Лабораторная работа №4

Тематика:

1. Параллельное выполнение
2. Веб-программирование
3. docker-compose
4. CI/CD

Общее описание

Основные задания:

- Дополнить проект из лабораторной №3 модулем для работы с параллельным кодом (можно использовать asyncio, multiprocessing или multithreading на выбор);
- API должно подыматься в production режиме;
- Разработать Dockerfile для вашего проекта (или частей проекта);
- docker-compose файл, с помощью которого возможно запустить проект локально (должен включать образы вашего проекта (образы сделать публичными, чтобы преподаватель мог запустить у себя) и базы данных + возможно какие-то сервисы, необходимые для проекта);
- (*) добавить в проект возможность масштабирования API. Пример: предположим у меня есть REST API, которое умеет обрабатывать какие-то задачи в асинхронном режиме, при этом на каждую задачу создается уникальный ID, который возвращается в ответе на запрос и по которому можно получить статус задачи. Тогда если развернуть несколько экземпляров API (несколько контейнеров), вы должны гарантировать, что при одинаковом запросе на каждый

из них касательно статуса задачи, все они вернут один и тот же результат. То есть например складывать результат всех задач в базу данных или какое-нибудь стороннее хранилище;
- (*) закрыть приложение каким-нибудь веб-сервером (например, NGINX) и поднять его в отдельном контейнере;
- (*) Настроить CI, которое будет уметь делать: 1) сборка проекта; 2) запуск тестов; 3) пуш в dockerhub, если сборка прошла успешно; (можно использовать github actions или бесплатные CI/CD проекты: travis, circleci, etc…)
- Развернуть проект в облаке (например можно использовать google cloud platform, которая предоставляет бесплатную квоту в 300$ на год; но можно использовать все, что угодно) (в зависимости от темпов сдачи, тут может появиться * :));
- (*) Настроить CD (непрерывное развертывание) на сервер в облаке при пуше в master(main) ветку на github.

Требования к защите:
- знать и уметь объяснить преимущества и недостатки каждого из подходов (asyncio, multiprocessing, multithreading);
- уметь продемонстрировать способность написания параллельного кода. Т.е. при защите преподаватель может попросить вас написать решение какой-нибудь задачи используя возможности параллельного выполнения;
- знать, что такое docker-compose и как им пользоваться;
- все требования предыдущих лабораторных.