Python Pet Project

Общее описание: Необходимо реализовать веб-сайт с back-end на Django.

Общие требования: Проект имеет уникальный код на github, коммиты имеют адекватное описание и временной промежуток (не в последний день все коммиты). При обнаружении сильного сходства будем дополнительно общаться :)

Требования к проекту: Проект представляет собой решение какой-либо бизнес-задачи, а не проект “пустышку” с набором требований. Согласовать тему и архитектуру проекта с преподавателем.

Требования к коду:

- Наличие docs-strings, type annotations
- Проход таких стадий как mypy, isort, black (наличие файлов настройки в проекте), можно добавить их в Makefile или pre-commit хук
- Понятная архитектура с разбиением на приложения
- Вынос зависимостей в requirements.txt
- Отсутствие “мусорных” файлов в репозитории
- README с описанием проекта
- Не использовать секретные ключи без переменных окружения

Технические требования (back):

- Создание моделей (реализация 1-to-1, 1-to-many, many-to-many связей)
- Реализация регистрации, авторизации, аутентификации (ваш API должен иметь разный уровень доступности)
- Добавление моделей в django-admin, для администрирования
- Оптимизация запросов на чтение за счет введения индексов на поля
- Реализация CRUD запросов
- Реализация JOIN запросов
- Реализация логирования (разный тип, считывается с конфигураций)
- Приемлемый внешний вид (использование css, bootstrap)
- Использовать мощнейший фреймворк VanillaJS. Например: реализация toaster в случае клика или ошибки при валидации
- Реализация тестирования (pytest, unittest, fixture, parametrized)
- Покрытие тестами выше 80%
- Валидация форм и полей как на back-end так и на front-end
- Реализация asyncio, multiprocessing или multithreading (на выбор)

Технические требования (back) 9-10:

- Все что было выше
- Использование JS-фреймворков (React, Angular, VueJS)
- Различный формат вывода логов (например в production mode) (plain text, json и тд)
- Использование NOSQL базы данных

Технические требования (devops):

- Разработка Docker, docker-compose файлов (приложение, БД и тд.)
- Реализация различных режимов (dev, production)
- Реализация конфигурации
- Развернуть проект в облаке. Выбирать можно на вкус студента, но рекомендуется использовать что-то из heroku, AWS, GCP, Azure Cloud. (Аккуратно с платными ресурсами). Заранее посмотрите какую квоту на бесплатное пользование дает каждый облачный провайдер

Технические требования (devops) 9-10:

- Все что было выше
- Использование веб-сервера NGINX
- 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;
- Настроить CI, которое будет уметь делать:
  1) сборка проекта;
  2) запуск тестов;
  3) пуш в dockerhub (другой container registry по желанию студента), если сборка прошла успешно; (можно использовать github actions или бесплатные CI/CD проекты: travis, circleci, etc…)
- Настроить CD (непрерывное развертывание) на сервер в облаке при пуше в master(main) ветку на github

- Будет супер-плюсом, но точно не обязательно использование Celery, Redis, Prometheus, Grafana, Jaeger, Zabbix

Требования к CI/CD 9-10:

- Стадия linter
- Стадия build
- Стадия tests
- Стадия push to dockerhub
- Стадия deploy
- Дополнительные стадии по желанию