Заняття №38. Асинхронні web-фреймворки. FastAPI.Tornado. API. REST

Корисні посилання (обов'язкові до опрацювання):

https://habr.com/ru/post/351890/

https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api

https://fastapi.tiangolo.com/

https://fastapi.tiangolo.com/tutorial/query-params/

https://fastapi.tiangolo.com/tutorial/sql-databases/

https://www.tornadoweb.org/en/stable/

Правила здачі:

- Назва файлу має відповідати назві завдання (програми), якщо явно не вказано протилежне. Файли з кодом мають бути з розширенням тої мови на якій написати код (.py, ...)
- Форматування файлу має відповідати **PEP8**.
- Файли з рішеннями потрібно запушити на ваш git репозиторій з назвою **itstep**, в окрему папку з назвою **lesson38**. Репозиторій має бути публічним, як це перевірити можна дізнатися в офіційній документації вашого git провайдера, або за допомогою Google.
- якщо ви хочете тестувати код в тому ж файлі, що пишете помістіть виклики в конструкцію
  if __name__ == "__main__":
  ...
- Посилання на репозиторій помістіть в окремий файл під назвою **solution_lesson38_<your_login>** де **<your_login> це ваш студентський логін** та прикріпіть як рішення в mystat. Уважно перевірте, що ви прикріпляєте саме рішення до поточного заняття!
- за будь-яку невідповідність до вищезазначених пунктів оцінка знижується.

Завдання.

В процесі виконання ДЗ вам потрібно буде побудувати API як логічне продовження MoneyTracker. Реалізувати АПІ ви можете за допомогою фреймворків (Flask, FastAPI, Tornado) та використовуючи сторонні бібліотеки. На сервері ви маєте розширити частину що стосується зберігання даних і додати власника запису (або як окреме поле в таблиці, або окрему таблицю і зв'язати їх), або взагалі використовуючи NoSQL DB. В кожному API запиті має приходити параметр **login** який і буде ідентифікувати власника.
Перелік API Endpoints:
GET /records - всі записи (список)
GET /records/<id> - конкретний запис, видається тільки у разі якщо власник запису та логін, що передається в запиті співпадає
GET /incomes - список всіх записів з додатними значеннями
GET /outcomes - список всіх записів з від'ємними значеннями
GET /logins - список усіх користувачів (усі логіни)
POST /records - створення нового запису
PATCH /records/<id> - зміна конкретного запису по id
DELETE /records/<id> - видалення конкретного запису по id

Розширена частина.
Додати можливість фільтрації даних за допомогою параметрів **start_date="дата"** та **end_date="дата"**.

Розширене завдання++
Напишіть клієнта для вашого АПІ, для відображення даних. Це може бути python скрипт використовуючи бібліотеку **requests** та консольний інтерфейс. Це може бути **HTML, CSS, JS**. Або навіть **VueJS, React** чи **Angular** застосунок. Також якщо у вас є ідеї для розширення АПІ якимось чином, все це ви можете зробити і описати в **README.md** щоб було зрозуміло, що саме ви додали та як цим користуватися.