Software Engineer Test Assignment

Description

The goal of this assignment is to familiarize candidates with working in Tuum and using the technologies we use daily.

Requirements

The tasks consist of implementing a small core banking solution:

- Account to keep track of current accounts, balances, and transaction history
- Capability to publish messages into RabbitMQ for other consumers

Test coverage

- The application must contain integration tests, and test coverage must be at least 80%.

Handover

- Application must be executable using Docker
- The application must be runnable without configuration changes on the code reviewer's computer (PATH variable changes, etc.)
- Please ensure that the code is shared with us via Source code management platforms (GitHub, GitLab, etc). and is accessible for code reviewers.

Technologies to be used

- Java 17+
- SpringBoot
- MyBatis
- Gradle
- Postgres
- RabbitMQ
- JUnit

Application description

Account

- Account service keeps track of accounts, their balances, and transactions.
- Account service must publish all insert and update operations to RabbitMQ.

REST APIs

The application provides the following REST APIs:

Create account

- Creates a bank account for the customer and returns an account object together with balance objects.
- Input:
    - Customer ID
    - Country
    - List of currencies (allowed values are EUR, SEK, GBP, USD)
- Output:
    - Account ID
    - Customer ID
    - List of balances:
        - Available amount
        - Currency
- Errors:
    - Invalid currency

Additional requirements:

- API must create balances for the account in the given currencies.

Get account

- Return the account object.
- Input:
    - Account ID
- Output:
    - Account ID
    - Customer ID
- List of balances:
    - Available amount
    - Currency
- Errors:
    - Account not found

Create transaction

- Create a transaction on the account and return the transaction object.
- Input:
    - Account ID
    - Amount
    - Currency
    - Direction of transaction (IN, OUT)
    - Description
- Output:

    - Account ID
    - Transaction ID
    - Amount
    - Currency
    - Direction of transaction
    - Description
    - Balance after transaction
- Errors:
    - Invalid currency
    - Invalid direction
    - Invalid amount (in case of a negative amount)
    - Insufficient funds
    - Account missing
    - Description missing

Additional requirements:

- Transactions with direction IN must increase the account balance by the transaction amount, and transactions with direction OUT must decrease the balance of the account in the respective currency.

Get transaction

- Return a list of transactions
- Input:
    - Account ID
- Output:
    - Account ID
    - Transaction ID
    - Amount
    - Currency
    - Direction of transaction
    - Description
- Errors:
    - Invalid account

Deliverables

- Source code
- Instructions on how to build and run applications
- Dockerfile and docker-compose.yml, which includes the database and RabbitMQ, and initializes the necessary database structure.
- Explanation of important choices in your solution
- Estimate how many transactions your account application can handle per second on your development machine
- Describe what you have to consider to be able to scale applications horizontally
- Explanation of the usage of AI