WE WATCH THE BEES

Пользовательские истории

Проект представляет собой веб-приложение «We Watch the Bees» – помощника для пчеловодов, объединяющего функции удалённого мониторинга состояния ульев и базы знаний по пчеловодству. Система позволяет пользователям регистрироваться, добавлять ульи с привязанными виртуальными датчиками (температура, влажность, вес), получать и визуализировать данные с этих датчиков в виде интерактивных графиков, а также пользоваться энциклопедией статей, разделённых по категориям, и оставлять комментарии к материалам.

Общая технологическая база (для всех вариантов)

- **Python 3.10+** – язык, на котором будет реализован весь бэкенд, а также скрипты эмуляции датчиков.
- **Django 4.x** – веб-фреймворк. Он берёт на себя:
  - Маршрутизацию (определение соответствия URL и представлений).
  - Работу с базой данных через ORM (объектно-реляционное отображение) – взаимодействие с таблицами как с классами Python.
  - Аутентификацию (регистрация, вход, выход) – практически готовые решения.
  - Административный интерфейс – встроенная админка для управления данными (пользователи, статьи, комментарии).
  - Формы – для обработки пользовательского ввода.
  - Шаблоны – для генерации HTML-страниц.
- **SQLite** – база данных, хранящаяся в одном файле. Не требует отдельной установки и настройки, идеальна для учебного проекта.
- **Bootstrap 5** – CSS-фреймворк. Подключается через CDN и предоставляет готовые стили для кнопок, карточек, сетки, что обеспечивает аккуратный и адаптивный дизайн.
- **HTML + Django Templates** – для вёрстки страниц. Шаблонизатор Django позволяет вставлять переменные, циклы и условия непосредственно в HTML.
- **JavaScript (ES6)** – для реализации интерактивных элементов:
  - **Fetch API** – встроенное средство браузера для асинхронных запросов к серверу без перезагрузки страницы.
  - **Chart.js** – библиотека для построения графиков, подключается через CDN.
- **Дополнительные пакеты Python (устанавливаются через pip)**:
  - `django-bootstrap5` – упрощает рендеринг форм в стиле Bootstrap (необязательно, но удобно). o `python-decouple` – для хранения секретных ключей и настроек в отдельном файле (повышает безопасность).
- **Эмулятор датчиков** – отдельный скрипт (или набор скриптов), написанный на Python, который имитирует работу реальных IoT-устройств. Он периодически генерирует случайные значения (температура, влажность, вес) и отправляет их на REST API сервера с помощью HTTP-запросов. Для этого используются стандартные библиотеки: `random`, `time`, `requests` (или `aiohttp` для асинхронности). Эмулятор не является частью основного Djangoприложения, а

работает параллельно, что позволяет тестировать систему в условиях, приближенных к реальным.

Функционал. Фокус на мониторинг (без базы знаний)

Функционал

- **Регистрация, вход, выход** – аналогично варианту 1.
- **Профиль пользователя** – редактирование личных данных.
- **Пасеки**: пользователь может создавать несколько пасек (название, местоположение).
- **Ульи**:
  - При добавлении улья указывается название и выбирается, какие датчики установлены (флажки: вес, температура, влажность). Каждому датчику присваивается уникальный идентификатор.
  - Для каждого выбранного датчика в БД создаётся соответствующая запись.
- **Эмуляция данных**:
  - Для каждого улья запускается отдельный экземпляр скрипта-эмулятора (или один многопоточный), который отправляет данные от имени всех датчиков этого улья. ○ Скрипт использует API сервера (`POST /api/v1/sensors/data`) и может работать с заданной периодичностью. ○ Альтернативно можно использовать один универсальный эмулятор, который перебирает все активные датчики в системе (получая их список через API) и отправляет показания.
- **Главная страница**: отображаются все ульи пользователя (с группировкой по пасекам) в виде плиток с последними показаниями и цветовой индикацией.
- **Карточка улья**:
  - Текущие показания по каждому датчику. ○ Графики за день, неделю, месяц (Chart.js, данные через Fetch API). ○ Список оповещений, относящихся к данному улью.
- **Оповещения**:
  - При обработке новых показаний сервер проверяет пороговые значения (например, температура > 38°C, резкое падение веса). При обнаружении аномалии создаётся запись в таблице `Alert`.
  - На главной странице отображается счётчик непрочитанных оповещений.
  - Доступна отдельная страница со списком всех оповещений, где можно фильтровать их по статусу и помечать как прочитанные.
- **Страницы из макета, не относящиеся к мониторингу**, могут быть оставлены заглушками с сообщением «В разработке».

Стек технологий

- Полностью повторяет стек варианта 1 (Django, SQLite, Bootstrap, Chart.js, Fetch API, djangobootstrap5, python-decouple).

- Дополнительно: эмулятор датчиков (Python + requests) может быть более сложным, поддерживающим многопоточность или асинхронность.

Преимущества

- Глубокая проработка мониторинга – основной ценности проекта.
- Поддержка нескольких пасек и гибкой конфигурации датчиков.
- Система оповещений добавляет практическую полезность.
- Эмуляция через внешний скрипт максимально приближена к реальной интеграции с IoTустройствами.

Недостатки

- Отсутствует база знаний, что делает макет не полностью реализованным.
- Требуется реализация логики проверки аномалий и цветовой индикации.
- Необходимо обеспечить надёжную работу эмулятора (управление несколькими датчиками).