📋 Тестовое задание: Сервис обмена мгновенными сообщениями

Цель: Разработать простой сервис для обмена мгновенными сообщениями между пользователями в реальном времени.

📌 Задачи:

1. **Регистрация и аутентификация пользователей:**
   - Реализуй возможность регистрации новых пользователей.
   - Обеспечь аутентификацию и авторизацию при работе с API.
2. **Отправка и получение сообщений:**
   - Пользователи могут отправлять сообщения друг другу.
   - Реализуй получение новых сообщений в реальном времени.
3. **Сохранение истории сообщений:**
   - Все сообщения должны сохраняться в базе данных.
   - Предусмотри возможность получения истории переписки между пользователями.
4. **Уведомления через Telegram-бота:**
   - Создай простого Telegram-бота с помощью Aiogram.
   - Бот должен уведомлять пользователя о новом сообщении, если он офлайн.
5. **Веб-интерфейс для тестирования:**
   - Разработай простую веб-страницу для взаимодействия с сервисом.
   - Дизайн не важен — можешь использовать любые готовые шаблоны.
   - Веб-интерфейс может быть реализован на любом языке и стеке по твоему выбору.

🛠️ Технические требования:

- **Язык программирования:** Python 3.10 или новее.
- **Фреймворк:** FastAPI для разработки RESTful API.
- **Асинхронность:**
  - Используй `async`/`await` для обработки запросов.
  - Реализуй реальное время с помощью WebSockets или другой технологии по твоему выбору.
- **Многопоточность:**
  - Используй многопоточность, где это необходимо для повышения производительности.
- **Базы данных:**

    - **PostgreSQL** для хранения пользователей и сообщений.
    - **Redis** для кэширования и хранения сессий.
- **ORM и миграции:**
    - SQLAlchemy для работы с базой данных.
    - Alembic для управления миграциями.
- **Фоновые задачи:**
    - Celery для обработки фоновых задач (например, отправка уведомлений через бота).
- **Контейнеризация:**
    - Docker для контейнеризации приложения.
- **Сервер:**
    - Nginx для обратного проксирования (можно использовать простой конфигурационный файл).

🎨 Веб-интерфейс:

- Должен позволять:
    - Регистрироваться и входить в систему.
    - Отправлять и получать сообщения.
- **Не трать много времени на дизайн.**
- Можешь использовать любые фреймворки или даже простую HTML-страницу.
- Готовые шаблоны и библиотеки приветствуются.

📄 Что мы ожидаем:

- **Код:** Чистый, структурированный, с понятными комментариями.
- **README:** Инструкции по запуску проекта и описанию его функциональности.
- **Git-репозиторий:** Пожалуйста, используй систему контроля версий и предоставь ссылку на репозиторий.
- **Документация API:** Можешь использовать встроенные возможности FastAPI (Swagger UI).

🔍 Дополнительная информация:

- **Не усложняй задачу больше необходимого.** Нам важно увидеть твоё понимание технологий и умение применять их на практике.
- **Golang:** Если у тебя есть базовые знания или желание изучить, можешь реализовать какую-либо небольшую часть на Go, но это не обязательно.
- **Вопросы:** Если что-то непонятно или нужны уточнения, не стесняйся спрашивать. Мы ценим открытое и честное общение.

  Telegram для связи: https://t.me/gptdns