а) Задание заключается в реализации простого API для управления сущностями (create, delete, read) и API для логики по передаче объектов между пользователями.

Есть два списка сущностей - пользователи (users / атрибуты: логин, пароль) и объекты (items / атрибуты: название объекта)

Они связаны с собой отношением "один ко многим" (пользователь имеет несколько объектов, объект привязан к одному пользователю).

Задача включает в себя следующее:

1. Реализовать API для регистрации пользователя: логин, пароль. Логин и пароль могут передаваться в открытом виде
2. Реализовать API для авторизации пользователя и получения временного токена. Логин и пароль могут передаваться в открытом виде
3. Реализовать API для создания и удаления объектов у пользователя, а также получение списка объектов (для определения пользователя - дополнительным параметром передается токен)
4. Реализовать API для передачи объекта другому пользователю. Работать это должно следующим образом:
   * передающий пользователь отправляет запрос с id объекта и логин пользователя, которому он передает этот объект
   * в ответ он получает ссылку подтверждения, по которой должен перейти принимающий пользователь (и только он, принадлежность определяется по токену, который также передается дополнительным параметром).
   * при корректной обработке объект должен быть перепривязан от передающего пользователя к принимающему

**Спецификация для методов API**

Формат для всех HTTP-запросов и ответов API: JSON

# Регистрация пользователя

* POST-запрос (/registration)
* параметры запроса – логин, пароль пользователя
* параметры ответа – сообщение об успешной регистрации

# Авторизация пользователя

* POST-запрос (/login)
* параметра запроса – логин, пароль пользователя
* параметры ответа – временный токен аутентификации

# Создание объекта

* POST-запрос (/items/new)
* параметры запроса – временный токен, атрибуты объекта
* параметра ответа – сообщение об успешном создании объекта, id объекта, атрибуты объекта

# Удаление объекта

* DELETE-запрос (/items/:id)
* параметры запроса – временный токен, id объекта
* параметра ответа – сообщение об успешном удалении объекта

# Получение списка объектов

* GET-запрос (/items)
* параметры запроса – временный токен
* параметра ответа – список объектов пользователя с их id и атрибутами

# Генерация ссылки для передачи объекта

* POST-запрос (/send)
* параметры запроса – id передаваемого объекта, логин принимающего пользователя, временный токен
* параметры ответа – ссылка, по которой принимающий пользователь должен совершить переход

# Переход по ссылке для получения объекта

* GET-запроса (/get)
* параметры запроса – ссылка, сгенерированная в ответе метода send и временный токен принимающего пользователя, для проверки принадлежности
* параметры ответа – сообщение об успешном получении объекта

б) Реализовать cicd, например, в gitlab, с выполнением тестов, сохранении образа в докер регистри(например, dockerhub), разворачивании в облачном сервере(aws, azure, heroku…)

в) Добавить load balancer и развернуть ваше приложение на нескольких физических серверах.

г) Использовать систему мониторинга SENTRY

д) Добавить отчет, содержащий описание ваших действий, ссылку на сайт, ссылку на VCS git (github, gitlab и пр.), трудности, с которыми столкнулись и как их преодалели.

Обратить внимание на:

* Корректное решение поставленной задачи
* Общая архитектура приложения, выбранный стек технологий
* Корректная обработка ошибок (коды и сообщения)
* Наличие спецификации API и тестов

Основной стек - Python 3.x. и любой фреймворк(flask, aiohhtp, fastapi, sanic ...)

В качестве хранилища можно выбрать любую удобную базу данных.

Для ведения разработки желательно использовать любой открытый ресурс с VCS git (github, gitlab и пр.)

Максимальный бал - 20