**Общая концепция проекта: "MeteoSphere"**

Название: MeteoSphere  
Миссия: Создать продвинутый погодный веб-сервис с персонализацией, предоставляющий не только базовый прогноз, но и подробные данные о качестве воздуха, УФ-индексе и погодные карты.

**Техническое задание и распределение задач**

Вот детальный план работ для вашей команды из 4 человек.

**Команда Бэкенда (Рома и Дидар)**

Ваша главная цель — создать надежный API, который будет "сердцем" приложения: обрабатывать запросы, взаимодействовать с внешними сервисами (OpenWeatherMap, Telegram) и управлять данными пользователей.

Рома (Основной бэкенд-разработчик):

* Задача 1: Расширение API погоды.
  + Описание: Модифицировать существующие функции для получения дополнительных данных с OpenWeatherMap API.
  + Детали:
    - Создать новый эндпоинт /api/air\_quality, который по координатам (lat, lon) города будет возвращать данные о загрязнении воздуха (индекс AQI, CO, NO, O3, SO2 и т.д.).
    - Создать эндпоинт /api/uv\_index, который будет возвращать текущий и прогнозируемый УФ-индекс.
    - Объединить все данные (погода, прогноз, качество воздуха, УФ-индекс) в один комплексный ответ от API по запросу на город, чтобы фронтенд мог получить всё за один раз.
* Задача 2: Разработка системы пользователей.
  + Описание: Спроектировать и реализовать базу данных для хранения информации о пользователях.
  + Детали:
    - Настроить базу данных (например, SQLite для простоты или PostgreSQL для масштабируемости).
    - Создать модель User со следующими полями: id, telegram\_id (уникальный), username, first\_name, auth\_code (для 6-значного кода), code\_expiry (время жизни кода), created\_at.
    - Реализовать функции для создания, поиска и обновления пользователей.

Дидар (Бэкенд-разработчик, интеграции):

* Задача 1: Интеграция с Telegram Bot API.
  + Описание: Создать Telegram-бота, который будет обрабатывать регистрацию и аутентификацию пользователей.
  + Детали:
    - Зарегистрировать нового бота в Telegram через @BotFather и получить токен.
    - Написать логику для бота (можно использовать библиотеки python-telegram-bot или aiogram).
    - Реализовать обработчик команды /start. При первом запуске бот должен регистрировать пользователя (сохранять его telegram\_id, username в базу данных) и отправлять приветственное сообщение.
    - Реализовать обработчик команды /login. В ответ на эту команду бот должен генерировать уникальный 6-значный код, сохранять его в базу данных для этого пользователя и отправлять его в чат.
* Задача 2: Реализация API для аутентификации.
  + Описание: Создать эндпоинты на Flask для завершения процесса входа.
  + Детали:
    - Создать эндпоинт POST /api/auth/telegram\_code, который принимает telegram\_id и 6-значный code.
    - Логика: эндпоинт проверяет, совпадает ли код с тем, что хранится в базе, и не истекло ли его время жизни (например, 5 минут).
    - В случае успеха — генерировать сессионный токен (JWT) и отправлять его фронтенду. В случае неудачи — возвращать ошибку.
    - Создать эндпоинт GET /api/user/me для проверки текущей сессии, который будет защищен JWT-токеном.

**Команда Фронтенда (Алихан и Наташа)**

Ваша цель — создать красивый, интуитивно понятный и функциональный интерфейс, который будет эффективно отображать все новые данные и обеспечивать плавный процесс входа для пользователя.

Алихан (Ведущий фронтенд-разработчик, логика):

* Задача 1: Разработка страницы входа.
  + Описание: Создать новый HTML-шаблон и CSS-стили для страницы аутентификации.
  + Детали:
    - На странице должны быть два варианта входа:
      1. QR-код и ссылка: Ведут на вашего Telegram-бота (t.me/your\_bot\_name).
      2. Поле для ввода 6-значного кода: Пользователь вводит код, полученный от бота.
    - Написать JavaScript-логику для отправки кода на бэкенд (POST /api/auth/telegram\_code).
    - При успешном ответе от бэкенда — сохранять полученный JWT-токен (в localStorage) и перенаправлять пользователя на главную страницу.
* Задача 2: Интеграция с API и управление состоянием.
  + Описание: Модифицировать главную страницу для работы с новыми данными от бэкенда.
  + Детали:
    - Переписать логику запросов к бэкенду: теперь все запросы на получение данных о погоде должны включать JWT-токен в заголовках (Authorization: Bearer <token>).
    - Реализовать логику "защищенных" страниц: если пользователь не авторизован (нет токена), его нужно перенаправлять на страницу входа.
    - Создать функцию "Выход", которая удаляет токен из localStorage и перенаправляет на страницу входа.

Наташа (Фронтенд-разработчик, UI/UX):

* Задача 1: Визуализация данных о качестве воздуха и УФ.
  + Описание: Создать новые UI-компоненты для отображения данных о загрязнении и УФ-индексе.
  + Детали:
    - Разработать компонент "Качество воздуха", который будет показывать индекс AQI в виде цветного круга (от зеленого до темно-красного) и детальные показатели (CO, O3 и т.д.).
    - Создать компонент "УФ-индекс" с цветной шкалой и рекомендациями (например, "Низкий - защита не требуется", "Очень высокий - необходима максимальная защита").
    - Интегрировать эти компоненты в основную карточку погоды или вынести в отдельный блок.
* Задача 2: Интерактивная карта погоды.
  + Описание: Добавить на страницу интерактивную карту для визуализации погодных явлений.
  + Детали:
    - Интегрировать библиотеку для карт, например, Leaflet.js (бесплатная и простая).
    - Настроить карту так, чтобы она центрировалась на выбранном городе.
    - Использовать слои от OpenWeatherMap. У них есть готовые тайлы для карт (облачность, осадки, температура). Ваш API-ключ должен это поддерживать. Пример URL слоя: https://tile.openweathermap.org/map/precipitation\_new/{z}/{x}/{y}.png?appid={API\_KEY}.
    - Добавить на карту переключатель слоев, чтобы пользователь мог выбирать, что отображать: осадки, облака, температуру, ветер.

**План работы (Дорожная карта)**

1. Спринт 1: Основа и Аутентификация (1-2 недели)
   * Бэкенд: Рома создает модели пользователей. Дидар создает базового Telegram-бота и API для генерации кода.
   * Фронтенд: Алихан создает страницу входа. Наташа верстает новые компоненты для AQI и УФ-индекса (пока со статичными данными).
2. Спринт 2: Интеграция и новые фичи (1-2 недели)
   * Бэкенд: Рома расширяет API погоды (AQI, UV). Дидар завершает логику аутентификации с JWT-токенами.
   * Фронтенд: Алихан "оживляет" страницу входа, подключая ее к API. Наташа интегрирует карту Leaflet.js.
3. Спринт 3: Завершение и Тестирование (1 неделя)
   * Все: Совместное тестирование всего функционала: регистрация -> вход -> просмотр погоды -> просмотр карты -> выход. Исправление багов.
   * Фронтенд: Финальная полировка UI/UX, добавление анимаций.
   * Бэкенд: Оптимизация запросов, возможное добавление кэширования.