tuum

Software Engineer Test Assignment

Description

Goal of this assignment is to familiarize candidates for 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

Account service must contain integration tests and test coverage must be at least 80%.

Technologies to use

- Java 11+
- SpringBoot
- MyBatis
- Gradle
- Postgres
- RabbitMQ
- JUnit

Applications

Account

Account service keeps track of accounts, their balances, and transactions.

Account service must publish all insert and update operations to RabbitMQ.

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

API must create balances for the account in given currencies.

Get account

Return the account object.

Input:

- Account ID

Output:

- Account ID
- Customer ID
- List of balances:
  - Available amount
  - Currency

Input:

- Account not found

tuum

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 (if negative amount for example)
- Insufficient funds
- Account missing
- Description missing

Transactions with direction IN must increase account balance with the transaction amount and transactions with direction OUT must decrease the balance of account in 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 database and RabbitMQ and initializes necessary database structure.
- Explanation of important choices in your solution
- Estimate on how many transactions can your account application can handle per second on your development machine
- Describe what do you have to consider to be able to scale applications horizontally