Background

In Cobre we have a transactional, cloud-native, event driven and microservices platform that manages resources available for the clients, such as, accounts, payments or transactions, enabling the capabilities to create, delete or update each one of them. Each account has its own currency then every payment and transaction has to be compliant with the definition of the account.

The Challenge.

Cross Border Money Movements (CBMM)

A new cross-border money movement (CBMM) feature, enabling transactions between currencies like USD and COP, has been defined by the product team. The asynchronous process for this feature involves the following steps:

1. **FX Quote Acquisition:** Clients obtain an FX quote by providing the origin currency, destination currency, and amount, receiving the amount for the CBMM.
2. **CBMM Request:** The client then submits a request to initiate the CBMM to the component A, referencing the acquired FX quote.
3. **Settlement Redirection:** The component A redirects the money movement to the component B for settlement.
4. **Balance Validation & Transactions:** The component B validates the origin account balance and generates corresponding balance transactions for the request.
5. **Final Notification:** Upon completion of the entire CBMM process, the component B sends a final notification detailing the outcome to the component A then the CBMM is updated to the final status (failed or completed=

Task 1. System Design:

- Design an event flow for the entire CBMM flow received, ensuring idempotency and eventual consistency. Designing an idempotent and eventually consistent event flow for the entire CBMM requires a robust architectural approach that handles distributed systems challenges.

Implementation Considerations:

- **Schema Evolution:** Define a clear schema for events and plan for graceful evolution to handle changes over time.
- **Versioning:** Version events and consumers to manage backward and forward compatibility.
- **Error Handling and Retry Mechanisms:** Implement robust error handling, including exponential backoff for retries and circuit breakers to prevent cascading failures.
- **Monitoring and Observability:** Comprehensive logging, tracing, and metrics are essential to understand the flow, identify bottlenecks, and debug issues in a distributed system.
- **Security:** Ensure secure communication channels and proper authentication/authorization for event producers and consumers.
- **Data Archiving and Retention:** Define policies for storing and archiving historical events.

Task 2. Implementation of accounts module.

Implement a robust accounts module. This module will leverage the principles of hexagonal architecture to ensure clear separation of concerns, testability, and maintainability. The chosen framework for development is Spring Boot, providing a comprehensive and efficient environment for building enterprise-grade applications.

Core Responsibilities:

- **Account Balance Management:** The module will be responsible for meticulously maintaining the current balance for each account. This includes ensuring data consistency and accuracy at all times.
- **Transaction Ledger:** A detailed ledger of all financial transactions (credits and debits) will be meticulously recorded and associated with the respective accounts. This ledger will serve as an auditable history of all movements.

Concurrent Event Processing:

A critical requirement for this module is its ability to concurrently process event structures. These event structures will be provided in the form of files, representing a series of financial activities. The system must be capable of:

- **Final Balance Calculation:** For each account, the module will process the incoming events to accurately calculate and update the final account balance. This processing

must be highly efficient to handle potentially large volumes of events.
- **Applied Transaction Listing:** As events are processed, the module will meticulously list all applied transactions within the ledger for each account. This ensures a comprehensive and up-to-date record of all financial activities.

Input Event Files:

The event files, crucial for testing and demonstrating the module's capabilities, will be supplied during the case presentation. These files will contain the structured data necessary for the module to perform its calculations and ledger updates.

CBMM event:

JSON

```
{
  "event_id": "cbmm_20250909_000123",
  "event_type": "cross_border_money_movement",
  "operation_date": "2025-09-09T15:32:10Z",
  "origin": {
    "account_id": "ACC123456789",
    "currency": "COP",
    "amount": 15000.50
  },
  "destination": {
    "account_id": "ACC987654321",
    "currency": "USD",
    "amount": 880.25
   }
  }
```

event_id: Unique event identifier for traceability.

event_type: Type of movement, in this case "cross_border_money_movement".

operation_date: Date and time of the operation in ISO 8601 (UTC).

origin: Object with the details of the origin account, including account_id, currency, and amount.

destination: Object with the details of the destination account.

Initial conditions:

```
ACC987654321:
	Initial Balance: $0.00 USD
ACC123456789:
	Initial Balance: $200,000.00 MXN
```

Additional information

- You are free to use any format to deliver your solution proposal, could be a document or presentation. Also, feel free to use any modeling framework to document your solution (UML, C4, etc.).
- Please be prepared to discuss any architectural decision and reasoning you have made.
- You will present your case solution to a panel of people, please consider to be ready for questions.
- About implementation tasks, please share a github repository. Last commit should be before the delivery date.
- At Cobre we promote AI usage as a compliment to enhance our productivity, so we encourage you to use it in your implementation, then please document in detail (prompts, screenshots, etc.) in what use cases you have used.
- Enjoy!