Тестовое задание для Python Backend

Вступление

Предположим, у нас разрабатывается проект с сервисно-ориентированной архитектурой. В качестве тестового задания предлагается сделать один из самых примитивных узлов этого проекта, согласно предложенной ниже схеме REST API. Сервис сужен до одной валюты ETH и урезан во всем, что можно было урезать.

Краткое описание сервиса

Сервис отвечает за создание кошельков, получение балансов по ним, хранение приватных ключей и трансфер денег с созданных кошельков.

Описание API

Сервис должен отвечать следующей схеме REST API (то есть ниже представлены endpoint`ы, которые нужно реализовать)

Создание кошелька

Сервис должен уметь создавать кошельки в блокчейне (далее БЧ) Ethereum и в БД хранить публичный и приватный ключи.

`POST /api/v1/wallets/`

REQUEST:

Параметр	Описание
currency: str, required	Можем принимать только ETH

RESPONSE:

Параметр	Описание
id: int	ID кошелька
currency: str	Валюта кошелька
public_key: str	Публичный ключ (адрес кошелька)

Получение списка кошельков

Сервис должен уметь отдавать список кошельков в БД. В ответ также должен быть

включен их баланс

`GET /api/v1/wallets/`

RESPONSE:

Список, состоящий из словарей, со следующей структурой

Параметр	Описание
id: int	ID кошелька
currency: str	Валюта кошелька
public_key: str	Публичный ключ (адрес кошелька)
balance: decimal	Баланс кошелька

Перевод с одного кошелька в системе на другой кошелек в системе

Сервис должен уметь совершать транзакции с одного кошелька системы на другой кошелек в нашей системе. Обратите внимание, что должны быть реализованы проверки на то, что переводить можно только между кошельками внутри системы и сумма перевода должна быть больше, чем баланс кошелька (плюс затраты на комиссию сети - газ)

`POST /api/v1/transactions/`

REQUEST:

Параметр	Описание
from: str, required	Кошелек отправителя
to: str, required	Кошелек получателя
currency: str, required	Можем принимать только ETH
amount: decimal, required	Сумма перевода

RESPONSE:

Параметр	Описание
hash: str	Хеш транзакции

Ожидаемый результат

1. Код должен быть выложен в репозиторий и предоставлен доступ
2. Сервис должен быть написан на базе Python3.11 + Django4 + DRF

3. Сервис должен быть покрыт юнит-тестами (с помощью pytest). **Это самое важное!** (юнит-тесты значит, что в тестах запросы к БЧ должны быть замоканы)
4. Должно быть описание АПИ с помощью drf-spectacular
5. Должно быть краткое описание как запустить сервис, тесты и как увидеть документацию к API
6. **Реализовать необходимо только 2 из 3 endpoint`ов - создание и получения списка кошельков.** Endpoint для перевода средств между кошельками **строго опционально и выполняется кандидатом по желанию.**

Это может быть полезным:

https://infura.io/ - нода для Ethereum БЧ.

https://pypi.org/project/ethereum/

https://web3py.readthedocs.io/en/stable/ - библиотека для работы с БЧ Ethereum для Python (в ней есть все необходимые методы для получения балансов, создания кошельков и совершения транзакций)