МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

# Разработка десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам

КУРСОВАЯ РАБОТА
по дисциплине «Программная инженерия»
ЮУрГУ – 09.03.04.2023.308-348.КР

Нормоконтролер, д.ф.-м.н., доцент, профессор каф. СП
_____ Т.А. Макаровских
«___»_____ 2023 г.

Научный руководитель:
д.ф.-м.н., доцент, профессор каф. СП
_____ Т.А. Макаровских

Автор работы:
студент группы КЭ-303
_____ Е.В. Ращупкин

Работа защищена
с оценкой: _____
«___»_____ 2023 г.

Челябинск, 2023 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
**«Южно-Уральский государственный университет (национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

УТВЕРЖДАЮ
Зав. кафедрой СП

_____
Л.Б. Соколинский
06.02.2023 г.

**ЗАДАНИЕ**
**на выполнение курсовой работы**
по дисциплине «Программная инженерия»
студенту группы КЭ-303
Ращупкину Евгению Владимировичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

**1. Тема работы**

Разработка десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам.

**2. Срок сдачи студентом законченной работы:** 29.05.2023 г.

**3. Исходные данные к работе**

3.1. The Rust Programming Language [Электронный ресурс] URL: https://doc.rust-lang.org/stable/book/

3.2. Tauri guides [Электронный ресурс] URL: https://tauri.app/v1/guides/

3.3. «Геоскан» - беспилотные технологии [Электронный ресурс] URL: https://www.geoscan.aero/ru/products/geoscan401

**4. Перечень подлежащих разработке вопросов[3]**

4.1. Изучить методы разделения сельскохозяйственного поля на сетку ячеек, расчета маршрута следования дрона по центрам этих ячеек. Методы учета характеристик дрона вносящих корректировки в маршрут;

4.2. Привести описание требований к разрабатываемому продукту на основе диаграмм вариантов использования UML. ;

4.3. Спроектировать структуру приложения и разработать необходимые модули для ее функционирования, связать модули для работы с графическим интерфейсом программы;
4.4. Протестировать возможности системы подав на вход реальные данные, координаты поля и характеристики дрона, сравнить результаты с ожидаемыми.
**5. Дата выдачи задания:** 06.02.2023 г.

**Научный руководитель,**
д.ф.-м.н., доцент, профессор каф. СП Т.А. Макаровских

**Задание принял к исполнению** Е.В. Ращупкин

**ГЛОССАРИЙ**

1. UAV (Unmanned Aerial Vehicle) – Беспилотный летательный аппарат, управляемый дистанционно или автономно посредством встроенных систем управления.

2. Валидация – процесс проверки данных на соответствие заданным критериям. В контексте программирования, валидация обычно включает проверку корректности и полноты введенных пользователем данных.

3. Фронтенд – часть программной системы, которая взаимодействует с пользователем. Включает в себя интерфейс пользователя и логику, обеспечивающую его функционирование.

4. Бэкенд – часть программной системы, которая выполняет основную обработку данных и не взаимодействует напрямую с пользователем. Включает в себя серверные компоненты, базы данных и другие системы.

5. Десктоп – термин, обычно используемый для описания приложений, разработанных для работы на персональных компьютерах или рабочих станциях.

6. Кроссплатформенность – свойство программного продукта, которое позволяет ему работать на разных операционных системах или устройствах без необходимости внесения изменений в исходный код.

7. Фреймворк – набор библиотек и инструментов, которые упрощают разработку программного обеспечения, предоставляя структуру и набор стандартных функций.

8. Методы – в программировании, это определенные блоки кода, которые выполняют конкретные задачи. Методы используются для организации кода и повторного использования функций в различных частях программы.

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

## Актуальность

Актуальность данной курсовой работы определяется растущим интересом к применению беспилотных летательных аппаратов (БПЛА) в сельском хозяйстве. В частности, дроны используются для мониторинга урожая, а также для создания точных карт полей.

Научно-технологический прорыв в сельскохозяйственном производстве невозможен без применения цифровых технологий точного земледелия (ТЗ), являющегося ключевым сегментом «умного сельского хозяйства». Направление точного земледелия – многопрофильное и характеризуется комплексностью и сложностью научных, инженерных, агрономических и организационных задач. Существующие методы и инструменты точного сельского хозяйства позволили передовым странам довольно быстро перевести сельское хозяйство на инновационный путь развития [1]. Одной их технологий, используемых для мониторинга состояния сельскохозяйственных угодий, является аэрофотосъемка при помощи БПЛА. Организация такой съемки зачастую осуществляется специализированными организациями и требует определенных вложений от заказчика. В частности, необходимо оплатить вызов бригады для осуществления съемки, а также амортизацию используемого оборудования (например, каждую перезарядку аккумуляторной батареи БПЛА).

С учетом этого, разработка десктопного приложения, способного оптимизировать маршруты полетов дронов на основе их характеристик, представляется актуальной и востребованной задачей. Данное приложение может помочь сельскохозяйственным предприятиям повысить эффективность использования БПЛА, уменьшить затраты на выезд специалистов за счет сокращения времени полета.

**Постановка задачи**

Целью курсовой работы является разработка десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам.

Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить анализ предметной области и произвести обзор существующих решений;
2) разработать базовую архитектуру приложения;
3) выполнить реализацию приложения;
4) выполнить тестирование.

**Структура и содержание работы**

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 52 страницы, объем списка литературы – 12 источников.

В первой главе проводится анализ предметной области и существующих работ, связанных с планированием маршрутов для дронов, что позволяет лучше понять контекст и потребности конечных пользователей.

Вторая глава посвящена анализу требований к программной системе. В ней определяются основные функциональные и нефункциональные требования, а также формируется диаграмма вариантов использования и спецификация основных прецедентов.

В третьей главе описана архитектура системы. Приведено общее описание архитектуры, описание компонентов и сервисов, составляющих систему, а также модель базы данных и процесс работы с системой.

В четвертой главе приведена реализация ключевых компонентов системы и пользовательского интерфейса, а также проводится тестирование системы для убедиться в ее корректной работе и соответствии заявленным требованиям.

В приложении А содержатся функции расчета сервиса Algorithms.

В приложении Б содержатся рисунки интерфейса приложения.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РАБОТ ПО ТЕМАТИКЕ КУРСОВОГО ПРОЕКТА

## 1.1. Предметная область проекта

Предметная область проекта связана с разработкой десктопного приложения, предназначенного для расчета маршрута полета сельскохозяйственного дрона. Основной задачей приложения является определение оптимального маршрута полета, учитывающего имеющиеся характеристики дрона и заданные параметры полета.

## 1.2. Анализ аналогичных проектов и существующих решений для реализации проекта

В рамках разработки десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам, необходимо реализовать возможность планировки маршрута для фотографирования посевов полей. Эти данные, после планировки маршрута, уже передаются в другое приложение для склейки фотографий поверхности и, при необходимости, анализа. Важным аспектом при планировке маршрута будет учет размеров поля и наличие препятствий на его территории. Кроме того, приложение должно иметь возможность визуализировать маршрут на карте и предоставлять информацию о параметрах полета, таких как высота, длительность полета и т.д. Все эти функции должны быть реализованы в удобном и интуитивно понятном интерфейсе, который позволит пользователям быстро и эффективно планировать маршруты для фотографирования посевов полей.

Существует множество инструментов для планирования полетов дронов и обработки полученных данных, которые могут быть полезны при разработке десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам. Рассмотрим некоторые из них.

DroneDeploy [2] – это один из наиболее распространенных инструментов для планирования полетов дронов и обработки данных. С помощью этого приложения пользователь может задавать параметры полета, такие как высоту, скорость, угол наклона камеры и другие, а также задавать целевые точки на карте, которые дрон должен посетить. Приложение автоматически расчитывает оптимальный маршрут полета и предоставляет возможность обработки полученных данных, например, создание карт и 3D-моделей местности.

Litchi [3] – еще один инструмент для планирования полетов дронов. Он позволяет пользователю задавать различные параметры полета, а также создавать специальные миссии, включающие несколько точек на карте, которые дрон должен посетить в определенном порядке. Litchi также предоставляет возможность записи видео и фото во время полета, а также позволяет управлять камерой дрона в режиме реального времени.

Pix4D Capture [4] – специализированный инструмент для создания карт и 3D-моделей с помощью дронов. Приложение позволяет задавать различные параметры полета, такие как высоту, скорость и угол наклона камеры, а также задавать целевые точки на карте. После полета Pix4D Capture обрабатывает полученные данные и создает точные 3D-модели местности.

UgCS [5] – это инструмент, который позволяет пользователю управлять не только дронами, но и другими беспилотными системами. Пользователь может задавать различные параметры полета, такие как скорость, высота полета и угол наклона камеры, а также задавать целевые точки на карте. Приложение автоматически расчитывает оптимальный маршрут полета и позволяет обрабатывать полученные данные.

Приведем таблицу с сравнением возможностей представленных продуктов (таблица 1).

Таблица 1 – Сравнительный анализ приложений

| Возможность | DroneDeploy | Litchi | Pix4D Capture | UgCS |
|---|---|---|---|---|
| Планирование маршрута полета | Да | Да | Да | Да |
| Управление полетом дрона | Да | Да | Да | Да |
| Обработка полученных данных | Да | Да | Да | Да |
| Визуализация карты | Да | Да | Да | Да |
| Ограниченный ряд поддерживаемых дронов | Да | Да | Да | Да |
| Добавление собственных дронов | Ограничение функционала | Ограничение функционала | Нет | Ограничение функционала |
| Поддержка ОС | iOS, Android + Windows, macOS, Linux | iOS, Android + Windows, macOS, Linux | iOS, Android | Windows, macOS, Linux, Android |
| Лицензия | Проприетарная | Проприетарная | Проприетарная | Проприетарная |
| Стоимость | $149+/месяц | $25 | Бесплатно | €790+ или €149+/месяц |

Рассмотрев аналогичные проекты и существующие решения для реализации проекта, можно заметить, что многие из них имеют проприетарную лицензию и не предоставляют возможность добавления собственных дронов. Кроме того, некоторые из них могут быть слишком дорогими для малого бизнеса, к тому же, стоит отметить, что все анализируемые аналоги приложения разработаны зарубежными компаниями. В свою очередь, наше приложение будет предоставлять открытый исходный код, что позволит пользователям настраивать и дополнять его функциональность. В приложении будет возможность добавления собственных дронов, что поможет людям с моделями дронов от малоизвестных компаний. Кроме того, распространение приложения будет бесплатным, что обеспечит его доступность и удобство использования для малых предприятий в сельском хозяйстве.

В целом, все эти решения могут быть полезны при разработке десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам. Также необходимо обеспечить возможность интеграции с другими системами, например, с системами отправки маршрута на дрон или экспорта в уже существующие форматы представления координат.

### 1.3. Особенности реализации

Для реализации десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам, было принято решение использовать веб технологии. Такой подход позволяет обеспечить кроссплатформенность приложения и увеличить его доступность для пользователей.

Для реализации приложения будет использоваться Tauri [6]. Он выбран из-за своей высокой производительности и возможности создания кроссплатформенных десктопных приложений на основе веб-технологий. Tauri, обеспечивает быстрое и эффективное взаимодействие между ядром приложения и веб-интерфейсом.

Для обработки данных будет использоваться язык программирования Rust [7]. Rust позволяет обеспечить высокую производительность и безопасность при обработке данных, что критически важно для такого типа приложений. Более того, Rust имеет большую и быстрорастущую экосистему, что обеспечивает доступность и готовность библиотек и инструментов, необходимых для реализации конкретных функциональных требований приложения.

В качестве фреймворка для разработки пользовательского интерфейса будет использоваться SvelteKit [8]. Этот инструмент позволяет создавать эффективные и быстрые веб-приложения с помощью компиляции кода во время сборки и минимизации размера бандла приложения.

Для визуализации карты будет использована OpenLayers [9] – библиотеку для работы с картами. Она предоставляет широкий спектр функциональности для работы с картами, таких как отображение маршрутов, меток, векторных слоев и т.д. Это позволит создать интерактивную карту, на которой можно будет легко планировать маршруты для фотографирования посевов полей.

Таким образом, использование Tauri, Rust, SvelteKit и OpenLayers обеспечивает высокую производительность, безопасность, готовность библиотек и инструментов, а также удобную и эффективную визуализацию карты. Все эти особенности сделают приложение более доступным для пользователей, а также обеспечат надежность в работе.

### 1.4. Заключение

В результате анализа предметной области и существующих работ по тематике курсового проекта был выявлен набор инструментальных средств для реализации поставленной задачи, наиболее полно удовлетворяющий требованиям к подобного рода системам. Было принято решение реализовать систему на основе на базе технологий Tauri, Rust, SvelteKit и OpenLayers.

## 2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ

В результате анализа предметной области и обзора существующих аналогов были сформированы следующие 2 два основных типа требований:

– функциональные требования – перечень сервисов, которые должна выполнять система;

– нефункциональные требования – описание характеристики системы и ее окружения, также содержит перечень ограничений.

### 2.1. Функциональные требования к проектируемой системе

– пользователь должен иметь возможность добавлять, просматривать, изменять и удалять записи о дронах и камерах в базе данных. Через пользовательский интерфейс системы;

– система должна иметь возможность строить маршрут на основе заданных характеристик дрона, камеры, координат старта съемки, координат границ, указываемых на карте и процента перекрытия полученных снимков;

– система должна иметь возможность визуализировать маршрут на карте.

### 2.2. Нефункциональные требования к проектируемой системе

1) система должна использовать язык программирования Rust;
2) система должна проверять корректность вводимых данных:

– параметры дронов (Величина полезной нагрузки, продолжительность полета, скорость полета, минимальная максимальная высоты полета), параметры камеры (Масса, угол обзора, разрешение съемки) принимают неотрицательные значения;

– параметр дронов, величина полевной нагрузки лежит в пределах: от 100 грамм до 10 кг. Продолжительность полета находится в пределах от 1 минуты до 8 часов. Скорость полета находится в пределах от 0.1 м/с до 50 м/с. Минимальная и максимальная высота полета находятся в пределах от 2 метров до 5 км. При этом, максимальная высота полета больше минимальной;

– параметр камеры, масса должна быть меньше 10 килограмм. Углы обзора по x и y должны быть меньше 180 градусов. Разрешение съемки по x и y – целочисленные;

3) система должна использовать веб технологии такие как Tauri, Svelte, TypeScript для визуализации интерфейса и библиотеку OpenLayers для отображения карты и маршрута.

### 2.3. Диаграмма вариантов использования

На основе требований, предъявляемых к разрабатываемому приложению, были разработаны варианты его использования, которые представлены на рисунке 1.

В ходе проектирования были выделены следующие актеры:

– пользователь – пользователь приложения, ему доступна возможность использовать весь функционал приложения.

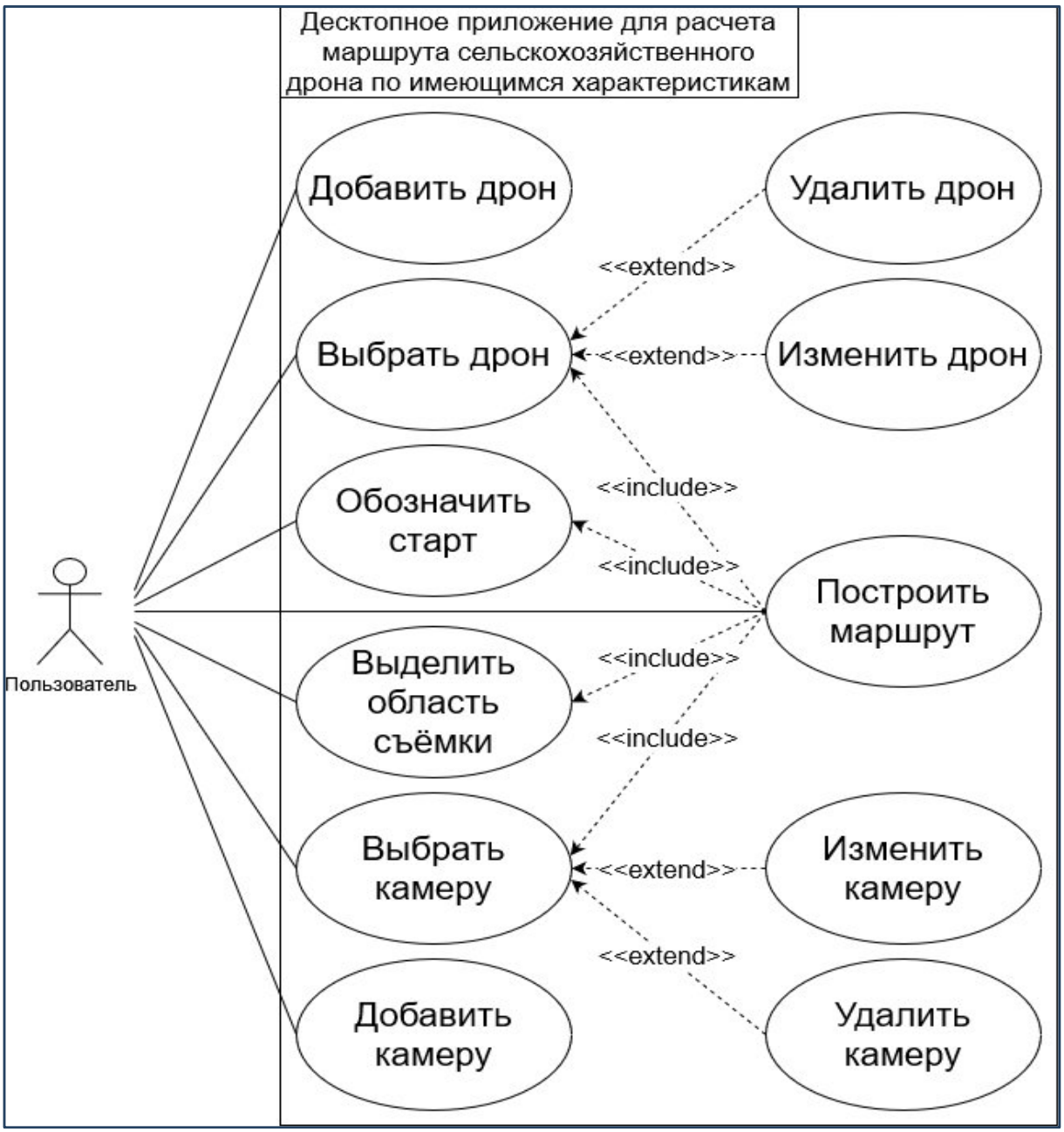


Рисунок 1 – Диаграмма вариантов использования

Пользователь может совершать следующие действия:

– добавить дрон – пользователь может добавить новый дрон с его характеристиками;

– выбрать дрон – пользователь может выбрать один из добавленных дронов для расчета маршрута или редактирования;

– удалить дрон – пользователь может удалить выбранный дрон;

– изменить дрон – пользователь может изменить характеристики выбранного дрона;

– обозначить старт – пользователь может задать стартовую точку маршрута;

– выделить область съемки – пользователь может указать область, которую требуется снять с помощью дрона;

– выбрать камеру – пользователь может выбрать камеру, которая будет установлена на дроне для расчетов или редактирования;

– изменить камеру – пользователь может изменить характеристики выбранной камеры;

– удалить камеру – пользователь может удалить выбранную камеру;

– добавить камеру – пользователь может добавить новую камеру с ее характеристиками;

– построить маршрут – пользователь может рассчитать маршрут дрона на основе введенных данных, включая выбранный дрон, стартовую точку, область съемки и камеру.

## 2.4. Спецификация основных прецедентов

Спецификация прецедентов представлена в таблицах

Таблица 2 – Спецификация прецедента «Добавить дрон»

| Прецедент: | Добавить дрон |
|---|---|
| *ID:* | ID1 |
| *Аннотация:* | Пользователь добавляет новый дрон с его характеристиками в систему. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | Пользователь открыл меню с изменением дрона |
| *Основной поток:* | 1. Пользователь вводит характеристики дрона<br>2. Пользователь нажимает кнопку добавления Дрона.<br>3. Система валидирует введенные данные.<br>4. Система сохраняет дрон в базе данных. |
| *Постусловия:* | Новый дрон добавлен в систему и доступен для выбора. |

Окончание таблицы 2

| | |
|---|---|
| *Альтернативные потоки:* | 4А. Ошибка валидации данных:<br>4А.1. Система отображает сообщение об ошибке и предлагает пользователю исправить введенные данные.<br>4А.2. Возврат к шагу 3 основного потока.<br>5А. Ошибка сохранения в базе данных:<br>5А.1. Система отображает сообщение об ошибке сохранения в базе<br>5А.2. Возврат к шагу 1 основного потока. |

Таблица 3 – Спецификация прецедента «Выбрать дрон»

| | |
|---|---|
| Прецедент: | Выбрать дрон |
| *ID:* | ID2 |
| *Аннотация:* | Пользователь выбирает дрон для расчета маршрута. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | В системе есть хотя бы один сохраненный дрон. |
| *Основной поток:* | 1. Пользователь нажимает на listbox с доступными дронами.<br>2. Система отображает список доступных дронов.<br>3. Пользователь выбирает дрон из списка.<br>4. Система устанавливает выбранный дрон как текущий. |
| *Постусловия:* | Дрон выбран и установлен как текущий для дальнейших операций. |
| *Альтернативные потоки:* | 3А. Пользователь не выбрал дрон:<br>3А.1. Возврат к шагу 1 основного потока. |

Таблица 4 – Спецификация прецедента «Удалить дрон»

| | |
|---|---|
| Прецедент: | Удалить дрон |
| *ID:* | ID2.1 |
| *Аннотация:* | Пользователь удаляет выбранный дрон из системы. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | Пользователь выбрал дрон (прецедент ID2). |
| *Основной поток:* | 1. Пользователь выбирает опцию «Удалить дрон».<br>2. Система удаляет дрон из базы данных. |
| *Постусловия:* | Дрон удален из системы. |
| *Альтернативные потоки:* | 2А. Ошибка удаления из базы данных:<br>2А.1. Система отображает сообщение об ошибке удаления.<br>2А.2. Возврат к шагу 1 основного потока. |

Таблица 5 – Спецификация прецедента «Изменить дрон»

| Прецедент: | Изменить дрон |
|---|---|
| *ID:* | ID2.2 |
| *Аннотация:* | Пользователь изменяет характеристики выбранного дрона. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | Пользователь выбрал дрон (прецедент ID2). |
| *Основной поток:* | 1. Пользователь изменяет характеристики выбранного дрона<br>2. Пользователь выбирает опцию по изменению дрона.<br>3. Система валидирует измененные данные.<br>4. Система сохраняет изменения в базе данных. |
| *Постусловия:* | Характеристики дрона успешно изменены. |
| *Альтернативные потоки:* | 2А. Пользователь отменяет изменения:<br>2А.1. Пользователь нажимает на кнопку отмены.<br>2А.2. Возврат к шагу 1 основного потока.<br>4А. Ошибка изменения в базе данных:<br>4А.1. Система отображает сообщение об ошибке.<br>4А.2. Возврат к шагу 1 основного потока. |

Таблица 6 – Спецификация прецедента «Обозначить старт»

| Прецедент: | Обозначить старт |
|---|---|
| *ID:* | ID3 |
| *Аннотация:* | Пользователь задает стартовую точку маршрута дрона. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | Пользователь находится в меню карты |
| *Основной поток:* | 1. Пользователь выбирает опцию «Обозначить старт».<br>2. Пользователь устанавливает маркер стартовой точки на карте.<br>3. Система сохраняет координаты стартовой точки. |
| *Постусловия:* | Стартовая точка маршрута успешно задана и сохранена в системе. |
| *Альтернативные потоки:* | 2А. Пользователь изменяет инструмент на карте:<br>2А.2. Возврат к шагу 1 основного потока. |

Таблица 7 – Спецификация прецедента «Выделить область съемки»

| Прецедент: | Выделить область съемки |
|---|---|
| *ID:* | ID4 |
| *Аннотация:* | Пользователь указывает область, которую требуется снять с помощью дрона. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | Пользователь находится в меню карты |

Окончание таблицы 7

| | |
|---|---|
| *Основной поток:* | 1. Пользователь выбирает опцию «Выделить область съемки».<br>2. Пользователь выделяет область на карте, которую требуется снять с помощью дрона.<br>3. Система сохраняет координаты стартовой точки. |
| *Постусловия:* | Область съемки успешно задана и сохранена в системе. |
| *Альтернативные потоки:* | 2А. Пользователь изменяет инструмент на карте:<br>2А.2. Возврат к шагу 1 основного потока. |

Таблица 8 – Спецификация прецедента «Выбрать камеру»

| | |
|---|---|
| Прецедент: | Выбрать камеру |
| *ID:* | ID5 |
| *Аннотация:* | Пользователь выбирает камеру для использования вместе с дроном. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | В системе есть хотя бы одна сохраненная камера. |
| *Основной поток:* | 1. Пользователь нажимает на listbox с доступными камерами.<br>2. Система отображает список доступных камер.<br>3. Пользователь выбирает дрон из списка.<br>4. Система устанавливает выбранную камеру, как текущую. |
| *Постусловия:* | Камера выбрана и установлена как текущая для дальнейших операций. |
| *Альтернативные потоки:* | 3А. Пользователь не выбрал камеру:<br>3А.1. Возврат к шагу 1 основного потока. |

Таблица 9 – Спецификация прецедента «Удалить камеру»

| | |
|---|---|
| Прецедент: | Удалить камеру |
| *ID:* | ID5.1 |
| *Аннотация:* | Пользователь удаляет выбранную камеру из системы. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | Пользователь выбрал камеру (прецедент ID5). |
| *Основной поток:* | 1. Пользователь выбирает опцию «Удалить камеру».<br>2. Система удаляет камеру из базы данных. |
| *Постусловия:* | *Камера* удалена из системы. |
| *Альтернативные потоки:* | 2А. Ошибка удаления из базы данных:<br>2А.1. Система отображает сообщение об ошибке удаления.<br>2А.2. Возврат к шагу 1 основного потока. |

Таблица 10 – Спецификация прецедента «Изменить камеру»

| Прецедент: | Изменить камеру |
|---|---|
| *ID:* | ID5.2 |
| *Аннотация:* | Пользователь изменяет характеристики выбранной камеры. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | Пользователь выбрал камеру (прецедент ID5). |
| *Основной поток:* | 1. Пользователь изменяет характеристики выбранной камеры.<br>2. Пользователь выбирает опцию по изменению камеры.<br>3. Система валидирует измененные данные.<br>4. Система сохраняет изменения в базе данных. |
| *Постусловия:* | Характеристики камеры успешно изменены. |
| *Альтернативные потоки:* | 2А. Пользователь отменяет изменения:<br>2А.1. Пользователь нажимает на кнопку отмены.<br>2А.2. Возврат к шагу 1 основного потока.<br>4А. Ошибка изменения в базе данных:<br>4А.1. Система отображает сообщение об ошибке.<br>4А.2. Возврат к шагу 1 основного потока. |

Таблица 11 Спецификация прецедента «Добавить камеру»

| Прецедент: | Добавить камеру |
|---|---|
| *ID:* | ID6 |
| *Аннотация:* | Пользователь добавляет новую камеру с ее характеристиками в систему. |
| *Главные актеры:* | Пользователь |
| *Второстепенные актеры:* | - |
| *Предусловия:* | Пользователь открыл меню с изменением камеры |
| *Основной поток:* | 1. Пользователь вводит характеристики камеры<br>2. Пользователь нажимает кнопку добавления камеры.<br>3. Система валидирует введенные данные.<br>4. Система сохраняет камеру в базе данных. |
| *Постусловия:* | Новая камера добавлена в систему и доступна для выбора. |
| *Альтернативные потоки:* | 4А. Ошибка валидации данных:<br>4А.1. Система отображает сообщение об ошибке и предлагает пользователю исправить введенные данные.<br>4А.2. Возврат к шагу 1 основного потока.<br>5А. Ошибка сохранения в базе данных:<br>5А.1. Система отображает сообщение об ошибке.<br>5А.2. Возврат к шагу 2 основного потока. |

# 3. АРХИТЕКТУРА СИСТЕМЫ

## 3.1. Общее описание архитектуры системы

На рисунке 2 изображена диаграмма компонентов системы для планирования маршрутов сельскохозяйственных дронов. Система состоит из следующих сервисов: сервис расчета маршрутов (Algorithms), сервис управления данными (Data Managment), а также из компонентов пользовательского интерфейса Map, в состав которого входит сервис Calculate, также пользовательский интерфейс составляют LMenu, RMenu. Система включает в себя компонента базы данных, который содержит данные о дронах и камерах. Сервис расчета маршрутов использует данные, предоставленные пользователем через компоненты интерфейса. Сервис управления данными осуществляет взаимодействие с базой данных для создания, чтения, обновления и удаления данных о дронах и камерах. Компоненты пользовательского интерфейса взаимодействуют с сервисами для отправки данных, получения результатов расчетов и отображения информации пользователю.

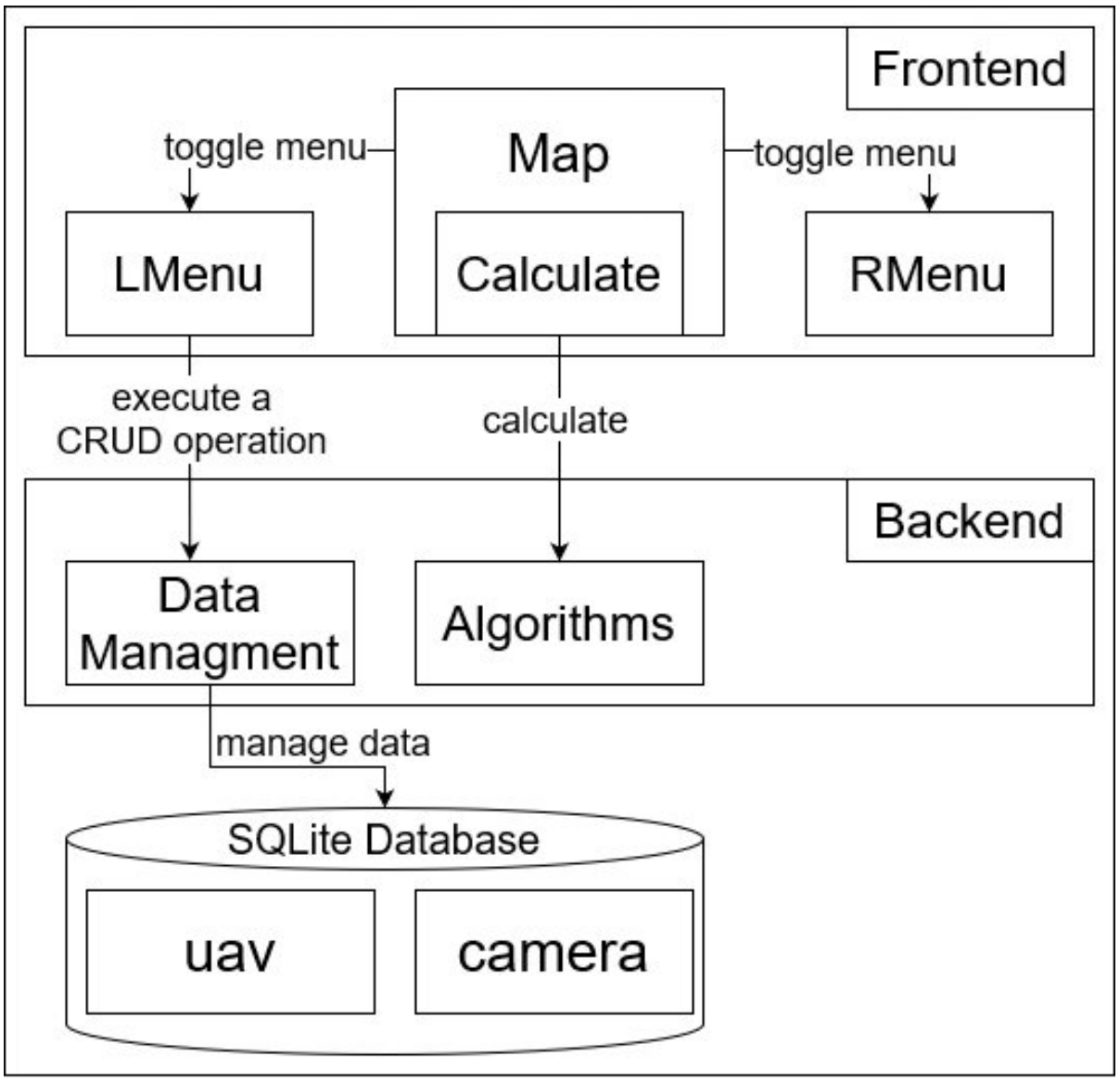


Рисунок 2 – Компоненты системы

### 3.2. Описание компонентов и сервисов, составляющих систему

Map – центральное меню с картой, компонент пользовательского интерфейса в его функционал входят:

- отображение карты и маршрута;
- панель инструментов для рисования маршрута;
- переключение отображения правого и левого меню;
- кнопка вызова Calculate.

Calculate – часть сервиса Map он включает в себя:

- валидацию данных, он проверяет корректность ввода всех необходимых данных;

– отправку данных на бэкенд для расчетов;
– обновление отображаемого маршрута на карте.

LMenu – Этот компонент позволяет пользователю работать с дронами и камерами, он предлагает следующий набор функций:

– выбор дрона для построения маршрута;
– CRUD операции над дроном;
– выбор камеры для построения маршрута;
– CRUD операции над камерой.

RMenu – Этот компонент позволяет пользователю управлять параметрами миссии, он предоставляет пользователю данный функционал:

– ввод высоты полета;
– выбор алгоритма расчета;
– отображение текущих параметров миссии;
– экспорт маршрута в формат GeoJSON.

Data Managment – Этот сервис взаимодействует с базой данных SQLite, управляя данными о дронах и камерах. Он выполняет операции CRUD на основе команд с фронтенда.

Algorithms – это сервис расчета маршрута, он отвечает за расчет маршрута дрона. Он обладает следующим набором возможностей:

– дискретизация области на основе полученных данных;
– алгоритм ближайшего соседа;
– алгоритм полного перебора;
– отправка результатов на фронтенд.

SQLite Database – база данных, является частью инфраструктуры системы, в которой находятся 2 таблицы uav и camera для хранения данных о дронах и камерах, хранится локально в файле mydatabase.db.

### 3.3. Модель базы данных

На рисунке 3 представлена модель базы данных приложения для планирования маршрутов сельскохозяйственных дронов. Модель отличается своей простотой: она состоит всего из двух таблиц – «uav» Дроны и «camera» Камеры. Эти таблицы содержат информацию о различных дронах и камерах, которые могут использоваться в миссиях. Важно отметить, что таблицы не связаны друг с другом.

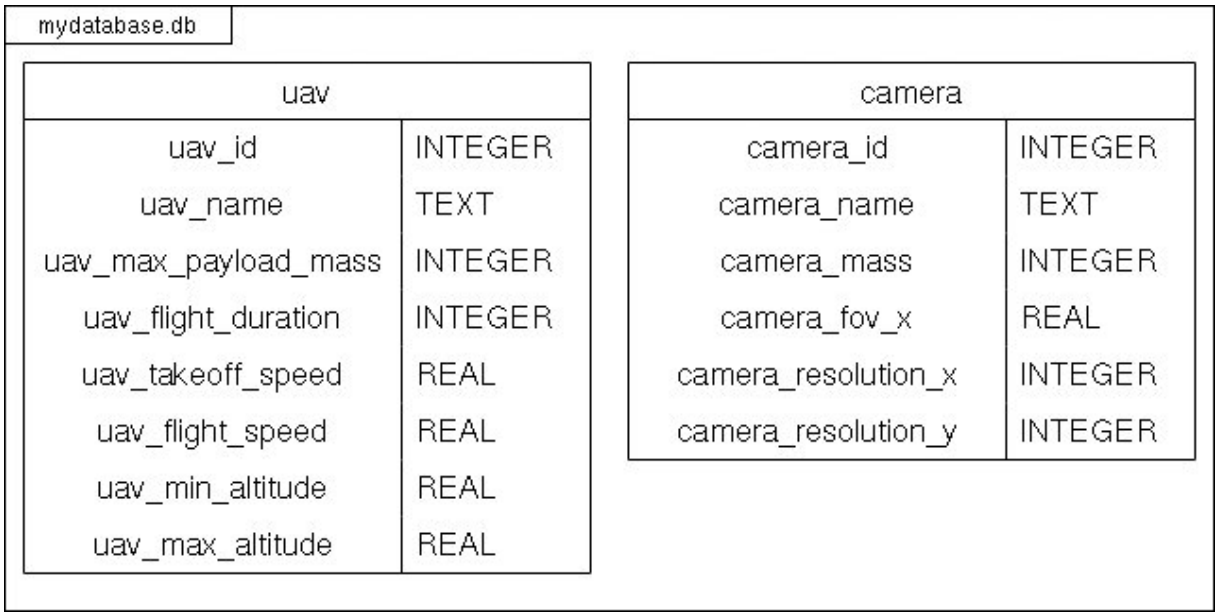


Рисунок 3 – Модель базы данных

Таблица uav – представляет собой экземпляр дрона и включает в себя следующие атрибуты:

- uav_id – идентификатор UAV;
- uav_name – название UAV;
- uav_max_payload_mass – максимальная полезная нагрузка дрона в граммах;
- uav_flight_duration – средняя продолжительность полета в секундах;
- uav_takeoff_speed – средняя скорость взлета в метрах в секунду;
- uav_flight_speed – средняя скорость полета в метрах в секунду;

– uav_min_altitude – минимальная безопасная высота полета в метрах;

– uav_max_altitude – максимальная безопасная высота полета в метрах.

Таблица camera – представляет собой экземпляр камеры и включает в себя следующие атрибуты:

– camera_id – идентификатор камеры;
– camera_name – название камеры;
– camera_mass – масса камеры в граммах;
– camera_fov_x – угол обзора камеры по оси x в градусах;
– camera_resolution_x – разрешение камеры по оси x в пикселях;
– camera_resolution_y – разрешение камеры по оси x в пикселях.

### 3.4. Процесс работы с системой

На рисунке 4 приведена диаграмма деятельности, которая подробно описывает процесс работы с системой планирования маршрутов для сельскохозяйственных дронов.

Процесс начинается одного из шести действий. Пользователь может установить точку запуска дрона, что является начальной точкой маршрута. Также, пользователь может обозначить область съемки, указывая на карте область, которую необходимо облететь. Пользователь может выбрать алгоритм, модель дрона и камеры из списка доступных в системе и установить процент перекрытия и определить высоту полета дрона. Здесь у пользователя есть два варианта: ввести высоту вручную или рассчитать ее. Во втором случае пользователю необходимо ввести разрешение съемки, после чего система сама рассчитывает оптимальную высоту полета.

После того как все параметры заданы, система производит расчет маршрута. Если все данные корректны, и валидация прошла успешно, система отображает маршрут на карте и предлагает пользователю экспортировать его в формате GeoJSON для дальнейшего использования.

GeoJSON [10] является открытым форматом данных, основанным на JSON, и используется для представления простых географических объектов вместе с их не пространственными атрибутами. GeoJSON файл, который экспортирует данное приложение включает в себя последовательность точек с координатами в формате WGS84, которые должен посетить дрон.

Если в процессе валидации обнаруживаются ошибки, система отображает сообщение об ошибке, давая пользователю возможность исправить введенные данные.

В то же время, независимо от основного процесса планирования маршрута, пользователь может выбрать и изменить характеристики модели дрона или камеры.

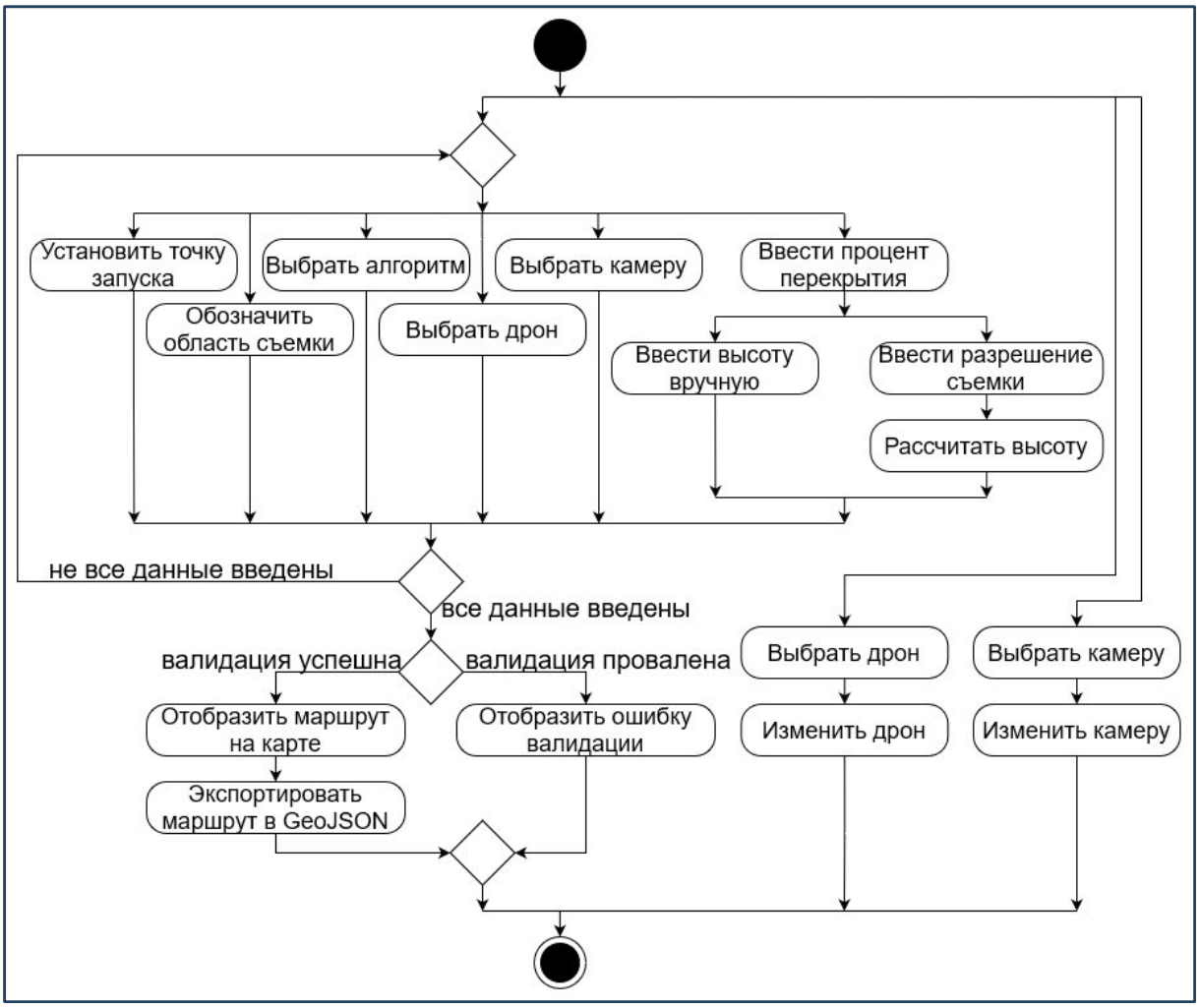


Рисунок 4 – Диаграмма деятельности

# 4. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ СИСТЕМЫ

## 4.1. Реализация компонентов системы

В реализации компонентов системы имеется сервис Data Management, который предоставляет функциональность для работы с информацией о дронах и камерах. Этот сервис состоит из следующих файлов:

uav/mod.rs – Основной файл, определяющий структуру Uav и методы для создания экземпляров дронов с заданными параметрами. Пример структуры Uav представлен в листинге 1.

Листинг 1 – Структура Uav

```
#[derive(Debug, Deserialize, Serialize)]
pub struct Uav {
    id: u64,                    // uav id
    pub name: String,           // uav name
    pub max_payload_mass: u64, // maximum payload in grams
    pub flight_duration: u64,  // average flight duration in seconds
    pub takeoff_speed: f64,    // average takeoff speed in meters per
second
    pub flight_speed: f64,     // average flight speed in meters per second
    pub min_altitude: f64,     // minimum safe flight altitude in meters
    pub max_altitude: f64,     // maximum safe flight altitude in meters
}
```

uav/uav_handle.rs – Файл содержит функции для работы с дронами через Tauri, такие как создание, обновление и удаление дронов, а также получение списка всех дронов. Пример функции для создания нового дрона представлен в листинге 2.

Листинг 2 – Функции для создания нового дрона

```
#[tauri::command]
pub fn new_uav(uav: Uav) -> String {
    let conn = Connection::open("mydatabase.db").expect("Cant open base");
    println!("Received new UAV: {:?}", uav);
    match uav_sql::insert(&uav, &conn) {
        Ok(_) => "Ok».to_string(),
        Err(e) => e.to_string(),
    }
}
```

uav/uav_sql.rs – Файл содержит функции для работы с базой данных, включая создание таблицы для хранения информации о дронах, а также вставка, обновление, удаление и получение списка всех дронов. Пример функции для создания таблицы в базе данных представлен в листинге 3.

Листинг 3 – Функции для создания таблицы uav

```
pub fn create_table(conn: &Connection) -> Result<usize> {
    conn.execute(
       "CREATE TABLE IF NOT EXISTS uav (
                uav_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
                uav_name TEXT NOT NULL,
                uav_max_payload_mass INTEGER NOT NULL CHECK
(uav_max_payload_mass >= 0),
                uav_flight_duration INTEGER NOT NULL CHECK
(uav_flight_duration >= 0),
                uav_takeoff_speed REAL NOT NULL CHECK (uav_takeoff_speed >=
0),
                uav_flight_speed REAL NOT NULL CHECK (uav_flight_speed >=
0),
                uav_min_altitude REAL DEFAULT 0 NOT NULL CHECK
(uav_min_altitude >= 0),
                uav_max_altitude REAL DEFAULT 0 NOT NULL CHECK
(uav_max_altitude >= 0)
                )",
        (), // empty list of parameters.
    )
}
```

camera/mod.rs – Основной модуль, который содержит структуру и методы для работы с камерой. Структура Camera содержит различные свойства камеры, включая ее идентификатор, имя, массу, угол обзора по оси X и разрешение. Здесь также определены метод для создания новых камер. Пример структуры Camera представлен в листинге 4.

Листинг 4 – Структура Camera

```
#[derive(Debug, Deserialize, Serialize)]
pub struct Camera {
    id: u64,               // id
    pub name: String,      // name
    pub mass: u64,         // mass in grams
    pub fov_x: f64,        // x-axis viewing angle in degrees
    pub resolution_x: u16, // camera resolution x
    pub resolution_y: u16, // camera resolution y
}
```

camera/camera_handle.rs – Этот модуль предоставляет функции для обработки запросов к базе данных. Здесь определены функции для создания новой камеры, обновления существующей камеры, удаления камеры и получения всех камер из базы данных. Каждая из этих функций открывает соединение с базой данных и вызывает соответствующую функцию из модуля camera_sql. Пример функции для обновления данных о камере представлен в листинге 5.

Листинг 5 – Функция для обновления данных о камере

```
#[tauri::command]
pub fn update_camera(camera: Camera) -> String {
 let conn = Connection::open("mydatabase.db").expect("Cant open base");
 println!("Received updated camera: {:?}", camera);
 match camera_sql::update(&camera, &conn) {
 Ok(_) =>"Ok».to_string(),
 Err(e) => e.to_string(),
 }
}
```

camera/camera_sql.rs – Этот модуль содержит функции для взаимодействия с базой данных SQLite. Здесь определены функции для создания таблицы в базе данных, вставки новой камеры в базу данных, обновления существующей камеры, удаления камеры из базы данных и получения всех камер из базы данных. Все эти функции работают непосредственно с базой данных, выполняя соответствующие SQL-запросы. Пример функции обновления данных о камере представлен в листинге 6.

Листинг 6 – Функция для обновления данных о камере в базе

```
pub fn update(camera: &Camera, conn: &Connection) -> Result<usize> {
    conn.execute(
        "
                UPDATE camera SET
                    camera_name = ?1,
                    camera_mass = ?2,
                    camera_fov_x = ?3,
                    camera_resolution_x = ?4,
                    camera_resolution_y = ?5
                WHERE camera_id = ?6",
        (
            &camera.name,
            &camera.mass,
            &camera.fov_x,
            &camera.resolution_x,
            &camera.resolution_y,
            &camera.id,
        ),
    )
}
```

Сервис Algorithms, описанный файлом algorithms.rs, представляет собой ключевую составляющую системы и выполняет алгоритмические функции. Он включает в себя следующие функции:

discretize_area: Эта функция принимает вектор кортежей, представляющих собой координаты x и y полигона, и размеры фотографии (ширину и высоту). Функция начинает с инициализации минимальных и максимальных значений x и y. Далее, функция проходит по всей области с шагом равным ширине и высоте фотографии и проверяет находится ли какой-либо из углов прямоугольника внутри полигона, основываясь на лучевом методе, в случае успеха, добавляет точку в результирующий вектор. После того как все области были проверены, функция возвращает вектор кортежей, представляющих собой координаты центров прямоугольников, которые составляют полигон. Реализация представлена в листинге 1 в приложении А.

nearest_neighbor: Эта функция принимает вектор точек и начальную точку, а затем использует алгоритм ближайшего соседа для определения оптимального пути через все точки. Функция инициализирует набор оставшихся точек и результатирующий вектор с начальной точкой. Функция затем входит в цикл, продолжающийся, пока все точки не будут обработаны. В каждой итерации цикла функция находит ближайшую точку к текущей точке, удаляет ее из оставшихся точек и добавляет ее в результатирующий вектор [12]. Реализация представлена в листинге 2 в приложении А.

euclidean_distance: Это вспомогательная функция, которая принимает две точки и возвращает евклидово расстояние между ними. Расстояние вычисляется как квадратный корень из суммы квадратов разностей координат x и y. Реализация представлена в листинге 7.

Листинг 7 – Функция расчета расстояния между двумя точками

```
// Helper function to calculate the Euclidean distance between two points
pub fn euclidean_distance(a: &(f64, f64), b: &(f64, f64)) -> f64 {
 let (x1, y1) = *a;
 let (x2, y2) = *b;
 ((x2 - x1).powi(2) + (y2 - y1).powi(2)).sqrt()
}
```

brute_force: Эта функция использует алгоритм полного перебора для поиска оптимального пути через набор точек. Функция начинает с создания отдельного потока для каждой точки, каждый из которых запускает функцию brute_force_helper. Эта вспомогательная функция рекурсивно исследует все возможные пути через оставшиеся точки, обновляя лучший путь при обнаружении более короткого пути. Реализация представлена в листинге 3 в приложении А.

calculate_distance: Эта функция вычисляет общую длину пути, проходящего через заданный набор точек Реализация представлена в листинге 8.

Листинг 8 – Функция расчета общей длины пути

```
#[tauri::command]
pub fn calculate_distance(points: Vec<(f64, f64)>) -> f64 {
 points
 .iter()
 .zip(points.iter().cycle().skip(1))
 .map(|(a, b)| euclidean_distance(a, b))
 .sum()
}
```

### 4.2. Реализация интерфейса системы

Пользовательский интерфейс системы состоит из трех основных меню: центрального меню, левого меню и правого меню. Центральное меню отображает карту и маршрут дрона, а также содержит набор кнопок для управления.

Центральное меню предоставляет основные инструменты и функции для работы с картой, панель инструментов, отображение маршрута и выполнения других операций, связанных с управлением маршрутом дрона, оно представлено на рисунке 5.


Рисунок 5 – Центральное меню

Центральное меню состоит из элементов:

1. Карта и маршрут: В центре экрана отображается карта (загружаемая с OpenStreetMap [11]), на которой отображается маршрут дрона. Маршрут представлен в виде линии, соединяющей точки съемки, по которым дрон должен пролететь. Это визуальное представление планируемого маршрута.

2. Кнопки переключения меню: В верхней части интерфейса расположены кнопки, которые позволяют переключать отображение левого и правого меню. Пользователь может скрыть или показать эти меню по своему усмотрению для освобождения места на экране.

3. Панель инструментов: На панели инструментов расположены кнопки для выполнения различных действий:

3.1.Отмена (Undo): Кнопка, позволяющая отменить последнее действие при построении полигона;

3.2.Навигация (Navigation): Кнопка, позволяющая пользователю вернуться к режиму навигации после использования других инструментов.

Это удобно, если пользователь не хочет случайно переместить точку полигона области съемки;

3.3.Рисование (Draw): Кнопка, активирующая режим построения полигона. В этом режиме пользователь может указать область, которую дрон должен снять;

3.4.Установка стартовой точки (Set Starting Point): Кнопка, позволяющая пользователю установить стартовую точку для маршрута дрона. После выбора этой опции пользователь может щелкнуть на карте, чтобы установить стартовую точку;

3.5.Расчет (Calculate): Кнопка, которая выполняет валидацию введенных данных и отправляет их на обработку на сервере. Результаты расчета маршрута отображаются на карте.

Левое меню включает два блока: блок UAV (Беспилотный Летательный Аппарат) и блок Camera (Камера). Левое меню представлено на рисунке 6.


Рисунок 6 – Левое меню

Блок UAV включает в себя:

1. Выбор и отображение списка UAV: В верхней части блока расположен выпадающий список (select), который позволяет пользователю выбрать один из доступных UAV. Каждый элемент списка содержит имя UAV. При выборе UAV из списка его параметры отображаются в полях ниже.

2. Кнопка «Fetch»: Рядом с выпадающим списком находится кнопка «Fetch», которая позволяет пользователю обновить список UAV, загрузив его с базы.

3. Кнопка «UAV details»: Эта кнопка позволяет пользователю переключать отображение блока с параметрами UAV. При нажатии на кнопку блок с параметрами UAV может быть скрыт или показан на экране.

4. Внутри блока UAV параметры UAV отображаются в виде полей для ввода которые позволяют пользователю просматривать и редактировать параметры UAV. Некоторые из доступных параметров UAV включают:

4.1.ID: Идентификатор UAV (только для чтения);

4.2.Name: Имя UAV;

4.3.Max Payload Mass: Максимальная масса груза, которую может нести UAV;

4.4.Flight Duration: Длительность полета UAV;

4.5.Takeoff speed: Скорость взлета UAV;

4.6.Flight speed: Скорость полета UAV;

4.7.Min altitude: Минимальная высота полета UAV;

4.8.Max altitude: Максимальная высота полета UAV;

5. Под блоком с параметрами UAV располагается панель инструментов, которая содержит следующие кнопки:

5.1.«Update»: Кнопка, которая позволяет пользователю обновить параметры выбранного UAV.

5.2.«New»: Кнопка, которая позволяет пользователю создать новый UAV с указанными в полях параметрами.

5.3.«Delete»: Кнопка, которая позволяет пользователю удалить выбранный UAV.

5.4.«Undo»: Кнопка, которая позволяет пользователю отменить последнее изменение параметров UAV.

Блок Camera аналогичен блоку UAV и включает в себя:

1. выпадающий список (select), который позволяет пользователю выбрать одну из доступных камер. Каждый элемент списка содержит имя камеры. При выборе камеры из списка ее параметры отображаются в полях ниже.

2. Кнопка «Fetch»: Рядом с выпадающим списком находится кнопка «Fetch», которая позволяет пользователю обновить список камер, загрузив его с базы.

3. Кнопка «Camera details»: Эта кнопка позволяет пользователю переключать отображение блока с параметрами камеры. При нажатии на кнопку блок с параметрами камеры может быть скрыт или показан на экране.

4. Внутри блока Camera параметры камеры отображаются в виде полей для ввода (input) и меток (label), которые позволяют пользователю просматривать и редактировать параметры камеры. Некоторые из доступных параметров камеры включают:

4.1.ID: Идентификатор камеры (только для чтения).

4.2.Name: Имя камеры.

4.3.Mass (grams): Масса камеры в граммах.

4.4.X-axis FOV (degrees): Угол обзора камеры по оси X в градусах.

4.5.Resolution X: Разрешение камеры по оси X.

4.6.Resolution Y: Разрешение камеры по оси Y.

5. Под блоком с параметрами камеры располагается панель инструментов, которая содержит следующие кнопки:

5.1.«Update»: Кнопка, которая позволяет пользователю обновить параметры выбранной камеры.

5.2.«New»: Кнопка, которая позволяет пользователю создать новую камеру с указанными в полях параметрами.

5.3.«Delete»: Кнопка, которая позволяет пользователю удалить выбранную камеру.

5.4.«Undo»: Кнопка, которая позволяет пользователю отменить последнее изменение параметров камеры.

Правое меню состоит из трех блоков: «Altitude Menu» (меню высоты), «Algorithm Menu» (меню алгоритма) и «Mission Parameters» (параметры миссии). Правое меню представлено на рисунке 8.

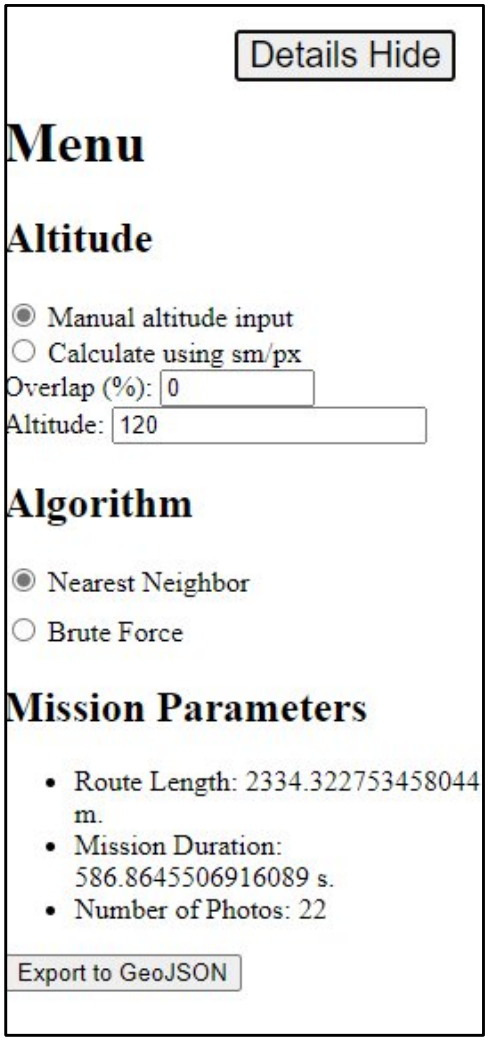


Рисунок 7 – Правое меню

Далее рассмотрен каждый из блоков подробнее:

1. Блок «Altitude Menu», здесь можно выбрать режим ввода высоты. Есть два варианта: «Manual altitude input» (ручной ввод высоты) и «Calculate using sm/px» (расчет высоты на основе sm/px):

1.1.Если выбран режим ручного ввода, можно указать процент перекрытия и ввести высоту вручную.

1.2.Если выбран режим расчета на основе sm/px, нужно указать процент перекрытия и ввести значение sm/px. После ввода всех параметров

можно нажать кнопку «Calculate Altitude» (рассчитать высоту) для получения результата.

2. Блок «Algorithm Menu», здесь можно выбрать алгоритм для расчета маршрута. Есть два варианта: «Nearest Neighbor» (ближайший сосед) и «Brute Force» (полный перебор).

3. Блок «Mission Parameters», здесь отображаются параметры миссии, такие как длина маршрута, продолжительность миссии и количество фотографий. Также есть кнопка «Export to GeoJSON» (экспорт в формате GeoJSON), которая позволяет экспортировать координаты миссии в этот формат для дальнейшего использования.

Дополнительные иллюстрации, демонстрирующие пользовательский интерфейс, представлены в Приложении Б.

Исходный код программы находится по ссылке: https://github.com/EvgenKot/uav_route_calculation

### 4.3. Тестирование системы

В ходе разработки и последующего программного обеспечения были разработаны и проведены комплексные тесты для проверки функциональности и стабильности работы системы. Проведенные тесты относятся к виду функционального тестирования, а также тестирования пользовательского интерфейса.

Тестирование проводилось в соответствии с предложенной методологией, при которой каждый из тестовых сценариев включал в себя конкретные шаги для воспроизведения действий пользователя, а также определенный ожидаемый результат. В процессе тестирования внимание было уделено как отдельным функциям, так и взаимодействию между различными компонентами системы.

Процесс тестирования осуществлялся с использованием различных входных данных, включая граничные значения и невалидные данные, чтобы оценить устойчивость и надежность системы в различных условиях, а также чтобы выявить и устранить возможные недостатки. Протоколы

тестирования приведены в таблице 12. Результат выполнения теста 5 «Проверка параметров миссии» представлен на рисунке 8. Результат выполнения теста 6 «Визуализация маршрута»представлен на рисунке 9.

Таблица 12 – Протоколы тестирования системы

| *№* | ***Название теста*** | ***Шаги*** | ***Ожидаемый результат*** | ***Тест прой ден?*** |
|---|---|---|---|---|
| *1* | Загрузка дрона | 1. В левом меню выбирать идентификатор дрона.<br>2. Нажать кнопку «Fetch». | В полях деталей дрона отображается информация о выбранном дроне. | да |
| *2* | Редактирование дрона | 1. Выбирать дрон.<br>2. Включить режим редактирования.<br>3. Изменить параметры дрона.<br>4. Нажать кнопку «Update». | Информация о дроне в базе данных обновляется. | да |
| *3* | Расчет маршрута | 1. Ввети необходимые параметры, высоту полета, алгоритм, обозначить точку старта, обозначить зону съемки, выбрать дрон, выбрать камеру.<br>2. Нажать кнопку. «Calculate» | На карте отображается рассчитанный маршрут. | да |
| *4* | Некорректная попытка расчета маршрута | 1. Не ввести необходимые параметры, высоту полета, алгоритм, точку старта, зону съемки, дрон, камеру.<br>2. Нажать кнопку «Calculate». | Отображается ошибка о том, что не все параметры выбраны | да |
| *5* | Проверка параметров миссии | 1. Просмотрите отображаемые параметры миссии в правом меню. | Параметры миссии (длина маршрута, продолжительность миссии, количество фотографий) отображаются корректно. | да |
| *6* | Визуализация маршрута | 1. После расчета маршрута проверьте визуализацию маршрута на карте. | Маршрут корректно отображается на карте. | да |
| *7* | Экспорт в GeoJSON | 1. После расчета маршрута нажать кнопку «Export to GeoJSON».<br>2. Выбрать путь сохранения | Файл GeoJSON с информацией о маршруте сохраняется. | да |
| *8* | Загрузка камеры | 1. В левом меню выберать идентификатор камеры.<br>2. Нажать кнопку «Fetch». | В полях деталей камеры отображается информация о выбранной камере. | да |

Окончание таблицы 12

| *№* | ***Название теста*** | ***Шаги*** | ***Ожидаемый результат*** | ***Тест прой ден?*** |
|---|---|---|---|---|
| *9* | Редактирование камеры | 1. Выбрать камеру.<br>2. Включить режим редактирования.<br>3. Изменить параметры камеры.<br>4. Нажать кнопку «Update». | Информация о камере в базе данных обновляется. | да |
| *10* | Выбор алгоритма | 1. В правом меню выберите желаемый алгоритм. | Выбранный алгоритм устанавливается как текущий. | да |
| *11* | Ручной ввод высоты | 1. В правом меню выберите режим ручного ввода высоты.<br>2. Введите желаемую высоту. | Введенная высота устанавливается как текущая. | да |
| *12* | Расчет высоты | 1. В правом меню выберите режим расчета высоты.<br>2. Введите значение sm/px.<br>3. Нажмите кнопку «Calculate Altitude». | Высота автоматически рассчитывается и устанавливается как текущая. | да |
| *13* | Изменение параметра перекрытия | 1. В правом меню ввести значение в поле «Overlap (%)». | Значение перекрытия изменяется в соответствии с введенным значением. | да |
| *14* | Проверка работы с отсутствующими данными | 1. Нажать кнопку «Calculate», когда база данных пустая. | Система сообщает о том, что дрон не выбран. | да |

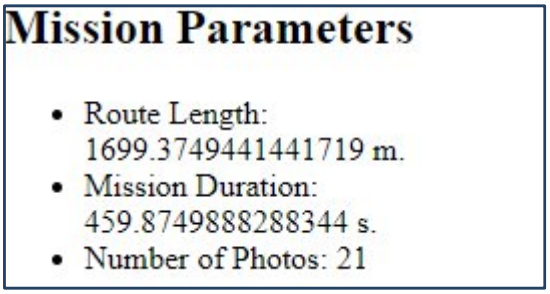


Рисунок 8 – Результат выполнения теста 5 «Проверка параметров миссии»

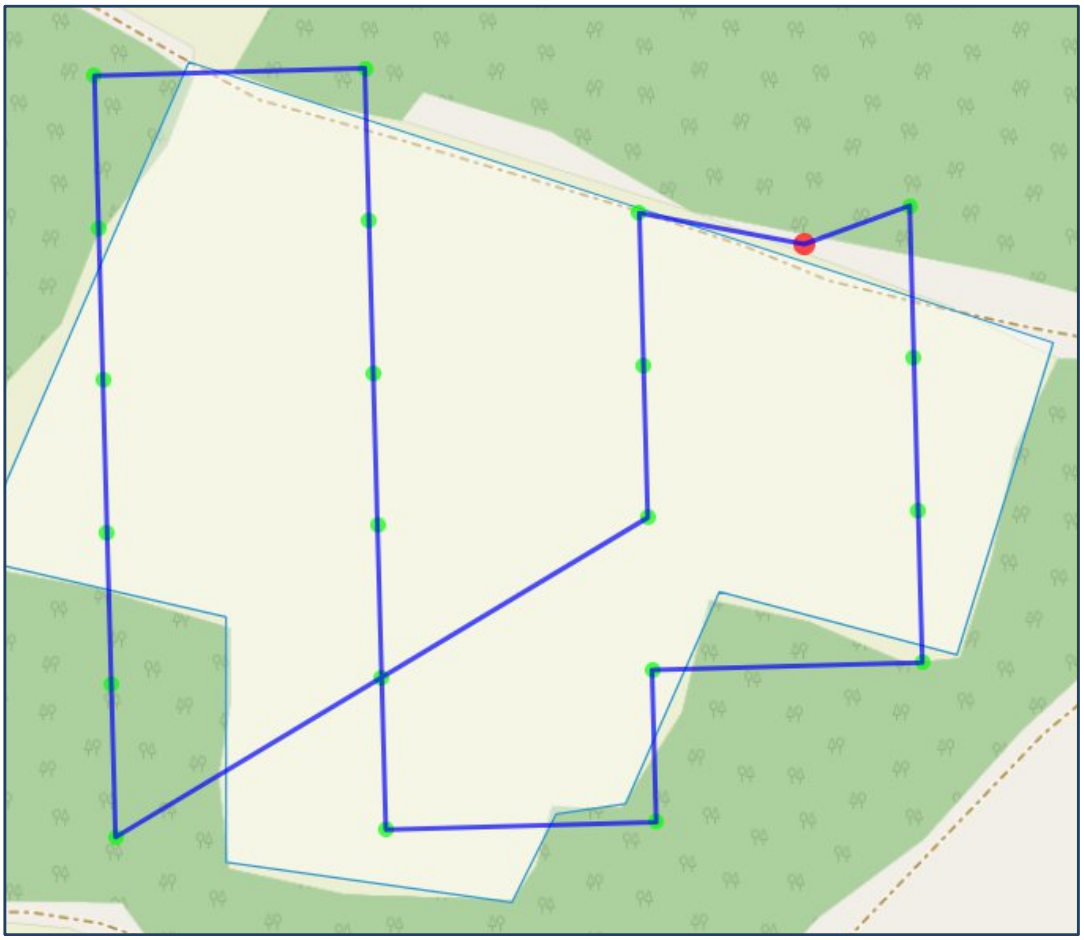

Рисунок 9 – Результат выполнения теста 6 «Визуализация маршрута»

## ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано десктопное приложение для расчета маршрута сельскохозяйственного дрона с учетом его характеристик. Решены были следующие задачи.

1. Был проведен анализ предметной области и изучены существующие решения, связанные с планированием маршрутов для дронов.
2. Была разработана архитектура приложения, обеспечивающая его масштабируемость и простоту внесения изменений в будущем.
3. Был выполнена реализация приложения. Разработаны компоненты бэкенда для расчета маршрута и взаимодействия с базой данных, а также компоненты пользовательского интерфейса.
4. Было проведено тестирование приложения, включая проверку корректности расчетов и работы пользовательского интерфейса.

В перспективе планируется дальнейшее развитие приложения, включая добавление функции построения маршрута нескольких полей, добавление новых и улучшение уже написанных алгоритмов планирования маршрута. Также возможна работа по интеграции приложения с другими системами для автоматического получения координат поля, и экспорта маршрута на дроны.

**ЛИТЕРАТУРА**

1. Якушев В. П., Якушев В. В., Матвеенко Д. А. Роль и задачи точного земледелия в реализации национальной технологической инициативы // Агрофизика. 2017. № 1. С. 51–65.

2. Официальный сайт проекта DroneDeploy. каЭлектронный ресурс] URL: https://www.dronedeploy.com/ (дата обращения: 11.05.2023 г.).

3. Официальный сайт проекта Litchi. [Электронный ресурс] URL: https://flylitchi.com/ (дата обращения: 11.05.2023 г.).

4. Официальный сайт проекта Pix4D. [Электронный ресурс] URL: https://www.pix4d.com/ (дата обращения: 11.05.2023 г.).

5. Официальный сайт проекта UgCS. [Электронный ресурс] URL: https://www.ugcs.com/ (дата обращения: 11.05.2023 г.).

6. Официальный сайт проекта Tauri. [Электронный ресурс] URL: https://tauri.app/ (дата обращения: 11.05.2023 г.).

7. Официальный сайт проекта Rust. [Электронный ресурс] URL: https://www.rust-lang.org/ (дата обращения: 11.05.2023 г.).

8. Официальный сайт проекта Svelte. [Электронный ресурс] URL: https://svelte.dev/ (дата обращения: 11.05.2023 г.).

9. Официальный сайт проекта OpenLayers. [Электронный ресурс] URL: https://openlayers.org/ (дата обращения: 11.05.2023 г.).

10. Официальный сайт GeoJSON. [Электронный ресурс] URL: https://geojson.org/ (дата обращения: 11.05.2023 г.).

11. Офицальный сайт проекта OpenStreetMap [Электронный ресурс] URL: https://www.openstreetmap.org (дата обращения: 11.05.2023 г.).

12. G. Gutin, A. Yeo and A. Zverovich Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP // Discrete Applied Mathematics. - 2002. - №117. - С. 81–86.

# ПРИЛОЖЕНИЯ

## Приложение А. Функции расчета сервиса Algorithms

Реализация функции дискретизации области съемки переведена в листинге 1. Реализация функции расчета маршрута с использованием алгоритма ближайшего соседа приведена в листинге 2. Реализация функций расчета маршрута с использованием алгоритма полного перебора приведена в листинге 3.

Листинг 1 – Функция дискретизации области

```
#[tauri::command]
pub fn discretize_area(
    // Vector of tuples representing x and y coordinates of the polygon.
    polygon: Vec<(f64, f64)>,
    // Width of the photo.
    photo_width: f64,
    // Height of the photo.
    photo_height: f64,
) -> Result<Vec<(f64, f64)>, String> {
    // Returns a Result containing either a vector of tuples representing
the discretized area or a String error.
    println!("Received polygon coordinates: {:?}", polygon);


    // Initialize min and max x and y values to extreme opposites.
    let (mut min_x, mut max_x, mut min_y, mut max_y) =
        (INFINITY, NEG_INFINITY, INFINITY, NEG_INFINITY);


    // Loop through polygon coordinates to find min and max x and y values.
    for (x, y) in &polygon {
        min_x = min_x.min(*x);
        max_x = max_x.max(*x);
        min_y = min_y.min(*y);
        max_y = max_y.max(*y);
    }


    // Inner function to check if a point is inside the polygon.
    fn is_inside_polygon(point: (f64, f64), polygon: &Vec<(f64, f64)>) ->
bool {
        // Initialize inside flag to false.
        let mut inside = false;
        let len = polygon.len();
        let mut j = len - 1;


        // Loop through the polygon's vertices, checking if the point
intersects.
        for i in 0..len {
            let (x_i, y_i) = polygon[i];
            let (x_j, y_j) = polygon[j];
```

```
            // Determine if the point is intersecting with the edge from
vertex i to vertex j.
            let intersect = (y_i > point.1) != (y_j > point.1)
                && point.0 < (x_j - x_i) * (point.1 - y_i) / (y_j - y_i) +
x_i;
            if intersect {
                inside = !inside;
            }
            j = i;
        }
        inside
    }


    // Initialize the result vector.
    let mut result = Vec::new();


    // Calculate half the camera width and height.
    let (half_camera_width, half_camera_height) = (photo_width / 2.0,
photo_height / 2.0);
    let mut x = min_x;


    // Iterate over x and y values from min to max, checking for
intersection with the polygon.
    while x <= max_x {
        let mut y = min_y;


        while y <= max_y {
            // Calculate the corners and midpoints of the rectangle at (x,
y).
            let points = vec![
                (x, y), // top-left corner
                (x + photo_width, y), // top-right corner
                (x, y + photo_height), // bottom-left corner
                (x + photo_width, y + photo_height), // bottom-right corner
                (x + half_camera_width, y), // top edge center
                (x + photo_width, y + half_camera_height), // right edge
center
                (x + half_camera_width, y + photo_height), // bottom edge
center
                (x, y + half_camera_height), // left edge center
            ];


            // Check if any corner or midpoint of the rectangle is inside
the polygon.
            let is_any_point_inside = points
                .into_iter()
                .any(|point| is_inside_polygon(point, &polygon));
```

```
            if is_any_point_inside {
                let center_x = x + half_camera_width;
                let center_y = y + half_camera_height;
                result.push((center_x, center_y));
            }


            y += photo_height;
        }


        // Move to the next position in x-axis.
        x += photo_width;
    }


    // Return the result vector containing the centers of the rectangles
that intersect with the polygon.
    Ok(result)
}
```

Листинг 2 – Функция расчета «Ближайший сосед»

```
#[tauri::command]
pub fn nearest_neighbor(
    points: Vec<(f64, f64)>,
    start_point: (f64, f64),
) -> Result<Vec<(f64, f64)>, String> {
    if points.is_empty() {
        return Err("The input points must not be empty».to_string());
    }


    let mut remaining_points: Vec<(f64, f64)> = points;
    let mut result: Vec<(f64, f64)> = Vec::new();
    let mut current_point = start_point;


    while !remaining_points.is_empty() {
        let (nearest_index, nearest_point) = remaining_points
            .iter()
            .enumerate()
            .min_by(|(_, a), (_, b)| {
                euclidean_distance(&current_point, a)
                    .partial_cmp(&euclidean_distance(&current_point, b))
                    .unwrap_or(std::cmp::Ordering::Equal)
            })
            .ok_or("Failed to find the nearest point».to_string())?;
        let nearest_point = *nearest_point;
        remaining_points.remove(nearest_index);
        if result.is_empty() {
            result.push(start_point);
        }
        result.push(nearest_point);
        current_point = nearest_point;
    }


    result.push(start_point);
    Ok(result)
}
```

Листинг 3 – Функции расчета «Полный перебор»

```
// Main function to find the shortest path using the brute force approach.
#[tauri::command]
pub fn brute_force(points: Vec<(f64, f64)>, start_point: (f64, f64)) ->
Vec<(f64, f64)> {
    // Wrap the points and best_path in Arc for shared ownership across
threads.
    let points = Arc::new(points);
    let best_path = Arc::new(Mutex::new((Vec::new(), f64::MAX)));


    let mut threads = Vec::new();


    // Iterate through each point, creating a separate thread for each.
    for (i, point) in points.iter().enumerate() {
        // Clone Arc references to share ownership with the spawned
threads.
        let points = Arc::clone(&points);
        let best_path = Arc::clone(&best_path);
        let start_point = start_point.clone();
        let removed_point = point.clone();


        // Spawn a thread for each point, removing it from the remaining
points.
        let thread = thread::spawn(move || {
            let mut remaining_points = points.to_vec();
            remaining_points.remove(i);


            let current_path = vec![start_point, removed_point];
            let current_distance = euclidean_distance(&start_point,
&removed_point);


            // Call the helper function in each thread.
            brute_force_helper(
                remaining_points,
                current_path,
                start_point,
                current_distance,
                &best_path,
            );
        });


        threads.push(thread);
    }


    // Wait for all threads to complete.
    for thread in threads {
        thread.join().unwrap();
    }

    // Extract the best path from the Arc<Mutex<_>>.
    let (best_path, _) =
Arc::try_unwrap(best_path).unwrap().into_inner().unwrap();
    best_path
}
```

```
// Recursive helper function to find the shortest path using the brute
force approach.
fn brute_force_helper(
    points: Vec<(f64, f64)>,
    current_path: Vec<(f64, f64)>,
    start_point: (f64, f64),
    current_distance: f64,
    best_path: &Arc<Mutex<(Vec<(f64, f64)>, f64)>>,
) {
    // Base case: if there are no points left, update the best path if the
current path is shorter.
    if points.is_empty() {
        let total_distance =
            current_distance + euclidean_distance(&start_point,
current_path.last().unwrap());


        let mut best_path = best_path.lock().unwrap();


        if total_distance < best_path.1 {
            best_path.0 = current_path.clone();
            best_path.0.push(start_point);
            best_path.1 = total_distance;
        }
    } else {
        // Iterate through each remaining point, removing it from the list
and updating the path.
        for (i, point) in points.iter().enumerate() {
            let mut remaining_points = points.clone();
            let removed_point = remaining_points.remove(i);


            let last_point = current_path.last().unwrap();
            let new_distance = current_distance +
euclidean_distance(last_point, &removed_point);


            // If the updated path is shorter than the current best path,
continue the search.
            if new_distance < best_path.lock().unwrap().1 {
                let mut new_path = current_path.clone();
                new_path.push(*point);


                brute_force_helper(
                    remaining_points,
                    new_path,
                    start_point,
                    new_distance,
                    best_path,
                );
            }
        }
    }
}
```

## Приложение Б. Скриншоты приложения

Скриншоты разработанного приложения приведены на рисунках 1–6.


Рисунок 1 – Отображение всех меню


Рисунок 2 – Расчет исходя из разрешения съемки

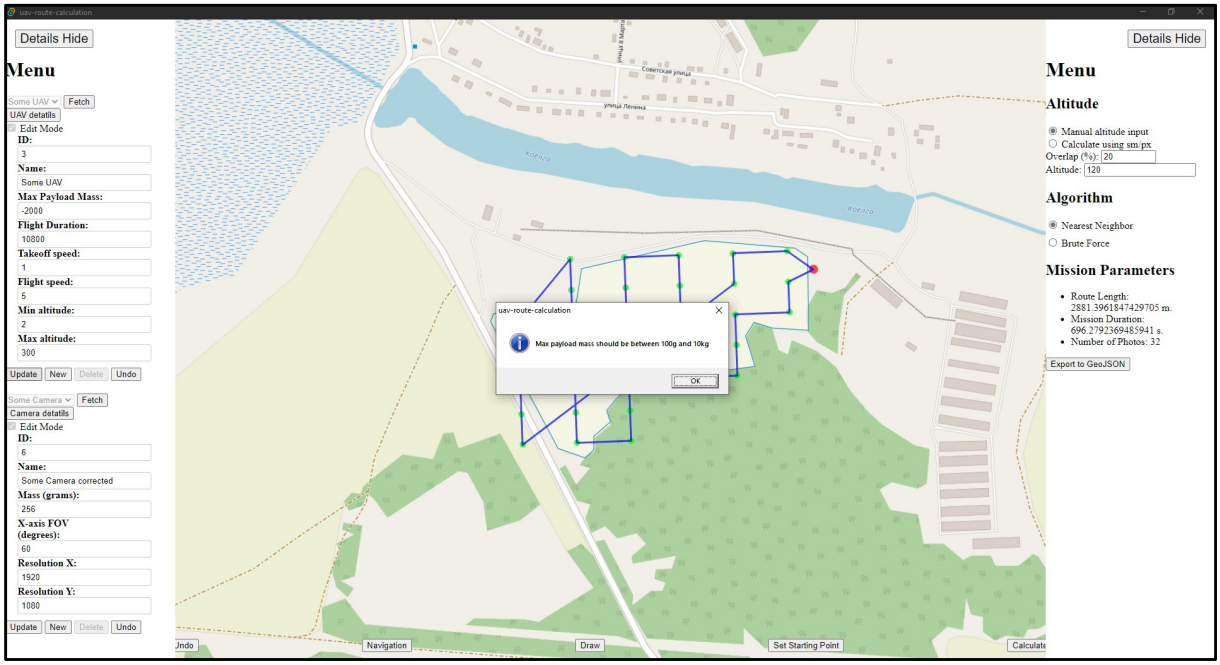


Рисунок 3 – Сообщение об ошибке

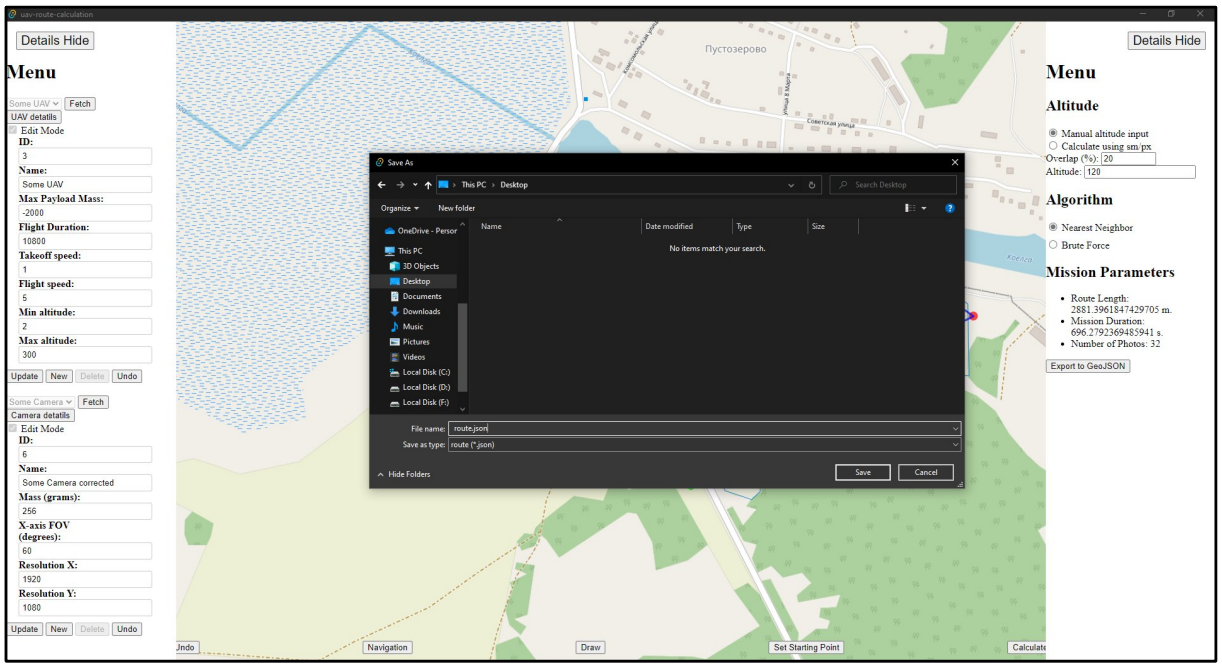


Рисунок 4 – Окно экспорта в GeoJSON


Рисунок 5 – Работа алгоритма ближайший сосед


Рисунок 6 – Работа алгоритма полный перебор