Interfaz común y contrato de datos

Cada filtro implementa `process(context: dict) -> dict`. El `context` transporta datos acumulados entre etapas (transacción, usuario, montos en fiat, comisión, total, estado, timestamps). Si un filtro falla, debe **lanzar una excepción** o **marcar estado de error** para detener el flujo, evitando efectos secundarios inconsistentes.

Filtros y responsabilidades

1. **Validación**
    - Verifica campos obligatorios: `user_id`, `btc_amount > 0`, `base_currency ∈ {USD, EUR, GBP}`.
    - Normaliza/estandariza formatos (e.g., moneda en mayúsculas).
    - *Salida:* `context['transaction']` válido.
2. **Autenticación**
    - Confirma identidad del usuario contra una **base simulada** (JSON local o diccionario en memoria). Puede extenderse a RDS/SQLite.
    - *Salida:* `context['user']` con metadatos (nombre, límites, etc.).
3. **Transformación (BTC→moneda base)**
    - Convierte `btc_amount` a fiat según la **moneda base**.
    - Puede usar **API REST pública** (e.g., exchange rates) o un **servicio simulado** con datos fijos (para entornos sin Internet o pruebas determinísticas).
    - *Salida:* `context['fiat_amount']` y `context['fx_used']`.
4. **Cálculo de Comisiones**
    - Agrega una **comisión fija** equivalente a `5.00` USD convertida a la moneda base.
    - *Salida:* `context['fee']` y `context['total'] = fiat_amount + fee`.
5. **Almacenamiento**
    - Persiste la transacción procesada en **almacenamiento durable** (archivo JSON o SQLite). Incluye `timestamp`, montos, tasas y estado.
    - *Salida:* `context['persisted'] = True` y `context['storage_path']`.

Comunicación entre filtros

- **Forma:** `context` inmutable por referencia (diccionario) que **se enriquece** en cada paso.
- **Acoplamiento:** Bajo. Cada filtro **solo depende** de entradas de `context` documentadas por filtros previos y de sus **propias dependencias** (p.ej., cliente de tasas, repositorio de usuarios, repositorio de transacciones).