Написать микросервис, удовлетворяющий следующим требованиям:

1. Сервис в одном асинхронном потоке получает раз в N минут из любого удобного открытого источника (например, отсюда:https://www.cbr-xml-daily.ru/daily_utf8.xml) данные о курсе доллара, рубля и евро (по умполчанию). N передаётся скрипту параметром вида --period N

2. В аргументах скрипту так же передаётся начальный объём средств для каждой из валют в произвольном порядке для каждой из используемых валют. Примеры запуска:

python3 test.py --rub 1000 --usd 2000 --eur 3000 --period 10

python3 test.py --eur 52.5 --period 5 --rub 23.1 --usd 234.77

3. В качестве дополнительного параметра может передаваться --debug с возможными значениями из списка:

0, 1, true, false, True, False, y, n, Y, N

В случае если параметр debug принимает положительное значение, выводить содержимое request/response для апи в консоль. В противном случае выводить сообщение о старте приложения, об успешном получении данных о курсах валют и о общей сумме средств (п.5). Использовать разные уровни логирования (DEBUG, INFO, WARNING и т.п.).

4. Во втором асинхронном потоке сервер отвечает на HTTP запросы на порту 8080. Необходимо реализовать REST api, отвечающее на запросы следующего вида (тип запроса, url, payload //комментарий):

GET /usd/get

GET /rub/get

GET /eur/get

GET /amount/get

POST /amount/set {"usd":10}

POST /amount/set {"rub":100.5, "eur":10, "usd":20}

POST /modify {"usd":5} // добавить к текущему количеству usd 5

POST /modify {"eur":10, "rub":-20} // добавить к текущему количеству eur 10, уменьшить текущее количество rub на 20

На запрос /amount/get нужно отвечать общей суммой средств для каждой из трёх валют с учётом текущего курса, количеством каждой из валют отдельно и текущим курсом. Разницей в курсе покупки/продажи можно принебречь. Пример вывода:

rub: 100

usd: 200

eur: 300

rub-usd: 65.5

rub-eur: 73.4

usd-eur: 1.12

sum: 35220.0 rub / 537.52 usd / 479.93 eur

К заголовкам ответа добавить заголовок content-type со значением text/plain.

5. В третьем асинхронном потоке раз в минуту выводить в консоль те же данные, что в п.4, в случае, если изменился курс какой-либо из валют или количество средств относительно предыдущего вывода в консоль.

6. В четвертом асинхронном потоке сервер отвечает на события со стороны телеграмм бота (отправку сообщений, команд пользователями), должны быть идентичны получаемым по апи.

7. Приложение должно быть реализовано в виде модуля с абстрактным классом и второго модуля, импортирующего этот класс. При инициализации должна быть возможность передать наименования валют.

8. Телеграм-бот должен использовать обычные и инлайн кнопки, механизм конечных автоматов.

9. Код должен быть офомлен по набору правил оформленному в pyproject.toml, проверка осуществялется через ruff

10. Код должен быть оформлен тестами, покрытие не менее 85%. Проверить можно через pytest-cov.

Библиотеки, рекомендуемые к использованию: asyncio, aiohttp, argparse, logging, json, requests

Вариант	Валюты
1	RUB AUD JPY
2	RUB AZN RSD
3	RUB GBP RON
4	RUB AMD PLN
5	RUB BYN NOK
6	RUB BGN NZD
7	RUB BRL MDL
8	RUB HUF CNY
9	RUB VND KGS
10	RUB HKD QAR
11	RUB GEL CAD
12	RUB DKK KZT
13	RUB AED IDR
14	RUB USD INR
15	RUB EUR EGP