Тестовое задание для Python разработчика (FastAPI)

Введение: Данное тестовое задание включает как обязательные, так и дополнительные, более сложные пункты. Выполнение основных задач — минимальное требование, а выполнение дополнительных задач — не обязательное условие, но будет большим плюсом и позволит более высоко оценить вашу квалификацию. Дополнительные задачи направлены на более глубокое использование технологий и усложнение логики приложения, однако вы можете выбрать, какие именно из них реализовать.

Задание: Необходимо разработать веб-приложение для взаимодействия с участниками. Приложение должно включать следующие функциональные возможности:

1. **Создание модели участников**:
    - **Минимум**: У участника должны быть следующие поля: аватарка, пол, имя, фамилия, почта.
    - **Дополнительно**: Добавить проверку на уникальность почты (простая проверка без особой сложной логики).
2. **Эндпоинт для регистрации нового участника**:
    - URL: `/api/clients/create`
    - **Минимум**: При регистрации нового участника необходимо обработать его аватарку: наложить водяной знак (можно использовать любую картинку). Реализовать совместимость с авторизацией модели участника, включая обработку пароля.
    - **Дополнительно**: Использовать асинхронное выполнение для обработки аватарки с водяным знаком (требуется реализовать асинхронную функцию, которая будет выполнять наложение водяного знака, что позволит повысить производительность и избежать блокировки основного потока выполнения).
3. **Эндпоинт оценивания участником другого участника**:
    - URL: `/api/clients/{id}/match`
    - **Минимум**: Если возникает взаимная симпатия, вернуть почту другого участника и отправить на почты участников сообщение вида: «Вы понравились <имя>! Почта участника: <почта>».
    - **Дополнительно**: Добавить:
        - Лимит оценок в день, чтобы предотвратить злоупотребление функцией оценивания.
4. **Эндпоинт для получения списка участников**:
    - URL: `/api/list`
    - **Минимум**: Реализовать фильтрацию списка по полу, имени, фамилии.
    - **Дополнительно**: Добавить возможность сортировки участников по дате регистрации.
5. **Определение дистанции между участниками**:
    - **Минимум**: Добавить в модель участников поля долготы и широты. В эндпоинт списка добавить фильтр, который показывает участников в

пределах заданной дистанции относительно авторизованного пользователя (для расчета дистанции использовать Great-circle distance).
    - **Дополнительно**: Реализовать кэширование результатов для улучшения производительности при повторных запросах (ожидается, что кэширование позволит избежать повторных дорогостоящих расчетов дистанции для тех же данных, что улучшит производительность приложения).
6. **Предоставление результатов**:
    - **Минимум**: Скинуть ссылку на репозиторий.
    - **Дополнительно**: Задеплоить проект на любом удобном для вас хостинге, сервисах PaaS (Heroku) и т.п.

Требования к выполнению задания:

- **Приоритет в реализации**: Использование **FastAPI**.
- **Документация API**: Реализовать автоматическую документацию с использованием **Swagger** (встроенная в FastAPI).
- **Написание тестов**: Написание тестов не является обязательным, но будет являться плюсом.
- **Чистота кода и структура проекта**: Придерживаться стандартов PEP8, соблюдать правильную организацию кода и файлов проекта.
- **Интерфейс**: Веб-интерфейс не требуется. Все взаимодействие должно осуществляться через REST API.

Оценка выполнения задания будет основываться на:

- Качество реализации функционала.
- Оформление кода и соответствие стандартам PEP8.
- Чистота кода и логичная структура проекта.

Просьба предоставить ссылку на GitHub или GitLab с репозиторием проекта, чтобы мы могли ознакомиться с решением.

Правила работы с репозиторием:

- Каждый пункт задачи должен быть реализован в отдельном коммите.
- После выполнения каждого пункта необходимо выполнить следующие команды для фиксации и отправки изменений в репозиторий:
    - `git add .`
    - `git commit -m "Сделал то-то и то-то"`
    - `git push origin {ветка}`