Тестовое задание для стажера на позицию «Go-разработчик»

Необходимо разработать приложение EWallet реализующее систему обработки транзакций платёжной системы. Приложение должно быть реализовано в виде HTTP сервера, реализующее REST API. Сервер должен реализовать 4 метода и их логику:

**Создание кошелька**

* Эндпоинт - POST /api/v1/wallet
* Параметры запроса отсутствуют
* Ответ содержит JSON-объект с состоянием созданного кошелька. Объект содержит параметры:
* id – строковый ID кошелька. Генерируется сервером
* balance – дробное число, баланс кошелька
* Созданный кошелек должен иметь сумму 100.0 у.е. на балансе

**Перевод средств с одного кошелька на другой**

* Эндпоинт - POST /api/v1/wallet/{walletId}/send
* Параметры запроса:
  + walletId – строковый ID кошелька, указан в пути запроса
  + JSON-объект в теле запроса с параметрами:
    - to – ID кошелька, куда нужно перевести деньги
    - amount – сумма перевода
* Статус ответа 200 если перевод успешен
* Статус ответа 404 если исходящий кошелек не найден
* Статус ответа 400 если целевой кошелек не найден или на исходящем нет нужной суммы

**Получение историй входящих и исходящих транзакций**

* Эндпоинт – GET /api/v1/wallet/{walletId}/history
* Параметры запроса:
  + walletId – строковый ID кошелька, указан в пути запроса
* Ответ с статусом 200 если кошелек найден. Ответ должен содержать в теле массив JSON-объектов с входящими и исходящими транзакциями кошелька. Каждый объект содержит параметры:
  + time – дата и время перевода в формате RFC 3339
  + from – ID исходящего кошелька
  + to – ID входящего кошелька
  + amount – сумма перевода. Дробное число
* Статус ответа 404 если указанный кошелек не найден

**Получение текущего состояния кошелька**

* Эндпоинт – GET /api/v1/wallet/{walletId}
* Параметры запроса:
  + walletId – строковый ID кошелька, указан в пути запроса
* Ответ с статусом 200 если кошелек найден. Ответ должен содержать в теле JSON-объект с текущим состоянием кошелька. Объект содержит параметры:
* id – строковый ID кошелька. Генерируется сервером
* balance – дробное число, баланс кошелька
* Статус ответа 404 если кошелек не найден

Для удобства к тестовому заданию приложено описание API в формате OpenAPI 3.0. Его можно открыть любым онлайн редактором, например Swagger Editor или [ReDoc](https://redocly.github.io/redoc/). Это описание дублирует написанное выше и уточняет его.

На что нужно обратить внимание при реализации:

* Безопасность: в приложении не должно быть уязвимостей, позволяющих произвольно менять данные в базе.
* Персистентность: данные и изменения не должны «теряться» теряться при перезапуске приложения.

Список используемых библиотек не ограничен, однако нужно учесть требования к стеку:

* Язык реализации – Go 1.21
* База данных – PostgreSQL, SQLite или MongoDB

Плюсом будет:

* Наличие в решении Dockerfile для сборки контейнера с приложением.
* Хранение исходного кода в системе контроля версий, например Git с публикацией на GitHub. В решении необходимо предоставить ссылку на репозиторий.