Backend Assessment Test

- Buatlah sebuah service transaksi yang berupa HTTP Server menggunakan REST API yang memiliki API sebagai berikut:

    1. `/tabung`: API untuk nasabah menabung dengan payload JSON berisi field `no_rekening`, dan `nominal`. API akan memberikan balikan dengan status 200 dan payload JSON berisi field `saldo` yang berisi data saldo nasabah saat ini. Jika `no_rekening` tidak dikenali, API akan memberikan balikan status 400 dan payload JSON berisi field `remark` yang berisi deskripsi kesalahan terkait data yg dikirim.

    2. `/tarik`: API untuk nasabah menarik dana nasabah dengan payload JSON berisi field `no_rekening` dan `nominal`. API akan memberikan balikan dengan status 200 dan payload JSON berisi field `saldo` yang berisi data saldo nasabah saat ini. Jika `no_rekening` tidak dikenali atau saldo tidak cukup, API akan memberikan balikan status 400 dan payload JSON berisi field `remark` yang berisi deskripsi kesalahan terkait data yg dikirim.

- Buatlah service mutasi yang menerima request mutasi dari service transaksi setiap kali ada transaksi tabung atau tarik menyimpannya di database. Request mutasi berisi field `tanggal_transaksi`, `no_rekening`, `jenis_transaksi`, `nominal`.

- Untuk seluruh API lakukan authentication nomor rekening dan PIN menggunakan middleware. Data `account_no` & `pin` dikirim melalui Authorization headers. Data PIN harus dienkripsi pada database.

- Buatlah test script untuk masing endpoint dengan skenario sebagai berikut:

    1. Request tabung nomor rekening dikenali

    2. Request tabung nomor rekening tidak dikenali

    3. Request tarik saldo kurang dari yang dimiliki

    4. Request tarik saldo lebih dari yang dimiliki

Catatan:

- Membuat log yang jelas dan terstruktur. Sertai data dan context yang berkait dengan proses yang sedang terjadi pada log. Minimal terdapat `timestamp`, `level`, `message`, dan informasi terkait proses bisnis. Gunakan level log (INFO, WARNING, ERROR, CRITICAL) yang sesuai dengan keadaan.
- HTTP Server dibuat menggunakan Golang (`echo` atau `fiber`) atau Python (`fastapi`).
- Data disimpan dalam database PostgreSQL menggunakan docker container.
- Gunakan ORM (`gorm` atau `sqlalchemy`) atau SQL compiler (`sqlc-go` atau `sqlc-python`) sebagai database client.
- Gunakan Redis Stream sebagai event stream untuk komunikasi antara service transaksi dan service mutasi.
- Gunakan environment variable dan argument parser untuk konfigurasi semua service. Environment variable digunakan untuk konfigurasi sensitif seperti user dan password database. Argument parser digunakan untuk konfigurasi non-sensitif seperti REST API host dan port.
- Gunakan Dockerfile untuk membuat image untuk service transaksi dan service mutasi.
- Gunakan Docker Compose untuk deployment service transaksi, service mutasi, database, dan event stream.
- Minimal terdapat 3 layer pada struktur code: `api` untuk modul API, `app` untuk modul business logic, dan `datastore` untuk modul penyimpanan data.
- Memasang system observability menggunakan Open Telemetry + Jaeger merupakan nilai plus.
- Buatkan README.md berisi konfigurasi aplikasi dan cara menjalankan aplikasi.
- Aplikasi harus dapat dijalankan menggunakan command `docker compose up`

Kriteria Penilaian:

- Log structure: Log yang terformat rapih dan memiliki informasi yang jelas
- Software Architecture: Variable naming yang jelas, separasi modul yang rapih
- Database schema: Penggunaan relasi antar table yang sesuai
- Database operation: Operasi database yang efektif
- REST API structure: Struktur endpoint, parameter, dan middleware yang sesuai
- Configuration management: Konfigurasi service yang fleksibel dan aman
- Deployment setup: Penggunaan Docker/Kubernetes dalam deployment aplikasi