Модель базы данных

Пользователь – репрезентация пользователей в приложении. Должны быть обычные и админ пользователи (админ назначается руками в базе или создаётся на старте приложения)

Товар – Состоит из заголовка, описания и цены

Счёт – Имеет идентификатор счёта и баланс. Привязан к пользователю. У пользователя может быть несколько счетов

Транзакция – история зачисления на счёт, хранит сумму зачисления и идентификатор счёта

Функциональные критерии

Весь описываемый ниже функционал должен быть осуществлён в формате REST API. Работа с шаблонами, HTML или фронтендом в любой форме не предусматривается**.**

Пользователь может:

1. Регистрация (по паролю и логину, возвращает ссылку активации)
2. Логин
3. Просмотр списка товаров
4. Покупка товара, просто списывает с баланса стоимость товара, при условии наличия на балансе счёта достаточного количества средств
5. Просмотр баланса всех счетов и историю транзакций
6. Зачисление средств на счёт, выполняется с помощью эндпоинта **[POST] /payment/webhook** симулирует начисление со стороннего сервиса.

Пример тела вебхука, с транзакцией (формат json):

{

“signature”: “f4eae5b2881d8b6a1455f62502d08b2258d80084”,

“transaction_id”: 1234567,

“user_id”: 123456,

“bill_id”: 123456,

“amount”: 100

}

Сигнатура формируются по правилу:

```
from Crypto.Hash import SHA1


signature = SHA1.new()\
        .update(f'{private_key}:{transaction_id}:{user_id}:{bill_id}:{amount}'.encode())\
        .hexdigest()
```

Где **private_key** – приватный ключ, задаётся в свойствах приложения, **transaction_id** – уникальный идентификатор транзакции, **user_id** – пользователь на чеё счёт произойдёт зачисление, **bill_id** – идентификатор счёта (если счёта с таким айди не существует, то но должен быть создан), **amount** – сумма транзакции.

Возможности админа:

1. Видеть все товары
2. Видеть всех пользователей и их счета
3. Включать/отключать пользователей
4. Создавать/редактировать/удалять товары

Не функциональные критерии

1. Логины пользователей уникальны
2. После регистрации пользователь создаётся в не активном состоянии. Становится активным переходя по ссылке полученной с регистрации
3. Авторизация должна быть сделана через **JWT**. Защищённые эндпоинты должны получать токен в заголовке Authorization в Bearer формате

Выполнить задачу с учётом особенностей асинхронной обработки данных. В особенности это касается обработки транзакций, приложение должно быть способно обработать сравнительно большой объём параллельных запросов (с поправкой на технические характеристики сервера).

Стек:

- язык программирования: **Python**
- фреймворк: **FastAPI**
- база данных: **PostgreSQL**