida saldo).
- **Transferência Pix (interna)**: enviar Pix para outra carteira usando **chave Pix** (gera `endToEndId`).
- **Webhook Pix (simulado)**: endpoint que recebe eventos `CONFIRMED`/`REJECTED` para um `endToEndId`. Eventos podem chegar **duplicados** e **fora de ordem**.

Requisitos Não Funcionais

- **Missão crítica**: evite inconsistências; efeito **exactly-once** no débito.
- **Rastreabilidade/Auditoria**: trilha completa de operações (ledger/eventos).
- **Concorrência**: o sistema deve resistir a **requisições simultâneas** do mesmo Pix.
- **Idempotência**
- **Observabilidade**: logs estruturados e métricas mínimas.

Cenários-Chave (Concorrência & Race)

1. **Duplo disparo**: duas requisições **simultâneas** de transferência Pix com **mesmo** `Idempotency-Key` e/ou **mesmo** `endToEndId` ⇒ **um** único débito efetivo.
2. **Webhook duplicado**: vários `POST /pix/webhook` com **mesmo** **`eventId`** ⇒ aplicar **uma** vez.
3. **Ordem trocada**: `REJECTED` chegando antes de `CONFIRMED` ⇒ respeitar **máquina de estados** e manter consistência.
4. **Reprocesso**: reexecutar o processamento do mesmo evento (simulate “at least once”) sem mudar o saldo final.

Sugestão de Endpoints (contratos simples)

- `POST /wallets` → cria carteira.
- `POST /wallets/{id}/pix-keys` → registra chave Pix.
- `GET /wallets/{id}/balance` → saldo atual.
- `GET /wallets/{id}/balance?at=2025-10-09T15:00:00Z` → saldo histórico.
- `POST /wallets/{id}/deposit` → depósito.
- `POST /wallets/{id}/withdraw` → saque.
- `POST /pix/transfers`
  Headers: `Idempotency-Key: <uuid>`

Body: `{ fromWalletId, toPixKey, amount }`
Resposta: `{ endToEndId, status }`

- `POST /pix/webhook`
  Body: `{ endToEndId, eventId, eventType, occurredAt }`
  Idempotente por `eventId`.

Obs.: Não precisa integrar ao Bacen; o “webhook” simula a confirmação do arranjo Pix.

Entregáveis (GitHub)

- Implementação do microserviço.
- Instruções para instalar, testar e executar (README claro; Docker Compose do Postgres opcional).
- Explicação das decisões de design e como atendem aos requisitos (funcionais e não-funcionais).
- Explicação de trade-offs/compromissos por limite de tempo.
- **Testes** unitários e testes integrados

Critérios de Avaliação (alto nível)

- **Corretude & Consistência** (efeito único, estados válidos, ledger fechando).
- **Concorrência & Falhas** (prevenção de corrida, idempotência demonstrada).
- **Qualidade de Arquitetura & Código** (Clean Architecture, coesão, clareza).
- **Observabilidade & Auditabilidade** (logs/metrics e trilha de eventos).
- **Documentação & Execução** (README, como subir/rodar/testar).

Pistas Técnicas (opcionais, escolha sua abordagem)

- **Idempotência**: tabela `idempotency(scope,key)` com **unique**; retornar o mesmo resultado ao detectar repetição.

- **Race condition**: **optimistic locking** (`version`) ou **pessimista** (`SELECT FOR UPDATE`) em saldo/ledger; retries curtos.

- **Ledger**: entradas imutáveis (+/−) vinculadas ao `endToEndId`; saldo derivado ou mantido com locking.

- **Máquina de estados**: `PENDING → CONFIRMED → (SETTLED opcional)` ou `PENDING → REJECTED`.

- **Observabilidade**: log com `endToEndId`, `eventId`, `idempotencyKey`.