Реализация страницы профиля с лентой постов

Frontend

Необходимо разработать **одну страницу**, которая включает:

1. **Блок профиля пользователя:**
    - Отображение информации о пользователе:
        - Аватар
        - Имя, фамилия
        - Дата рождения
        - Краткая информация («О себе»)
        - Контактные данные (email, телефон)
    - Возможность редактирования профиля (кнопка "Редактировать профиль")
    - Возможность смены аватара (клик по аватару → загрузка нового изображения)
2. **Лента постов (стена):**
    - Отображение списка постов пользователя
    - Каждый пост содержит:
        - Текст
        - Одно или несколько изображений
        - Дату публикации
    - Возможность взаимодействия с постами:
        - Добавление нового поста (текст + фото)
        - Удаление поста
        - Редактирование поста (изменение текста, добавление/удаление фото)
    - Пагинация, сортировка и фильтрация:
        - Пагинация (кнопка "Загрузить еще" или бесконечный скролл)
        - Сортировка по дате (новые / старые)

Технические требования:

- **React.js + TypeScript**
- **State-менеджмент:** Tanstack Query / SWR для работы с API, Zustand / Redux (по необходимости)
- **UI-библиотека:** Любая (shadcn/ui, Material-UI, Ant Design и др.)
- **Загрузка изображений:** Drag & Drop или через модальное окно

Backend

Необходимо разработать API для работы с профилем и постами пользователя.

Функционал API:

1. **Управление профилем:**
    - **Получение данных профиля**
    - **Обновление данных профиля**
    - **Обновление аватара** (загрузка фото и привязка к профилю)
2. **Управление постами (CRUD):**
    - **Получение списка постов**
        - Пагинация (limit, offset)
        - Сортировка (по дате)
    - **Создание поста**
        - Передача текста + изображений
    - **Редактирование поста**
        - Обновление текста
        - Добавление / удаление изображений
    - **Удаление поста**

Технические требования:

- **Nest.js**
- **База данных:** Любая SQL (PostgreSQL + TypeORM предпочтительно)
- **Хранение изображений:**
    - В файловой системе (`uploads/`)
    - **Аутентификация:** JWT (refresh + access tokens)
- **Docker:**
    - База данных и сервер должны быть завернуты в контейнеры
    - `docker-compose` для быстрой развертки

Дополнительные требования:

- Код должен быть **чистым и читаемым**, с разделением на компоненты
- Эндпоинты API должны соответствовать RESTful-архитектуре
- Желательно использовать Prettier для форматирования кода

Результат:

- Код должен быть выложен на GitHub
- Инструкция по запуску проекта (`README.md`) обязательна