Тестовое задание Junior Java разработчика

Цель: Создать микросервис, который включает два API, клиент для работы с внешним API и базу данных для хранения данных о бизнесе и валютных курсах.

Требования:

1. **Структура сервиса. Сервис должен содержать два REST API:**
   a. Для обработки транзакций (условная интеграция с банковскими сервисами).
   b. Клиентский API для внешних запросов: получения списка транзакций, превысивших лимит, установление нового лимита, получение всех лимитов.
2. **Документация и контейнеризация. Необходимо:**
   a. Описать шаги по запуску сервиса в файле README.md
   b. Подготовить сервис к запуску в Docker, желательно добавить Docker Compose.
3. **База данных.**
   a. Выбрать любую реляционную БД. Разработать модель данных самостоятельно. Пример в конце задания.
   b. Включить скрипты миграций БД в репозиторий. Можно использовать Liquibase или FlyWay.
4. **Тестирование и публикация.**
   a. Написать один юнит тест (использовать Junit5, Mockito, AssertJ)
   b. Опубликовать работу на GitHub

Описание задания:

Необходимо разработать микросервис со следующей функциональностью:

1. **Запись транзакций**: Получать и сохранять информацию о каждой расходной операции в разных валютах, таких как тенге, рубли в базе данных.
2. **Месячный лимит расходов**: Хранить месячный лимит по расходам в долларах США (USD) для двух категорий расходов: товары и услуги.
3. **Получение курсов валют**: Запрашивать дневные курсы валютных пар KZT/USD и RUB/USD. Сохранять эти курсы и базе данных, используя данные закрытия (close). Если данные на текущий день недоступны (например, выходной), использовать данные последнего закрытия (previous_close). Рассчитывать сумму расходов в USD нужно по курсу на день расхода или по последнему доступному курсу. Апи с курсами валют выбрать самостоятельно: twelvedata.com alphavantage.co openexchangerates.org или любое другое.

полученные курсы валют следует сохранять в своей базе данных и использовать их в первую очередь, чтобы не делать лишние запросы в сервис.

4. **Флаг превышения лимита**: Помечать транзакции, превысившие месячный лимит, флагом limit_exceeded. Дефолтный месячный лимит взять за $1000, который сбрасывается первого числа каждый месяц. Новый лимит не должен влиять на старые транзакции. Например, сели лимит 1000 USD был установлен 1.01.2022 и превышен двумя транзакциями на 500 и 600 USD, то у второй транзакции должен быть флаг limit_exceeded = true. Если пользователь установил новый лимит 11.01.2022 и совершил третью транзакцию на 100 USD 12.01.2022, у нее должен быть флаг limit_exceeded = false.
5. **Установка нового лимита:** Позволять клиентам устанавливать новый лимит расходов. При установке нового лимита автоматически записывать текущую дату. Не допускать установку даты в прошлом или будущем. Запрещать обновление существующих лимитов.
6. **Отчет о превышении лимита:** Предоставлять клиентам список транзакций, превысивших лимит, с указанием суммы лимита, валюты (USD) и даты установления.

Пример структуры данных:

п/п	Наименование параметра	Имя параметра	Тип данных	Пример
1	Банковский счет клиента	account_from	Целочисленный, 10 знаков	0000000123
2	Банковский счет контрагента	account_to	Целочисленный, 10 знаков	9999999999
3	Валюта счета	currency_shortname	Строка	KZT
4	Сумма транзакции	sum	Число с плавающей точкой, округление до сотых	10000,45
5	Категория расхода	expense_category	Строка	product/service
6	Дата и время	datetime	Отметка времени с временной зоной	2022-01-30 00:00:00+06
7	Сумма установленного лимита	limit_sum	Число с плавающей точкой, округление до сотых	1000.00
8	Дата и время установленного лимита	limit_datetime	Отметка времени с временной зоной	2022-01-10 00:00:00+06
9	Валюта лимита	limit_currency_shortname	Строка	USD

В случае вопросов/сомнений: сделать самостоятельный выбор и объяснить почему сделан именно он. Пояснить целесообразность в Readme.