МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Сервис для огородников

Студенты гр. 2383 _____ Анищенко А.И.

_____ Бараева Е.Н.

Студенты гр. 2381 _____ Двиков Д.В.

_____ Кривов С.А.

_____ Потапова Д.М.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2025

ЗАДАНИЕ

НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студенты

Анищенко А.И.

Бараева Е.Н.

Двиков Д.В.

Кривов С.А.

Потапова Д.М.

Группы 2383, 2381

Тема работы: Разработка сервиса для огородников

Исходные данные:

Необходимо реализовать сервис, в котором продвинутый огородник может вести записи (какие у него грядки, когда и что сажал, как ухаживал, как это выглядит на фото) и получать полезные рекомендации (когда что делать на участке). Стек: Flask, MongoDB, PyMongo.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарий использования»

«Модель данных»

«Разработанное приложение»

«Вывод»

«Список использованных источников»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 15.02.2025

Дата сдачи реферата: 22.05.2025

Дата защиты реферата: 22.05.2025

Студенты гр. 2383	_____	Анищенко А.И.
	_____	Бараева Е.Н.
Студенты гр. 2381	_____	Двиков Д.В.
	_____	Кривов С.А.
	_____	Потапова Д.М.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках данного проекта предлагалось разработать сервис для огородников, позволяющий удобно управлять записями о растениях, их посадках и уходе за ними. Для реализации был выбран стек технологий: MongoDB в качестве основной СУБД, PyMongo для работы с базой данных и Flask для создания веб-интерфейса. Во внимание будут приниматься такие аспекты как удобное взаимодействие с сервисом, быстрый доступ к данным и гибкость в управлении контентом. Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql1h25-garden

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность проекта обусловлена отсутствием удобных решений для садоводов, сочетающих функции учета посадок, полезных рекомендаций и платформы для обмена опытом. Основная задача заключается в разработке веб-сервиса, объединяющего личный дневник для фиксации посадок и ухода за растениями, систему персонализированных рекомендаций, а также возможность для общения между пользователями. Предлагаемое решение включает создание интуитивно понятной платформы с личным кабинетом для управления грядками, подсказками по уходу за растениями на основе введенных данных, возможностью публикации достижений в общей ленте, и поддержкой визуализации статистики. Реализация проекта позволит существенно упростить планирование садовых работ и повысить эффективность ухода за растениями за счет объединения рекомендаций приложения и опыта пользователей.

1. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

1.1. Макет UI

Макета пользовательского интерфейса изображен на Рисунке 1.

Рисунок 1 – Макет UI

1.2. Описание сценариев использования

Импорт данных:

Действующее лицо – Пользователь.

Основной сценарий:

1. Пользователь заходит в личный кабинет.
2. Нажимает кнопку «Импортировать данные».
3. Выбирает JSON-файл с сохранёнными данными.
4. Система проверяет данные и загружает их в базу.
5. Пользователь получает сообщение об успешном импортировании данных.

Альтернативный сценарий:

- Файл имеет неверную структуру – сервис сообщает об ошибке.
- В файле есть ошибки (например, отсутствуют обязательные поля) – сервис сообщает об ошибках и не загружает данные.

Представление данных:

«Просмотр записи в личном дневнике»

Действующее лицо – Пользователь.

Основной сценарий:

1. Пользователь заходит в личный кабинет.
2. Переходит в раздел «Личный Дневник».
3. Выбирает фильтр -- публичные или приватные записи.
4. Вводит название записи.
5. Выбирает период создания записи.
6. Выбирает дату редактирования записи.
7. Выбирает интересующую его запись.
8. Переходит на страницу просмотра конкретной записи.

Альтернативный сценарий:

- Личный дневник пустой – записей нет.

«Поиск участка»

Действующее лицо – Пользователь.

Основной сценарий:

1. Пользователь заходит в личный кабинет.

2. Переходит в раздел «Мои участки».

3. Вводит название участка.

4. Выбирает период создания участка.

5. Выбирает дату редактирования участка.

6. Находит необходимый участок.

Альтернативный сценарий:

- Пользователь вводит данные, но не находит соответствующий участок – список пуст.

«Поиск грядки»

Действующее лицо – Пользователь.

Основной сценарий:

1. Пользователь заходит в личный кабинет.

2. Переходит в раздел «Мои участки».

3. Выбирает необходимый участок.

4. Вводит название грядки.

5. Вводит название культуры.

6. Выбирает период создания грядки.

7. Выбирает период посева.

8. Выбирает дату редактирования грядки.

9. Находит необходимую грядку.

Альтернативный сценарий:

- Пользователь вводит данные, но не находит соответствующую грядку – список пуст.

«Поиск записи об уходе»

Действующее лицо – Пользователь.

Основной сценарий:

1. Пользователь заходит в личный кабинет.

2. Переходит в раздел «Уход за грядками».

3. Вводит название участка.

4. Вводит название грядки.

5. Выбирает период ухода.

6. Выбирает тип ухода.

7. Находит необходимую запись.

Альтернативный сценарий:

- Пользователь вводит данные, но не находит соответствующую запись – список пуст.

«Просмотр новостей»

Действующее лицо – Пользователь.

Основной сценарий:

1. Пользователь заходит в личный кабинет.

2. Пользователь заходит в раздел «Новости».

3. Выставляет фильтр (Самые новые или самые популярные).

4. Вводит название новости.

5. Вводит имя автора новости.

6. Выбирает период публикации новости.

7. Просматривает найденные новости.

«Просмотр статистики»

Действующее лицо – Пользователь.

Основной сценарий:

1. Пользователь заходит в личный кабинет.

2. Переходит в раздел «Статистика».

3. Выбирает вид культуры.

4. Выбирает период посадки.

5. Выбирает количество поливов в месяц.

6. Выбирает использование удобрений.

7. Выбирает, какие данные расположить по осям X и Y.

8. Нажимает создать диаграмму.

9. Получает интересующую его статистику.

Альтернативный сценарий:

- В системе нет данных для статистики – сервис сообщает об этом.

Анализ данных:

Действующее лицо – Пользователь.

Основной сценарий:

1. Пользователь заходит в личный кабинет.
2. Переходит в раздел «Статистика».
3. Выбирает вид культуры.
4. Выбирает период посадки.
5. Выбирает количество поливов в месяц.
6. Выбирает использование удобрений.
7. Выбирает, какие данные расположить по осям X и Y.
8. Нажимает создать диаграмму.
9. Получает интересующую его статистику.

Альтернативный сценарий:

- В системе нет данных для статистики – сервис сообщает об этом.

Экспорт данных:

Действующее лицо – Пользователь.

Основной сценарий:

1. Пользователь заходит в личный кабинет.
2. Нажимает кнопку «Экспортировать данные».
3. Система формирует JSON-файл со всеми участками, грядками и культурами.
4. Файл автоматически скачивается на устройство пользователя.

Альтернативный сценарий:

- В системе нет данных – сервис сообщает об этом.

2. МОДЕЛЬ ДАННЫХ

2.1. Нереляционная модель данных

Графическое представление модели представлено на Рисунке 2.

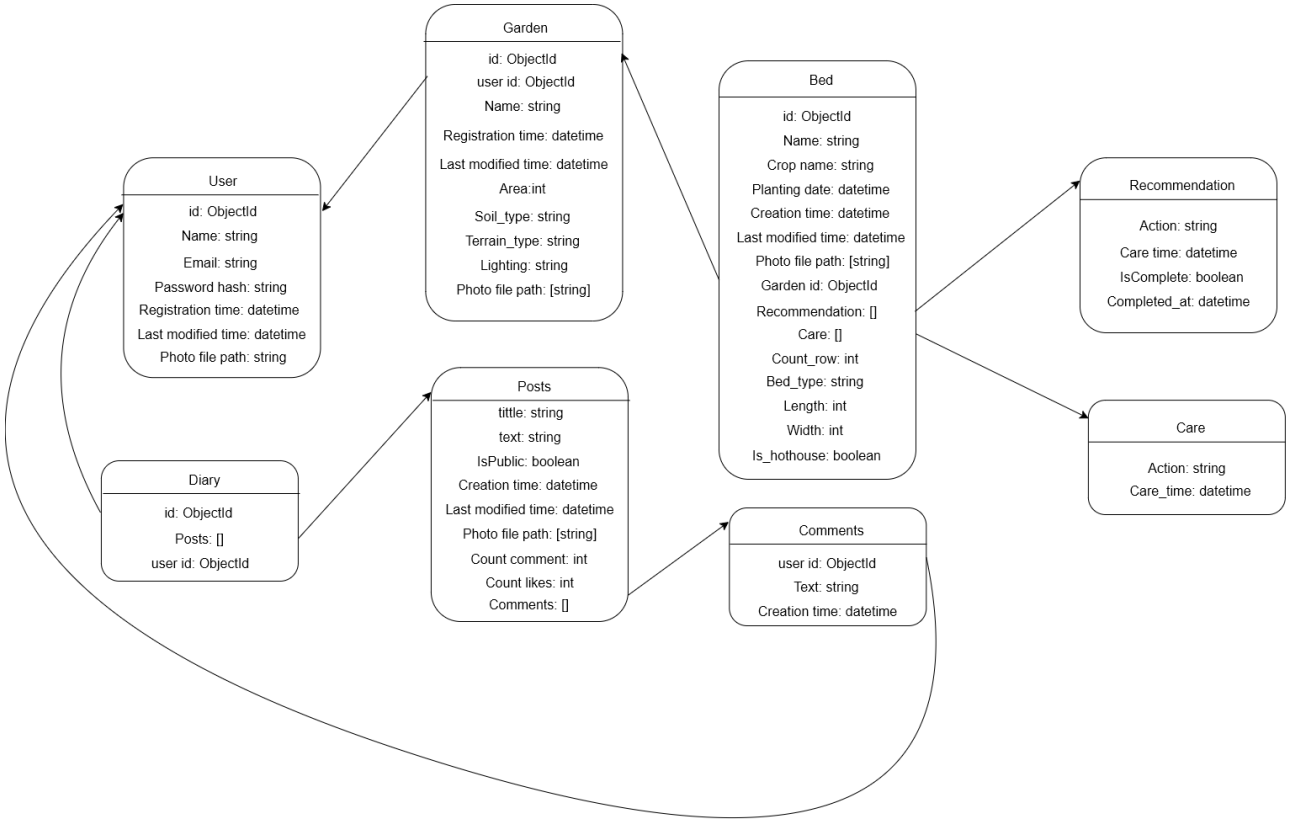

Рисунок 2 – Графическое представление модели

Описание назначений коллекций, типов данных и сущностей:

1. User

Назначение: Хранение данных о пользователях сервиса (учётная запись, личная информация).

- id: уникальный идентификатор. *Тип:* ObjectId (24 байта). *Назначение:* используется для ссылок на пользователя в других коллекциях.
- email: адрес электронной почты. *Тип:* String (в среднем 50 байт). *Назначение:* логин для аутентификации.
- password hash: хэш пароля. *Тип:* String (в среднем 60 байт). *Назначение:* хранение безопасного хэша пароля.

- name: имя пользователя. *Тип:* String (в среднем 45 байт). *Назначение:* отображается в профиле и при взаимодействиях.
- registration time: дата регистрации. *Тип:* Datetime (8 байт). *Назначение:* фиксирует момент создания учётной записи.
- last modified time: время последнего изменения данных пользователя. *Тип:* Datetime (8 байт). *Назначение:* помогает отслеживать, когда профиль в последний раз обновлялся.
- photo file path: путь к файлу с фотографией пользователя (у каждого пользователя одна аватарка). *Тип:* String (около 100 байт). *Назначение:* хранит путь к изображению профиля.

2. Garden

Назначение: Модель участка, принадлежащего конкретному пользователю.

- id: уникальный идентификатор. *Тип:* ObjectId (24 байта).
- name: название участка. *Тип:* String (в среднем 50 байт).
- registration time: время создания участка. *Тип:* Datetime (8 байт).
- last modified time: время последнего изменения данных об участке. *Тип:* Datetime (8 байт).
- userId: ссылка на пользователя, владеющего участком. *Тип:* ObjectId (24 байта). *Назначение:* устанавливает связь «User → Garden».
- area: площадь участка (в метрах квадратных). *Тип:* Double (8 байт).
- soil_type: тип почвы. *Тип:* String (в среднем 10 байт).
- terrain_type: тип местности (ровный / на склоне). *Тип:* String (в среднем 9 байт).
- lighting: тип освещения (затененный / солнечный). *Тип:* String (в среднем 10 байт).
- photo file path[]: список путей к фотографиям участка. *Тип:* Array (в среднем на один участок – 1 фотография → 100 байт). *Назначение:* хранит пути к изображениям участка.

3. Bed

Назначение: Модель «грядки», которая находится внутри участка (Garden).

- id: уникальный идентификатор. *Тип:* ObjectId (24 байта).
- name: название грядки. *Тип:* String (в среднем 50 байт).
- crop name: название посаженной культуры. *Тип:* String (в среднем 20 байт).
- planting date: время посева. *Тип:* Datetime (8 байт).
- creation time: время создания участка. *Тип:* Datetime (8 байт).
- last modified time: время последнего изменения информации о грядке. *Тип:* Datetime (8 байт).
- photo file path[]: список путей к фотографиям грядки. *Тип:* Array (в среднем на одну грядку – 1 фотография → 100 байт). *Назначение:* хранит пути к изображениям грядки.
- gardenId: ссылка на участок, к которому относится грядка. *Тип:* ObjectId (24 байта). *Назначение:* связь «Garden → Bed».
- recommendation[]: массив рекомендаций. *Тип:* Array<> (в среднем на одну грядку – 1 рекомендация → 27 байт). *Назначение:* вложенный список рекомендаций к грядке (и участку).
- care[]: массив записей об уходе. *Тип:* Array<> (в среднем на одну грядку (обнуление во время сбора урожая) – 60 записей → 900 байт). *Назначение:* вложенный список записей об уходе.
- count_row: количество рядов на грядке. *Тип:* Int32 (4 байта).
- length: длина грядки (в метрах). *Тип:* Double (8 байта).
- width: ширина грядки (в метрах). *Тип:* Double (8 байта).
- bed_type: тип грядки (высокая / низкая). *Тип:* String (в среднем 7 байт).
- is_hothouse: тепличная грядка или нет. *Тип:* Boolean (1 байт).

4. Recommendation

Назначение: Рекомендации для пользователя по уходу за участками и грядками.

- action: рекомендованное действие. *Тип:* String (в среднем 10 байт).
- Care_time: дата и время, когда рекомендация должна быть выполнена. *Тип:* Datetime (8 байт).

- isComplete: признак выполнения рекомендации. *Тип:* Boolean (1 байт).
- Completed_at: дата и время, когда рекомендация была выполнена. *Тип:* Datetime (8 байт).

5. Care

Назначение: Запись об уходе за грядкой на конкретном участке.

- action: совершенное действие – на выбор из Полив / Удобрение / Рыхление / Прополка / Срез / Сбор. *Тип:* String (в среднем 7 байт).
- care_time: дата и время, когда пользователь выполнил уход. *Тип:* Datetime (8 байт).

6. Diary

Назначение: Хранение структуры дневника пользователя.

- id: уникальный идентификатор. *Тип:* ObjectId (24 байта).
- posts: массив постов. *Тип:* Array<> (в среднем у пользователя – 10 постов, 1 из них – публичный → 8524 байта). *Назначение:* список постов в дневнике пользователя.
- userId: ссылка на пользователя, владеющего дневником *Тип:* ObjectId (24 байта).

7. Post

Назначение: Пост в дневнике или в новостях (блоге).

- title: заголовок поста. *Тип:* String (в среднем 50 байт).
- text: текст поста. *Тип:* String (в среднем 500 байт).
- isPublic: флаг видимости поста (приватный или публичный). *Тип:* Boolean (1 байт).
- creation time: время создания поста. *Тип:* Datetime (8 байт).
- last modified time: время последнего изменения поста. *Тип:* Datetime (8 байт).
- photo file path[]: список путей к фотографиям поста. *Тип:* Array (в среднем на один пост – 2 фотографии → 200 байт). *Назначение:* хранит пути к изображениям, приложенным к посту.

- count likes: количество лайков (в случае, если isPublic – False → всегда 0). *Тип:* Int32 (4 байта).
- count comment: количество комментариев (в случае, если isPublic – False → всегда 0). *Тип:* Int32 (4 байта).
- comments: массив комментариев к посту (в случае, если isPublic – False → пустой список). *Тип:* Array<> (в среднем на один пост – 3 комментария -> 846 байт). *Назначение:* список комментариев к посту в новостях.

8. Comments

Назначение: Комментарий к посту-новости.

- userId: ссылка на автора комментария. *Тип:* ObjectId (24 байта).
- text: текст комментария. *Тип:* String (в среднем 250 байт).
- creation time: время создания комментария. *Тип:* Datetime (8 байт).

Оценка объема информации, хранимой в модели:

Ниже приведён пример оценки объёма памяти, необходимой для хранения всех объектов нашей модели, с выражением общей зависимости через число пользователей (Nu). В расчётах использованы усреднённые размеры полей, указанные в описании сущностей, а также сделаны следующие предположения о количестве объектов:

- User: Количество записей = Nu.
- Garden: *Предположение:* в среднем у каждого пользователя 2 участка. Количество = $2 \cdot Nu$.
- Bed: *Предположение:* в среднем в каждом участке 8 грядок. Количество = $8 \cdot (\text{количество Garden}) = 8 \cdot 2 = 16 \cdot Nu$.
- Diary: У каждого пользователя ровно 1 дневник. Количество = Nu.

Далее приведём расчёт среднего размера одной записи для каждой коллекции, исходя из размеров типов данных, указанных выше (в байтах):

- User: $24 + 50 + 60 + 45 + 8 + 8 + 100 = 295$ байт
- Garden: $24 + 50 + 8 + 8 + 24 + 8 + 10 + 9 + 10 + 100 = 251$ байт

- Bed: $24 + 50 + 20 + 8 + 8 + 8 + 100 + 24 + 27 + 900 + 4 + 8 + 8 + 7 + +1 = 1197$ байт
- Diary: $24 + 8524 + 24 = 8572$ байт

Теперь подсчитаем общий объём для каждой коллекции, выражая результат через Nu:

- User: $295 \cdot Nu$
- Garden: $251 \cdot (2 \cdot Nu) = 502 \cdot Nu$
- Bed: $1197 \cdot (16 \cdot Nu) = 19152 \cdot Nu$
- Diary: $8572 \cdot Nu$

Складываем все вместе:

$V(Nu) = 295 \cdot Nu + 502 \cdot Nu + 19152 \cdot Nu + 8572 \cdot Nu$

Таким образом, общий объём хранения всех объектов:

$V(Nu) = 28521 \cdot Nu$ байт, где Nu – число пользователей.

Примеры запросов:

1. Регистрация пользователя

Проверка существующего email

```
db.user.findOne({ email: "ivanov@example.com" });
```

Вставка нового пользователя (если email свободен)

```
db.user.insertOne({
  name: "Новый пользователь",
  email: "new_user@example.com",
  password_hash: "scrypt$...",
  registration_time: new Date(),
  last_modified_time: new Date()
});
```

Кол-во запросов: 2. Коллекции: user.

2. Вход в личный кабинет

```
db.user.findOne({
  email: "ivanov@example.com",
  password_hash: "scrypt$ln=16384$r=8$p=1$dK4e..."
});
```

Кол-во запросов: 1. Коллекции: user.

3. Редактирование имени

```
db.user.updateOne(
  { _id: ObjectId("661a1d5e3b4e8a7f4c3b2a1d") },
```

```
    {
      $set: {
        name: "Иван Петров",
        last_modified_time: new Date()
      }
    }
  );
```

Кол-во запросов: 1. Коллекции: user.

4. Создание заметки/поста

Создание поста

```
  db.diary.updateOne(
    { userId: ObjectId("aa1b2c3d4e5f6a7b8c9d0e1f2") },
    {
      $setOnInsert: {
        _id: new ObjectId(), // Можно опустить, чтобы MongoDB
сгенерировал сам
        userId: ObjectId("aa1b2c3d4e5f6a7b8c9d0e1f2"),
        posts: []
      },
      $push: {
        posts: {
          title: "Новый урожай огурцов",
          text: "Собрали первые огурцы с лесного участка",
          isPublic: true,
          creation_time: new Date(),
          last_modified_time: new Date(),
          photo_file_path: [
          "/uploads/posts/cucumbers_2025.jpg",
          "/uploads/posts/cucumbers_2025_2.jpg"
          ],
          count_likes: 0,
          count_comment: 0,
          comments: []
        }
      }
    },
    { upsert: true }
  );
  db.diary.createIndex({ userId: 1 }, { unique: true });
```

Кол-во запросов: 2. Коллекции: diary.

5. Просмотр записей дневника

Параметры (пример):

```
  db.diary.aggregate([
    {
      $match: {
        userId: ObjectId("aa1b2c3d4e5f6a7b8c9d0e1f2")
      }
    },
    { $unwind: "$posts" },
    {
      $match: {
        "posts.creation_time": {
```

```
            $gte: ISODate("2025-04-01"),
            $lte: ISODate("2025-04-30")
          },
          "posts.last_modified_time": {
            $gte: ISODate("2025-04-15"),
             $lte: ISODate("2025-04-30")
          },
          "posts.isPublic": true,
          "posts.title": {
            $regex: "урожай",    // Поиск по части названия
            $options: "i"
          }
        }
      },
      {
        $sort: {
          "posts.creation_time": -1
        }
      },
      {
        $project: { // Опциональная проекция
          "posts.title": 1,
          "posts.text": 1,
          "posts.creation_time": 1,
          "posts.last_modified_time": 1
        }
      }
    ]);
```

Кол-во запросов: 1. Коллекции: diary.

6. Добавление участка

```
    db.garden.insertOne({
      userId: ObjectId("661a1d5e3b4e8a7f4c3b2a1d"),
      name: "Новый участок",
      registration_time: new Date(),
      last_modified_time: new Date(),
      area: 150.0,
      soil_type: "песчаная",
      terrain_type: "на склоне",
      lighting: "солнечный",
      photo_file_path: ["/uploads/garden/moscow_house.jpg",
"/uploads/garden/moscow_house.jpg_2"]
    });
```

Кол-во запросов: 1. Коллекции: garden.

7. Удаление участка

```
    db.garden.deleteOne({
      _id: ObjectId("55f3b8a9d4e6c1b2a8d3e5f2")
    });
```

Кол-во запросов: 1. Коллекции: garden.

8. Редактирование названия участка

```
    db.garden.updateOne(
      { _id: ObjectId("d4e5f6a7b8c9d0e1f2aa1b2c") },
      {
```

```
    $set: {
      name: "Обновленное название",
      last_modified_time: new Date()
    }
  }
);
```

Кол-во запросов: 1. Коллекции: garden.

9. Поиск участка

```
db.garden.aggregate([
  {
    $match: {
      userId: ObjectId("aa1b2c3d4e5f6a7b8c9d0e1f2"),
      name: { $regex: "Лесной", $options: "i" },
      registration_time: {
        $gte: ISODate("2025-01-01"),
        $lte: ISODate("2025-04-30")
      },
      last_modified_time: {
        $gte: ISODate("2025-04-20"),
        $lte: ISODate("2025-04-25")
      }
    }
  },
  {
    $project: { // опциональная проекция
      name: 1,
      registration_time: 1,
      last_modified_time: 1,
      area: 1,
      soil_type: 1,
      terrain_type: 1,
      lighting: 1,
      photo_file_path: 1
    }
  }
]);
```

Кол-во запросов: 1. Коллекции: garden.

10. Добавление грядки

```
db.bed.insertOne({
  name: "Клубничная грядка",
  crop_name: "Клубника",
  gardenId: ObjectId("d4e5f6a7b8c9d0e1f2aa1b2c"),
  planting_date: new Date("2025-05-01"),
  creation_time: new Date(),
  last_modified_time: new Date(),
  photo_file_path: ["/uploads/bed/strawberry.jpg"],
  recommendations: [],
  care: [],
  count_row: 10,
  length: 5.0,
  width: 4.0,
  bed_type: "низкая",
  is_hothouse": true
});
```

Кол-во запросов: 1. Коллекции: bed.

11. Удаление грядки

```
db.bed.deleteOne({
  _id: ObjectId("5a4b3c2d1e0f9a8b7c6d5e4f"),
  gardenId: ObjectId("d4e5f6a7b8c9d0e1f2aa1b2c")
});
```

Кол-во запросов: 1. Коллекции: bed.

12. Редактирование грядки

```
db.bed.updateOne(
  {
    _id: ObjectId("5a4b3c2d1e0f9a8b7c6d5e4f"),
    gardenId: ObjectId("d4e5f6a7b8c9d0e1f2aa1b2c")
  },
  {
    $set: {
      name: "Обновленная клубничная грядка",
      crop_name: "Ремонтантная клубника",
      planting_date: new Date("2025-05-10"),
      last_modified_time: new Date(),
      photo_file_path: ["/uploads/bed/new_strawberry.jpg"],
      count_row: 15,
      length: 9.0,
      width: 6.0,
      bed_type: "низкая",
      is_hothouse: false
    }
  }
);
```

Кол-во запросов: 1. Коллекции: bed.

13. Поиск грядки

```
db.bed.aggregate([
  {
    $match: {
      gardenId: ObjectId("d4e5f6a7b8c9d0e1f2aa1b2c"),
      name: { $regex: "клубничная", $options: "i" },
      crop_name: "Клубника",
      creation_time: {
        $gte: ISODate("2025-04-01"),
        $lte: ISODate("2025-05-31")
      },
      last_modified_time: {
        $gte: ISODate("2025-05-02"),
        $lte: ISODate("2025-06-10")
      },
      planting_date: {
        $gte: ISODate("2025-05-01"),
        $lte: ISODate("2025-05-10")
      }
    }
  },
  {
    $project: {
```

```
      name: 1,
      crop_name: 1,
      planting_date: 1,
      creation_time: 1,
      last_modified_time: 1
    }
  }
]);
```

Кол-во запросов: 1. Коллекции: bed.

14. Уход за грядками

```
db.bed.updateOne(
  {
    _id: ObjectId("5a4b3c2d1e0f9a8b7c6d5e4f"), // ID грядки
    gardenId: ObjectId("d4e5f6a7b8c9d0e1f2aa1b2c") // ID участка
  },
  {
    $push: {
      care: {
        action: "Полив",
        care_time: ISODate("2025-04-25T12:00:00Z")
      }
    },
    $set: { last_modified_time: new Date() }
  }
);
```

Кол-во запросов: 1. Коллекции: bed.

15. Удаление записи об уходе

```
db.bed.updateOne(
  {
    _id: ObjectId("5a4b3c2d1e0f9a8b7c6d5e4f"),
    gardenId: ObjectId("d4e5f6a7b8c9d0e1f2aa1b2c")
  },
  {
    $pull: {
      care: {
        action: "Полив",
        care_time: ISODate("2025-04-25T12:00:00Z")
      }
    }
  }
);
```

Кол-во запросов: 1. Коллекции: bed.

16. Поиск записи об уходе

```
db.bed.aggregate([
  {
    $match: {
      gardenId: ObjectId("d4e5f6a7b8c9d0e1f2aa1b2c"),
      name: "Клубничная грядка"
    }
  },
  { $unwind: "$care" }, // Разворачиваем массив care
  {
```

```
    $match: {
      "care.care_time": {
        $gte: ISODate("2025-04-01"),
        $lte: ISODate("2025-04-30")
      },
      "care.action": { $regex: "Полив", $options: "i" }
    }
  },
  {
    $project: {
      "gardenId": 1,
      "name": 1,
      "care.action": 1,
      "care.care_time": 1,
    }
  }
]);
```

Кол-во запросов: 1. Коллекции: bed.

17. Получение рекомендаций

```
db.bed.find(
  {
    "recommendations.isComplete": false,
    gardenId: ObjectId("d4e5f6a7b8c9d0e1f2aa1b2c")
  },
  {
    name: 1,                // Показать название грядки
    "recommendations.$": 1, // Только НЕвыполненные рекомендации
    _id: 0                  // Скрыть системное поле _id
  }
)
```

Кол-во запросов: 1. Коллекции: bed.

18. Отметка рекомендации как выполненной

```
db.bed.updateOne(
  {
    _id: ObjectId("5a4b3c2d1e0f9a8b7c6d5e4f"),
    "recommendations.action": "Подкормка азотом"
  },
  {
    $set: {
      "recommendations.$.isComplete": true,
      "recommendations.$.completed_at": new Date()
    }
  }
);
```

Кол-во запросов: 1. Коллекции: bed.

19. Просмотр новостей

```
db.diary.aggregate([
  { $unwind: "$posts" },
  {
    $match: {
      "posts.isPublic": true,
      "posts.creation_time": { $gte: ISODate("2025-04-01") }
```

```
      }
    },
    {
      $sort: {
        "posts.creation_time": -1 // Сортировка по дате (новые
сначала)
        // ИЛИ для сортировки по популярности:
        // "posts.count_likes": -1
      }
    }
  ]);
```

Кол-во запросов: 1. Коллекции: diary.

20. Комментирование новости

```
db.diary.updateOne(
  { "posts._id": ObjectId("5a4b3c2d1e0f9a8b7c6d5e4f") },
  {
    $push: {
      "posts.$.comments": {
        userId: ObjectId("f3e2d1c0b9a8d7e6f5a4b3c2"),
        text: "Интересный пост!",
        creation_time: new Date()
      }
    },
    $inc: { "posts.$.count_comment": 1 }
  }
);
```

Кол-во запросов: 1. Коллекции: diary.

21. Удаление комментария

```
db.diary.updateOne(
  { "posts.comments._id": ObjectId("5a4b3c2d1e0f9a8b7c6d5e4f") },
  {
    $pull: {
      "posts.$[].comments": {
        userId: ObjectId("f3e2d1c0b9a8d7e6f5a4b3c2")
      }
    },
    $inc: { "posts.$.count_comment": -1 }
  }
);
```

Кол-во запросов: 1. Коллекции: diary.

22. Оценивание новости

```
db.diary.updateOne(
  { "posts._id": ObjectId("5a4b3c2d1e0f9a8b7c6d5e4f") },
  { $inc: { "posts.$.count_likes": 1 } }
);
```

Кол-во запросов: 1. Коллекции: diary.

23. Экспорт данных (резервное копирование)

Получение данных пользователя

```
const user = db.user.findOne({ _id:
ObjectId("aa1b2c3d4e5f6a7b8c9d0e1f2") });
```

Получение всех участков (garden) пользователя

```
const garden = db.garden.find({ userId: user._id }).toArray();
```

Получение всех грядок (bed) для участков пользователя

```
const gardenIds = garden.map(g => g._id);
const bed = db.bed.find({ gardenId: { $in: gardenIds }
}).toArray();
const gardenIds = usergarden.map(g => g._id); // Получаем массив
_id участков
const userbed = db.bed.find({ gardenId: { $in: gardenIds }
}).toArray();
```

Получение данные дневника пользователя:

```
const diary = db.diary.findOne({  userId: user._id });
const backupData = {
  user,
  garden,
  bed,
  diary
};
```

Кол-во запросов: 4. Коллекции: garden, bed, diary, user.

24. Импорт данных

Очистка старых данных (опционально):

```
db.garden.deleteMany({ userId:
ObjectId("aa1b2c3d4e5f6a7b8c9d0e1f2") });
db.bed.deleteMany({ gardenId: { $in: gardenIds } });
db.diary.deleteMany({ userId: ObjectId("aa1b2c3d4e5f6a7b8c9d0e1f2")
});
```

Импорт новых данных

```
db.user.updateOne(
  {_id: ObjectId("aa1b2c3d4e5f6a7b8c9d0e1f2")},
  {$set: {
    name: backupData.user.name,
    email: backupData.user.email,
    photo_file_path: backupData.user.photo_file_path,
    last_modified_time: new Date()
  }}
);
db.garden.insertMany(backupData.garden);
db.bed.insertMany(dbackupData.bed);
db.diary.insertMany(backupData.diary);
```

Кол-во запросов: 7. Коллекции: garden, bed, diary, user.

2.2. Аналог модели данных для SQL СУБД

Графическое представление модели представлено на Рисунке 3.

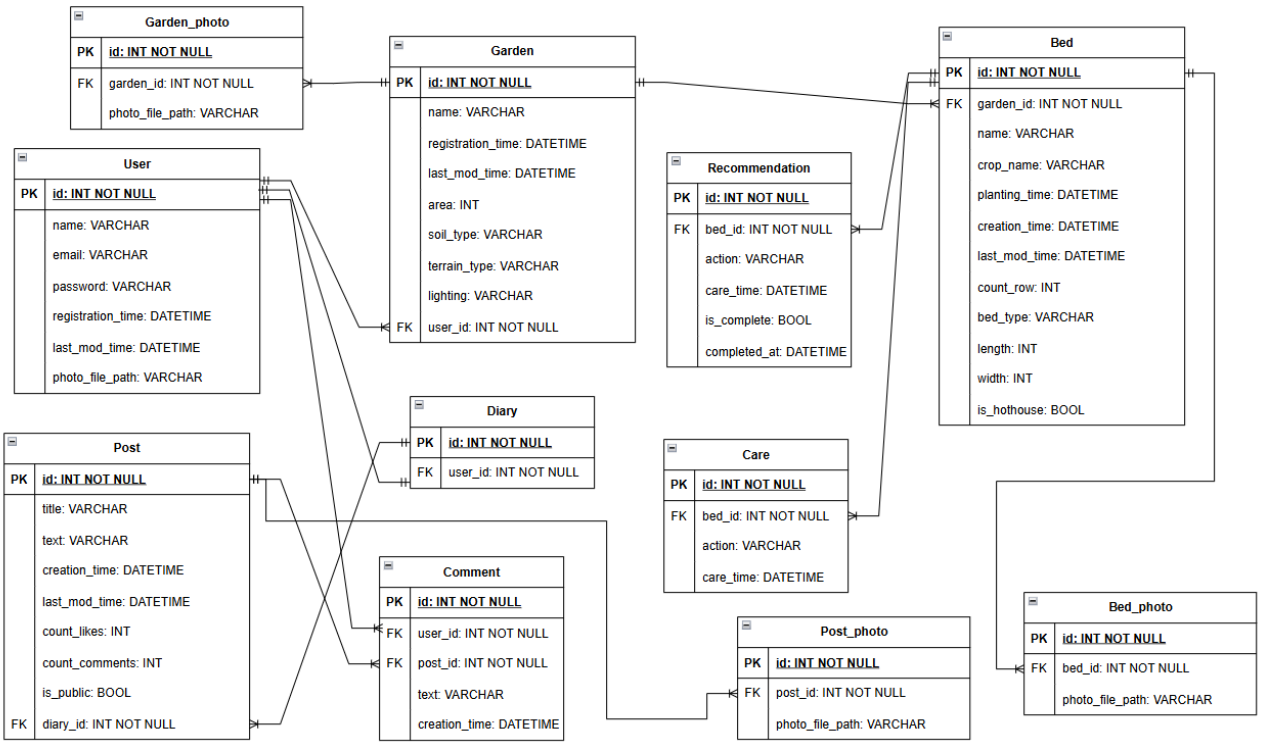

Рисунок 3 – Графическое представление модели

Описание назначений коллекций, типов данных и сущностей:

1. User

Назначение: Хранение данных о пользователях сервиса (учётная запись, личная информация).

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта). *Назначение:* используется для ссылок на пользователя в других сущностях.
- email: адрес электронной почты. *Тип:* VARCHAR (в среднем 50 байт). *Назначение:* логин для аутентификации.
- password: хэш пароля *Тип:* VARCHAR (в среднем 60 байт). *Назначение:* хранение безопасного хэша пароля.
- name: имя пользователя. *Тип:* VARCHAR (в среднем 45 байт). *Назначение:* отображается в профиле и при взаимодействиях.
- registration time: дата регистрации. *Тип:* DATETIME (8 байт). *Назначение:* фиксирует момент создания учётной записи.

- last modified time: время последнего изменения данных пользователя. *Тип:* DATETIME (8 байт). *Назначение:* помогает отслеживать, когда профиль в последний раз обновлялся.
- photo file path: путь к файлу с фотографией пользователя. *Тип:* VARCHAR (около 100 байт). *Назначение:* хранит путь к изображению профиля.

2. Garden

Назначение: Модель участка, принадлежащего конкретному пользователю.

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- name: название участка. *Тип:* VARCHAR (в среднем 50 байт).
- registration time: время создания участка. *Тип:* DATETIME (8 байт).
- last modified time: время последнего изменения данных об участке. *Тип:* DATETIME (8 байт).
- area: площадь участка (в метрах квадратных). *Тип:* FLOAT (4 байта).
- soil_type: тип почвы. *Тип:* VARCHAR (в среднем 10 байт).
- terrain_type: тип местности (ровный / на склоне). *Тип:* VARCHAR (в среднем 9 байт).
- lighting: тип освещения (затененный / солнечный). *Тип:* VARCHAR (в среднем 10 байт).
- user_id: внешний ключ. *Тип:* INT (4 байта). *Назначение:* устанавливает связь с User.id.

3. Bed

Назначение: Модель «грядки», которая находится внутри участка (Garden).

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- name: название грядки. *Тип:* VARCHAR (в среднем 50 байт).
- crop_name: название посаженной культуры. *Тип:* VARCHAR (в среднем 20 байт).
- planting_time: время посева. *Тип:* DATETIME (8 байт).
- creation_time: время создания участка. *Тип:* DATETIME (8 байт).

- last_mod_time: время последнего изменения информации о грядке. *Тип:* DATETIME (8 байт).
- count_row: количество рядов на грядке. *Тип:* INT (4 байта).
- length: длина грядки (в метрах). *Тип:* FLOAT (4 байта).
- width: ширина грядки (в метрах). *Тип:* FLOAT (4 байта).
- bed_type: тип грядки (высокая / низкая). *Тип:* VARCHAR (в среднем 7 байт).
- is_hothouse: тепличная грядка или нет. *Тип:* BOOL (1 байт).
- garden_id: внешний ключ *Тип:* INT (4 байта). *Назначение:* связь с Garden.id.

4. Recommendation

Назначение: Рекомендации для пользователя по уходу за участками и грядками.

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- action: рекомендованное действие. *Тип:* VARCHAR (в среднем 20 байт).
- care_time: дата и время, когда рекомендация должна быть выполнена. *Тип:* DATETIME (8 байт).
- is_complete: признак выполнения рекомендации. *Тип:* BOOL (1 байт).
- completed_at: дата и время, когда рекомендация была выполнена. *Тип:* DATETIME (8 байт).
- bed_id: внешний ключ *Тип:* INT (4 байта). *Назначение:* связь с Bed.id.

5. Care

Назначение: Запись об уходе за грядкой на конкретном участке.

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- action: совершенное действие - на выбор из Полив / Удобрение / Рыхление / Прополка / Срез / Сбор. *Тип:* VARCHAR (в среднем 7 байт).
- care_time: дата и время, когда пользователь выполнил уход. *Тип:* DATETIME (8 байт).
- bed_id: внешний ключ *Тип:* INT (4 байта). *Назначение:* связь с Bed.id.

6. Diary

Назначение: Хранение структуры дневника пользователя.

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- user_id: внешний ключ. *Тип:* INT (4 байта) *Назначение:* связь с User.id.

7. Post

Назначение: Пост в дневнике или в новостях (блоге).

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- title: заголовок поста. *Тип:* VARCHAR (в среднем 50 байт).
- text: текст поста. *Тип:* TEXT (в среднем 500 байт).
- is_public: флаг видимости поста (приватный или публичный). *Тип:* BOOL (1 байт).
- creation_time: время создания поста. *Тип:* DATETIME (8 байт).
- last_mod_time: время последнего изменения поста. *Тип:* DATETIME (8 байт).
- count likes: количество лайков (в случае, если is_public – False → всегда 0). *Тип:* INT (4 байта).
- count comment: количество комментариев (в случае, если is_public – False → всегда 0). *Тип:* INT (4 байта).
- diary_id: внешний ключ. *Тип:* INT (4 байта). *Назначение:* связь с Diary.id.

8. Comments

Назначение: Комментарий к посту-новости.

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- user_id: внешний ключ. *Тип:* INT (4 байта). *Назначение:* связь с User.id.
- post_id: внешний ключ. *Тип:* INT (4 байта). *Назначение:* связь с Post.id.
- text: текст комментария. *Тип:* TEXT (в среднем 250 байт).
- creation time: время создания комментария. *Тип:* DATETIME (8 байт).

9. Garden_photo

Назначение: Фотографии к участкам.

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- garden_id: внешний ключ. *Тип:* INT (4 байта) *Назначение:* связь с Garden.id.

- photo file path: путь к файлу с фотографией. *Тип:* VARCHAR (около 100 байт).

10. Post_photo

Назначение: Фотографии к постам.

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- post_id: внешний ключ. *Тип:* INT (4 байта). *Назначение:* связь с Post.id.
- photo file path: путь к файлу с фотографией. *Тип:* VARCHAR (около 100 байт).

11. Bed_photo

Назначение: Фотографии грядок.

- id: уникальный идентификатор (первичный ключ). *Тип:* INT (4 байта).
- bed_id: внешний ключ. *Тип:* INT (4 байта). *Назначение:* связь с Bed.id.
- photo file path: путь к файлу с фотографией. *Тип:* VARCHAR (около 100 байт).

Оценка объема информации, хранимой в модели:

Ниже приведён пример оценки объёма памяти, необходимой для хранения всех объектов нашей модели, с выражением общей зависимости через число пользователей (Nu). В расчётах использованы усреднённые размеры полей, указанные в описании сущностей, а также сделаны следующие предположения о количестве объектов:

- User: Количество записей = Nu.
- Garden: *Предположение:* в среднем у каждого пользователя 2 участка. Количество = $2 \cdot Nu$.
- Garden_photo: *Предположение:* в среднем у каждого участка 1 фотография. Количество = $2 \cdot Nu$.
- Bed: *Предположение:* в среднем в каждом участке 8 грядок. Количество = $8 \cdot (\text{количество Garden}) = 8 \cdot 2 \cdot Nu = 16 \cdot Nu$.
- Bed_photo: *Предположение:* в среднем у каждой грядки 1 фотография. Количество = $16 \cdot Nu$.

- Recommendation: *Предположение:* в среднем на одну грядку одна рекомендация. Количество = $16 \cdot Nu$.
- Care: *Предположение:* в среднем на одну грядку (обнуление во время сбора урожая) – 60 записей об уходе. Количество = $60 \cdot 16 \cdot Nu = 960 \cdot Nu$.
- Diary: у каждого пользователя ровно 1 дневник. Количество = Nu.
- Post: *Предположение:* в среднем у пользователя – 10 постов, 1 из них – публичный. Количество = $10 \cdot Nu$.
- Post_photo: *Предположение:* в среднем у каждого поста 2 фотографии. Количество = $20 \cdot Nu$.
- Comment: *Предположение:* в среднем у пользователя 1 публичный пост, у которого в среднем 3 комментария. Количество = $3 \cdot Nu$.

Далее приведём расчёт среднего размера одной записи для каждой коллекции, исходя из размеров типов данных, указанных выше (в байтах):

- User: $4 + 50 + 60 + 45 + 8 + 8 + 100 = 275$ байт
- Garden: $4 + 50 + 8 + 8 + 4 + 10 + 9 + 10 + 4 = 107$ байт
- Bed: $4 + 50 + 20 + 8 + 8 + 8 + 4 + 4 + 4 + 7 + 1 + 4 = 122$ байт
- Diary: $4 + 4 = 8$ байт
- Recommendation: $4 + 20 + 8 + 8 + 1 + 4 = 45$ байт
- Care: $4 + 4 + 7 + 8 = 23$ байт
- Post: $4 + 50 + 500 + 1 + 8 + 8 + 4 + 4 + 4 = 583$ байт
- Comment: $4 + 4 + 4 + 250 + 8 = 270$ байт
- Garden photo: $4 + 4 + 100 = 108$ байт

  Аналогично
- Bed photo: 108 байт
- Post photo: 108 байт

Теперь подсчитаем общий объём для каждой коллекции, выражая результат через Nu:

- User: $275 \cdot Nu$
- Garden: $107 \cdot (2 \cdot Nu) = 214 \cdot Nu$

- Bed: $122 \cdot (16 \cdot Nu) = 1952 \cdot Nu$
- Diary: $8 \cdot Nu$
- Recommendation: $45 \cdot (16 \cdot Nu) = 720 \cdot Nu$
- Care: $23 \cdot (960 \cdot Nu) = 22080 \cdot Nu$
- Post: $583 \cdot (10 \cdot Nu) = 5830 \cdot Nu$
- Comment: $270 \cdot (3 \cdot Nu) = 810 \cdot Nu$
- Garden_photo: $108 \cdot (2 \cdot Nu) = 216 \cdot Nu$
- Bed_photo: $108 \cdot (16 \cdot Nu) = 1728 \cdot Nu$
- Post_photo: $108 \cdot (20 \cdot Nu) = 2160 \cdot Nu$

Складываем все вместе:

$V(Nu) = 275 \cdot Nu + 214 \cdot Nu + 1952 \cdot Nu + 8 \cdot Nu + 720 \cdot Nu + 22080 \cdot$

$\cdot Nu + 5830 \cdot Nu + 810 \cdot Nu + 216 \cdot Nu + 1728 \cdot Nu + 2160 \cdot Nu$

Таким образом, общий объём хранения всех объектов:

$V(Nu) = 35993 \cdot Nu$ байт, где Nu – число пользователей.

Примеры запросов:

1. Регистрация пользователя

Проверка существующего email

```
SELECT id FROM User WHERE email = 'ivanov@example.com';
```

Вставка нового пользователя

```
INSERT INTO User (id, name, email, password_hash,
registration_time, last_mod_time)
VALUES (
   1,
  'Новый пользователь',
  'new_user@example.com',
  'scrypt$...',
  CURRENT_TIMESTAMP,
  CURRENT_TIMESTAMP
);
```

Кол-во запросов: 2. Таблицы: User.

2. Вход в личный кабинет

```
SELECT id FROM User
WHERE
  email = 'ivanov@example.com'
  AND password_hash = 'scrypt$ln=16384$r=8$p=1$dK4e...';
```

Кол-во запросов: 1. Таблицы: User.

3. Редактирование имени

```
UPDATE User
SET
  name = 'Иван Петров',
  last_mod_time = CURRENT_TIMESTAMP
WHERE id = 1;
```

Кол-во запросов: 1. Таблицы: User.

4. Создание заметки/поста

Создание поста

```
INSERT INTO Post (
  id, title, text, is_public, creation_time,
  last_mod_time, count_likes, count_comments
) VALUES (
  1,
  'Новый урожай огурцов',
  'Собрали первые огурцы с лесного участка',
  TRUE,
  CURRENT_TIMESTAMP,
  CURRENT_TIMESTAMP,
  0,
  0
);

INSERT INTO Post_photo (
  id, post_id, photo_file_path
) VALUES (
  1, 1, '/uploads/Post/cucumbers_2025.jpg'
);
```

Кол-во запросов: 2. Таблицы: Post, Post_photo.

5. Просмотр записей дневника

```
SELECT
  title, text, creation_time, last_mod_time, photo_file_path
FROM Post p
JOIN Post_photo pp ON p.id = pp.post_id
WHERE
  AND creation_time BETWEEN '2025-04-01' AND '2025-04-30'
  AND last_mod_time BETWEEN '2025-04-15' AND '2025-04-30'
  AND is_public = TRUE
  AND title ILIKE '%урожай%'
ORDER BY creation_time DESC;
```

Кол-во запросов: 1. Таблицы: Post, Post_photo.

6. Добавление участка

```
INSERT INTO Garden (id, name, registration_time, last_mod_time,
area, soil_type, terrain_type, lighting, user_id)
VALUES (
  1,
  'Новый участок',
  CURRENT_TIMESTAMP,
  CURRENT_TIMESTAMP,
  10,
```

```
 'песчаная',
 'ровная',
 'солнечный',
  1
);

INSERT INTO Garden_photo (
  id, garden_id, photo_file_path
) VALUES
  (1, 1, '/uploads/Garden/new_2025.jpg'),
  (2, 1, '/uploads/Garden/new_2025_2.jpg')
);
```

Кол-во запросов: 2. Таблицы: Garden, Garden_photo.

7. Удаление участка

```
DELETE FROM Garden WHERE id = 1;
```

Кол-во запросов: 1. Таблицы: Garden.

8. Редактирование названия участка

```
UPDATE Garden
SET
  name = 'Обновленное название',
  last_mod_time = CURRENT_TIMESTAMP
WHERE id = 1;
```

Кол-во запросов: 1. Таблицы: Garden.

9. Поиск участка

```
SELECT
  name, registration_time, last_mod_time, area, soil_type,
terrain_type, lighting,
  photo_file_path
FROM Garden g
JOIN Garden_photo gh ON g.id = gp.garden_id
WHERE
  AND name ILIKE '%Лесной%'
  AND registration_time BETWEEN '2025-01-01' AND '2025-04-30'
  AND last_mod_time BETWEEN '2025-04-20' AND '2025-04-25';
```

Кол-во запросов: 1. Таблицы: Garden, Garden_photo.

10. Добавление грядки

```
INSERT INTO Bed  (
  id, name, crop_name, planting_date,
  creation_time, last_mod_time, count_row, bed_type, length, width,
is_hothouse
) VALUES (
  1,
  'Клубничная грядка',
  'Клубника',
  '2025-05-01',
  CURRENT_TIMESTAMP,
  CURRENT_TIMESTAMP,
  3,
  'низкая',
  2,
```

```
  4,
  False
);

INSERT INTO Bed_photo (
  id, bed_id, photo_file_path
) VALUES
  (1, 1, '/uploads/Bed/new_2025.jpg')
);
```

Кол-во запросов: 2. Таблицы: Bed, Bed_photo.

11. Удаление грядки

```
DELETE FROM Bed
WHERE id = 1;
```

Кол-во запросов: 1. Таблицы: Bed.

12. Редактирование грядки

```
UPDATE Bed
SET
  name = 'Обновленная клубничная грядка',
  crop_name = 'Ремонтантная клубника',
  planting_date = '2025-05-10',
  last_mod_time = CURRENT_TIMESTAMP,
WHERE id = 1;

UPDATE Bed_photo
SET
  photo_file_path = '/uploads/beds/new_strawberry.jpg'
WHERE bed_id = 1;
```

Кол-во запросов: 2. Таблицы: Bed, Bed_photo.

13. Поиск грядки

```
SELECT
  name, crop_name, planting_date, creation_time, last_mod_time,
count_row, bed_type,
  length, width, is_hothouse, photo_file_path
FROM Bed b
JOIN Bed_photo bp ON b.id = bp.bed_id
WHERE
  AND name ILIKE '%клубничная%'
  AND crop_name = 'Клубника'
  AND creation_time BETWEEN '2025-04-01' AND '2025-05-31'
  AND last_mod_time BETWEEN '2025-05-02' AND '2025-06-10'
  AND planting_date BETWEEN '2025-05-01' AND '2025-05-10';
```

Кол-во запросов: 1. Таблицы: Bed, Bed_photo.

14. Уход за грядками

Добавление записи об уходе

```
INSERT INTO Care (bed_id, action, care_time)
VALUES (1, 'Полив', '2025-04-25 12:00:00');
```

Кол-во запросов: 1. Таблицы: Care.

15. Удаление записи об уходе

```
DELETE FROM Care
WHERE
  bed_id = 1
  AND action = 'Полив'
  AND care_time = '2025-04-25 12:00:00';
```
Кол-во запросов: 1. Таблицы: Care.

16. Поиск записи об уходе

```
SELECT
  c.action, c.care_time, b.name
FROM Care c
JOIN Bed b ON c.bed_id = b.id
WHERE
  AND b.name = 'Клубничная грядка'
  AND c.care_time BETWEEN '2025-04-01' AND '2025-04-30'
  AND c.action ILIKE '%Полив%';
```
Кол-во запросов: 1. Таблицы: Care, Bed.

17. Получение рекомендаций

```
SELECT
  b.name, r.action
FROM Recommendation r
JOIN Bed b ON r.bed_id = b.id
WHERE
  r.is_complete = FALSE
  AND b.id = 1;
```
Кол-во запросов: 1. Таблицы: Recommendation, Bed.

18. Отметка рекомендации как выполненной

```
UPDATE Recommendation
SET
  is_complete = TRUE,
  completed_at = CURRENT_TIMESTAMP
WHERE id = 1;
```
Кол-во запросов: 1. Таблицы: Recommendation.

19. Просмотр новостей

```
SELECT
  title, text, creation_time, count_likes, photo_file_path
FROM Post p
JOIN Post_photo pp ON p.id = pp.post_id
WHERE
  is_public = TRUE
  AND creation_time >= '2025-04-01'
ORDER BY creation_time DESC;
```
Кол-во запросов: 1. Таблицы: Post, Post_photo.

20. Комментирование новости

Добавление комментария

```
INSERT INTO Comment (id, user_id, post_id, text, creation_time)
VALUES (1, 1, 1, 'Интересный пост!', CURRENT_TIMESTAMP);
```
Обновление счетчика комментариев

```
UPDATE Post
SET count_comments = count_comments + 1
WHERE id = 1;
```

Кол-во запросов: 2. Таблицы: Comment, Post.

21. Удаление комментария

Удаление комментария

```
DELETE FROM Comment
WHERE id = 1;
```

Обновление счетчика комментариев

```
UPDATE Post
SET count_comments = count_comments - 1
WHERE id = 1;
```

Кол-во запросов: 2. Таблицы: Comment, Post.

22. Оценивание новости

```
UPDATE Post
SET count_likes = count_likes + 1
WHERE id = 1;
```

Кол-во запросов: 1. Таблицы: Post.

23. Экспорт данных

Получение данных пользователя

```
SELECT * FROM User WHERE id = 1;
```

Получение участков пользователя

```
SELECT * FROM Garden WHERE user_id = 1;
```

Получение фото участков пользователя

```
SELECT * FROM Garden_photo WHERE garden_id IN (SELECT id FROM
Garden WHERE user_id = 1)
```

Получение грядок пользователя

```
SELECT * FROM Bed WHERE garden_id IN (SELECT id FROM gardens WHERE
user_id = 1);
```

Получение фото грядок пользователя

```
SELECT * FROM Bed_photo WHERE bed_id IN (SELECT id FROM Bed WHERE
garden_id IN (SELECT id FROM gardens WHERE user_id = 1))
```

Получение постов пользователя

```
SELECT * FROM Post WHERE diary_id = (SELECT id FROM diary WHERE
user_id = 1);
```

Получение фото с постов пользователя

```
SELECT * FROM Post_photo WHERE post_id IN (SELECT id FROM Post
WHERE diary_id = (SELECT id FROM diary WHERE user_id = 1))
```

Получение записей об уходе

```
SELECT * FROM Care WHERE bed_id IN (SELECT id FROM Bed WHERE
garden_id IN (SELECT id FROM gardens WHERE user_id = 1);
```

Получение рекомендаций

```
SELECT * FROM Recommendation WHERE bed_id IN (SELECT id FROM Bed
WHERE garden_id IN (SELECT id FROM gardens WHERE user_id = 1);
```

Получение комментариев

```
SELECT * FROM Comment WHERE post_id IN (SELECT id FROM Post WHERE
diary_id = (SELECT id FROM diary WHERE user_id = 1));
```

Кол-во запросов: 23 (учитывая вложенные). Таблицы: User, Garden, Bed, Post, Diary, Comment, Care, Recommendation, Garden_photo, Bed_photo, Post_photo.

24. Импорт данных

Очистка данных

```
DELETE FROM Garden WHERE user_id = 1;
DELETE FROM Garden_photo WHERE garden_id IN (SELECT id FROM Garden
WHERE user_id = 1);
DELETE FROM Bed WHERE garden_id IN (SELECT id FROM Garden WHERE
user_id = 1);
DELETE FROM Bed_photo WHERE bed_id IN (SELECT id FROM Bed WHERE
garden_id IN (SELECT id FROM gardens WHERE user_id = 1);
DELETE FROM Post WHERE diary_id = (SELECT id FROM diary WHERE
user_id = 1);
DELETE FROM Post_photo WHERE post_id IN (SELECT id FROM Post WHERE
diary_id = (SELECT id FROM diary WHERE user_id = 1))
DELETE FROM Care WHERE bed_id IN (SELECT id FROM Bed WHERE
garden_id IN (SELECT id FROM gardens WHERE user_id = 1);
DELETE FROM Recommendation WHERE bed_id IN (SELECT id FROM Bed
WHERE garden_id IN (SELECT id FROM gardens WHERE user_id = 1);
DELETE FROM Comment WHERE post_id IN (SELECT id FROM Post WHERE
diary_id = (SELECT id FROM diary WHERE user_id = 1));
```

Импорт данных

```
UPDATE User
SET
  name = 'Имя из бэкапа',
  email = 'email из бэкапа',
  photo_file_path = 'путь из бэкапа',
  last_mod_time = CURRENT_TIMESTAMP
WHERE id = 1;

INSERT INTO gardens (id, name, ...) VALUES (...);
INSERT INTO beds (id, name, ...) VALUES (...);
INSERT INTO Post (id, title, ...) VALUES (...);
```

Кол-во запросов: 33 (учитывая вложенные). Таблицы: User, gardens, beds, Post, Diary, Comment, Care, Recommendation, Garden_photo, Bed_photo, Post_photo.

2.3. Сравнение моделей

Удельный объем информации:

Параметр	Нереляционная модель	Реляционная модель
Фактический объем	28385Nu	35993Nu
Чистый объем	27449Nu	27609Nu
Избыточность	1.034	1.304

Запросы по отдельным юзкейсам:

Название запроса	NoSql, кол-во запросов	Sql, кол-во запросов	NoSql, кол-во коллекций	Sql, кол-во коллекций
Регистрация пользователя	2	2	1	1
Вход в личный кабинет	1	1	1	1
Редактирование имени	1	1	1	1
Создание заметки/поста	2	2	1	2
Просмотр записей дневника	1	1	1	2
Добавление участка	1	2	1	2
Удаление участка	1	1	1	1
Редактирование названия участка	1	1	1	1
Поиск участка	1	1	1	2
Добавление грядки	1	2	1	2
Удаление грядки	1	1	1	1
Редактирование грядки	1	2	1	2
Поиск грядки	1	1	1	2
Уход за грядками	1	1	1	1
Удаление записи об уходе	1	1	1	1

Поиск записи об уходе	1	1	1	2
Получение рекомендаций	1	1	1	2
Отметка рекомендации как выполненной	1	1	1	1
Просмотр новостей	1	1	1	2
Комментирование новости	1	2	1	2
Удаление комментария	1	2	1	2
Оценивание новости	1	1	1	1
Экспорт данных (резервное копирование)	4	23	4	11
Импорт данных	7	33	4	11

Вывод:

Коэффициент избыточности в реляционной модели оказался сильно выше, чем в нереляционной. При этом объем чистых данных почти одинаковый. Это связано с тем, что возможности NoSql позволяют нам хранить не структурированные данные, в то время как Sql требует создания отдельных сущностей для комментариев, рекомендаций и т.д.

В большинстве случаев количество запросов одинаково, однако Sql сильно проигрывает в сценариях импорта и экспорта данных, а также в количестве задействованных коллекций.

Модель данных NoSql оказалась эффективнее для данного проекта. Она выигрывает у Sql как по объему данных, так и по количеству запросов.

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

3.1. Краткое описание

Веб-приложение «Огородник» реализовано так: фронтенд отвечает за взаимодействие с пользователем через браузер, а бэкенд обеспечивает функциональность интерфейса и взаимодействует с базой данных. В нереляционной базе данных хранятся документы с информацией о пользователях, постах и комментариях, участках, грядках, личном дневнике, рекомендациях и действиях по уходу. Структура данных организована в коллекции: user, garden, bed, diary, post, care, recommendation, comments. Пользователь может регистрироваться, создавать и управлять участками и грядками, вести личный дневник с публичными и приватными постами, комментировать и оценивать новости, получать автоматические рекомендации по уходу и отмечать их выполнение, визуализировать кастомизированную статистику о себе. Данные могут быть экспортированы и импортированы в формате JSON. Приложение построено по REST-подходу: клиент отправляет запросы к API, а сервер обрабатывает их и возвращает JSON-ответы. Модель данных спроектирована так, чтобы легко масштабироваться при росте числа пользователей.

3.2. Использованные технологии

БД: MongoDB

Backend: Python, Flask

Frontend: HTML, CSS, JavaScript

3.3. Снимки экрана приложения

Экраны приложения отображены на Рисунках 4-21.

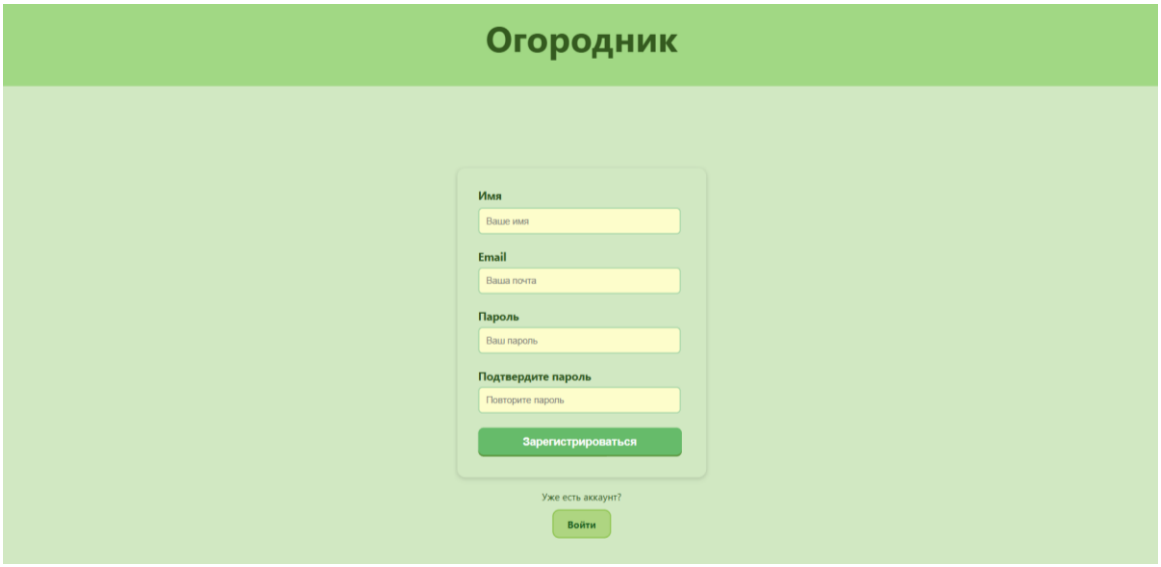

Рисунок 4 – Регистрация

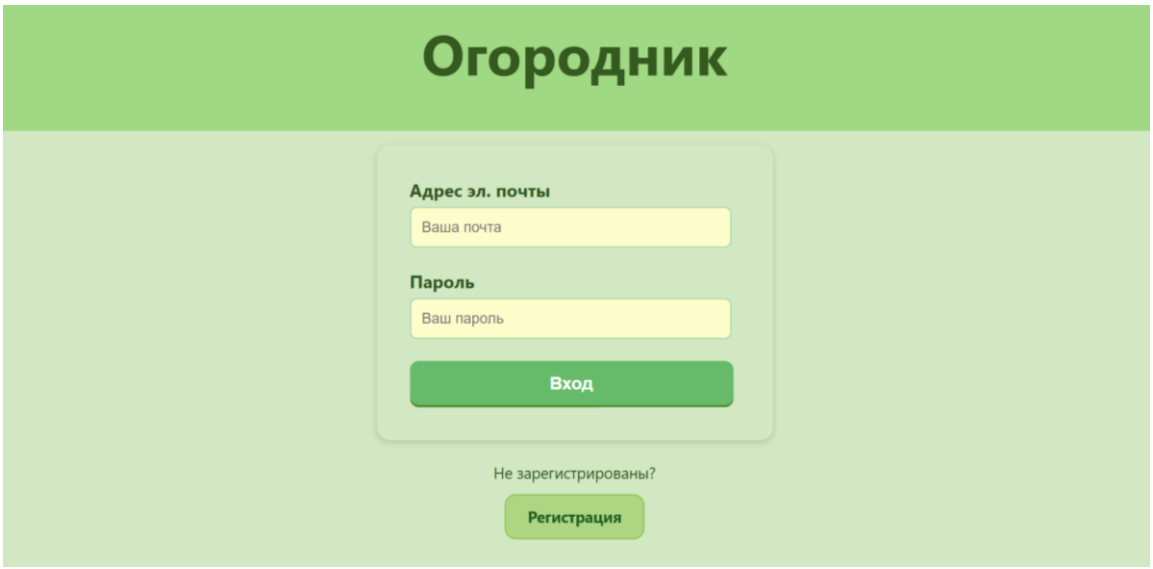

Рисунок 5 – Вход

Рисунок 6 – Профиль

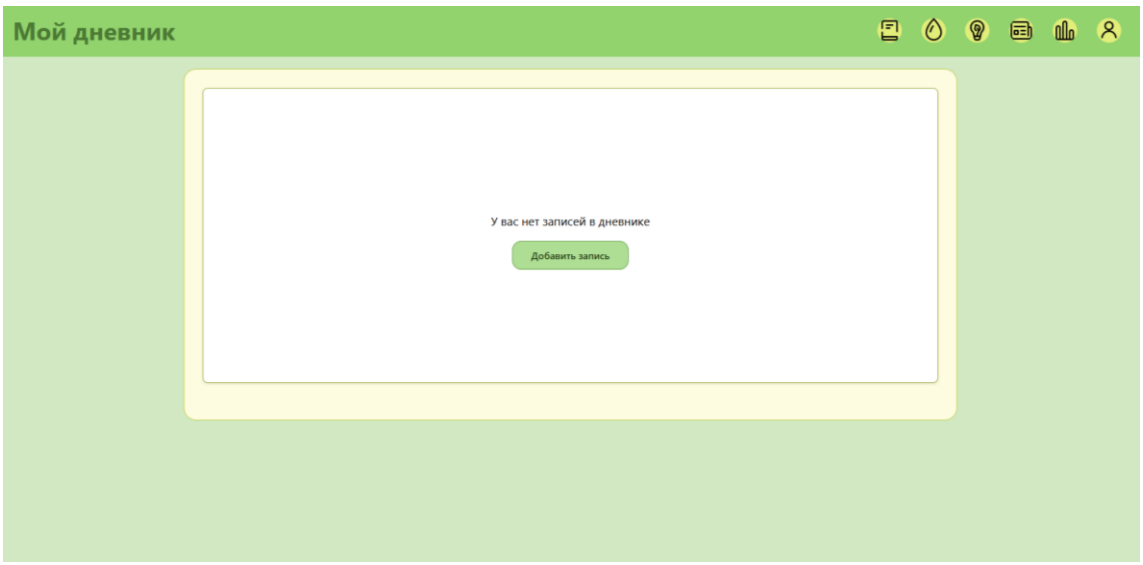

Рисунок 7 – Пустой личный дневник

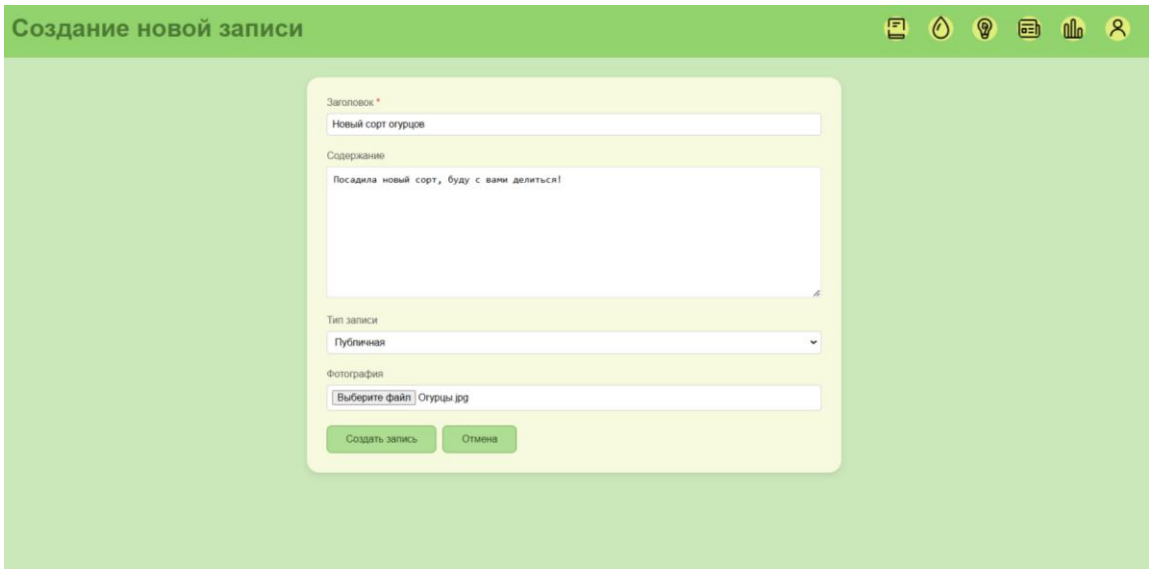

Рисунок 8 – Создание записи в личном дневнике

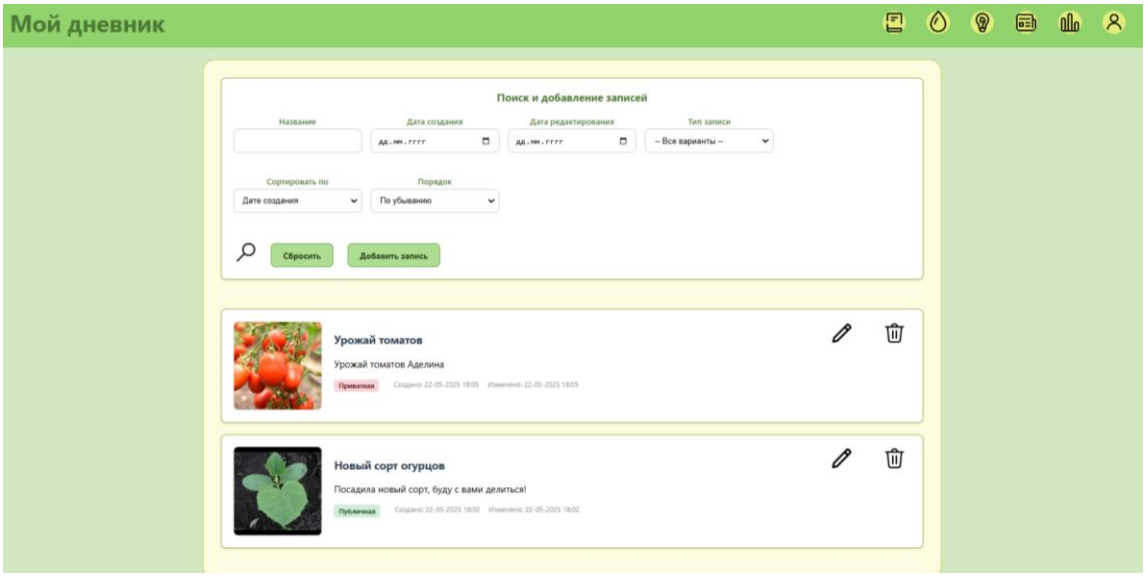

Рисунок 9 – Записи в личном дневнике

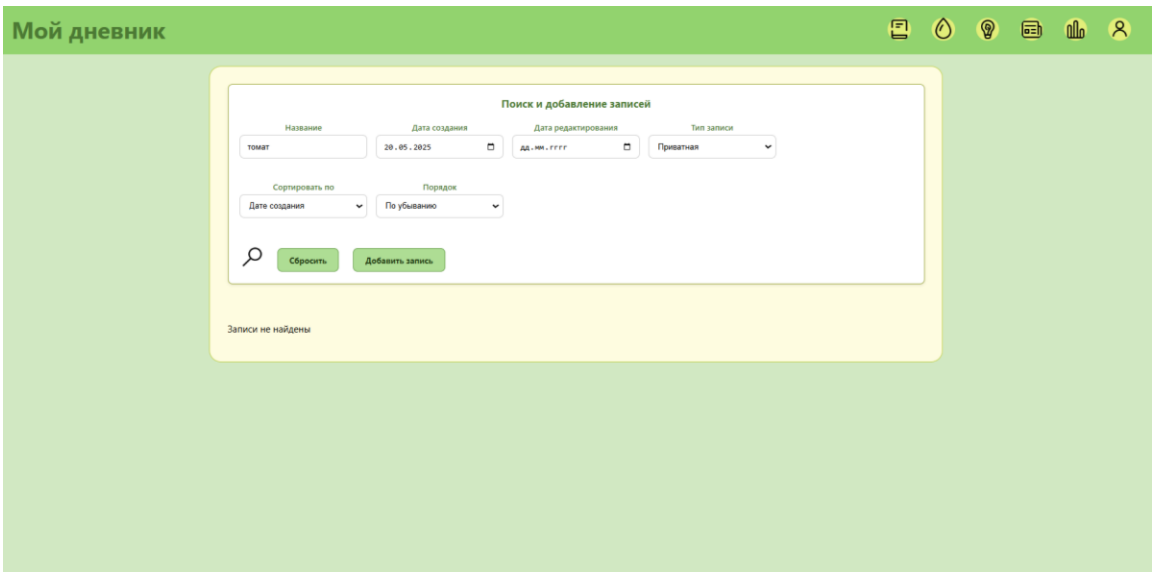

Рисунок 10 – Использование фильтров в личном дневнике

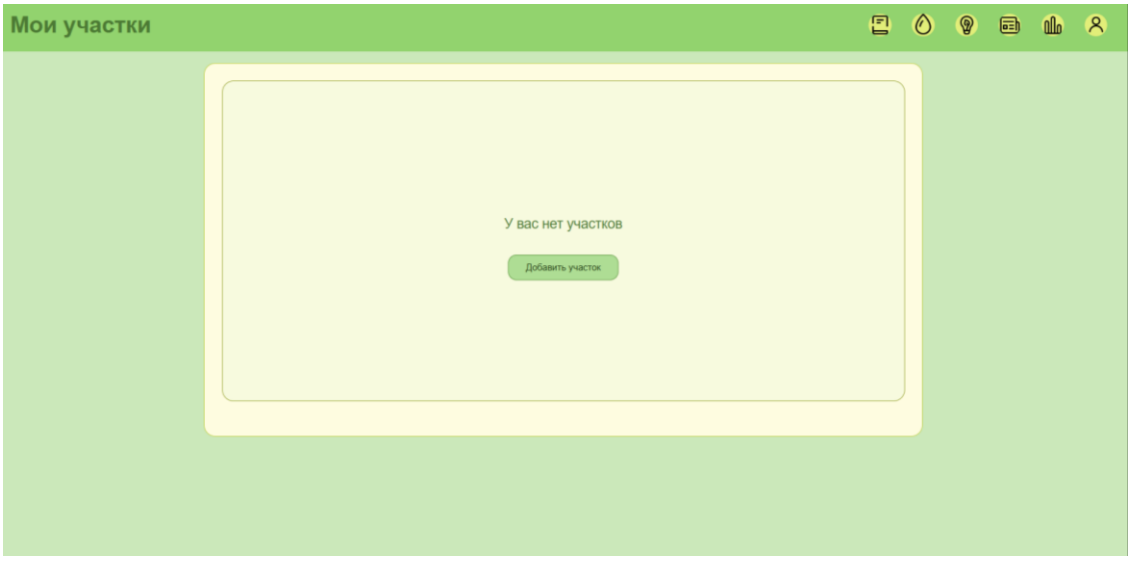

Рисунок 11 – Пустой раздел участков

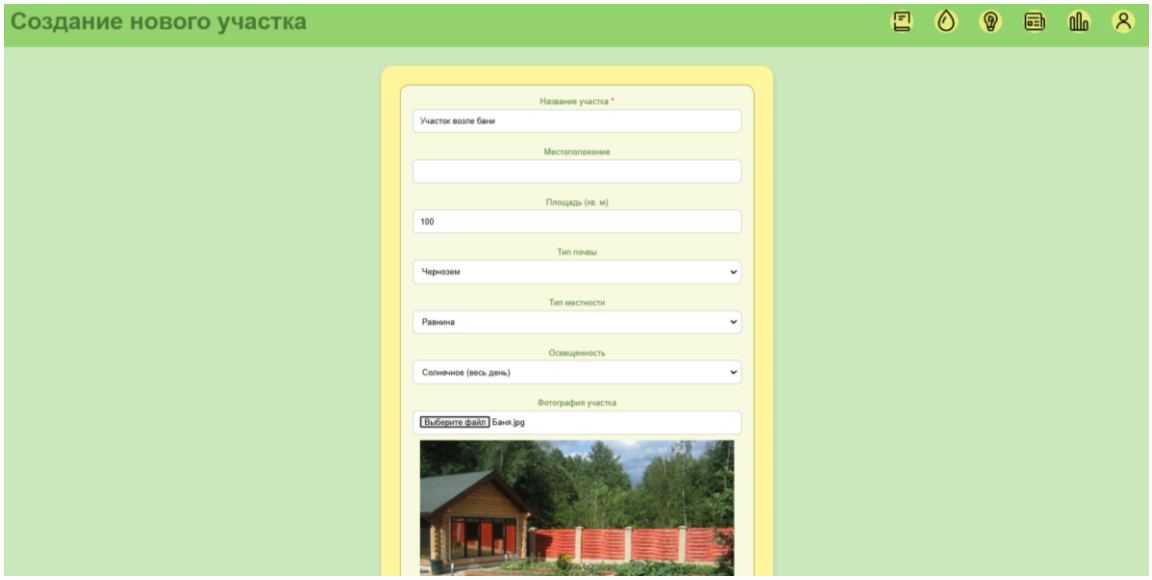

Рисунок 12 – Создание участка

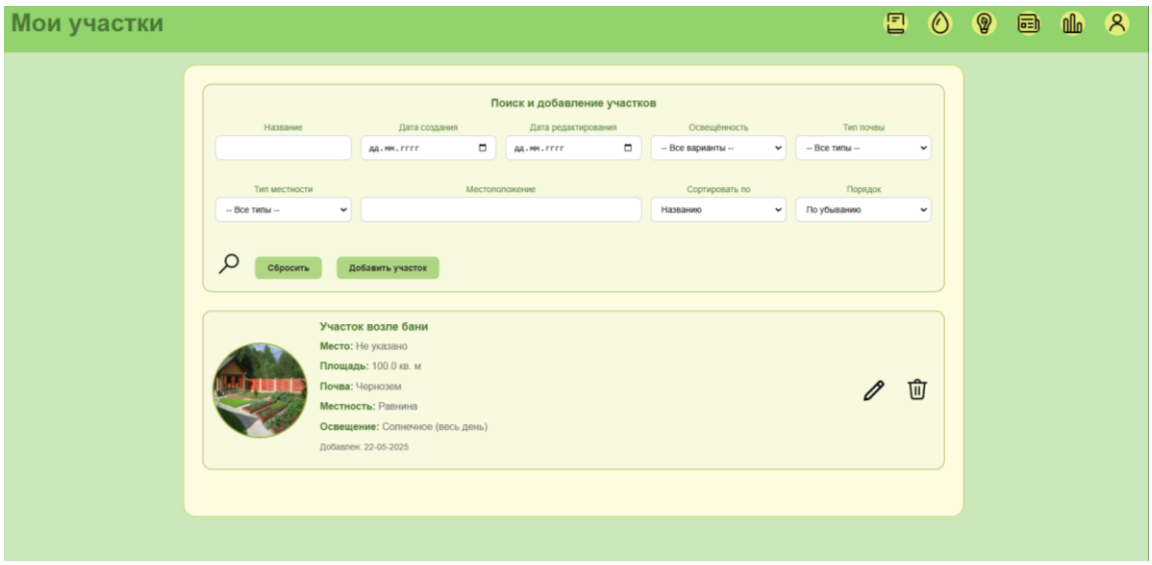

Рисунок 13 – Раздел участков

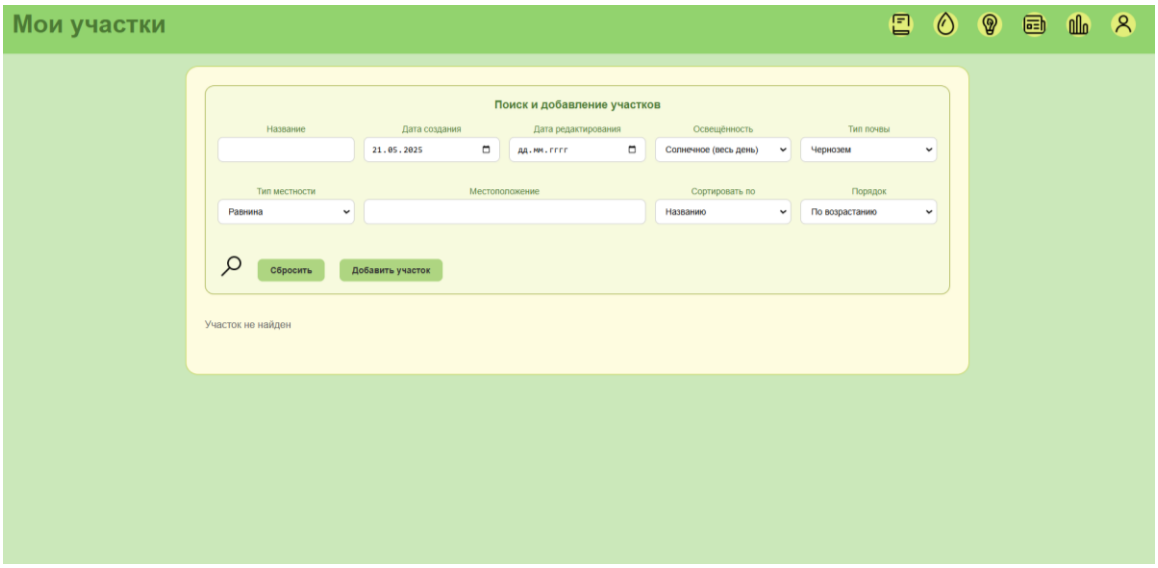

Рисунок 14 – Использование фильтров в разделе участков

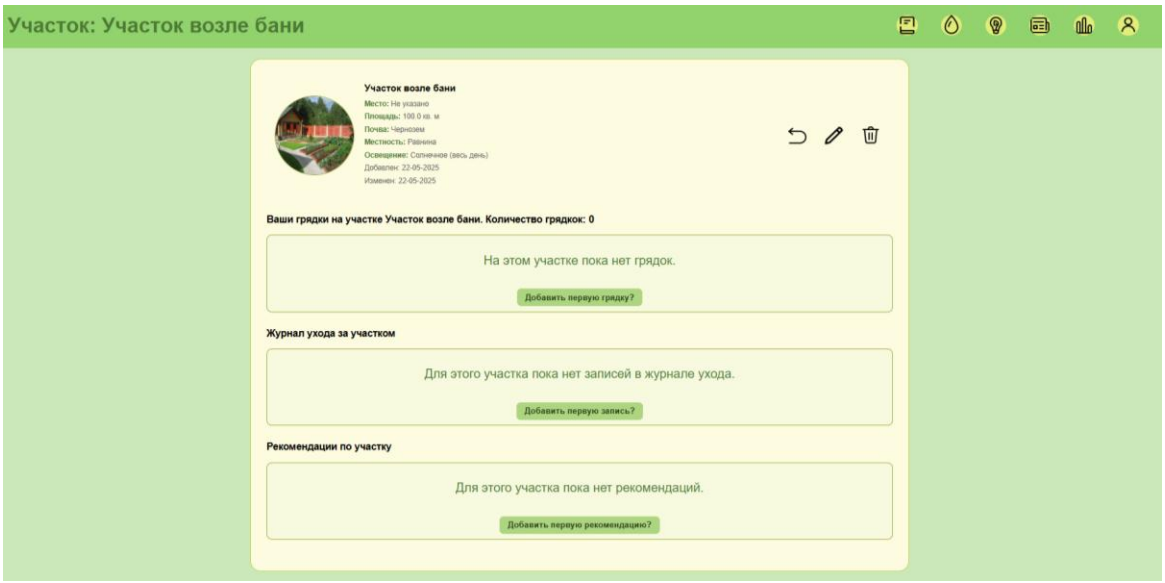

Рисунок 15 – Пустой участок

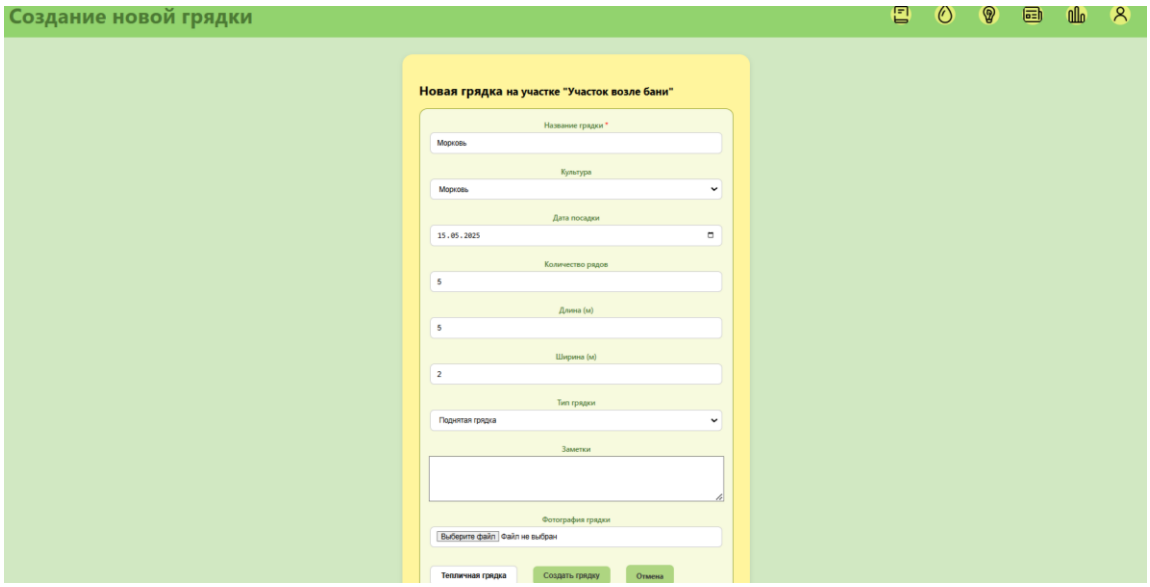

Рисунок 16 – Создание грядки на участке

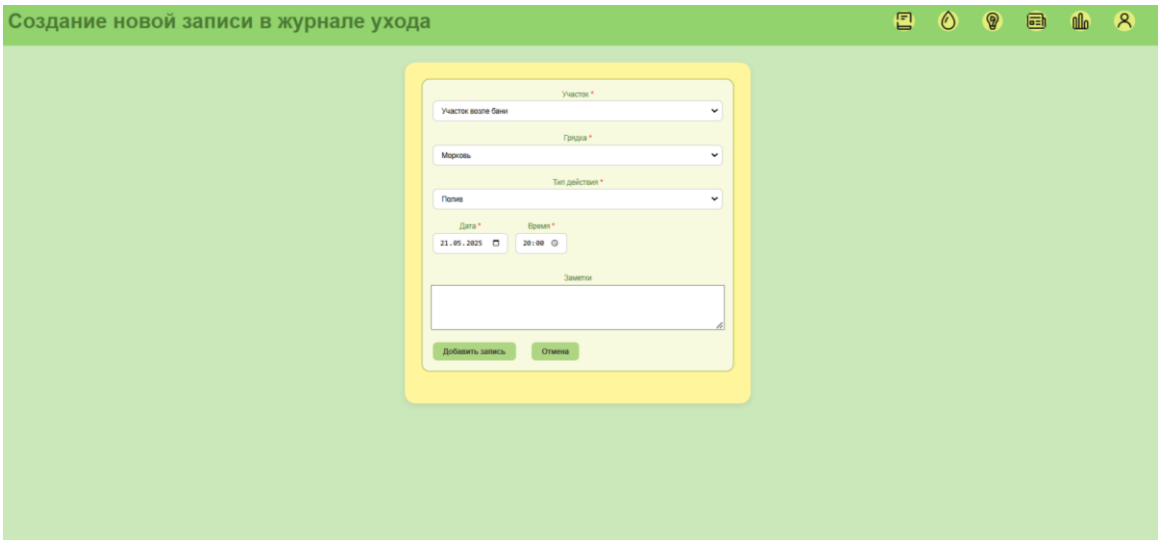

Рисунок 17 – Создание записи об уходе

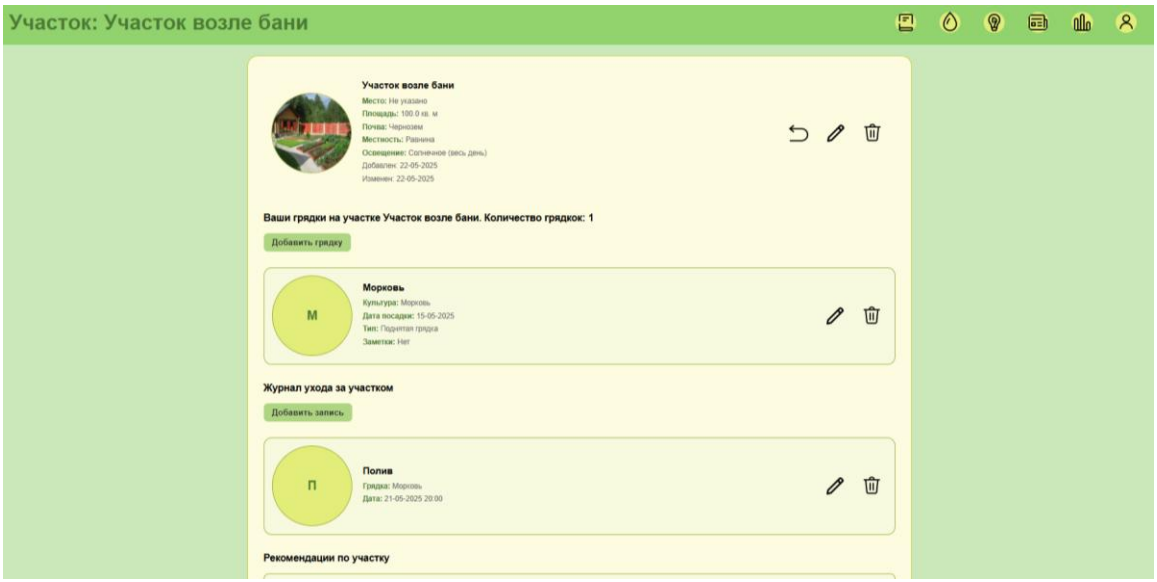

Рисунок 18 – Участок

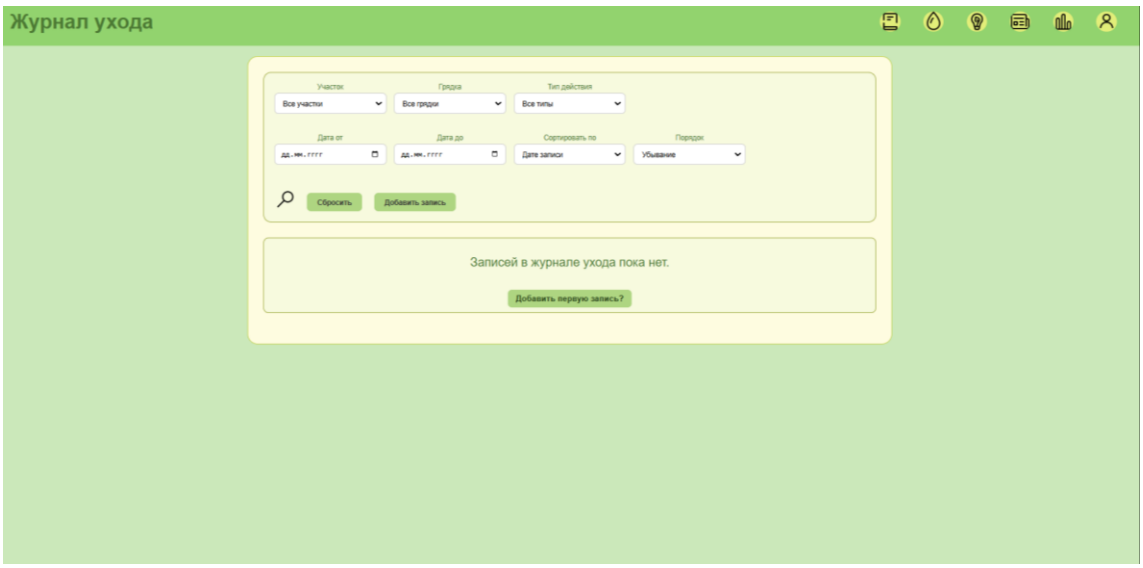

Рисунок 19 – Пустой журнал ухода

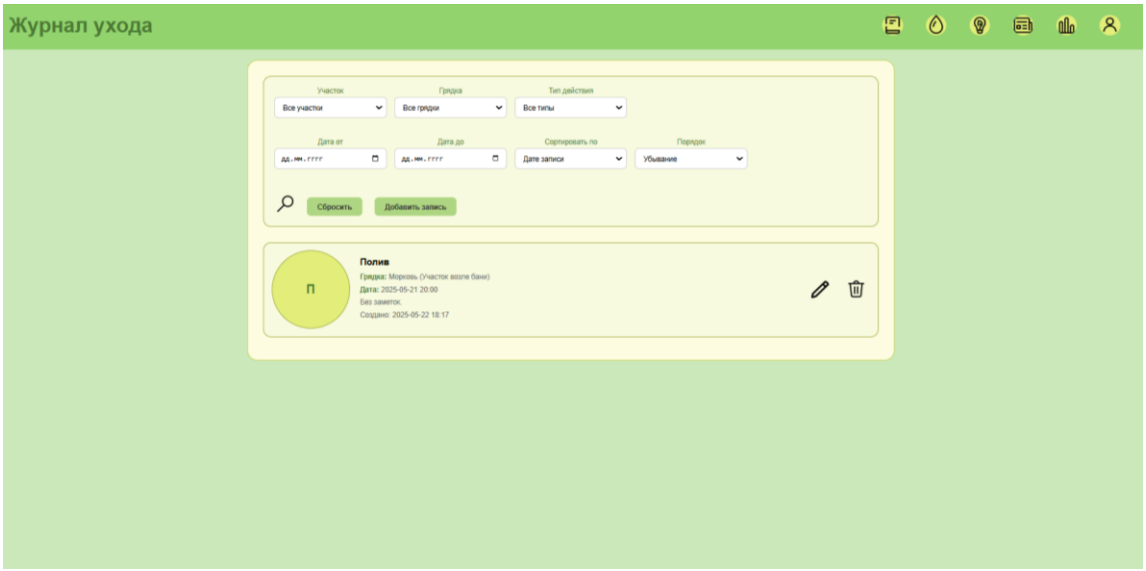

Рисунок 20 – Журнал ухода

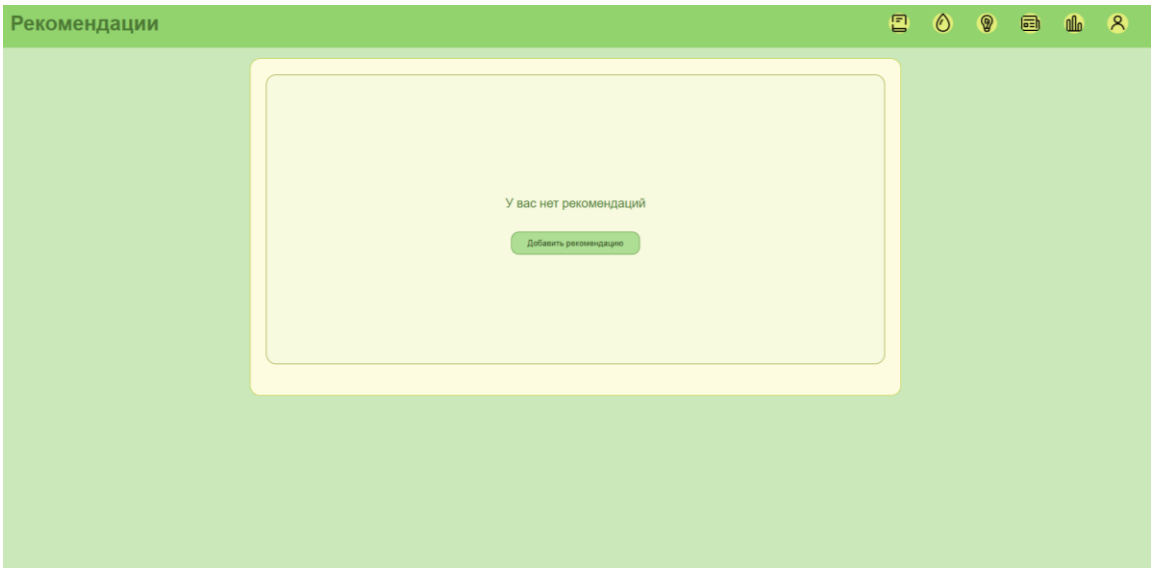

Рисунок 21 – Пустой раздел рекомендаций

ВЫВОД

В результате разработки веб-приложения «Огородник» реализован полноценный сервис для ведения личного дневника садовода с возможностью управления участками, грядками и действиями по уходу за растениями. Система поддерживает регистрацию и авторизацию пользователей, работу с публичными и приватными записями, публикацию новостей, комментирование и получение рекомендаций. Архитектура построена на технологиях: Python, Flask, MongoDB, HTML, CSS, JS, что обеспечивает гибкость, масштабируемость и удобство расширения функциональности. Реализованы функции импорта и экспорта данных, что повышает удобство использования и надежность хранения пользовательской информации.

Несмотря на реализованный функционал, имеются недоработки: отсутствие вкладки новостей и статистики, отсутствие адаптации под мобильные устройства, текущий пользовательский интерфейс реализован на базовом уровне, что ограничивает удобство работы с приложением. Возможным направлением улучшения является добавление системы уведомлений, более развитой фильтрации и статистики.

В перспективе возможно интегрировать ИИ-модули для предсказания оптимального времени ухода или диагностики заболеваний растений по фото, добавить мобильное приложение, реализовать совместный доступ к участкам и грядкам. Также перспективным направлением является подключение внешних API (например, прогноз погоды) для динамического формирования рекомендаций.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ссылка на репозиторий: https://github.com/moevm/nosql1h25-garden
2. Современный учебник JavaScript: https://learn.javascript.ru/
3. Документация для разработчиков CSS: https://doka-guide.vercel.app/css/
4. Документация для разработчиков HTML: https://doka-guide.vercel.app/html/
5. Руководство Flask: https://flask.palletsprojects.com/en/stable/
6. Руководство MongoDB: https://metanit.com/nosql/mongodb/

ПРИЛОЖЕНИЕ А

ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

Клонировать репозиторий:

```
git clone https://github.com/moevm/nosql1h25-garden.git
cd nosql1h25-garden/
```

Собрать и запустить проект:

```
docker-compose up --build
```

Проект будет доступен по адресу: `localhost:5000/`

ПРИЛОЖЕНИЕ Б
ИНСТРУКЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЯ

1. Регистрация и вход
   - Перейдите на главную страницу приложения по адресу: `localhost:5000/`.
   - Нажмите кнопку «Регистрация», введите ваше имя, email и пароль. После подтверждения вы автоматически попадёте в личный кабинет. Если у вас уже есть аккаунт – используйте форму входа, введя email и пароль.
2. Профиль пользователя. Вы можете изменить имя и загрузить аватарку.
3. Работа с участками в разделе «Мои участки», вы можете:
   - Добавить новый участок, указав его название и характеристики.
   - Редактировать существующие участки.
   - Удалить участок.
   - Искать участки, используя фильтры.
4. Для работы с грядками нажмите на нужный участок, чтобы перейти к списку грядок, вы можете:
   - Добавить новую грядку.
   - Редактировать данные грядки.
   - Удалить грядку.
   - Найти нужную грядку по фильтрам.
5. Уход за растениями в разделе «Уход за грядками», вы можете:
   - Добавить запись об уходе.
   - Удалить запись.
   - Найти нужную запись по фильтрам.
6. Раздел «Личный дневник» позволяет:
   - Создавать записи (посты) с заголовком, текстом, фотографиями и указанием приватности.
   - Просматривать и фильтровать записи.
   - Изменять или удалять посты.

7. Во вкладке «Новости» отображаются публичные записи других пользователей. Вы можете:

- Просматривать посты, ставить «лайки» и оставлять комментарии.
- Удалять собственные комментарии.
- Искать записи, используя фильтры.

8. Раздел «Рекомендации» показывает, какие действия по уходу следует выполнить.

9. Импорт и экспорт данных. Для резервного копирования данных нажмите «Экспортировать данные» – будет скачан json-файл с вашими данными. Для восстановления данных выберите «Импортировать данные» и загрузите ранее сохранённый json-файл. Система проверит структуру и добавит данные в ваш аккаунт.