

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Новосибирский национальный исследовательский государственный
университет»
(Новосибирский государственный университет, НГУ)
Структурное подразделение Новосибирского государственного университета –
Высший колледж информатики Университета (ВКИ НГУ)
КАФЕДРА ИНФОРМАТИКИ

**РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ ВАЛИДАЦИИ
ТЕМПЕРАТУРНЫХ ДАННЫХ КА LANDSAT И TERRA/AQUA НА
ОСНОВЕ НАЗЕМНЫХ ИЗМЕРЕНИЙ**

Квалификация программист

Руководитель

к.ф.-м.н., старший научный сотрудник ФИЦ ИВТ

Студент курса 107сб2

Мамаш Е.А.

« 7 » июня 2024 г.

Косинова А.И.

« 7 » июня 2024 г.

Нормоконтроль

Горячкина М.А.

« 7 » июня 2024 г.

Новосибирск

2023

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ	4
ВВЕДЕНИЕ	6
1 ПОСТАНОВКА ЗАДАЧИ ВКР	8
1.1 Бизнес-требования	8
1.2 Пользовательские требования.....	8
1.3 Системные требования	9
1.4 Требования к графическому пользовательскому интерфейсу.....	10
1.5 План-график выполнения ВКР	11
2 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ	13
2.1 Описание предметной области задачи ВКР	13
2.1.1 Информационные объекты предметной области и взаимосвязи между ними	13
2.1.2 Информационные и функциональные потребности пользователей разрабатываемого модуля	17
2.1.3 Методы работы с информационными объектами предметной области	17
2.1.4 Обзор существующих программных реализаций решения задачи	22
2.1.5 Концептуальное обоснование разработки	24
2.2 Классы и характеристики пользователей	24
2.3 Функциональные требования.....	26
2.3.1 Определение функциональных возможностей программного модуля.....	26
2.3.2 Описание прецедентов.....	27
2.4 Нефункциональные требования.....	30
3 ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ.....	32
3.1 Сравнительный анализ имеющихся возможностей по выбору средств разработки	32
3.2 Характеристика выбранных программных сред и средств.....	32
4 АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ	34
4.1 Этапы реализации модуля	34
4.2 Пользовательский интерфейс модуля	34
4.2.1 Взаимодействие пользователей с модулем.....	34
4.2.2 Проектирование пользовательских сценариев	38
4.2.3 Определение операций пользователей	40
4.2.4 Составление функциональных блоков	40
4.2.5 Проектирование структуры экранов модуля и схемы навигации	42
4.2.6 Низкоуровневое проектирование	43
4.3 Входные и выходные данные.....	44
4.4 Разработка механизмов хранения данных и структуры файловой системы, реализуемых в рамках модуля	50

4.5 Алгоритмы реализации используемых математических моделей	51
4.6 Алгоритмы использования применяемых программных технологий	55
4.7 Архитектура и схема функционирования модуля	56
5 ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ.....	59
5.1 План тестирования	59
5.2 Результаты тестирования	60
5.3 Оптимизация модуля	62
6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	63
ЗАКЛЮЧЕНИЕ.....	65
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	67
ПРИЛОЖЕНИЯ	68
Приложение А	68
Приложение Б	69
Приложение Г	69
Приложение Д	72

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ

Валидация (validation) – процесс подтверждения или проверки правильности и точности результатов, методов, моделей или данных, используемых в научном исследовании, с целью обеспечения их достоверности и надежности.

Спектрорадиометр (spectroradiometer) – прибор, предназначенный для измерения интенсивности излучения в различных участках электромагнитного спектра, обычно в видимом, инфракрасном или микроволновом диапазонах.

GEE (Google Earth Engine) – облачная платформа для анализа больших объемов геопространственных данных.

API (Application Programming Interface) – набор инструментов, протоколов и определений, которые позволяют различным программным приложениям взаимодействовать друг с другом.

MODIS (Moderate Resolution Imaging Spectroradiometer) – датчик, установленный на спутниках Terra и Aqua, который собирает данные о Земле.

Landsat – серия спутников, предназначенных для изучения земной поверхности и мониторинга, предоставляющая многолетние наблюдения Земли в видимом и инфракрасном спектре.

Продукт спутника — обработанный набор данных, полученных со спутника, представленный в определенном формате для конкретных целей исследования или применения. Каждый продукт имеет уникальный идентификатор, указывающий на источник данных и их обработку.

Terra и Aqua – спутники, входящие в программу Earth Observing System (EOS), предназначенные для мониторинга окружающей среды Земли.

ST (Surface Temperature) – измерение температуры земной поверхности, которое имеет значение для оценки климатических изменений и состояния окружающей среды.

Collection / IC (ImageCollection) – набор или стек изображений, связанных по своему типу или характеристикам.

Coords / Coor (Coordinates) – числовые значения, которые определяют положение точки или объекта в пространстве.

PNG (PortableNetworkGraphics) – растровый формат хранения изображений на основе алгоритмов сжатия.

XLSX/XLS – расширение для программы Microsoft Excel.

MBE (MeanBiasError) – средняя ошибка смещения в математике и статистике. Это метрика, используемая для измерения средней ошибки между оценками и фактическими значениями.

RMSE (RootMean Squared Error) – среднеквадратичная ошибка в математике и статистике. Это метрика, используемая для измерения средней квадратичной разницы между оценками и фактическими значениями.

ВВЕДЕНИЕ

Изучение температуры поверхности Земли представляет собой ключевой аспект научных исследований во многих областях. Анализ динамики температуры поверхности позволяет выявлять климатические особенности различных территорий и определять зоны, потенциально пригодные для сельского хозяйства и промышленного использования.

Измерение температуры на поверхности может производиться как при помощи наземных измерительных устройств, так и с использованием данных инфракрасных спутниковых снимков. Например, спутниковые данные, полученные при помощи спектрорадиометров MODIS на спутниках Terra и Aqua, предоставляют информацию о температуре поверхности (LANDSAT) и регистрируются четыре раза в сутки. Эти данные считаются стандартными для MODIS, однако их использование требует проверки и обоснования, особенно при работе с различными типами поверхностей. В настоящее время проводятся исследования, сравнивающие данные MODIS с результатами наземных наблюдений в различных регионах мира, таких как США, Португалия, Намибия и Китай [1]. Авторы статьи обращают внимание на диаграммы спутниковых и наземных данных, а также на основные статистические показатели, такие как MBE и RMSE. Полученные результаты указывают на наличие смещений (MBE), обусловленных различиями в геометрии съемки и на местности [1]. Согласно [1], значения RMSE составляют менее 2°K ($1\text{K} \approx 1^{\circ}\text{C}$).

Также в исследованиях используется сравнение данных LANDSAT Aqua/MODIS с результатами шести измерительных станций, аналогичных предыдущим, расположенных в США. Анализ 10-летних данных подтверждает достоверность продуктов LANDSAT Aqua/MODIS. Кроме того, в работах используются наземные терморегистраторы типа Thermocron. Их сопоставление с данными LANDSAT MODIS/Terra, полученными при изучении поверхности Арктики, указывает на различия в измерениях температуры поверхности ($\text{RMSE} = 3,1^{\circ}\text{C}$, смещение = $-3,4^{\circ}\text{C}$).

При решении задач, где требуется высокая точность вычислений, критически важно использовать инструменты, способные её обеспечить. Программа LANDSAT, предоставляя спутниковые данные высокой точности, заслуживает особого внимания и широкого использования в научных исследованиях. Серия спутников LANDSAT, благодаря своей надежности и качеству данных, стала преимущественным поставщиком для проведения различных исследований.

Благодаря съемкам с периодичностью в две недели спутники LANDSAT успешно предоставляют тепловую информацию о земной поверхности в течение почти четырех

десятилетий, что делает их надёжным и удобным инструментом для анализа и проведения исследований на основе полученных данных. Начиная с сентября 2021 года в коллекцию данных LANDSAT добавили новые данные со спутника LANDSAT-9.[2] Температурный инфракрасный датчик (TIRS) на спутниках LANDSAT-8 и LANDSAT-9 снимает в двух тепловых каналах (10,60–11,19 мкм) и 11 (10,6-11,19 мкм) с номинальным пространственным разрешением 100 метров. Несмотря на доступность спутниковых данных, требуется тщательная проверка их возможности использования для анализа температурных характеристик поверхности в соответствии с особенностями исследуемых регионов.

В связи с необходимостью проведения быстрых и эффективных исследований на основе спутниковых данных Terra, Aqua/MODIS и KA Landsat возникла потребность в создании модуля, который автоматически валидирует наземные и спутниковые данные для анализа корреляции и оценки смещения показателей. Целью этого модуля является использование алгоритма, который сравнивает время по минутам и датам, принимает временной промежуток и сверяет температурные данные. Обычно валидация коэффициентов между наземными и спутниковыми показателями при измерении температуры на одной территории осуществлялась на основе наземных данных, которые вручную сопоставлялись со спутниковыми. Этот процесс потребляет много человеческих ресурсов, особенно при работе с большими объемами данных и продолжительными временными периодами. Таким образом, актуальность модуля подчеркивается потребностью в автоматизации процесса вычислений, фильтрации входных данных и визуализации процессов с учетом заданных параметров.

Целью данной выпускной квалификационной работы (ВКР) является разработка программного модуля, направленного на автоматическую валидацию наземных и спутниковых данных для анализа температурных характеристик поверхности Земли.

1 ПОСТАНОВКА ЗАДАЧИ ВКР

1.1 Бизнес-требования

Разработка программного модуля направлена на предоставление специалистам в области исследований климата комплекса функциональных инструментов, включая загрузку, визуализацию и анализ температурных данных. Это позволяет оптимизировать использование времени и ресурсов, а также обеспечивает сохранение и выгрузку результатов для дальнейших исследований.

Модуль ориентирован на автоматизацию специализированных исследований, сокращая временные и трудовые затраты и повышая точность результатов.

Основная цель приложения - повысить эффективность и точность научных исследований. Источник подтверждает, что автоматизация сокращает сроки, уменьшает трудозатраты и повышает качество исследований.[3]

Использование технологий GEE API обеспечивает более точные результаты анализа и укрепляет позицию модуля в современном информационном пространстве.

Система, направленная на уменьшение временных и ресурсных затрат при проведении исследований, имеет важное значение для эффективной работы исследовательских групп.

Таким образом, разработка модуля предоставляет специалистам современные инструменты для более точных и эффективных научных исследований.

1.2 Пользовательские требования

В рамках проектирования модуля выделены следующие пользовательские требования, которые должны быть обеспечены для работы и взаимодействия с пользователем.

Пользователь должен иметь возможность вводить координаты точки в формате "широта, долгота".

1. Пользователь должен иметь возможность выбирать источник данных для анализа (MODIS/Landsat). Также должна быть предусмотрена возможность выбора между данными с спутников Terra и Aqua, а также определение периода за день или ночь.

2. Программа должна обрабатывать выбранные пользователем координаты и получать данные о температуре с земли и со спутника.

3. Пользователь должен видеть отображение данных о температуре на земле и со спутника для выбранной точки.
4. Данные должны быть представлены в удобочитаемом формате с указанием времени и значения температуры.
5. Программа должна предоставлять возможность визуализации полученных данных в виде графиков.
6. Программа должна вычислять статистические показатели, такие как среднеквадратическая ошибка (RMSE) и средняя ошибка (MBE).
7. Должна быть предоставлена опция для сохранения таблицы со значениями и отчета в формате PDF.
8. Пользователь должен иметь возможность управлять программой с помощью консоли, без необходимости интерактивного интерфейса.
9. Программа должна быть способна обрабатывать ошибки, такие как некорректный ввод координат или выбор несуществующего источника данных.

Исходя из данного массива требований к ПО, приложение должно представлять из себя не только набор функций, а целую систему, которая удобна, проста в использовании, стабильна и безопасна для пользователя и его данных.

1.3 Системные требования

Системные требования для программного модуля на Python, включающего несколько подсистем, описывают высокоуровневые компоненты и функциональные особенности системы. Модули и связи между ними представлены на рисунке 1.3.1.

1. Модуль для работы с данными MODIS и Landsat через Google Earth Engine (GEE) API обеспечивает доступ к спутниковым данным для последующего анализа и сравнения с наземными данными.
2. Модуль для расчета RMSE (Root Mean Square Error) и MBE (Mean Bias Error) позволяет оценить точность и смещение между спутниковыми и наземными данными.
3. Модуль для сравнения временных и температурных данных анализирует временные ряды и температурные характеристики, выявляя различия между наземными и спутниковыми данными.
4. Модуль для подготовки визуализации создает графики и другие визуальные представления результатов анализа для наглядного представления данных.
5. Модуль для работы с Excel позволяет экспортировать и импортировать данные из Excel для удобства анализа и обработки.

6. Модуль для работы с выходными данными обеспечивает сохранение и выгрузку результатов анализа для последующего использования и документирования.

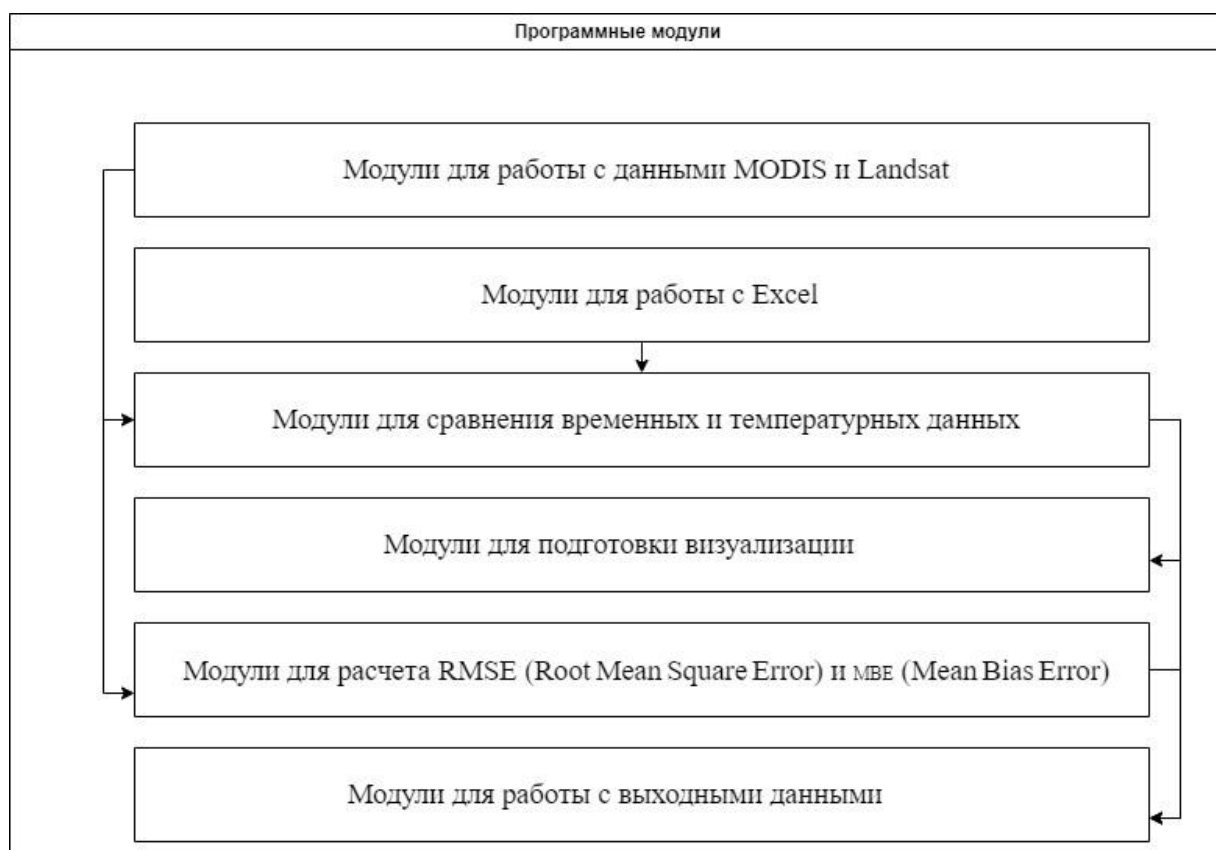


Рисунок 1.3.1 – Модульная архитектура программы

Эти требования учитывают функциональное разбиение системы на подсистемы и обеспечивают ее эффективную организацию и работу.

1.4 Требования к графическому пользовательскому интерфейсу

Требования к графическому пользовательскому интерфейсу:

1. Консольный диалог: взаимодействие с программным модулем осуществляется через консольный диалог, предлагающий пользователю последовательность шагов для выполнения задачи валидации данных.
2. Выбор формата выходного файла: пользователю предоставляется возможность выбора формата выходного файла. Опции форматов выводятся в консоль с соответствующими расширениями, и пользователь выбирает одну из них, введя соответствующее значение.
3. Указание вводного каталога: программа запрашивает у пользователя путь к каталогу, содержащему файлы наземных данных.

4. Указание выходного каталога: пользователю предлагается ввести путь к каталогу, куда будут сохранены результаты валидации данных. Это позволяет пользователю легко управлять местом сохранения результатов.

Консольный пользовательский интерфейс обеспечивает простоту и удобство использования программного модуля, позволяя пользователям легко и эффективно проводить валидацию температурных данных КА Landsat и Terra/Aqua на основе наземных измерений.

1.5 План-график выполнения ВКР

В ходе проектирования модуля выделены следующие контрольные точки и составлен календарный план, который представлен в таблице 1.5.1.

Таблица 1.5.1 – График работ

Этап	Виды работ	Срок завершения
1	Анализ требований	Ноябрь, 2023
2	Изучение API Google Earth Engine	Декабрь, 2023
3	Проектирование архитектуры приложения	Декабрь, 2023
4	Разработка модуля работы с Excel	Декабрь, 2023
5	Разработка модуля для получения спутниковых данных	Январь, 2024
6	Написание алгоритмов для и валидации данных	Март, 2024
7	Написание алгоритмов для работы с визуализацией	Март, 2024
8	Реализация выгрузки результатов	Март, 2024

9	Разработка системы сохранения данных	Апрель, 2024
10	Тестирование и отладка	Апрель, 2024
11	Разработка пользовательских инструкций	Май, 2024
12	Оформление пояснительной записки	Май, 2024

Таким образом, разработка модуля, вместе с соответствующей документацией будет завершена к маю 2024 года. Сам модуль будет готово к эксплуатации к концу апреля 2024 года.

2 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ

2.1 Описание предметной области задачи ВКР

2.1.1 Информационные объекты предметной области и взаимосвязи между ними

В ходе разработки модуля выделены следующие информационные объекты и сущности предметной области, взаимосвязанные в рамках одной системы:

1) Спутниковые данные

Представляют собой набор входных данных, получаемых в рамках функционирования модуля с платформы GEE посредством взаимодействия с GEE API. Спутниковые данные, полученные с LANDSAT[3] и Terra, Aqua/MODIS[4], предоставляют информацию о температуре на поверхности Земли. Информация, получаемая на основе спутниковых данных, является ключевой в вопросах исследования земной поверхности. Аппараты отправляют на станции данные с определенной частотой и имеют свой ряд особенностей, что отражено в таблице 2.1.1.1.

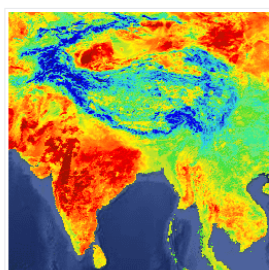
Таблица 2.1.1.1 – Технические характеристики спутников LANDSAT-8 и Terra, Aqua/MODIS

Характеристики	LANDSAT-8	Terra, Aqua/MODIS
Тип данных	оптические	оптические
Режим съемки	Моносъемка	Моносъемка
Спектральные каналы	панхроматический; мультиспектральные: VNIR (6), SWIR (2), TIR (2)	36 спектральных каналов в видимом, ближнем, среднем и тепловом инфракрасном диапазонах

Пространственное разрешение в надире, м	15 (панхроматический); 30 (VNIR, SWIR); 100 (TIR)	250 (каналы 1-2); 500 (каналы 3-7);1000 (каналы 8-36)
Динамический диапазон, бит/пиксель	16	12
Ширина полосы съемки в надире, км	185	2330
Период повторной съемки	1 раз в 16 суток	1-2 раза в сутки, в зависимости от широты места съемки

Для того, чтобы получить данные с GEE необходимо обращаться к продуктам источника спутниковых данных. Для Terra MODIS продукт, используемый в модуле представлен на рисунках 2.1.1.2[5].

MOD11A1.061 Terra Land Surface Temperature and Emissivity Daily Global 1km



Dataset Availability

2000-02-24T00:00:00Z–2024-04-11T00:00:00Z

Dataset Provider

[NASA LP DAAC at the USGS EROS Center](#)

Earth Engine Snippet

```
ee.ImageCollection("MODIS/061/MOD11A1")
```

Tags

daily emissivity global lst modis nasa surface-temperature terra usgs

mod11a1

Рисунок 2.1.1.2 – Продукт для Terra/MODIS

Для Aqua/MODIS используется продукт, изображенный на рисунке 2.1.1.3[6]

MYD11A1.061 Aqua Land Surface Temperature and Emissivity Daily Global 1km 

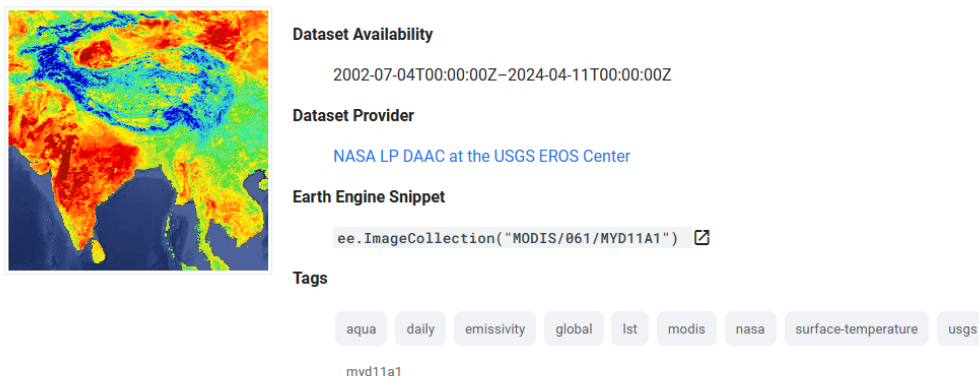


Рисунок 2.1.1.3 – Продукт для Aqua/MODIS

Для Landsat 8 используется продукт на рисунке 2.1.1.4[7]

USGS Landsat 8 Level 2, Collection 2, Tier 1 

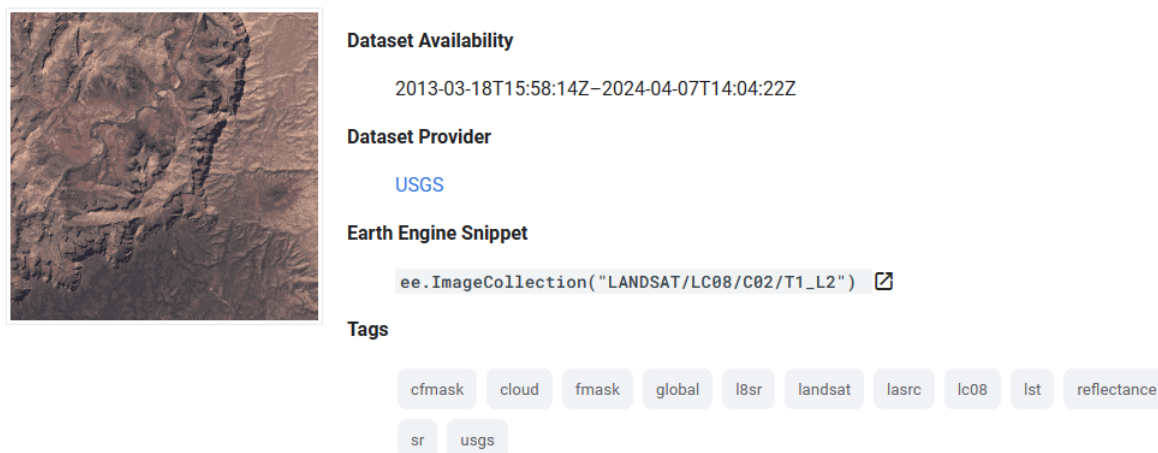


Рисунок 2.1.1.4 – Продукт для Landsat 8

2) Наземные данные

Наземные данные представляют собой собранный на земле при помощи терморегистраторов материал, состоящий из даты, времени и значений прибора, в названии файла указывается номер датчика. Датчики и координаты местности, на которой проводились исследования прикладываются в виде ведомости. Пример ведомости изображен на рисунке 2.1.1.5.

Номер разреза	координаты	номер датчика	Примечание
A1/22	49.90720 088.96675	1M	под Modis
		1ML	под ModisLandsat
		1B	температура воздуха
A2/22	49.91028 088.94305	2M	под Modis
		2ML	под ModisLandsat
		2B	температура воздуха
A3/22	49.85173 088.67678	3M	под Modis
		3ML	под ModisLandsat
		3B	температура воздуха
A4/22	49.96667 088.69635	4M	под Modis
		4ML	под ModisLandsat
		4B	температура воздуха
A5/22	50.02446 088.58833	5M	под Modis
		5B	температура воздуха
A6/22	49.91297 088.57922	6M	под Modis
		6ML	под ModisLandsat
		6B	температура воздуха
A7/23	49.99358 088.56201	7M	под Modis
		7ML	под ModisLandsat
A8/24	50.15606 087.83836	8M	под Modis
		8ML	под ModisLandsat
		8B	температура воздуха
A9/25	50.16241 087.86137	9M	под Modis
		9ML	под ModisLandsat
		9B	температура воздуха
A10/25	50.20907 088.02195	10M	под Modis
		10ML	под ModisLandsat

Рисунок 2.1.1.5 – Ведомость с информацией о данных, полученных с Термохрона

Также наземные данные занесены в таблицу. Пример данной таблицы изображен на рисунке 2.1.1.6.

15.07.2022	0:01	4,5
15.07.2022	4:01	2
15.07.2022	8:01	7,5
15.07.2022	12:01	23
15.07.2022	16:01	27
15.07.2022	20:01	12,5
16.07.2022	0:01	10
16.07.2022	4:01	8,5
16.07.2022	8:01	15,5
16.07.2022	12:01	33,5
16.07.2022	16:01	22
16.07.2022	20:01	18
17.07.2022	0:01	10,5
17.07.2022	4:01	7,5
17.07.2022	8:01	15,5
17.07.2022	12:01	17,5
17.07.2022	16:01	19,5
17.07.2022	20:01	13

Рисунок 2.1.1.6 – Таблица с наземными данными

В рамках модуля наземные данные необходимы для валидации их со значениями, полученными со спутников.

2.1.2 Информационные и функциональные потребности пользователей разрабатываемого модуля

Выделены следующие информационные потребности:

- 1) Получение спутниковых данных LANDSAT-8, Terra, Aqua/MODIS, а также их загрузка файла формата .XLSX через консоль.
- 2) Отображение спутниковых данных на карте, визуализация данных в виде графиков.
- 3) Выполнение валидации данных с использованием алгоритмов обработки. Анализ и вывод статистических показателей.

Функциональные потребности:

- 1) Ввод координат пользователем для выбора интересующего участка.
- 2) Возможность выбора источника спутниковых данных (MODIS, LANDSAT-8).
- 3) Получение данных с Google Earth Engine по указанным параметрам.
- 4) Применение алгоритмов MBE и RMSE для оценки точности данных.
- 5) Построение графиков и карт для анализа данных.
- 6) Подготовка выходного отчёта в формате PDF.

2.1.3 Методы работы с информационными объектами предметной области

2.1.3.1 Используемые математические модели

Для оценки расстояния между наборами спутниковых и наземных данных вычисляется среднеквадратическая ошибка RMSE[8].

Ошибка RMSE рассчитывается по формуле, представленной ниже:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (X_{\text{наземные},i} - X_{\text{спутниковые},i})^2}$$

Ошибка (сдвиг данных) MBE рассчитывается по формуле:

$$MBE = \frac{1}{N} \sum_{i=1}^N (X_{\text{наземные},i} - X_{\text{спутниковые},i})$$

Где:

N - количество измерений.

$X_{\text{наземные},i}$ - i-е наземное измерение.

$X_{\text{спутниковые},i}$ - i-е спутниковое измерение.

RMSE измеряет среднеквадратичное отклонение между соответствующими значениями наземных и спутниковых данных. Это позволяет оценить, насколько близки данные, полученные от спутников, к данным, собранным на земле.

Причины использования математической модели RMSE:

- 1) RMSE является мерой точности данных. Чем меньше значение RMSE, тем точнее соответствие между измеренными и предсказанными значениями.
- 2) При сравнении спутниковых данных с наземными значениями, RMSE помогает выявить систематические ошибки или отклонения в данных.

MBE оценивает среднюю разницу между наземными и спутниковыми данными, что позволяет определить смещение между двумя наборами данных.

Причины использования MBE:

- 1) MBE показывает, есть ли систематическое смещение в предсказаниях спутниковых данных по сравнению с наземными данными.
- 2) Если MBE отличен от нуля, это может указывать на необходимость коррекции спутниковых данных для уменьшения смещения.

Комбинированное использование RMSE и MBE позволяет оценить как точность, так и смещение данных между источниками, обеспечивая более полное представление о сопоставлении спутниковых и наземных данных.

2.1.3.2 Применяемые программные технологии, основанные на математических моделях

В языке программирования Python, который используется в разработке модуля существуют библиотеки NumPy, pandas и SciPy позволяющие использовать математические модели RMSE и MBE.

Пример использования библиотеки NumPy в коде для нахождения RMSE и MBE с комментариями изображен на рисунке 2.1.3.2.1.

```

1 usage
def calculate_rmse(ground_truth, satellite_data):
    # Расчет разницы между наземными и спутниковыми данными
    differences = ground_truth - satellite_data
    # Расчет среднеквадратичной ошибки
    rmse = np.sqrt(np.mean(differences ** 2))
    return rmse

1 usage
def calculate_mbe(ground_truth, satellite_data):
    # Расчет разницы между наземными и спутниковыми данными
    differences = ground_truth - satellite_data
    # Расчет средней ошибки (сдвига данных)
    mbe = np.mean(differences)
    return mbe

ground_truth_data = np.array([15, 20, 25, 30, 35])
satellite_data = np.array([14, 19, 24, 31, 34])

rmse_result = calculate_rmse(ground_truth_data, satellite_data)
mbe_result = calculate_mbe(ground_truth_data, satellite_data)

print(f"RMSE: {rmse_result}")
print(f"MBE: {mbe_result}")

```

Рисунок 2.1.3.2.1 – Использование библиотеки NumPy

В данном примере используется NumPy для расчёта данных, передаваемых в параметры методов. Алгоритм работы приведенного в пример кода изображен на рисунке 2.1.3.2.2.

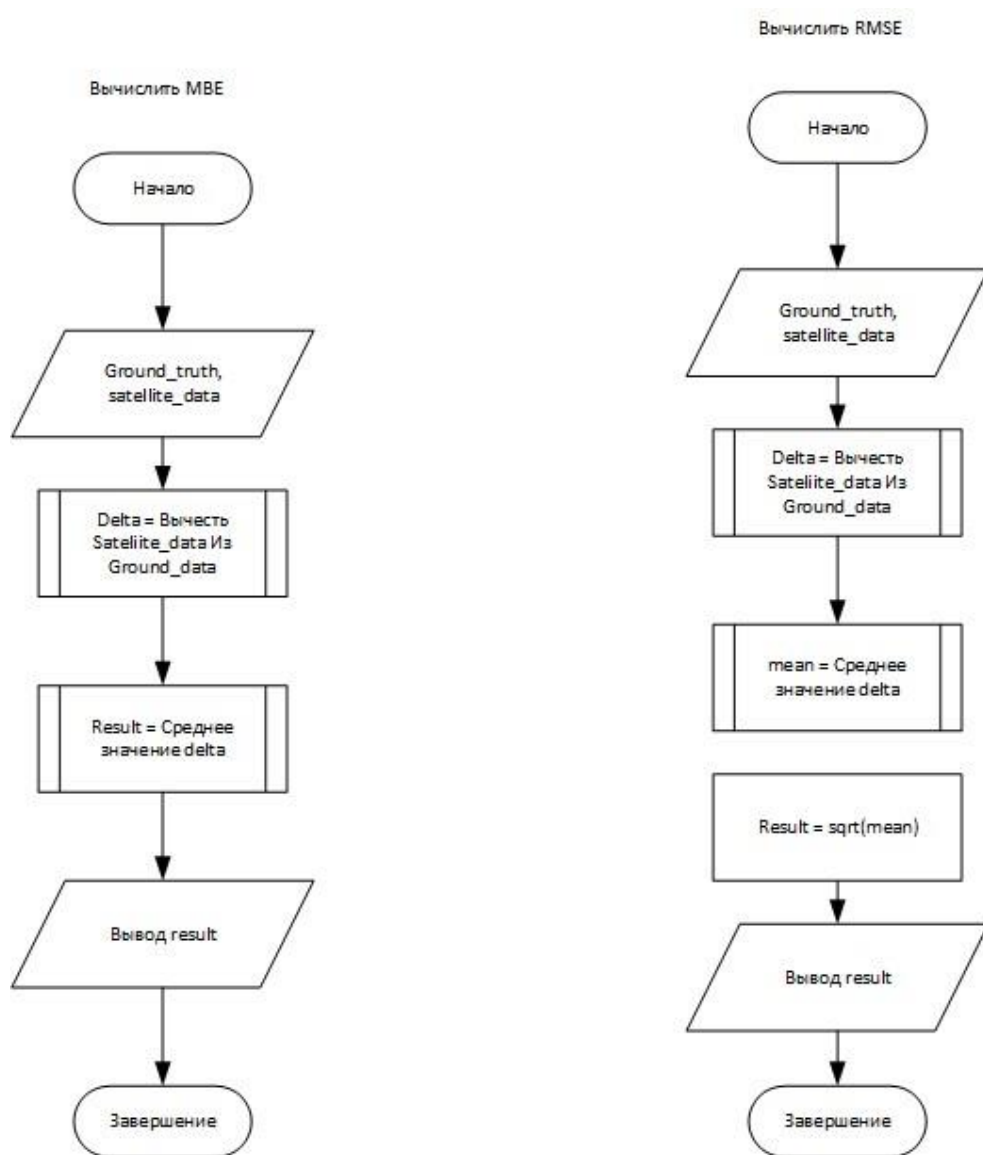


Рисунок 2.1.3.2.2 – Алгоритм расчёта статистических данных

Рисунок 2.1.3.2.3 позволяет более детально изучить алгоритм вычисления посредством представления кода в шести процессах, внутри которых происходят подпроцессы, такие как работа с массивами и математические вычисления.

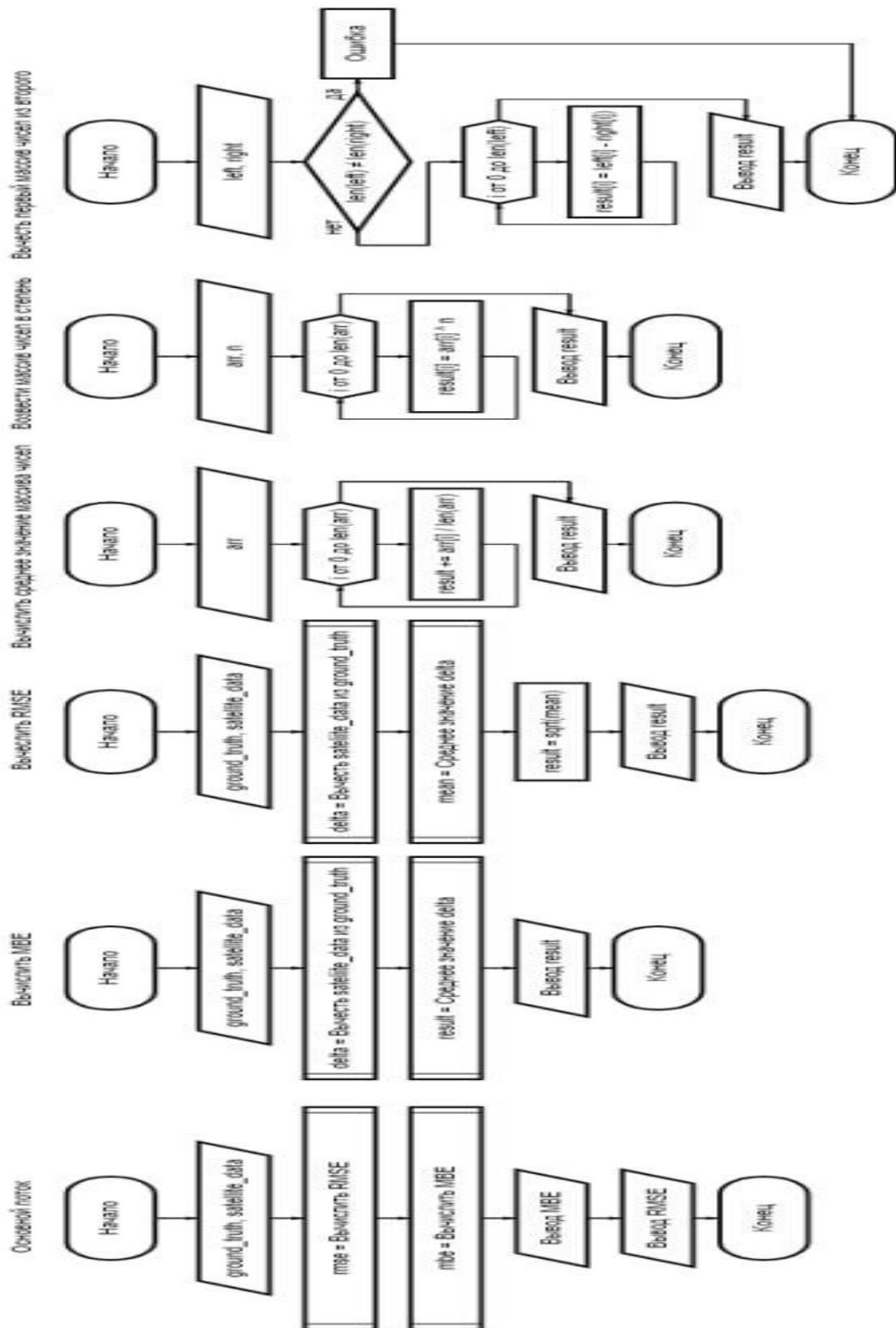


Рисунок 2.1.3.2.3 – Детальное описание алгоритма

Использование библиотек NumPy, pandas и SciPy в языке программирования Python значительно упрощает реализацию математических моделей, таких как RMSE и MBE, в

контексте разработки программных средств. Эти библиотеки предоставляют эффективные инструменты для обработки данных, вычислений и анализа, что особенно важно при работе с геопространственными и временными данными, характерными для спутниковых и наземных измерений.

Приведенный пример кода демонстрирует простоту использования NumPy для расчета RMSE и MBE, что делает их доступными и понятными даже для разработчиков, не специализирующихся в области математического моделирования. Это способствует повышению эффективности и качества разработки модуля для валидации данных спутников и наземных измерений.

2.1.4 Обзор существующих программных реализаций решения задачи

В ходе разработки были изучены аналогичные по функционалу программные средства. Все продукты базируются на открытых данных со спутников, отображая погоду с помощью интерактивных карт на страницах.

1. Веб-приложение MeteoBlue[9]

Данный сайт предоставляет данные о погоде, в том числе о температуре поверхности Земли. Также интерфейс данного программного средства имеет гибкие инструменты для работы с отображением необходимой информации, например, карта температуры почвы и выбор глубины измерения до 200 см. Интерфейс данного сайта изображен на рисунке 2.1.4.1.

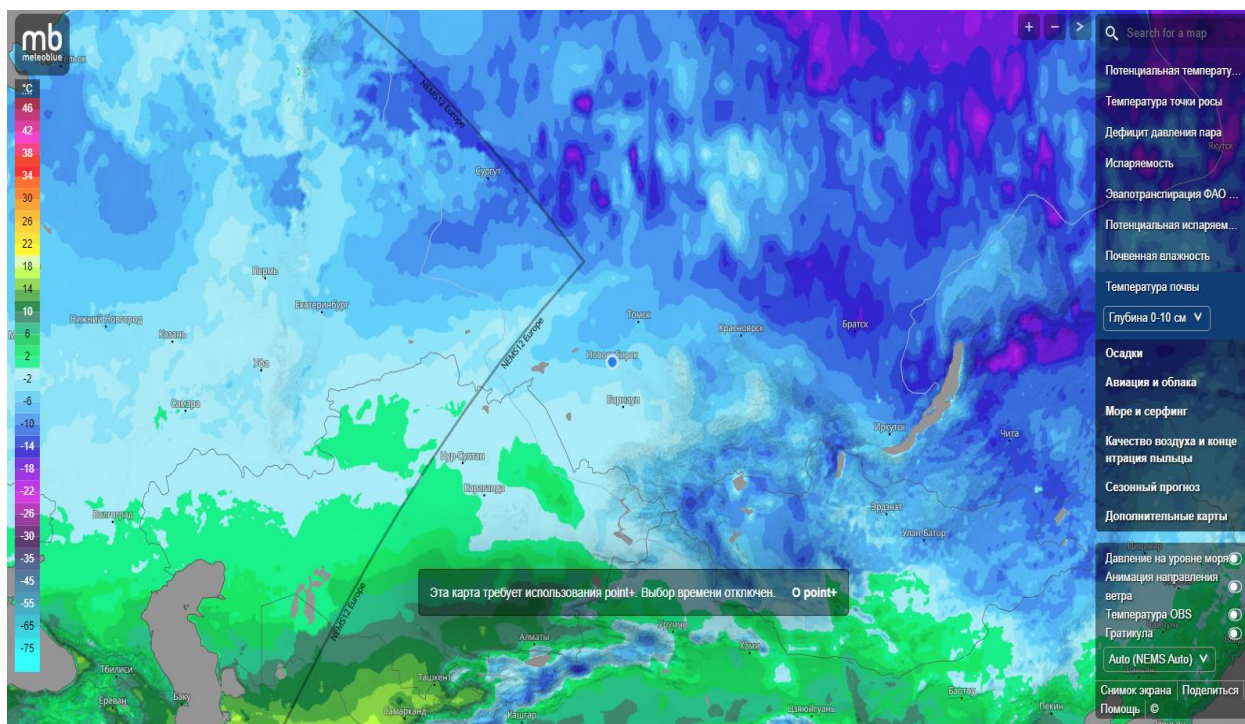


Рисунок 2.1.4.1 – Интерфейс сайта MeteoBlue

2. Веб-приложение Cosmos-Online[10]

Портал предоставляет разнообразные материалы, посвященные космосу, исследованию вселенной, а также деятельности Международной космической станции (МКС). Сайт предоставляет актуальные температурные данные со спутников в реальном времени, что показано на рисунке 2.1.4.2.

Этот ресурс обладает разнообразными настройками и инструментами для отображения карты, что делает его удобной платформой для любителей изучения космической области.

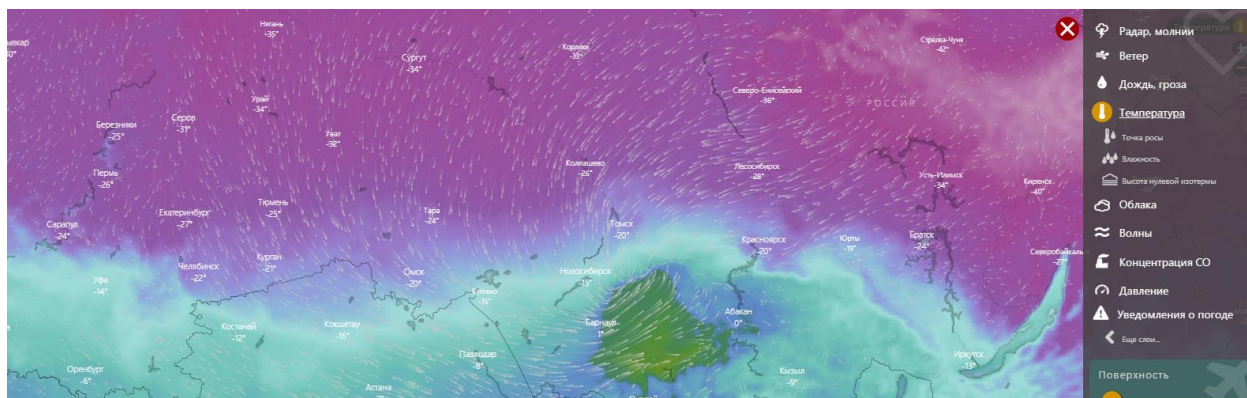


Рисунок 2.1.4.2 – Интерфейс сайта Cosmos-Online

Несмотря на достаточно широкий функционал приведенных выше программных средств, в них отсутствует возможность загрузки наземных измерений для валидации их с данными, полученными из космоса, а также выбор спутников для просмотра информации, что является приоритетной задачей в разрабатываемом приложении.

2.1.5 Концептуальное обоснование разработки

Разработка данного программного модуля основана на анализе проблем, с которыми сталкиваются исследователи при валидации данных как со спутников КА Landsat и Terra, Aqua/MODIS, так и с наземных измерений. В процессе исследования были выявлены различные недостатки, связанные с этим процессом.

Концепция разрабатываемого модуля опирается на идею интеграции спутниковых данных и наземных измерений с использованием методов визуализации и алгоритмов. Проект MODLAND NASA служит примером подобных усилий, начиная с 1998 года, когда начались исследования в области валидации данных. MODLAND продолжает активно заниматься вопросами валидации данных как со спутников, так и наземных измерений, поскольку это критически важно для понимания состояния Земли и ее изменений во времени.

Основная цель разработки данного программного модуля заключается в обеспечении точности и достоверности данных, полученных от спутников MODIS, путем сравнения их с результатами наземных измерений. Это позволит исследователям использовать данные MODIS с уверенностью в их качестве для множества научных и практических целей, включая мониторинг климатических изменений, изучение экосистем и оценку природных ресурсов.

Этот программный продукт называется модулем, а не полноценной программой, по ряду причин, основанных на его архитектуре и предназначении. Программный модуль представляет собой компонентную систему, состоящую из множества независимых скриптов, каждый из которых выполняет специализированные функции, такие как обработка данных, вычисление статистических показателей, визуализация результатов и создание отчетов. Это обеспечивает гибкость использования и возможность интеграции отдельных скриптов в более широкие системы или рабочие процессы. Модульный подход позволяет легко обновлять и расширять функциональность без необходимости полной переработки всего программного обеспечения, что способствует его адаптивности и долговечности в различных исследовательских и прикладных сценариях.

Разрабатываемый программный модуль представляет собой инструмент, созданный на основе уникальной концепции, что делает его более эффективным и перспективным решением в области валидации данных MODIS и наземных измерений.

2.2 Классы и характеристики пользователей

Для обеспечения точности и надежности температурных данных, получаемых от спутников КА LANDSAT и Terra/Aqua, а также для их валидации с использованием наземных измерений, было предложено разработать программное средство. Это программное средство будет полезно исследователям и экологическим организациям, занимающимся мониторингом климатических изменений.

1) Пользователь "Исследователь"

Социальные характеристики: Исследователи — это специалисты, которые проявляют интерес к анализу данных, полученных со спутников, в рамках научных исследований.

Мотивационная среда: их основной мотивацией является возможность проводить научные исследования и мониторинг температурных данных, а также отслеживать изменения температур для изучения климатических тенденций.

Навыки и умения: Исследователи обладают профессиональными навыками в проведении экологических исследований, анализе данных и использовании специализированных инструментов.

Требования к ПО: исследователи требуют высокой функциональности программного обеспечения для анализа и визуализации данных, а также доступа к базам данных и географической информации.

Задачи пользователя: их задачами являются сбор данных, анализ экологических показателей, мониторинг изменений и публикация научных результатов.

Рабочая среда: исследователи работают на стандартизированных персональных компьютерах в локальной сети.

2) Пользователь "Учебные заведения"

Социальные характеристики: Учебные заведения включают учеников и студентов, преподавателей и исследователей из образовательных учреждений.

Мотивационная среда: их главной мотивацией является образование, научные исследования и учебные цели.

Навыки и умения: Учебные заведения имеют разнообразные навыки - от учебных до исследовательских.

Требования к ПО: для учебных заведений требуются образовательные ресурсы, доступ к научным данным и материалам для обучения.

Задачи пользователя: Их задачами являются доступ к учебным материалам, участие в исследовательских проектах и обмен научными данными.

2.3 Функциональные требования

2.3.1 Определение функциональных возможностей программного модуля

Для предоставления полного функционала модуля пользователю были сформулированы требования, касающиеся того, что конкретно должна делать и предоставлять система. Ниже представлена таблица с описанием функциональных требований к системе.

Таблица 2.3.1 – Функциональные требования

Выбор файла с наземными данными из проводника	Пользователю открывается проводник для выбора файла в формате XLSX/XLS, содержащего наземные данные.
Ввод координат	Пользователь вводит координаты точки в формате "широта, долгота".
Ввод названия спутника	Пользователь выбирает спутник для запроса данных (MODIS или Landsat).
Получение и отображение результатов	Модуль получает данные о температуре как с наземных измерений, так и с выбранного спутника. Результаты отображаются в консоли, включая данные о температуре с различных временных точек и их соответствие данным наземных измерений.
Вычисление статистических показателей	Модуль вычисляет и отображает среднеквадратичное отклонение (RMSE) и среднюю абсолютную ошибку (MBE) для оценки точности данных спутника по сравнению с наземными измерениями.
Сохранение результатов	Пользователю предоставляется возможность сохранить полученные результаты, включая графики и таблицы с

	данными, в формате, удобном для последующего использования.
--	---

Данные требования необходимы для разработки и реализации программного средства. Наличие данных требований является обязательным и наиважнейшим для полного функционирования системы.

2.3.2 Описание прецедентов

В результате изучения предметной области выделена роль – Пользователь. Актер задействован в информационной системе.

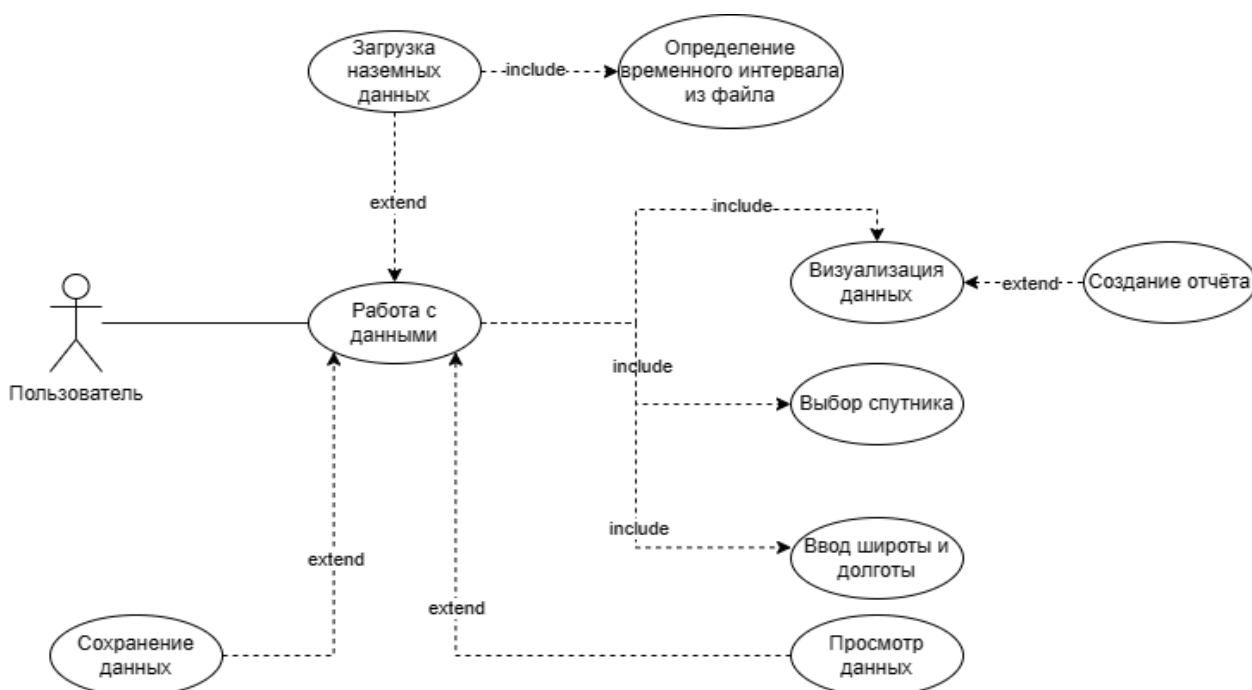


Рисунок 2.3.1 – Функциональные возможности программного средства

Описания прецедентов представлены в таблицах 2.3.2 - 2.3.8.

Таблица 2.3.2 – Описание прецедента «Загрузка наземных данных»

Название прецедента	Загрузка наземных данных
Исполнитель	Пользователь
Цель	Позволить пользователю выбрать файл с наземными данными.
Основной успешный сценарий	1. Пользователь запускает программный модуль.

	2. Пользователь выбирает файл из открывшегося окна
--	--

Таблица 2.3.3 – Описание прецедента «Определение временного интервала»

Название прецедента	Определение временного интервала
Исполнитель	Пользователь
Цель	Определить временной интервал на основе данных из Excel.
Основной успешный сценарий	<ol style="list-style-type: none"> 1. Программа получает данные из Excel. 2. Программа определяет первую и последнюю даты в данных. 3. Программа определяет временной интервал на основе первой и последней дат.

Таблица 2.3.4 – Описание прецедента «Ввод долготы и широты»

Название прецедента	Ввод долготы и широты
Исполнитель	Пользователь
Цель	Позволить пользователю ввести координаты долготы и широты.
Основной успешный сценарий	<ol style="list-style-type: none"> 1. Пользователь запускает программный модуль. 2. Пользователь выбирает файл с наземными данными. 3. Пользователь вводит координаты долготы и широты в консоль.

Таблица 2.3.5– Описание прецедента «Выбор спутника»

Название прецедента	Выбор спутника
---------------------	----------------

Исполнитель	Пользователь
Цель	Позволить пользователю выбрать спутник (Terra/Aqua MODIS (Day/Night) или Landsat)
Основной успешный сценарий	<ol style="list-style-type: none"> 1. Пользователь запускает программный модуль. 2. Пользователь выбирает файл с наземными данными. 3. Пользователь вводит координаты долготы и широты в консоль. 4. Пользователь выбирает спутник.

Таблица 2.3.6 – Описание прецедента «Визуализация данных»

Название прецедента	Визуализация данных
Исполнитель	Пользователь
Цель	Создать визуализацию данных исходя из предоставленных данных.
Основной успешный сценарий	<ol style="list-style-type: none"> 1. Программа получает данные о выбранных координатах. 2. Программа создает файл для просмотра карты. 3. Программа создает выходной файл Excel. 4. Программа создает отчет в PDF. 5. Программа создает графики на основе данных.

Таблица 2.3.7 – Описание прецедента «Сохранение данных»

Название прецедента	Сохранение данных
Исполнитель	Пользователь

Цель	Сохранить все визуализированные данные.
Основной успешный сценарий	<ol style="list-style-type: none"> 1. Пользователь запускает программный модуль. 2. Пользователь выбирает файл с наземными данными. 3. Пользователь вводит координаты долготы и широты в консоль. 4. Пользователь выбирает спутник. 5. Программа запрашивает необходимость сохранения полученных результатов. 6. В случае согласия пользователя, данные сохраняются.

Исходя из данного описания прецедентов, данное модуль является эффективным и удобным инструментом работы для разного вида пользователей, предоставляет широкий функционал и техническую поддержку в виде руководства пользователя.

2.4 Нефункциональные требования

В ходе проектирования программного модуля для валидации температурных данных КА Landsat и Terra/Aqua на основе наземных измерений были выявлены следующие нефункциональные требования:

1. Модуль должен обеспечивать высокую производительность и быстрый отклик, даже при обработке больших объемов данных и выполнении сложных вычислительных задач.
2. Система должна быть способной масштабироваться вертикально и горизонтально для обеспечения увеличения нагрузки без потери производительности.
3. Модуль должен обеспечивать безопасность данных и пользовательских сессий. Для работы модуля необходимо указать продукт на сервере поставщика спутниковых данных Google Earth Engine.
4. Модуль должен быть совместим с различными браузерами и устройствами, а также должен корректно функционировать на операционных системах Windows. Требуется также наличие установленного интерпретатора Python на устройстве пользователя.

5. Процесс установки и развертывания модуля должен быть документирован и автоматизирован для облегчения процесса установки на устройстве пользователя.

6. Должна предоставляться подробная документация, объясняющая пользователям процесс использования модуля, его функциональные возможности и особенности работы с ним.

3 ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ

3.1 Сравнительный анализ имеющихся возможностей по выбору средств разработки

Сравнительный анализ сред разработки необходим для выбора оптимальных инструментов при разработке модуля. Рассмотрим несколько популярных сред разработки для Python, а также их преимущества и недостатки.

1. Visual Studio Code (VSCode):

Преимущества: бесплатный, легковесный, сильное сообщество, множество расширений.

Недостатки: может потребовать дополнительной настройки для работы.

2. Jupyter Notebook:

Преимущества: интерактивность, визуализация данных, гибкость.

Недостатки: сложность отладки, зависимость от последовательности выполнения.

3. Spyder:

Преимущества: научные вычисления, удобство использования.

Недостатки: ограниченность возможностей для разработки приложений, ограниченность в расширениях.

Каждая из рассмотренных сред разработки имеет свои преимущества и недостатки. Выбор оптимальной среды зависит от конкретных потребностей проекта. Visual Studio Code подходит для разнообразных задач, Jupyter Notebook отличен для анализа данных, а Spyder специализируется на научных вычислениях.

В процессе сравнительного анализа сред разработки для Python выявлены их основные преимущества и недостатки. Visual Studio Code (VSCode) предоставляет удобную и расширяемую среду с отличной поддержкой Python, Jupyter Notebook подходит для интерактивного анализа данных, а Spyder специализируется на научных вычислениях. Однако каждая из них имеет свои ограничения и выявленные преимущества не подходят под данный проект.

3.2 Характеристика выбранных программных сред и средств

PyCharm — это интегрированная среда разработки (IDE) для языка программирования Python, предоставляющая множество инструментов и функциональности для эффективной разработки. Она обладает возможностью

автоматической подсветки синтаксиса, автодополнения кода, интеграции с системами контроля версий и другими функциями, что делает ее мощным инструментом для разработки приложений.[12]

Python (Py): Python - высокоуровневый, интерпретируемый язык программирования, используемый в проекте для разработки -приложений. PyCharm обеспечивает удобную среду разработки для Python, включая поддержку отладки, автодополнение, инструменты анализа кода и управление зависимостями проекта.

Google Earth Engine API (GEE API): GEE API — это мощный инструмент для работы с геопространственными данными, предоставляемый компанией Google. В проекте используется для получения данных с Google Earth Engine и их последующего анализа. PyCharm обеспечивает удобную интеграцию и разработку с использованием Google Earth Engine API, что делает процесс работы с геопространственными данными более эффективным.[13]

Библиотеки:

earthengine-api: библиотека для работы с Google Earth Engine API, позволяющая получать и анализировать геопространственные данные.

pandas: библиотека для работы с данными, предоставляющая высокоуровневые структуры данных и операции для их манипуляции.

matplotlib: библиотека для создания статических, интерактивных и анимированных графиков в Python.

folium: библиотека для визуализации данных на интерактивных картах с использованием Leaflet.js.

reportlab: библиотека для создания динамических PDF-документов в Python.

openpyxl: библиотека для работы с файлами Excel в формате .XLS.

4 АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

4.1 Этапы реализации модуля

Разработка программного модуля для валидации температурных данных КА Landsat и Terra/Aqua на основе наземных измерений является многоэтапным процессом, требующим внимательного планирования и последовательного выполнения шагов. Ниже представлены ключевые этапы реализации проекта:

1. Проведение предварительного анализа: на данном этапе осуществляется детальное изучение предметной области, выявление требований к функциональности модуля и определение основных источников данных.

2. Проектирование функциональности: определяются основные функции модуля, включая загрузку и обработку данных, выбор спутника, ввод координат и сохранение результатов.

3. Реализация бизнес-логики: на этом этапе осуществляется написание программного кода, отвечающего за обработку данных, их валидацию и выполнение основных функций модуля.

4. Тестирование: модуль проходит комплексное тестирование, включающее в себя функциональное, безопасное и производительное тестирование, с целью выявления и исправления возможных ошибок.

5. Разворачивание и поддержка: после успешного прохождения тестирования модуль разворачивается в рабочей среде и обеспечивается последующая поддержка, включая внесение изменений и решение выявленных проблем.

Эффективная реализация модуля требует системного подхода, начиная с анализа требований и проектирования, и заканчивая тестированием и разворачиванием. Каждый из этих этапов играет важную роль в обеспечении функциональности, надежности и удобства использования разрабатываемого модуля.

4.2 Пользовательский интерфейс модуля

4.2.1 Взаимодействие пользователей с модулем

Для модуля разработана диаграмма, отображающая полное взаимодействие пользователя с модулем. Она изображена на рисунке 4.2.1.1.

1. Пользователь запускает программу:
Пользователь открывает модуль.

2. Пользователь выбирает наземные данные в формате .XLS:

Система принимает только формат файла, который она сможет прочесть, в ином случае, возвращается ошибка.

3. Ввод координат

Пользователь вводит координаты, соответствующие проекции WGS84, иные будут возвращать ошибку.

4. Ввод спутника:

Пользователь указывает источник спутниковых данных. Если введено неожиданное для системы значение, она запрашивает спутник снова.

5. Вывод совпавших строк с временным промежутком.

В консоль выводятся все значения из двух массивов данных, которые совпали по дате, а также у которых разница составляет введенный пользователем временной промежуток.

6. Показ карты с выбранными параметрами.

Система открывает браузер, в котором изображена карта, показывающая данные из продукта GEE MODIS/006/MYD11A1, а также сохраняет преобразованный HTML-файл в папку Maps.

7. Сохранение графиков.

Система сохранит графики в формате .PNG в папку Graphics.

8. Сохранение таблиц

Система сохранит все полученные значения в таблицы в папке Tables.

9. Сохранение отчёта.

Система сохранит все полученные данные в один PDF-файл.

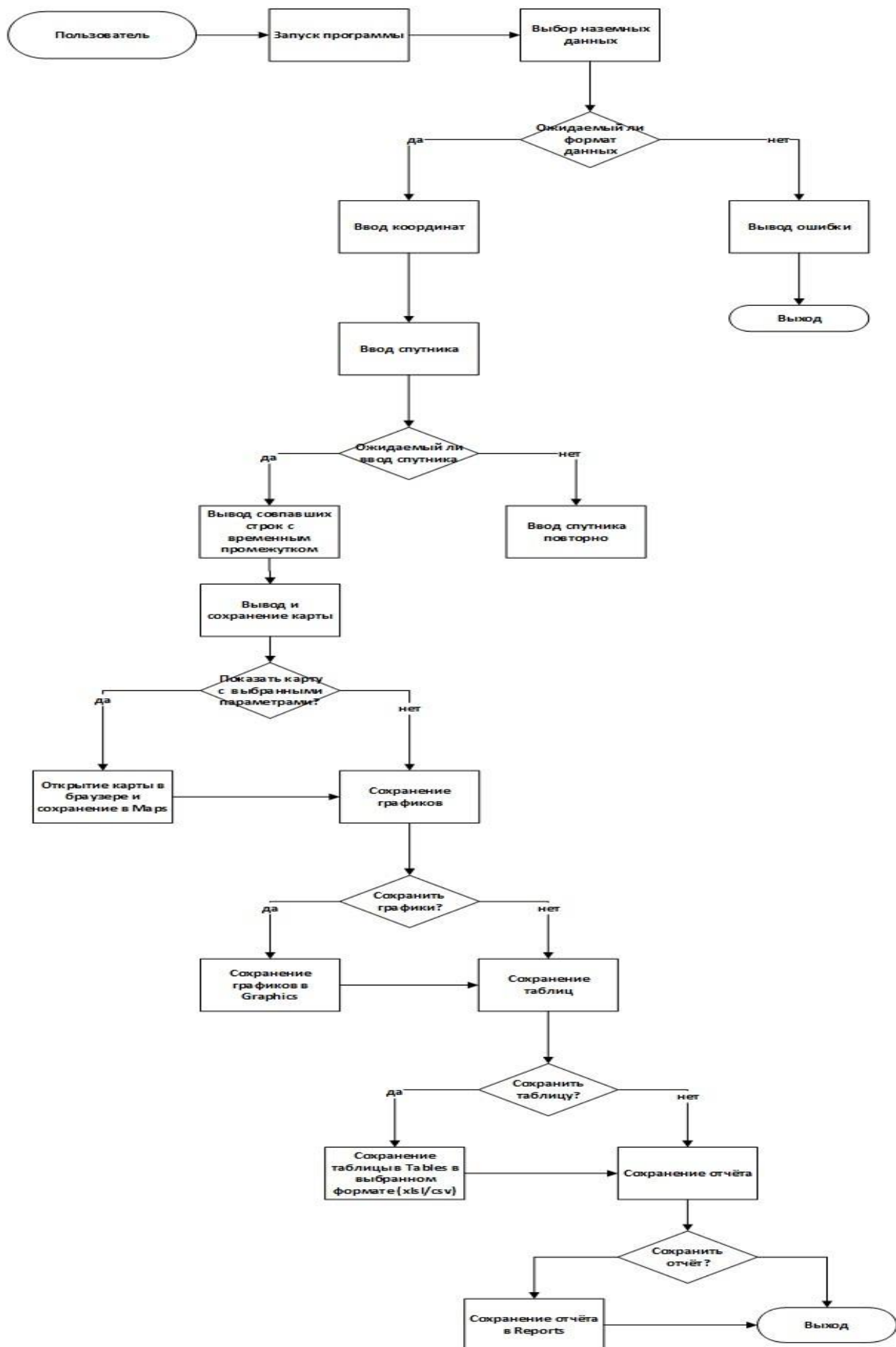


Рисунок 4.2.1.1 – Взаимодействие пользователя с модулем

Подробный и последовательный план взаимодействия с модулем обеспечивает удобство и простоту использования для пользователя. Каждый этап процесса явно структурирован, что помогает пользователям легко понять, как использовать программу, даже если они не имеют опыта работы с подобными приложениями ранее. Например, открытие модуля и последующий выбор данных в формате .XLS являются стандартными шагами, которые понятны для большинства пользователей.

Кроме того, проверки на корректность ввода данных, такие как формат файла или координат, помогают предотвратить ошибки и уменьшить вероятность возникновения проблем в процессе работы. Это важно в случае работы с большим объемом данных, где даже небольшие ошибки могут привести к искажениям результатов.

Система также предоставляет пользователю выбор источника спутниковых данных и отображает совпадающие значения с возможностью дополнительного анализа. После этого пользователь может выбрать отображение карты или сохранение графиков, таблиц и отчета, что позволяет эффективно организовать и сохранить результаты для последующего анализа и исследования.

Такой подход к взаимодействию с модулем упрощает процесс работы с данными, уменьшает необходимость вручную выполнять рутинные операции и обеспечивает надежное сохранение результатов, что в свою очередь способствует более качественному исследованию и анализу данных.

4.2.2 Проектирование пользовательских сценариев

Исследователь данных (Data Researcher)

Сценарий 1: Анализ данных со спутника Terra, Aqua/MODIS.

- 1) Исследователь запускает программный модуль.
- 2) Выбирает файл с наземными данными.
- 3) Указывает координаты для анализа данных.
- 4) Выбирает источник данных как MODIS.
- 5) Система обрабатывает запрос, а также отправляет запрос в GEE.
- 6) Программное средство анализирует данные и выводит визуализированные данные.
- 7) Исследователь может сохранить выходные данные в форматах .gif, .png, .XLS.
- 8) Выходит из системы.

Диаграмма, описывающая данный сценарий представлена на рисунке 4.2.2.1

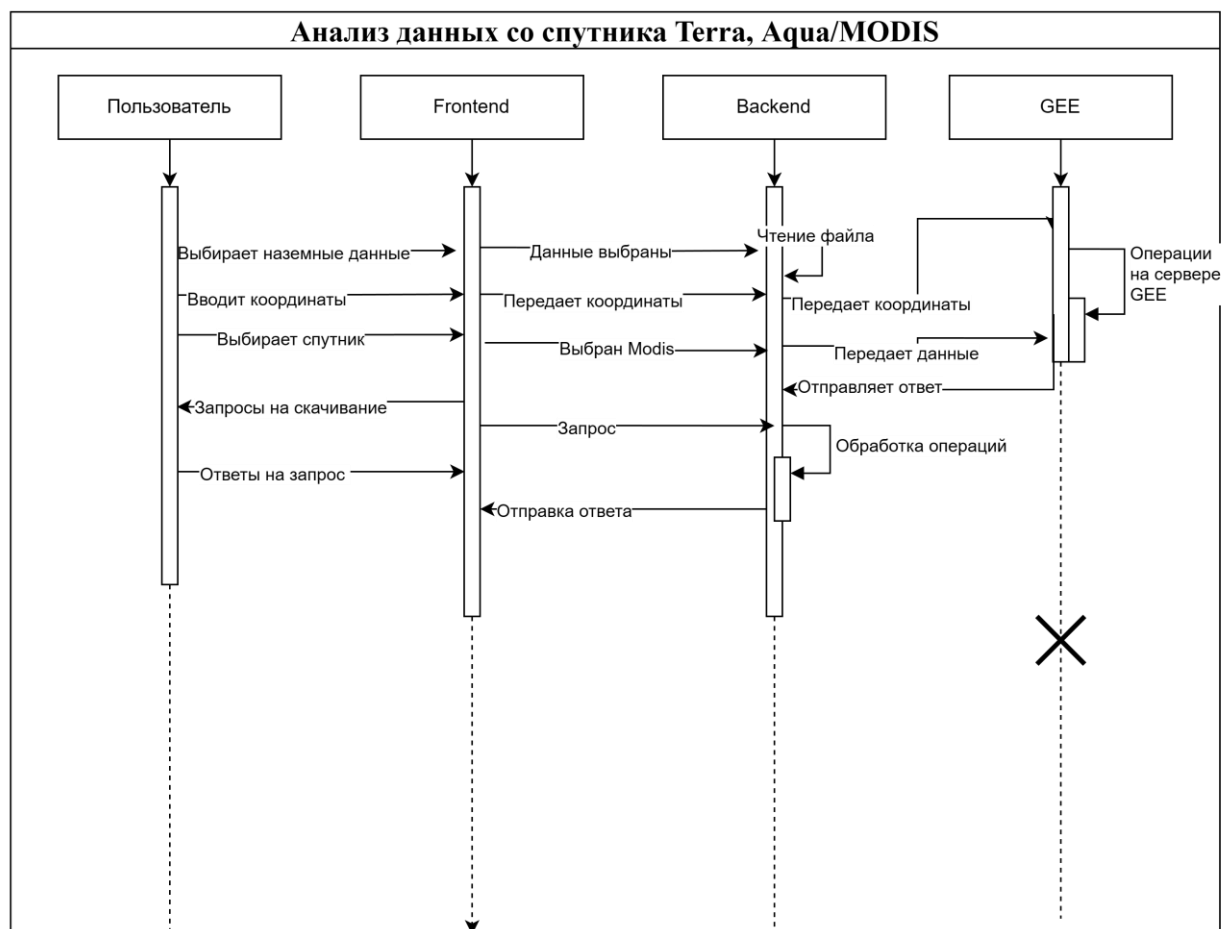


Рисунок 4.2.2.1– Сценарий использования модуля с использованием MODIS

Сценарий 2: Анализ данных со спутника Landsat.

- 1) Исследователь запускает программный модуль.
- 2) Выбирает файл с наземными данными.
- 3) Указывает координаты для анализа данных.
- 4) Выбирает источник данных как спутник Landsat.
- 5) Система обрабатывает запрос, а также отправляет запрос в GEE.
- 6) Программное средство анализирует данные и выводит визуализированные данные.
- 7) Исследователь может сохранить выходные данные в форматах .gif, .png, .XLS.
- 8) Выходит из системы.

Диаграмма, описывающая данный сценарий представлена на рисунке 4.2.2.2

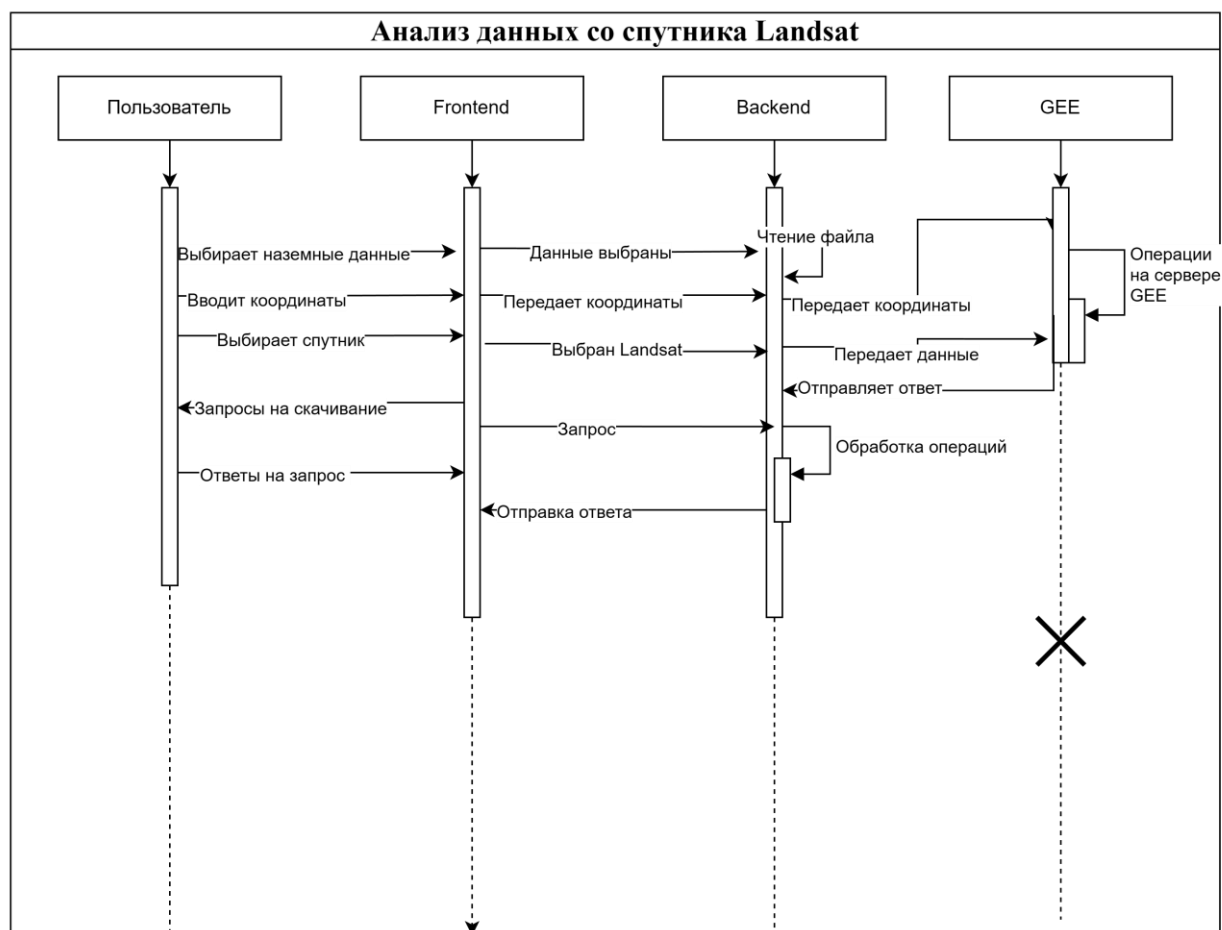


Рисунок 4.2.2.2– Сценарий использования модуля с использованием Landsat

В результате проектирования пользовательских сценариев для исследователей данных (Data Researchers), были выделены два ключевых сценария использования модуля для валидации температурных данных.

Первый сценарий описывает процесс анализа данных со спутников Terra, Aqua/MODIS. Пользователь может провести анализ, выбрав параметры, такие как координаты и источник данных. Результаты анализа визуализируются в виде графиков и прочих форматов визуализации, с возможностью сохранения в различных форматах.

Второй сценарий представляет валидацию данных со спутника LANDSAT. Пользователь загружает наземные данные, проводит анализ и получает результаты в удобном для скачивания виде.

Оба сценария обеспечивают пользователям интуитивно понятный и удобный интерфейс для работы с модулем, а диаграммы UML визуализируют шаги выполнения каждого сценария, обеспечивая ясное представление о процессе взаимодействия пользователей с системой.

4.2.3 Определение операций пользователей

В рамках прецедентов пользователи могут выполнять следующие операции:

1. Выбор параметров анализа:

Указание координат для анализа данных.

Выбор источника данных (спутник Terra, Aqua/MODIS или LANDSAT).

2. Загрузка наземных данных:

Выбор файла в формате .XLS, который содержит наземные данные.

3. Выполнение анализа:

Инициирование запроса на анализ данных. Передача запроса на сервер Google Earth Engine для получения соответствующих данных.

4. Просмотр результатов:

Визуализация результатов анализа в виде графиков, моды, среднего значения и тренда данных.

5. Возможность сохранения результатов в форматах .gif, .png, .XLS, .pdf.

6. Выход из системы:

Завершение сеанса работы модуля.

Эти операции предоставляют пользователям гибкость и контроль над процессом работы с модулем, обеспечивая им возможность проводить анализ и валидацию температурных данных в соответствии с их потребностями и целями исследования.

4.2.4 Составление функциональных блоков

На данном этапе выделяются следующие функциональные блоки программы:

1. Модуль для работы с данными MODIS и Landsat

Данный модуль содержит два класса: Aqua_Manager и Landsat_Manager. Они отвечают за получение и пре-обработку входных данных. Внутри содержатся функции для получения коллекции изображений. Классы отправляют запрос на получение определенных продуктов из GEE и конвертируют данные со спутника в необходимый для дальнейшего анализа формат.

2. Модуль для работы с Excel

Данный модуль необходим для работы с файлами Excel, операций считывания данных, а также оформлений данных в более удобные для дальнейшей работы структуры. Именно в этом модуле происходит сбор наземных данных.

3. Модуль для сравнения временных и температурных данных

Модуль содержит алгоритм сравнения двух массивов данных (спутниковые и наземные), преобразования их, а также создания выборки только из необходимых строк с временными интервалами.

4. Модуль для подготовки визуализации

Модуль содержит классы и функции, обеспечивающие визуализацию всех полученных значений на временных промежутках. Визуализация включает в себя отображение карты, графиков, таблиц, отчетов.

5. Модуль для расчёта RSME и MBE

Модуль содержит функцию расчёта RSME и MBE. На вход функции подаются преобразованные спутниковые и наземные данные.

6. Модуль для аутентификации

Модуль содержит функцию для доступа к зарегистрированным продуктам GEE.

7. Модуль для работы с выходными данными

Модуль использует функции других модулей, преобразуя значения в необходимые для составления выходных данных.

8. Модуль для работы с входными данными

Модуль обеспечивает диалог с пользователем, работает с введенными координатами, открывает Проводник Windows для выбора файла, позволяет выбрать спутник.

Модули вместе с классами и функциями, представлены на рисунке 4.2.4

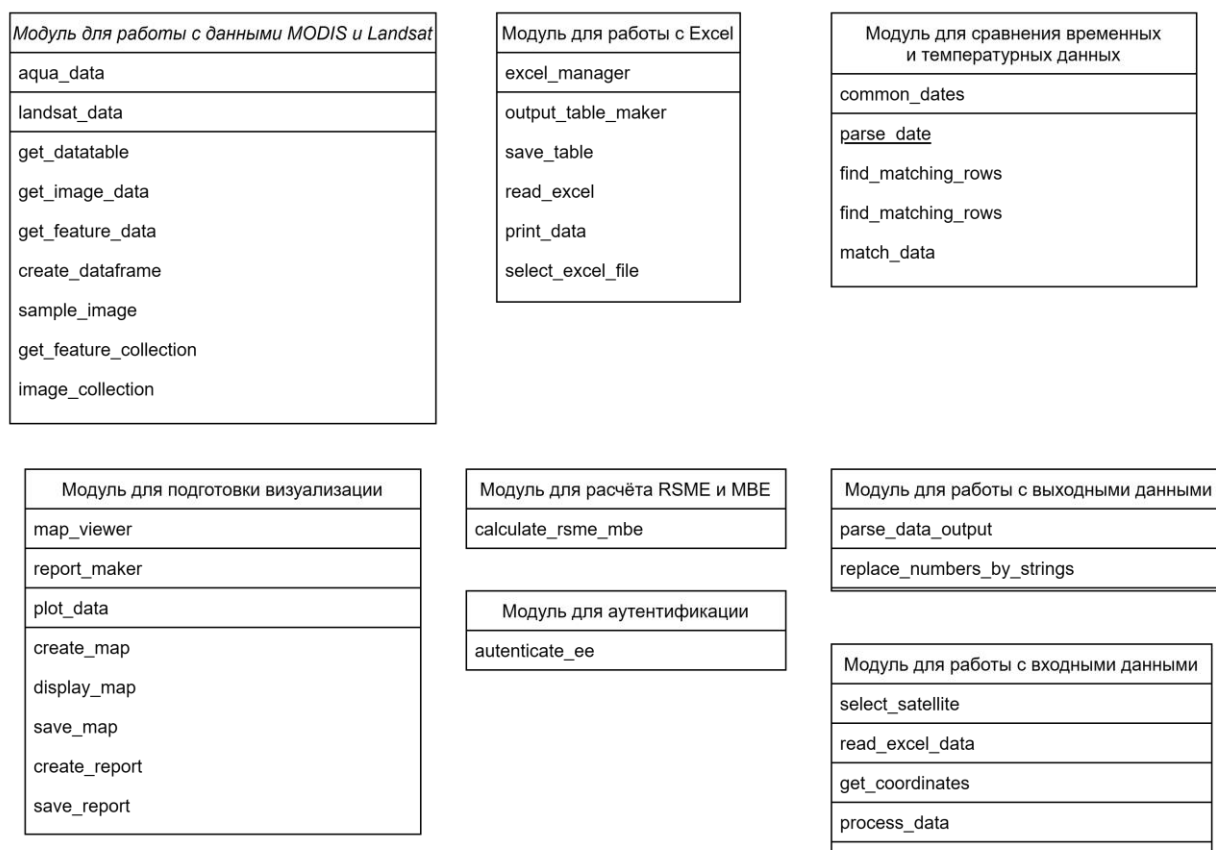


Рисунок 4.2.4 – Модули в программном средстве

Представленные функциональные блоки программы обеспечивают комплексную обработку данных, начиная с получения и предварительной обработки информации со спутников MODIS и Landsat, и заканчивая визуализацией результатов и формированием отчетности. Эта структура позволяет программному средству эффективно взаимодействовать с пользователями, обеспечивая удобство использования, точность анализа и сохранность результатов для последующего исследования.

4.2.5 Проектирование структуры экранов модуля и схемы навигации

На этапе проектирования структуры экранов модуля формируется навигационная схема, обеспечивающая логичное и удобное перемещение пользователя внутри консольного диалога. Для этого была разработана диаграмма навигации, которая является визуальным отображением механизма распределения функций и задач.

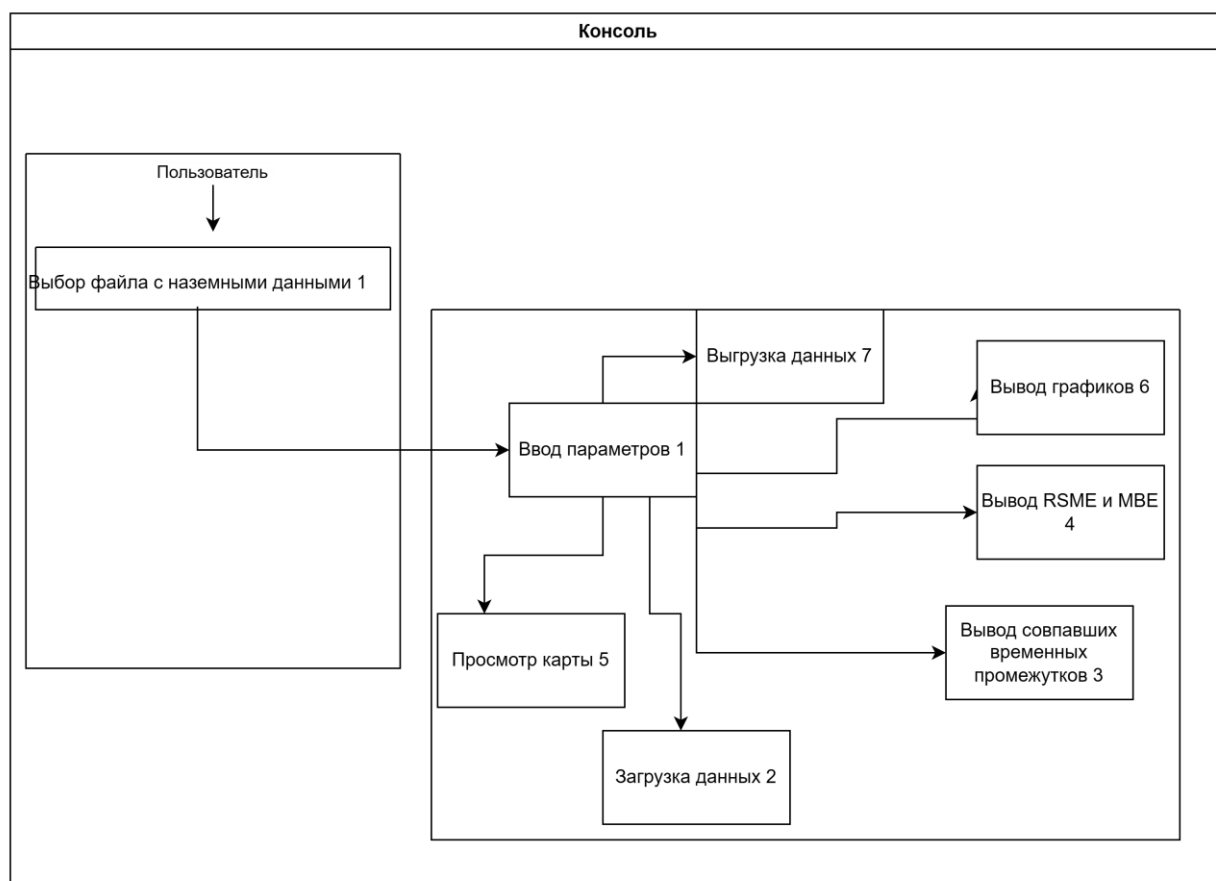


Рисунок 4.2.5.1 – Структура элементов модуля

Таким образом, программное средство имеет собственную удобную схему навигации, благодаря которой пользователь может быстро перемещаться по ресурсу.

4.2.6 Низкоуровневое проектирование

В процессе низкоуровневого проектирования консольного интерфейса программного модуля для валидации температурных данных КА Landsat и Terra/Aqua были определены следующие шаги взаимодействия с пользователем:

1. Запуск программы: пользователь запускает модуль, открывая консольное окно или исполняемый файл программы.
2. Выбор наземных данных: программа ожидает от пользователя указания пути к файлу наземных данных в формате .XLS. После ввода пользователем пути, система проверяет его корректность и продолжает выполнение программы. В случае ошибки пользователю выводится соответствующее сообщение.

3. Ввод координат: пользователь вводит координаты в соответствии с проекцией WGS84 через консольный интерфейс. Программа осуществляет проверку введенных данных на соответствие формату и правильность значений.

4. Ввод спутника: после ввода координат пользователь указывает источник спутниковых данных также через консоль. Программа проверяет корректность ввода и при необходимости запрашивает у пользователя повторный ввод.

5. Вывод совпадающих данных: после завершения ввода пользовательских данных программа выводит в консоль все значения из двух массивов данных, которые совпали по дате и времени, а также имеют разницу во времени, которую вводит пользователь.

6. Дополнительные действия: По запросу пользователя программа может выполнить дополнительные действия, такие как отображение карты с выбранными параметрами, сохранение графиков, таблиц или отчета в соответствующие файлы и директории.

Консольный интерфейс обеспечивает простоту использования модуля и удобство взаимодействия с пользователем, позволяя ему управлять процессом работы программы через стандартный текстовый ввод и вывод.

4.3 Входные и выходные данные

1) Входные данные:

1. Файлы формата .XLSX с наземными данными: входные данные предоставляются в виде файлов формата XLSX, содержащих информацию о дне, времени и показателях с приборов. Структура файла строго определена, включая обязательные поля и формат данных. Эти файлы загружаются и используются в процессе валидации. Пример файла с наземными данными изображен на рисунке 4.3.1

15.07.2022	0:01	3,5
15.07.2022	1:01	2,5
15.07.2022	2:01	2
15.07.2022	3:01	1
15.07.2022	4:01	1
15.07.2022	5:01	0,5
15.07.2022	6:01	0,5
15.07.2022	7:01	1,5
15.07.2022	8:01	6,5
15.07.2022	9:01	12,5
15.07.2022	10:01	17,5
15.07.2022	11:01	20
15.07.2022	12:01	23
15.07.2022	13:01	27
15.07.2022	14:01	30,5
15.07.2022	15:01	31,5
15.07.2022	16:01	27
15.07.2022	17:01	23
15.07.2022	18:01	20,5
15.07.2022	19:01	17
15.07.2022	20:01	13,5
15.07.2022	21:01	12
15.07.2022	22:01	11
15.07.2022	23:01	10,5
16.07.2022	0:01	10,5
16.07.2022	1:01	10
16.07.2022	2:01	9,5
16.07.2022	3:01	9,5

Рисунок 4.3.1 – Формат наземных данных

2. Спутниковые данные: в модуле используется два источника спутниковых данных – Terra, Aqua/MODIS и Landsat. Каждый спутник имеет свой продукт, благодаря которому появляется возможность использовать данные. В программном модуле для MODIS используется продукт «MYD11A1.061 Aqua Land Surface Temperature and Emissivity Daily Global 1km» MODIS/061/MYD11A1, «MOD11A1.061 Terra Land Surface Temperature and Emissivity Daily Global 1km» Terra/MODIS, а также каналы, которые используются для получения LANDSAT. Для дневного времени используется LANDSAT_Day_1km для получения значений LANDSAT, для получения времени Day_view_time, а для ночного другой канал - LANDSAT_Night_1km и Night_view_time. Свойство system:time_start используется для получения временной информации о снимках или измерениях, сделанных в различные моменты времени при съемке спутниковых данных. Это позволяет определять, когда конкретное изображение было зафиксировано или когда измерение было проведено. Такая временная информация может быть использована для анализа временных трендов, создания анимации, фильтрации данных по времени и других операций, связанных с временной динамикой явлений на Земле. Схема использования Aqua/MODIS изображена на рисунке 4.3.2.

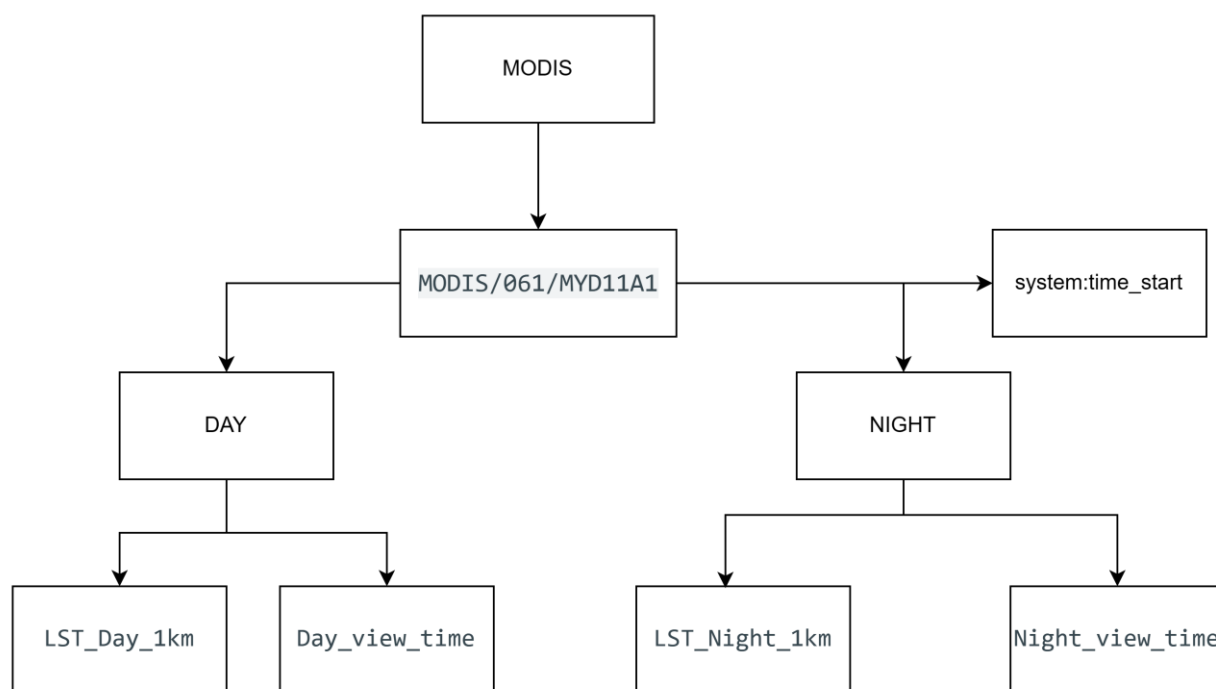


Рисунок 4.3.2 – Схема использования MODIS в программном модуле

Для спутника Landsat используется продукт LANDSAT/LC08/C02/T1_L2. Схема использования Landsat представлена на рисунке 4.3.3.

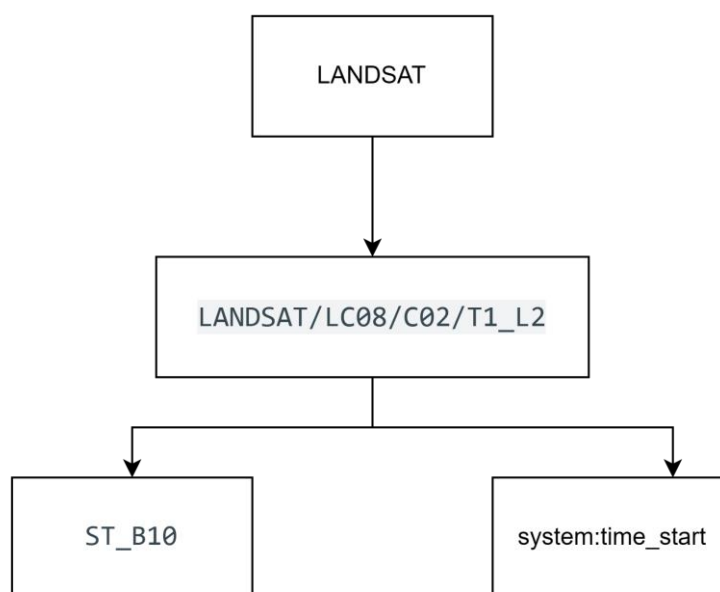


Рисунок 4.3.2 - Схема использования Landsat в программном модуле

Для создания таблиц также есть другие каналы спутниковых данных, такие как коэффициента пропускания, нисходящее сияние, стандартное отклонение и прочее.

2) Выходные данные:

Выходными данными являются таблицы Excel, графики, карты, а также PDF-файлы.

1) Таблицы Excel как выходные данные программного модуля представляют собой набор данных, собранных из наземных и спутниковых источников данных. Столбцы с продуктами, общими датами, значениями генерируются автоматически. Пример подобной таблицы представлен на рисунке 4.3.3

Clear_day	cover_night	view_any	view_time	Emis_31	Emis_32	T_Day_1kt	Night_1kt	view_aht	view_t	C_Dat_Nig	date	source	Common Date	value	Ground Value	Ground Dat
9.90	10.00	29.00	13:18:00	0.98	0.99	31.23	15.33	33.00	02:48:00	65	0 2022-07-20	Satellite - modis Day	2022-08-14 13:06:00	41.95	15,3	2022-07-15 02:4
4.76	9.81	-42.00	12:24:00	0.99	0.99	34.27	13.59	-39.00	00:18:00	65	0 2022-07-21	Ground	2022-08-14 12:54:00	37,9	34,2	2022-07-20 12:2
3.96	10.65	12.00	13:06:00	0.98	0.99	32.41	13.89	18.00	02:36:00	65	0 2022-07-22	Satellite - modis Night	2022-08-14 02:36:00	10.93	13,5	2022-07-20 00:1
20.00	19.86	-52.00	12:12:00	0.99	0.98	36.29	14.83	-50.00	00:06:00	65	0 2022-07-23	Ground	2022-08-14 02:24:00	10,3	32,4	2022-07-21 13:0
10.00	9.79	-7.00	12:54:00	0.98	0.99	45.61	15.97	0.00	02:24:00	0	0 2022-07-24	Satellite - modis Day	2022-08-19 13:24:00	31.97	13,9	2022-07-21 02:5
10.00	20.00	39.00	13:24:00	0.98	0.98	36.43	11.51	-63.00	01:18:00	0	65 2022-08-03	Ground	2022-08-19 13:12:00	34,6	36,2	2022-07-22 12:1
17.75	10.00	-34.00	12:30:00	0.98	0.99	27.83	14.29	-29.00	02:00:00	65	0 2022-08-04	Satellite - modis Night	2022-08-19 02:54:00	11.77	14,8	2022-07-22 00:0
10.00	19.94	24.00	13:12:00	0.98	0.99	41.03	13.45	29.00	02:42:00	65	0 2022-08-05	Ground	2022-08-19 02:42:00	6,7	45,6	2022-07-23 12:5
9.99	10.00	-46.00	12:18:00	0.98	0.99	33.91	13.65	-43.00	00:12:00	65	0 2022-08-06	Satellite - modis Day	2022-09-01 12:48:00	43.93	16	2022-07-23 02:2
10.00	9.86	-12.00	12:48:00	0.98	0.99	38.45	13.77	-6.00	02:18:00	0	0 2022-08-09	Ground	2022-09-01 12:48:00	43,9	36,4	2022-07-24 13:2
14.53	9.94	29.00	13:18:00	0.98	0.99	38.71	12.05	34.00	02:48:00	0	0 2022-08-12	Satellite - modis Night	2022-09-01 02:18:00	14.99	11,5	2022-07-24 01:1
10.00	16.76	-42.00	12:24:00	0.98	0.99	37.39	14.83	-39.00	00:18:00	0	0 2022-08-13	Ground	2022-09-01 02:18:00	15	27,8	2022-08-03 12:5
9.93	9.87	13.00	13:06:00	0.98	0.99	41.95	10.93	19.00	02:36:00	0	0 2022-08-14	Satellite - modis Day	2022-09-03 12:36:00	39.25	14,3	2022-08-03 02:0
10.00	10.00	-6.00	12:54:00	0.98	0.99	37.91	10.33	0.00	02:24:00	65	0 2022-08-16	Ground	2022-09-03 12:36:00	39,2	41	2022-08-04 13:1
9.99	10.00	34.00	13:24:00	0.99	0.99	31.97	11.77	38.00	02:54:00	0	65 2022-08-19	Satellite - modis Night	2022-09-03 02:06:00	8.97	13,4	2022-08-04 02:4
10.00	10.00	19.00	13:12:00	0.98	0.99	34.69	6.73	24.00	02:42:00	0	0 2022-08-21	Ground	2022-09-03 02:06:00	9	33,9	2022-08-05 12:1
20.00	9.93	50.00	13:42:00	0.98	0.99	33.57	9.99	51.00	03:12:00	65	0 2022-08-24	Satellite - modis Day	2022-09-18 11:48:00	27.89	13,7	2022-08-05 00:1
10.00	10.00	-18.00	12:42:00	0.98	0.99	40.59	15.21	-12.00	02:18:00	0	65 2022-08-25	Ground	2022-09-18 11:48:00	27,8	38,4	2022-08-06 12:4
10.00	10.00	24.00	13:12:00	0.98	0.99	39.21	13.39	29.00	02:48:00	0	0 2022-08-28	Satellite - modis Night	2022-09-18 03:00:00	9.85	13,8	2022-08-06 02:1
9.93	14.59	-45.00	12:18:00	0.98	0.98	40.17	10.99	61.00	03:30:00	0	65 2022-08-29	Ground	2022-09-18 03:00:00	9,9	38,7	2022-08-09 13:1
9.87	10.00	7.00	13:00:00	0.98	0.99	40.37	12.49	13.00	02:36:00	0	0 2022-08-30	Satellite - modis Day	2022-09-19 12:30:00	33.55	12	2022-08-09 02:4
6.27	9.96	-12.00	12:48:00	0.98	0.99	43.93	14.99	-6.00	02:18:00	65	65 2022-09-01	Ground	2022-09-19 12:30:00	33,5	37,3	2022-08-12 12:2
9.97	10.00	-29.00	12:36:00	0.98	0.99	39.25	8.97	-24.00	02:06:00	0	65 2022-09-03	Satellite - modis Night	2022-09-19 02:06:00	7.35	14,8	2022-08-12 00:1
20.00	9.96	-63.00	11:48:00	0.98	0.99	27.89	9.85	42.00	03:00:00	0	0 2022-09-18	Ground	2022-09-19 02:06:00	7,4	41,9	2022-08-13 13:0
10.00	10.00	-34.00	12:30:00	0.98	0.99	33.55	7.35	-30.00	02:06:00	0	0 2022-09-19	Satellite - modis Day	2022-09-20 13:12:00	33.15	10,9	2022-08-13 02:5
10.00	10.00	34.00	13:12:00	0.98	0.99	33.15	4.75	30.00	02:48:00	0	0 2022-09-20	Ground	2022-09-20 13:12:00	33,1	37,0	2022-08-14 13:0

Рисунок 4.3.2 – Выходной Excel-файл

2) Карты представляют собой выходной файл формата HTML, который создается с помощью нескольких продуктов в виде масок. Карта интерактивная, центрируется исходя из введенных пользователем координат. Пример карты изображен на рисунке 4.3.3.

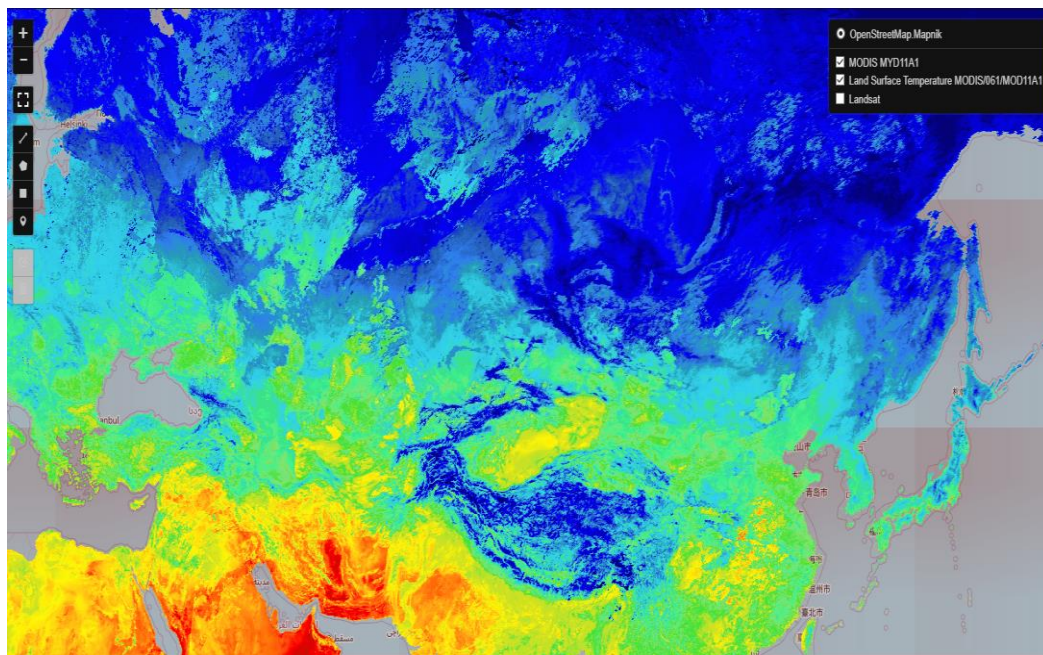


Рисунок 4.3.3 – Выходной файл в виде карты

3) Графики имеют две расцветки для точек – синяя для наземных данных и красная для спутниковых. График изображает только совпавшие значения, только совпавшие по дате и времени. Пример графика изображен на рисунке 4.3.4

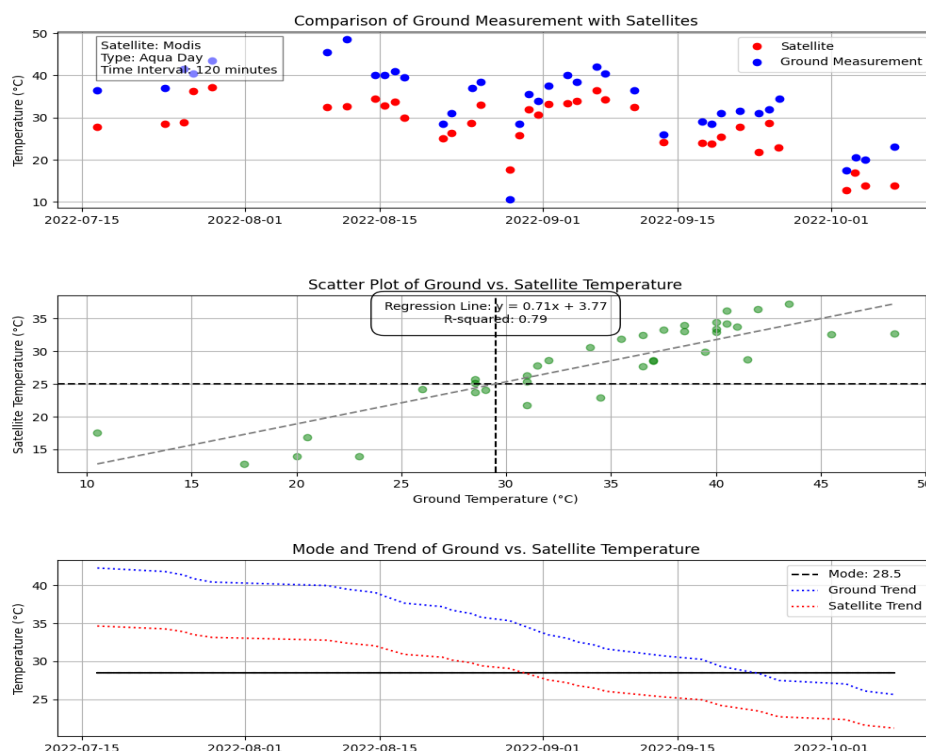


Рисунок 4.3.4 – Выходной файл в виде графика

4) Отчёт в формате PDF. Отчёт включает в себя все таблицы, полученные значения и графики. Пример отчёта изображен на рисунке 4.3.5

Report from data modis

Satellite: Modis Type: Aqua Time Interval: 35 minutes Coordinates: [88.58833, 50.02446] Date Range: 2022-07-15 to 2022-10-08

Clear_day_cov	Clear_night_cov	Day_view_angle	Day_view_time	Emis_31	Emis_32	LST_Day_1km
9.84	10.00	-25.00	12:42:00	0.98	0.99	40.63
20.00	9.88	-40.00	12:30:00	0.98	0.99	42.71
19.99	19.90	-53.00	12:12:00	0.98	0.99	36.11
9.83	9.92	-6.00	12:54:00	0.98	0.99	33.17
9.82	10.00	20.00	13:12:00	0.98	0.99	29.21
10.00	16.03	-50.00	12:18:00	0.98	0.99	31.71
20.00	10.00	52.00	13:42:00	0.98	0.99	34.95
10.00	10.00	-18.00	12:48:00	0.98	0.99	38.67
10.00	11.28	26.00	13:18:00	0.98	0.99	22.53
10.00	14.73	-47.00	12:24:00	0.98	0.99	32.43
9.80	10.00	-43.00	12:30:00	0.98	0.99	37.21
12.00	9.81	14.00	13:12:00	0.98	0.99	41.61
10.00	8.08	49.00	13:42:00	0.98	0.99	28.75
9.97	10.00	20.00	13:12:00	0.98	0.99	30.63
10.00	9.84	-19.00	12:48:00	0.98	0.99	27.11

Рисунок 4.3.5 – Выходной файл в виде отчёта

В разделе представлено полное описание данных, используемых и генерируемых программным модулем. Входные данные включают файлы наземных измерений в формате .XLSX и спутниковые данные из продуктов MODIS и Landsat. Каждый тип данных имеет свою специфику и формат, который строго определен для корректной обработки. Выходные данные включают в себя таблицы Excel, графики, интерактивные карты и отчеты в формате PDF. Таблицы Excel содержат собранные и обработанные данные из наземных и спутниковых источников, что обеспечивает удобство анализа и последующего использования. Графики представляют собой наглядное визуальное отображение результатов, а интерактивные карты позволяют визуализировать пространственную зависимость данных. Отчеты в формате PDF собирают все полученные значения, графики и аналитическую информацию для более удобного представления результатов и последующего исследования. Такой подход обеспечивает полный цикл обработки и анализа данных с минимальными усилиями со стороны пользователя.

4.4 Разработка механизмов хранения данных и структуры файловой системы, реализуемых в рамках модуля

В рамках разработки программного модуля для валидации температурных данных КА Landsat и Terra/Aqua на основе наземных измерений необходимо предусмотреть механизмы хранения данных и определить структуру файловой системы.

Файловая структура программного модуля представляет собой директории, в которые сохраняются готовые выходные данные. Программа сама даёт им название в формате `type{current_time}` для удобства и определения пользователем, когда запрашивались те или иные данные.

Для хранения наземных данных не требуется отдельной директории так как программный модуль открывает Проводник Windows. Для графиков есть папка Graphics, где название файла имеет свою структуру для удобства использования: название спутника, тип спутника, временной интервал в минутах и дата и время создания графика. Пример хранения файлов изображен на рисунке 4.4.1

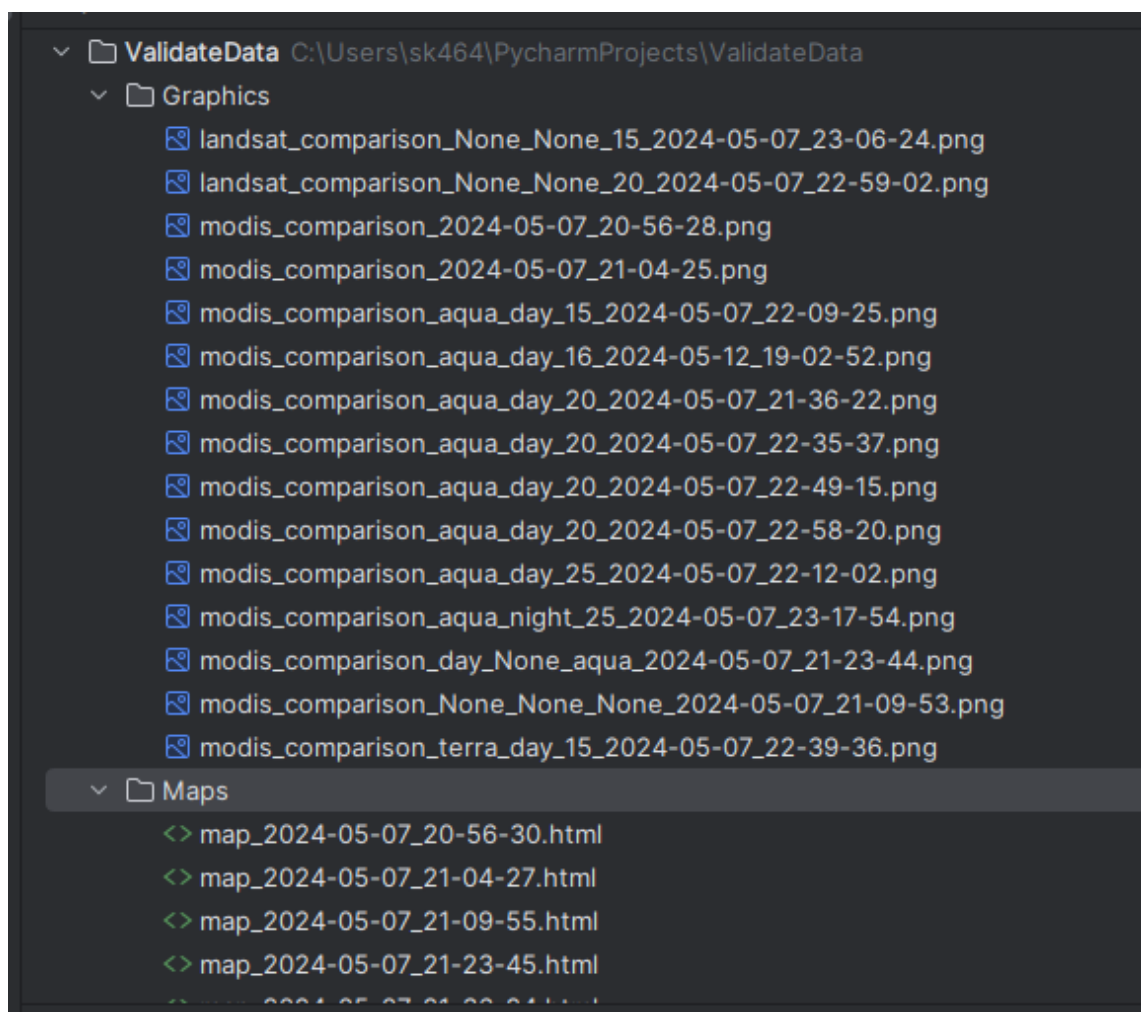


Рисунок 4.4.1 – Файловая структура

Файловая структура программного модуля также включает в себя поддиректории для временного хранения промежуточных файлов, а также логов работы программы. Эти поддиректории создаются автоматически при запуске программы и удаляются после завершения её работы для обеспечения чистоты и порядка в файловой системе.

Программный модуль предусматривает механизмы проверки доступности и целостности данных, включая проверку наличия необходимых файлов и их соответствие формату. В случае обнаружения ошибок или несоответствий программа генерирует соответствующие сообщения об ошибке и сохраняет их в лог-файл для последующего анализа.

Таким образом, файловая структура программного модуля обеспечивает эффективное управление и хранение как входных, так и выходных данных.

4.5 Алгоритмы реализации используемых математических моделей

В программном модуле задействованы алгоритмы вычисления RSME и MBE, а также алгоритм построчного сравнения данных из двух массивов. Алгоритм, который сравнивает данные, получаемые со спутника и из файла Excel необходим для дальнейшей валидации значений. Схема работы алгоритма представлена на рисунке 4.5.1.

Алгоритм работы класса CommonDates предназначен для сопоставления данных, полученных со спутника, с наземными данными из файла Excel.

1. Инициализация объекта CommonDates: происходит инициализация объекта CommonDates с полученными данными: DataFrame (df), содержащий наземные измерения, DataFrame (excel_data), содержащий данные из Excel, и строка satellite, указывающая на выбранный спутник. Выбираются необходимые столбцы в зависимости от спутника.

2. Метод match_data:

Начинается цикл по выбранным данным из файла Excel.

Для каждой строки в файле Excel: извлекаются данные из выбранных столбцов: дата и время спутника. Происходит поиск совпадающих строк в данных Excel по дате и времени спутника. У времени высчитывается разность, которую ввёл пользователь. Если найдены совпадающие строки, значения спутника и наземные данные добавляются в список совпадений. Завершается итерация.

3. Возврат списка совпадений: метод возвращает список, содержащий данные о совпадениях между спутником и наземными измерениями.

Этот алгоритм позволяет эффективно сопоставлять данные, полученные со спутника, с наземными измерениями, и выявлять временные совпадения для дальнейшего анализа и валидации.

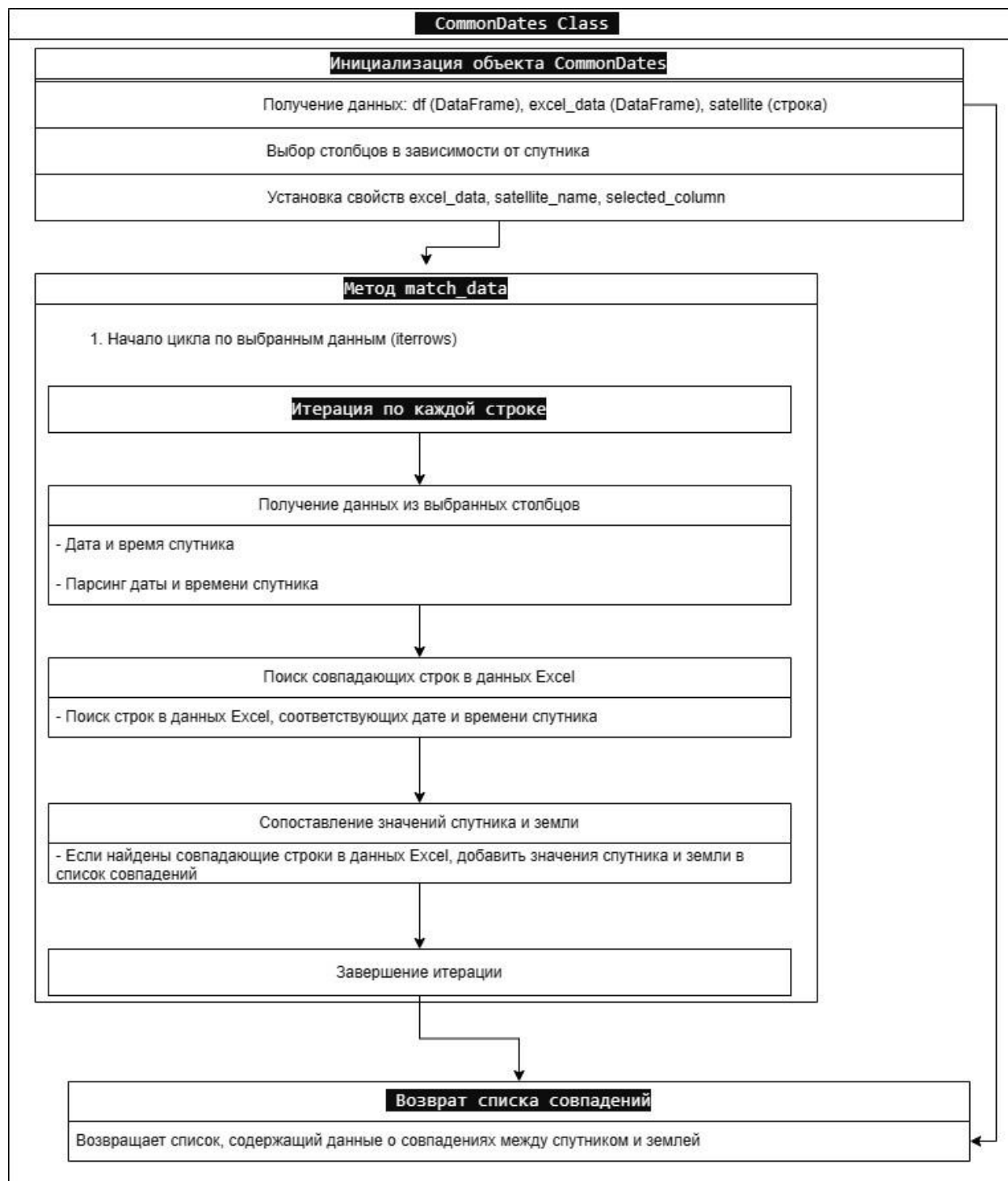


Рисунок 4.5.1 – Алгоритм сравнения строк

Алгоритм для расчета среднеквадратичной ошибки (RMSE) и средней абсолютной ошибки (MBE) между значениями, полученными со спутника, и наземными измерениями температуры представлен на рисунке 4.5.2.

Шаги алгоритма:

1. Инициализация списков: создаются два пустых списка: `satellite_values` и `ground_values`, в которые будут добавляться значения температуры со спутника и с наземных измерений соответственно.
2. Обход совпавших строк: происходит обход всех совпавших строк `Matches` из `Common_dates`, то есть строк, в которых есть данные как со спутника, так и с наземных измерений.
3. Для каждой пары значений спутника и земли:
 - 1) Извлекаются значения температуры со спутника и с наземных измерений из совпавших строк.
 - 2) Значения добавляются в соответствующие списки.
 - 3) Вычисляется разность между значениями спутника и земли.
 - 4) Вычисляется квадрат разности.
 - 5) Рассчитывается сумма квадратов разностей.
 - 6) Вычисляется корень из среднеквадратичной ошибки (RMSE) и средняя абсолютная ошибка (MBE).

Этот алгоритм позволяет оценить точность данных, полученных со спутника, по сравнению с наземными измерениями температуры, что является важным этапом валидации и анализа данных спутниковых наблюдений.

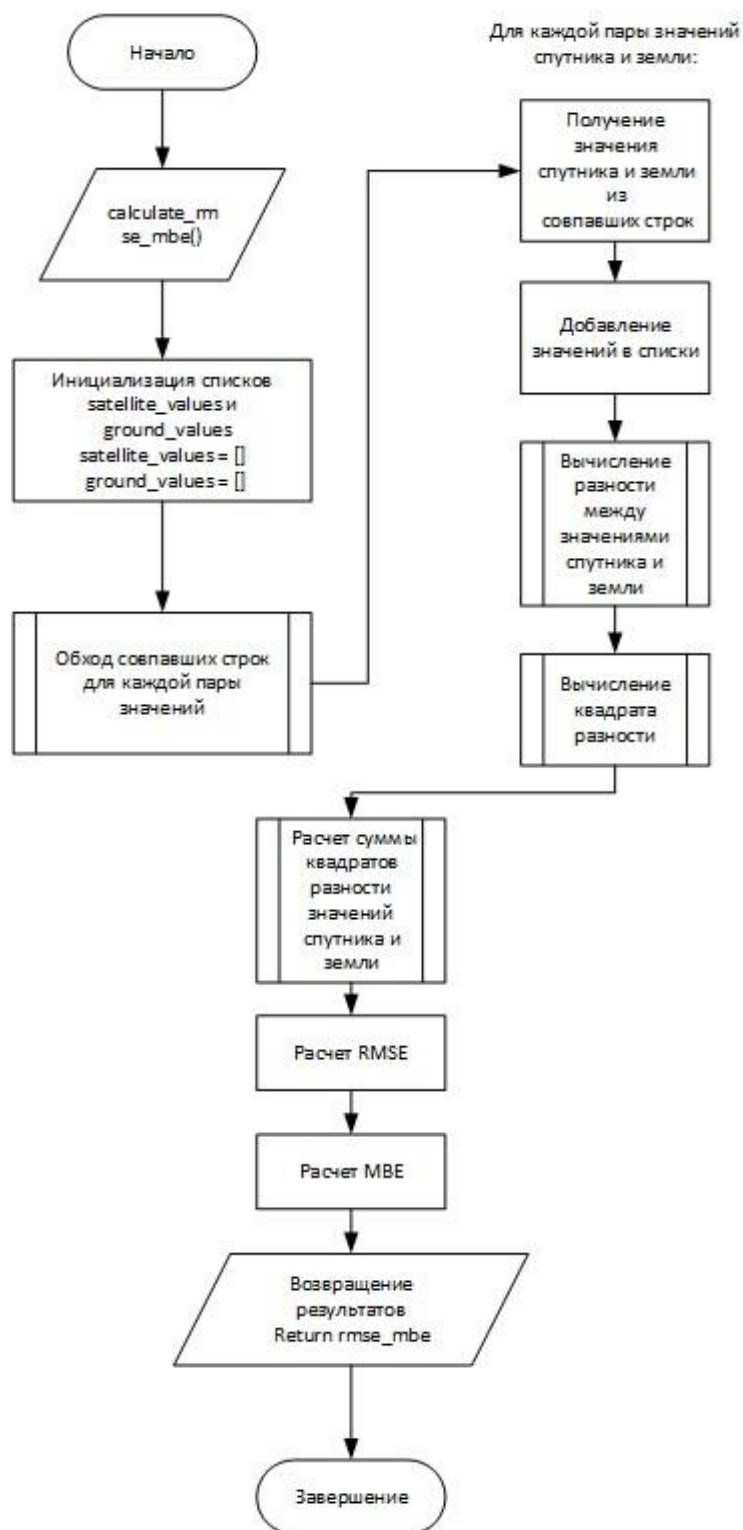


Рисунок 4.5.2 – Алгоритм расчёта каждой пары значений

Реализация этих алгоритмов в программном модуле позволяет эффективно сопоставлять и оценивать точность данных, полученных со спутника, по сравнению с наземными измерениями температуры. Такой подход является важным шагом в процессе валидации и анализа данных спутниковых наблюдений.

4.6 Алгоритмы использования применяемых программных технологий

Для решения задачи валидации температурных данных, основанной на сравнении спутниковых и наземных измерений, применяются следующие программные технологии, включая библиотеки matplotlib, pandas и numpy:

1. Подготовка данных в Google Earth Engine API:
2. Выбор района интереса: определение географической области, для которой необходимо получить спутниковые данные.
3. Запрос данных из GEE API: формирование запроса к Google Earth Engine API для получения спутниковых данных в заданной области и интересующем временном диапазоне.
4. Предварительная обработка данных: обработка полученных данных для удобства сравнения с наземными данными.
5. Использование Excel и Python:
 - 1) Подготовка данных: запись наземных данных в формате, совместимом с Python (например, CSV).
 - 2) Считывание данных в Python: использование библиотеки pandas для считывания данных из Excel и их предварительной обработки.
 - 3) Вычисление RMSE и MBE: разработка алгоритма на языке программирования Python для расчета RMSE и MBE согласно математической модели.
 - 4) Сравнение с данными из GEE: сопоставление результатов, полученных с наземными данными, с результатами, полученными из GEE API.
6. Визуализация результатов с помощью Matplotlib:
 - 1) Использование библиотек для визуализации: применение библиотек Python (например, Matplotlib, Seaborn) для визуализации результатов сравнительного анализа.
 - 2) Генерация графиков: создание графиков, диаграмм и других визуальных элементов, отображающих точность и смещение данных.
7. Автоматизация процесса с помощью Python:
 - 1) Создание скрипта: разработка скрипта или приложения на Python для автоматизации процесса считывания, обработки и анализа данных.
 - 2) Использование библиотеки NumPy для вычислений: применение функций библиотеки numpy для выполнения математических операций, таких как вычисление RMSE и MBE.

3) Использование механизмов очистки данных от выбросов (Правило трёх сигм, очистка от аномальных выбросов).

8. Работа с библиотекой NumPy и вычисление RMSE и MBE

1) Для вычисления RMSE и MBE используются функции библиотеки numpy, которая обеспечивает эффективные операции с массивами данных. Конкретно, используются следующие выражения:

Пример скрипта, использующий математическую модель, представлен на рисунке 4.6.1.

```
import pandas as pd
import numpy as np
import ee

# Загрузка данных из Excel
excel_data = pd.read_csv('MB3.csv', delimiter='\\t')
ground_truth = excel_data['наземные'].str.replace(',', '.').astype(float)
satellite_data = excel_data['спутниковые'].str.replace(',', '.').astype(float)

# Вычисление RMSE
rmse = np.sqrt(np.mean((ground_truth - satellite_data)**2))
print(f'RMSE: {rmse}')

ee.Initialize()

image = ee.Image()

gee_data = np.array([1.0, 2.0, 3.0, 4.0, 5.0])

# Вычисление RMSE с данными из GEE API
rmse_gee = np.sqrt(np.mean((ground_truth - gee_data)**2))
print(f'RMSE with GEE data: {rmse_gee}')
```

Рисунок 4.6.1 – Пример вычисления RMSE

Эти шаги предоставляют алгоритм использования программных технологий для решения задач ВКР, связанных с обработкой и анализом данных.

4.7 Архитектура и схема функционирования модуля

В данном разделе представлена архитектура и схема функционирования консольного программного модуля (ПМ) для обработки данных спутниковых наблюдений и их сравнения с наземными измерениями с использованием Google Earth Engine API (GEE API). Программный модуль состоит из следующих ключевых компонентов: User Services, Backend Component и GEE API.

1. Backend Component:

Backend Component является центральным компонентом программного модуля, отвечающим за обработку данных и взаимодействие с GEE API. Этот компонент состоит из следующих функциональных блоков:

1) Data Processing Module (Модуль обработки данных): отвечает за загрузку спутниковых данных из GEE API, обработку полученных данных и подготовку их к сравнению с наземными измерениями.

2) Comparison Module (Модуль сравнения): реализует алгоритмы сравнения спутниковых данных с наземными измерениями, включая расчеты RMSE и MBE. Этот модуль использует данные, предоставленные Data Processing Module, для выполнения сравнения.

2. GEE API:

Google Earth Engine API предоставляет доступ к спутниковым данным и функциям обработки геопространственных данных, необходимых для работы программного модуля.

3. User Services:

User Services представляют собой интерфейс взаимодействия пользователя с программным модулем через консольный диалог. Пользователь взаимодействует с программой, вводя команды и получая результаты обработки данных.

Пользователь взаимодействует с консольным диалогом, отправляя команды и получая ответы.

Консольный диалог передает команды на обработку Backend Component.

Backend Component взаимодействует с GEE API для получения и обработки спутниковых данных.

После обработки данных Backend Component возвращает результаты пользовательскому сервису через консольный диалог.

Таким образом, архитектура программного модуля обеспечивает эффективное взаимодействие между компонентами для обработки и сравнения спутниковых данных с наземными измерениями.

На рисунке 4.7.1 представлена диаграмма развертывания, иллюстрирующая архитектурную организацию модуля. Модуль разделен на следующие основные компоненты:

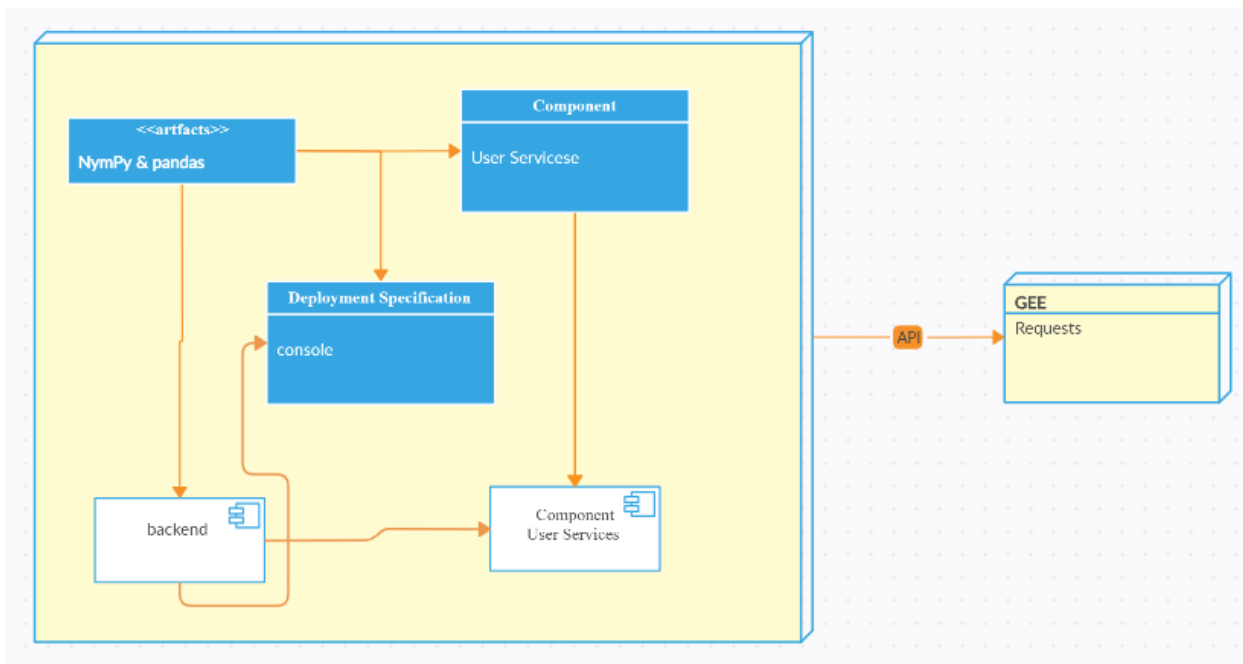


Рисунок 4.7.1 – Архитектурная организация системы

Диаграмма показывает физическое взаимодействие между компонентами, подчеркивая ключевые элементы, такие как браузер пользователя, сервера и внешние сервисы, которые необходимы для функционирования приложения.

5 ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ

5.1 План тестирования

При тестировании программного модуля, предназначенного для обработки спутниковых данных и сравнения их с наземными измерениями с использованием Google Earth Engine API (GEE API), необходимо учитывать особенности функциональности модуля и его взаимодействие с внешними компонентами. План тестирования включает в себя следующие этапы и методы тестирования:

1. Модульное тестирование (Unit Testing):

Цель: проверка корректности работы отдельных функций и модулей программного модуля.

Методы: использование фреймворка для модульного тестирования (например, unittest в Python) для написания и выполнения тестовых случаев для каждой отдельной функции и модуля.

Обоснование: модульное тестирование позволит выявить и исправить возможные ошибки на ранних этапах разработки и обеспечит надежность отдельных компонентов программы.

2. Интеграционное тестирование (Integration Testing):

Цель: проверка взаимодействия между компонентами программного модуля.

Методы: выполнение тестовых случаев для проверки взаимодействия и совместной работы различных компонентов модуля.

Обоснование: интеграционное тестирование позволит удостовериться в корректности взаимодействия между модулями и компонентами программы.

3. Функциональное тестирование (Functional Testing):

Цель: проверка соответствия функциональности программного модуля заявленным требованиям и спецификациям.

Методы: выполнение тестовых сценариев для проверки функциональности модуля с учетом всех вариантов использования.

Обоснование: функциональное тестирование поможет удостовериться, что модуль выполняет все свои функции в соответствии с требованиями пользователя.

4. Нагрузочное тестирование (Load Testing):

Цель: оценка производительности и устойчивости программного модуля при различных нагрузках.

Методы: создание тестовых сценариев для проверки работы модуля при максимальной и повышенной нагрузке.

Обоснование: нагрузочное тестирование позволит выявить узкие места и проблемы производительности программного модуля.

5. Тестирование безопасности (Security Testing):

Цель: проверка наличия уязвимостей и обеспечение защищенности программного модуля от внешних угроз.

Методы: анализ кода на предмет наличия уязвимостей, проверка защищенности передаваемых данных.

Обоснование: тестирование безопасности поможет предотвратить возможные атаки и утечки данных.

6. Автоматизированное тестирование (Automated Testing):

Цель: ускорение и оптимизация процесса тестирования с помощью автоматизированных средств.

Методы: написание скриптов для автоматизации выполнения тестовых сценариев.

Обоснование: автоматизированное тестирование позволит ускорить процесс тестирования, сократить время на ручное выполнение тестов и обеспечить повторяемость результатов.

Для обеспечения высокого уровня качества программного модуля необходимо провести комплексное тестирование, включающее в себя модульное, интеграционное, функциональное, нагрузочное, тестирование безопасности и автоматизированное тестирование. Это позволит выявить и устранить ошибки, обеспечить соответствие требованиям и обеспечить надежность и производительность программного модуля.

5.2 Результаты тестирования

Результаты тестирования представлены в таблицах ниже.

Таблица 5.2.1 – Модульное тестирование

Название теста	Описание	Статус
test_read_excel	Проверка корректности чтения данных из Excel	Пройден

test_process_data	Проверка обработки данных модулем	
test_calculate_rmse	Проверка корректности расчета RMSE	Пройден
test_calculate_mbe	Проверка корректности расчета MBE	Пройден

Таблица 5.2.2 – Интеграционное тестирование

Название теста	Описание	Статус
test_gee_integration	Проверка взаимодействия с Google Earth Engine	Пройден
test_data_comparison	Проверка сравнения данных с наземными измерениями	Пройден

Таблица 5.2.3 – Функциональное тестирование

Название теста	Описание	Статус
test_functionality_plot	Тестирование функции создания графика	Пройден
test_functionality_matches	Тестирование функции сравнения строк	Пройден

Таблица 5.2.4 – Нагрузочное тестирование

Название теста	Описание	Статус
test_load_handling	Тестирование обработки данных при максимальной нагрузке	Пройден

Таблица 5.2.5 – Тестирование безопасности

Название теста	Описание	Статус
----------------	----------	--------

test_security_scan	Сканирование кода на предмет уязвимостей	Пройден
--------------------	--	---------

В результате проведенного тестирования было установлено, что программный модуль успешно прошел все запланированные виды тестирования. Все функции работают корректно, обнаруженные ошибки были устранены, а производительность модуля соответствует требованиям. Тестирование безопасности показало отсутствие уязвимостей. Таким образом, программный модуль готов к внедрению в реальные условия эксплуатации.

5.3 Оптимизация модуля

В результате тестирования выявлены различные недочеты в работе модуля, включая проблемы с алгоритмом сопоставления спутниковых и наземных данных, обработкой дат, выводом PDF-отчетов, обработкой ошибок и исключений. Для улучшения производительности и исправления этих проблем была проведена тщательная оптимизация модуля. Ниже представлен процесс оптимизации с соответствующими скриншотами.

1. Исправление алгоритма сопоставления данных: в ходе оптимизации был пересмотрен и улучшен алгоритм сопоставления спутниковых и наземных данных. Это позволило увеличить точность сопоставления и сократить время выполнения операции.
2. Улучшение работы с датами: были внесены изменения в обработку дат для устранения возможных ошибок и улучшения точности анализа временных данных.
3. Улучшение вывода PDF-отчетов: в результате оптимизации была улучшена процедура формирования и вывода PDF-отчетов для более наглядного и информативного представления результатов анализа.
4. Обработка ошибок и исключений: были внесены исправления для более эффективной обработки ошибок и исключений, что повысило стабильность и надежность модуля.
5. Прочие улучшения: в процессе оптимизации были также внесены различные другие улучшения, направленные на повышение производительности, улучшение пользовательского опыта и общей функциональности модуля.

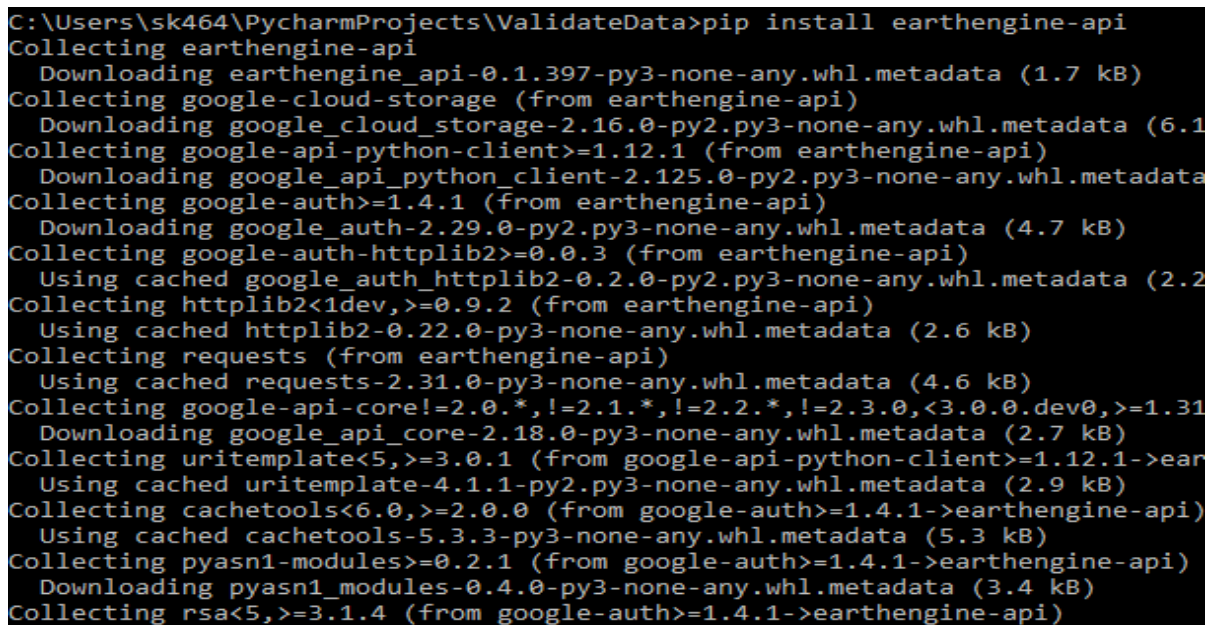
В результате проведенных оптимизаций модуль стал работать более эффективно и стабильно. Улучшенные алгоритмы, обработка дат и вывод результатов способствуют более точному анализу данных и улучшению пользовательского опыта. Теперь модуль готов к более продуктивной и надежной работе в различных сценариях использования.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. Установка и настройка программного модуля

Для установки и настройки программного модуля на вашем компьютере, убедитесь, что предварительно установлен интерпретатор Python версии, совместимой с модулем (3.10 и выше). Интерпретатор скачивается на официальном сайте, либо для Windows в Microsoft Store. Для того, чтобы установить необходимые компоненты, нужно использовать команду «pip install». Пример установки изображен на рисунке 6.1. Подобных библиотек необходимо экспортировать шесть:

- 1) pip install earthengine-api
- 2) pip install pandas
- 3) pip install matplotlib
- 4) pip install folium
- 5) pip install reportlab
- 6) pip install openpyxl
- 7) pip install scipy
- 8) pip install geemap



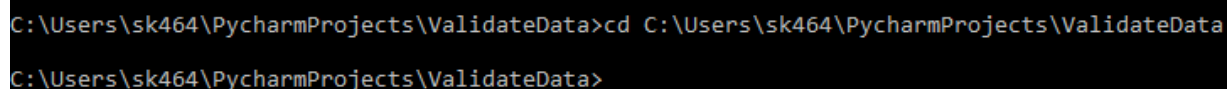
```
C:\Users\sk464\PycharmProjects\ValidateData>pip install earthengine-api
Collecting earthengine-api
  Downloading earthengine_api-0.1.397-py3-none-any.whl.metadata (1.7 kB)
Collecting google-cloud-storage (from earthengine-api)
  Downloading google_cloud_storage-2.16.0-py2.py3-none-any.whl.metadata (6.1 kB)
Collecting google-api-python-client<=1.12.1 (from earthengine-api)
  Downloading google_api_python_client-2.125.0-py2.py3-none-any.whl.metadata (2.2 kB)
Collecting google-auth>=1.4.1 (from earthengine-api)
  Downloading google_auth-2.29.0-py2.py3-none-any.whl.metadata (4.7 kB)
Collecting google-auth-httplib2>=0.0.3 (from earthengine-api)
  Using cached google_auth_httplib2-0.2.0-py2.py3-none-any.whl.metadata (2.2 kB)
Collecting httplib2<1dev,>=0.9.2 (from earthengine-api)
  Using cached httplib2-0.22.0-py3-none-any.whl.metadata (2.6 kB)
Collecting requests (from earthengine-api)
  Using cached requests-2.31.0-py3-none-any.whl.metadata (4.6 kB)
Collecting google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.0 (from earthengine-api)
  Downloading google_api_core-2.18.0-py3-none-any.whl.metadata (2.7 kB)
Collecting uritemplate<5,>=3.0.1 (from google-api-python-client<=1.12.1->earthengine-api)
  Using cached uritemplate-4.1.1-py2.py3-none-any.whl.metadata (2.9 kB)
Collecting cachetools<6.0,>=2.0.0 (from google-auth>=1.4.1->earthengine-api)
  Using cached cachetools-5.3.3-py3-none-any.whl.metadata (5.3 kB)
Collecting pyasn1-modules>=0.2.1 (from google-auth>=1.4.1->earthengine-api)
  Downloading pyasn1_modules-0.4.0-py3-none-any.whl.metadata (3.4 kB)
Collecting rsa<5,>=3.1.4 (from google-auth>=1.4.1->earthengine-api)
```

Рисунок 6.1 – Установка earthengine-api

Затем, загрузите программный модуль с официального репозитория и распакуйте его содержимое в удобную для вас директорию. После этого откройте командную строку (терминал) на вашем устройстве, перейдите в каталог с распакованным модулем и

выполните установку необходимых зависимостей с помощью команды `pip install -r requirements.txt`.

Для запуска проекта также необходимо перейти в директорию с проектом командой «`cd`». Пример этой команды изображен на рисунке 6.2.



```
C:\Users\sk464\PycharmProjects\ValidateData>cd C:\Users\sk464\PycharmProjects\ValidateData
C:\Users\sk464\PycharmProjects\ValidateData>
```

Рисунок 6.2 – Переход в директорию с проектом

Для того чтобы запустить исполняемый файл необходимо ввести команду `python main.py`. После введения команды стоит подождать при первом запуске для обработки кода. Далее высветится окно Проводника для загрузки Excel файла.

2. Основные задачи пользователя

Одной из основных задач пользователей является анализ данных со спутника Terra, Aqua/MODIS. Для этого запустите программу, выполнив команду `python main.py` в командной строке. При запуске укажите файл с наземными данными и координаты для анализа. Далее, выберите источник данных как MODIS, следуя инструкциям, которые появятся в консоли. Программа обработает ваш запрос, отправит его в Google Earth Engine (GEE) и визуализирует данные. После завершения обработки, результаты будут выведены в консоль, и вы сможете сохранить их в форматах .png, используя соответствующие инструкции.

3. Действия пользователя в случае сбоя программы

В случае возникновения сбоя программы, вы получите сообщение об ошибке в консоли. Попробуйте перезапустить программу и повторить необходимые действия.

ЗАКЛЮЧЕНИЕ

В результате выполнения данного проекта была разработана программная система для анализа данных, полученных со спутников Landsat и Terra, Aqua/MODIS. Этот процесс включал в себя не только создание и тестирование программного кода, но и оптимизацию его работы, а также разработку руководства пользователя для облегчения работы с системой. В рамках проекта было написано 11 скриптов, каждый из которых выполняет определенную функцию в системе.

Программная система успешно выполняет основную задачу — анализ данных со спутников. Кроме того, в систему была интегрирована поддержка данных со спутников Landsat, что значительно расширяет её функциональность. Также была реализована очистка данных, что обеспечивает высокую точность результатов анализа.

Однако в ходе разработки и тестирования были выявлены несколько проблем, которые требуют дальнейшего внимания. Первая проблема заключается в структурированности кода. Несмотря на то, что программа работает корректно, она может быть более легко читаемой и поддерживаемой, если разделить её на более мелкие функциональные блоки и модули.

Вторая проблема связана с ограниченным функционалом программы. В настоящее время система поддерживает только два основных источника данных и ограниченное количество типов графиков. Для повышения её привлекательности для пользователей следует рассмотреть возможность добавления поддержки большего количества спутников, видов графиков и математических вычислений.

Третья проблема касается зависимостей от сторонних библиотек и модулей. Сложности могут возникнуть при установке и запуске программы из-за необходимости установки всех зависимостей, которые не всегда присутствуют на компьютере пользователя. Это особенно актуально, так как для корректной работы системы требуется операционная система Windows 10.

Для улучшения программной системы можно улучшить структурированность кода, разбив его на более мелкие функциональные блоки и модули. Расширить функциональность системы, добавив поддержку большего количества спутников, видов графиков и математических вычислений. Также следует обеспечить автоматическую установку всех необходимых зависимостей при первом запуске программы, чтобы упростить процесс установки для пользователей.

Программная система для анализа данных со спутников Terra и Aqua/MODIS, а также Landsat, представляет собой значимый инструмент для исследователей и

специалистов в области геоинформатики. Однако её потенциал ещё не полностью реализован, и существует много возможностей для улучшения и дальнейшего развития.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Дуана С., Либз., Лиц Х., Гёттшед Ф., Вые Х., Жаоф В., Ленга П., Чжангби Х., Кол С. Validation of Collection 6 MODIS land surface temperature product using in situ measurements // Remote Sensing of Environment. 2019. Vol. 225. P. 16–29.
2. Данные Landsat 9 [Электронный ресурс] // GISProxima. – Режим доступа: https://gisproxima.ru/Dannye_LANDSAT9 (Дата обращения: Январь 2024).
3. Технические характеристики спутников LANDSAT-8 [Электронный ресурс] // ScanEx. – Режим доступа: <https://www.scanex.ru/data/satellites/LANDSAT-8/> (Дата обращения: Февраль 2024).
4. Технические характеристики спутников Terra, Aqua/MODIS [Электронный ресурс] // ScanEx. – Режим доступа: <https://www.scanex.ru/data/satellites/terra-aqua-modis/> (Дата обращения: Март 2024).
5. MOD11A1.061 Terra Land Surface Temperature and Emissivity Daily Global 1km | Earth Engine Data Catalog | Google for Developers.
6. MYD11A1.061 Aqua Land Surface Temperature and Emissivity Daily Global 1km | Earth Engine Data Catalog | Google for Developers.
7. USGS Landsat 8 Level 2, Collection 2, Tier 1 | Earth Engine Data Catalog | Google for Developers USGS Landsat 8 Level 2, Collection 2, Tier 1.
8. Среднеквадратичная ошибка (RMSE) [Электронный ресурс] // GIS-Lab. – Режим доступа: <https://gis-lab.info/qa/rmse.html> (Дата обращения: Октябрь 2023).
9. MeteoBlue [Электронный ресурс] // MeteoBlue. – Режим доступа: <https://www.meteoblue.com/ru/%D0%BF%D0%BE%D0%B3%D0%BE%D0%B4%D0%B0/%D0%BA%D0%B0%D1%80%D1%82%D1%8B/index#coords=3.26/52.31/85.5&map=satellite~r~adar~none~none~none> (Дата обращения: Декабрь 2023).
10. Cosmos-Online [Электронный ресурс] // Cosmos-Online. – Режим доступа: <https://cosmos-online.ru/weather-from-space> (Дата обращения: Январь 2024).
11. Проект MODLAND NASA [Электронный ресурс] // NASA. – Режим доступа: https://modis.gsfc.nasa.gov/data/atbd/land_val.pdf (Дата обращения: Февраль 2024).
12. PyCharm [Электронный ресурс] // JetBrains. – Режим доступа: <https://www.jetbrains.com/pycharm> (Дата обращения: Март 2024).
13. GEE API [Электронный ресурс] // Google Earth Engine. – Режим доступа: <https://earthengine.google.com/> (Дата обращения: Май 2024).

ПРИЛОЖЕНИЯ

Приложение А

Техническое задание на разработку программного модуля для валидации температурных данных КА Landsat и Terra/Aqua на основе наземных измерений

1. Требования к консольному интерфейсу:

- Пользовательский интерфейс должен быть простым и интуитивно понятным.
- Предусмотреть возможность ввода координат точки для анализа.
- Обеспечить выбор источника данных (Landsat или Aqua/Terra MODIS).
- Выводить информацию о ходе выполнения операций и сообщения об ошибках.

2. Требования к формированию графиков:

- Создание графика с основными показателями температуры.
- Формирование графика с облаком рассеяния для визуализации данных.
- Построение графика с трендом и модой для анализа временных рядов.

3. Требования к карте:

- Предоставление возможности отображения данных на карте.
- Маскирование областей.
- Центрирование карты на введенной пользователем координате.

4. Требования к excel-файлу выходному:

- Формирование excel-файла с результатами анализа.
- Столбцы с данными из GEE, совпавшими значениями, датой и временем.

5. Требования к очистке данных:

- Применение правила трех сигм для определения аномалий.
- Удаление явных выбросов из данных.
- Введение допустимого диапазона температур для фильтрации данных.

6. Требования к RMSE и MBE:

- Расчет RMSE и MBE с использованием правильных формул.

7. Требования к отчету PDF:

- Создание PDF-отчета с подробной информацией о совпавших значениях и результатами анализа.

- Читаемое оформление отчета с использованием заголовков, таблиц и графиков.

8. Требования к наименованию и созданию папок:

- Наименование папок должно содержать необходимые данные для идентификации результатов анализа.
- Корректное создание и структурирование папок для хранения результатов.

9. Требования к источникам данных:

- Использование двух источников данных: Landsat и Aqua/Terra MODIS.
- Разделение MODIS на данные для дневного и ночного времени.
- Убедиться в верности коэффициентов и значений смещения в источниках данных.

Приложение Б

Для данной программы необходимо наличие входного файла в Excel-формате. Структура файла строго определена следующим форматом: дата, время, значение. Пример входного файла изображен на рисунке 1Б.

А	В	С
7/15/2022	0:01	5
7/15/2022	1:01	4
7/15/2022	2:01	3.5
7/15/2022	3:01	3
7/15/2022	4:01	2
7/15/2022	5:01	2
7/15/2022	6:01	2
7/15/2022	7:01	4
7/15/2022	8:01	8.5
7/15/2022	9:01	13
7/15/2022	10:01	17.5
7/15/2022	11:01	16.5
7/15/2022	12:01	20.5
7/15/2022	13:01	26
7/15/2022	14:01	27
7/15/2022	15:01	29.5
7/15/2022	16:01	30
7/15/2022	17:01	23.5
7/15/2022	18:01	20
7/15/2022	19:01	16.5
7/15/2022	20:01	14
7/15/2022	21:01	12.5
7/15/2022	22:01	11.5
7/15/2022	23:01	11
7/16/2022	0:01	11
7/16/2022	1:01	10
7/16/2022	2:01	10
7/16/2022	3:01	10
7/16/2022	4:01	10
7/16/2022	5:01	8
7/16/2022	6:01	7.5
7/16/2022	7:01	10.5
7/16/2022	8:01	14
7/16/2022	9:01	18

Рисунок 1Б – Входной файл для точки АЗ

Стоит обратить внимание, что файл принимается как с заголовком, так и без него. Формат даты должен быть строго в формате «ММ/ДД/ГГГГ». Файл открывается из Проводника Windows.

Приложение Г

Программа выполняет расчёты из двух источников спутниковых данных – Landsat и Terra/Aqua MODIS и выбранного Excel-файла. Для Terra/MODIS в точке A1 с координатами [88.96675, 49.90721] и временным интервалом 25 минут в дневное время получены выходные данные, представленные на рисунках ниже.

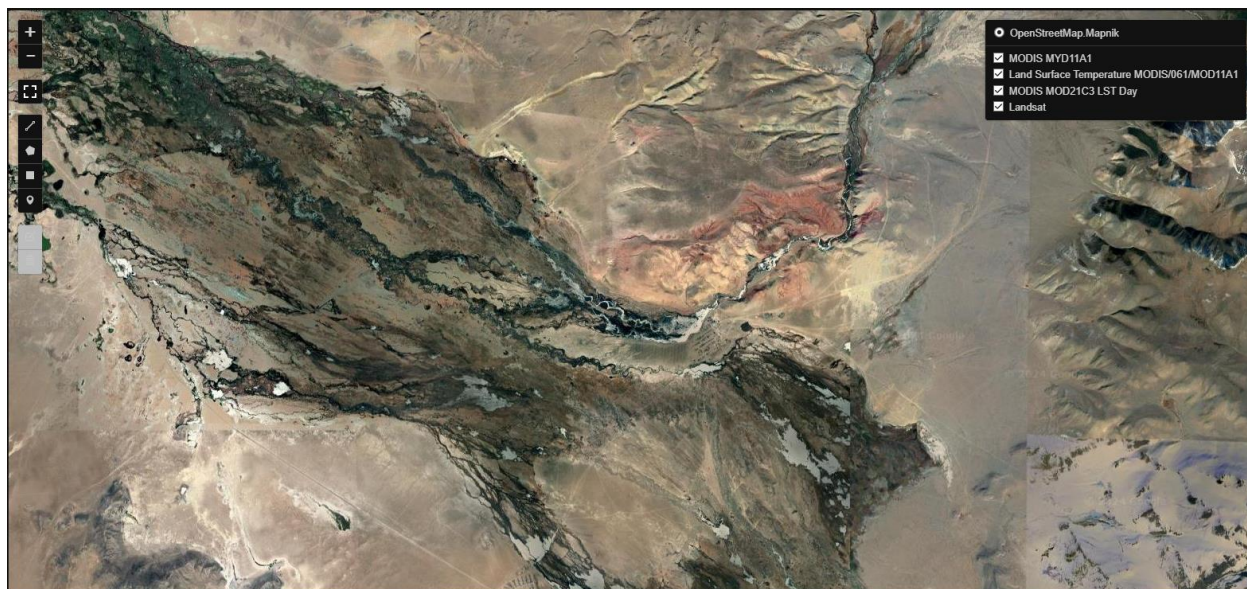


Рисунок Г1 – Карта с маской Landsat на точке A1

Для введенных координат формируется карта с центрированием на них. К ней приложены маски, детально показывающие местность и температурные особенности. Маска для MODIS/061/MOD21C3 представлена на рисунке Г2.

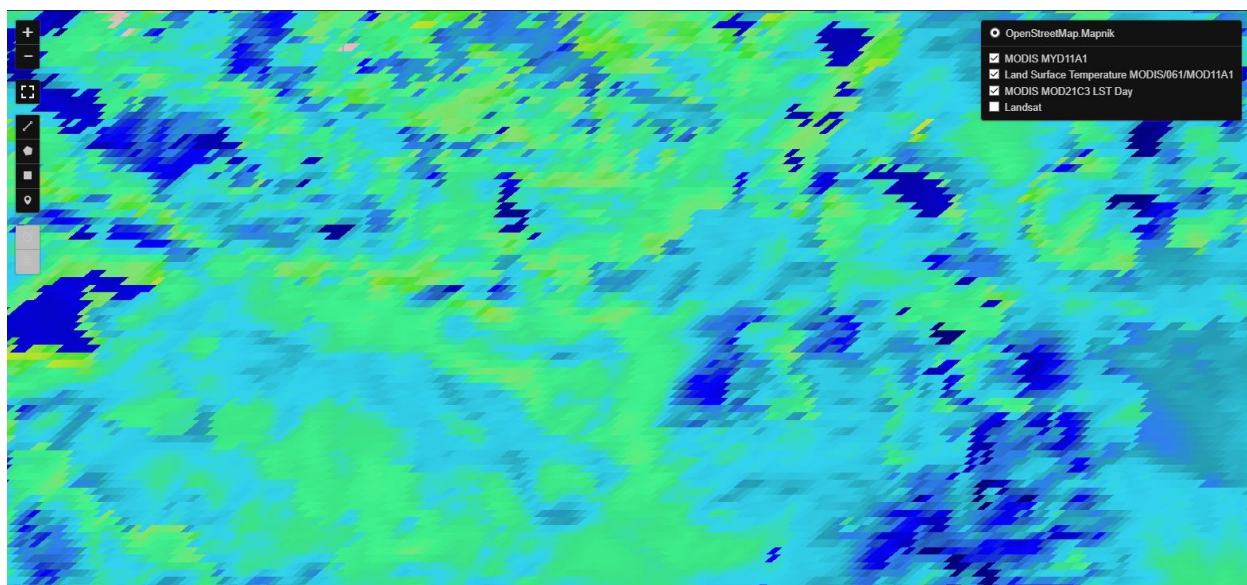


Рисунок Г2 – Карта с маской MODIS/061/MOD21C3

Продолжение приложения Г

Помимо карт создается отдельный файл в PDF-формате. Файл включает в себя всю собранную информацию, результаты валидации и дополнительные каналы, например, Emis. Пример отчёта изображен на рисунке Г3.

Report from data modis

Satellite: Modis Type: Terra Time Interval: 25 minutes Coordinates: [88.96675, 49.90721] Date Range: 2022-07-15 to 2022-10-08

Clear_day_cov	Clear_night_cov	Day_view_angle	Day_view_time	Emis_31	Emis_32	LST_Day_1km
1.54	0.88	54.00	11:42:00	0.98	0.99	21.93
1.97	1.00	32.00	11:18:00	0.98	0.99	29.31
1.00	1.00	-3.00	10:54:00	0.98	0.99	34.45
1.62	2.00	25.00	11:12:00	0.98	0.99	29.03
0.98	0.98	-29.00	10:36:00	0.98	0.99	29.89
0.99	0.93	16.00	11:06:00	0.98	0.99	31.49
2.00	1.00	-54.00	10:12:00	0.98	0.99	22.07
1.00	1.00	-21.00	10:42:00	0.98	0.99	31.83
1.64	0.66	39.00	11:24:00	0.98	0.99	30.91
1.00	0.66	-37.00	10:30:00	0.98	0.99	28.83
1.00	1.99	25.00	11:12:00	0.98	0.99	24.39
0.99	0.71	-12.00	10:48:00	0.98	0.99	24.39
1.00	0.99	-29.00	10:36:00	0.98	0.99	27.29

LST_Night_1km	Night_view_angle	Night_view_time	QC_Day	QC_Night	date
5.89	-32.00	21:18:00	0	65	2022-07-20
0.81	57.00	22:30:00	0	65	2022-07-24
8.43	37.00	22:06:00	0	65	2022-07-28
7.45	53.00	22:24:00	0	65	2022-08-02
10.53	13.00	21:48:00	0	65	2022-08-08
6.97	49.00	22:18:00	0	65	2022-08-11
10.17	-24.00	21:24:00	0	65	2022-08-12
7.83	22.00	21:54:00	0	65	2022-08-15
4.51	61.00	22:36:00	0	65	2022-08-16
4.77	3.00	21:42:00	0	65	2022-08-17

Рисунок Г3 – Отчёт в PDF-формате, сформированный программным модулем на основе полученных данных

Также в отчёт включена таблица всех совпавших строк, имеющая структуру «Имя спутника вместе с типом (если есть) – спутниковое значение – спутниковое время и дата – наземное значение – наземное дата и время». Под таблицей находятся рассчитанные RMSE и MBE. На рисунке Г8 изображены данные ручной валидации для сравнения их с анализом при помощи программного кода.

Оценка смещения (bias), среднеквадратичная ошибка (RMSE) и коэффициент детерминации (R^2) между MODIS / (Terra+Aqua) LST и LST in situ в дневное и ночное время

	Terra/MODIS				Aqua/MODIS			
	N	bias (°C)	RMSE (°C)	R^2	N	bias (°C)	RMSE (°C)	R^2
Точка 2 (день)	22	-0,92	2,47	0,64	27	-1,65	3,36	0,37
Точка 3 (день)	21	-0,09	2,53	0,63	21	-1,50	2,45	0,73
Точка 2 (ночь)	21	-1,82	2,70	0,61	24	-0,80	2,84	0,53
Точка 3 (ночь)	23	-2,30	2,71	0,84	22	-0,85	2,09	0,73
Среднее		-1,28	2,60			-1,20	2,68	

Рисунок Г8 – Результаты ручной валидации

Как можно видеть, значения RMSE и MBE, просчитанные программным модулем, вполне можно считать достоверными. Таблица из отчёта изображена на рисунке Г4.

ID	Satellite name	Satellite value	Satellite Datetime	Ground value	Ground Datetime
1	modis Terra Day	29.31	2022-07-24 11:18:00	31.5	2022-07-24 11:01:00
2	modis Terra Day	34.45	2022-07-28 10:54:00	38.5	2022-07-28 11:01:00
3	modis Terra Day	29.03	2022-08-02 11:12:00	31.0	2022-08-02 11:01:00
4	modis Terra Day	31.49	2022-08-11 11:06:00	36.5	2022-08-11 11:01:00
5	modis Terra Day	22.07	2022-08-12 10:12:00	27.0	2022-08-12 10:01:00
6	modis Terra Day	31.83	2022-08-15 10:42:00	36.5	2022-08-15 11:01:00
7	modis Terra Day	24.39	2022-08-18 11:12:00	21.0	2022-08-18 11:01:00
8	modis Terra Day	24.39	2022-08-22 10:48:00	26.5	2022-08-22 11:01:00
9	modis Terra Day	27.29	2022-08-24 10:36:00	29.0	2022-08-24 11:01:00

RMSE: 3.577113982590497
MBE: -2.5833333333333335

Рисунок Г4 – Отчёт в PDF-формате с совпавшими строками

После RMSE и MBE находится отдельный график, изображающий спутниковые и наземные значения. Он представлен на рисунке Г5.

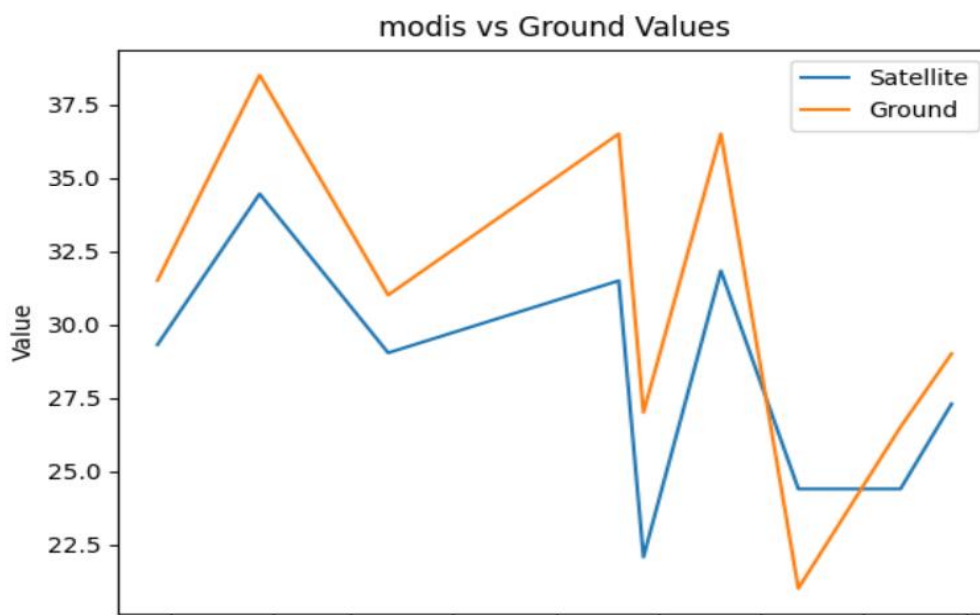


Рисунок Г5 – График из PDF-отчёта

Помимо этого, формируется Excel-таблица, позволяющая исследователям более гибко работать с полученными значениями. Она представлена на рисунке Г6.

Clear_day_cov	Clear_night_cov	view_angle	view_time	Emis_31	Emis_32	T_Day_1kt	Night_1kt	ht_view	ht_view_t	QC_Day	QC_Night	date	source	Common Date	value	round Value	Ground Date
1.54	0.88	54.00	11:42:00	0.98	0.99	21.93	5.89	-32.00	21:18:00	0	65	2022-07-21	Satellite	2022-07-24 11:18:00	29.31	4.5	2022-07-15 01:01:00
1.97	1.00	32.00	11:18:00	0.98	0.99	29.31	0.81	57.00	22:30:00	0	65	2022-07-21	Ground	2022-07-24 11:01:00	31.5	4.5	2022-07-15 02:01:00
1.00	1.00	-3.00	10:54:00	0.98	0.99	34.45	8.43	37.00	22:06:00	0	65	2022-07-21	Satellite	2022-07-28 10:54:00	34.45	3.5	2022-07-15 03:01:00
1.62	2.00	25.00	11:12:00	0.98	0.99	29.03	7.45	53.00	22:24:00	0	65	2022-08-01	Ground	2022-07-28 11:01:00	38.5	2.5	2022-07-15 04:01:00
0.98	0.98	-29.00	10:36:00	0.98	0.99	29.89	10.53	13.00	21:48:00	0	65	2022-08-01	Satellite	2022-08-02 11:12:00	29.03	2	2022-07-15 05:01:00
0.99	0.93	16.00	11:06:00	0.98	0.99	31.49	6.97	49.00	22:18:00	0	65	2022-08-11	Ground	2022-08-02 11:01:00	31	2	2022-07-15 06:01:00
2.00	1.00	-54.00	10:12:00	0.98	0.99	22.07	10.17	-24.00	21:24:00	0	65	2022-08-11	Satellite	2022-08-11 11:06:00	31.49	4	2022-07-15 07:01:00
1.00	1.00	-21.00	10:42:00	0.98	0.99	31.83	7.83	22.00	21:54:00	0	65	2022-08-11	Ground	2022-08-11 11:01:00	36.5	7.5	2022-07-15 08:01:00
1.64	0.66	39.00	11:24:00	0.98	0.99	30.91	4.51	61.00	22:36:00	0	65	2022-08-11	Satellite	2022-08-12 10:12:00	22.07	12	2022-07-15 09:01:00
1.00	0.66	-37.00	10:30:00	0.98	0.99	28.83	4.77	3.00	21:42:00	0	65	2022-08-11	Ground	2022-08-12 10:01:00	27	16.5	2022-07-15 10:01:00
1.00	1.99	25.00	11:12:00	0.98	0.99	24.39	-0.19	-58.00	20:48:00	0	65	2022-08-11	Satellite	2022-08-15 10:42:00	31.83	19.5	2022-07-15 11:01:00
0.99	0.71	-12.00	10:48:00	0.98	0.99	24.39	3.05	30.00	22:00:00	0	65	2022-08-21	Ground	2022-08-15 11:01:00	36.5	22.5	2022-07-15 12:01:00
1.00	0.99	-29.00	10:36:00	0.98	0.99	27.29	5.13	13.00	21:48:00	0	65	2022-08-21	Satellite	2022-08-18 11:12:00	24.39	26.5	2022-07-15 13:01:00
													Ground	2022-08-18 11:01:00	21	30	2022-07-15 14:01:00
													Satellite	2022-08-22 10:48:00	24.39	30.5	2022-07-15 15:01:00
													Ground	2022-08-22 11:01:00	26.5	28	2022-07-15 16:01:00
													Satellite	2022-08-24 10:36:00	27.29	23	2022-07-15 17:01:00
													Ground	2022-08-24 11:01:00	29	20.5	2022-07-15 18:01:00

Рисунок Г6 – Сформированная Excel-таблица

На основе всех полученных данных программный модуль выводит графики, которые помогают наглядно оценить значения для Terra/MODIS на основе наземных измерений в точке А1. Графики представлены на рисунке Г7.

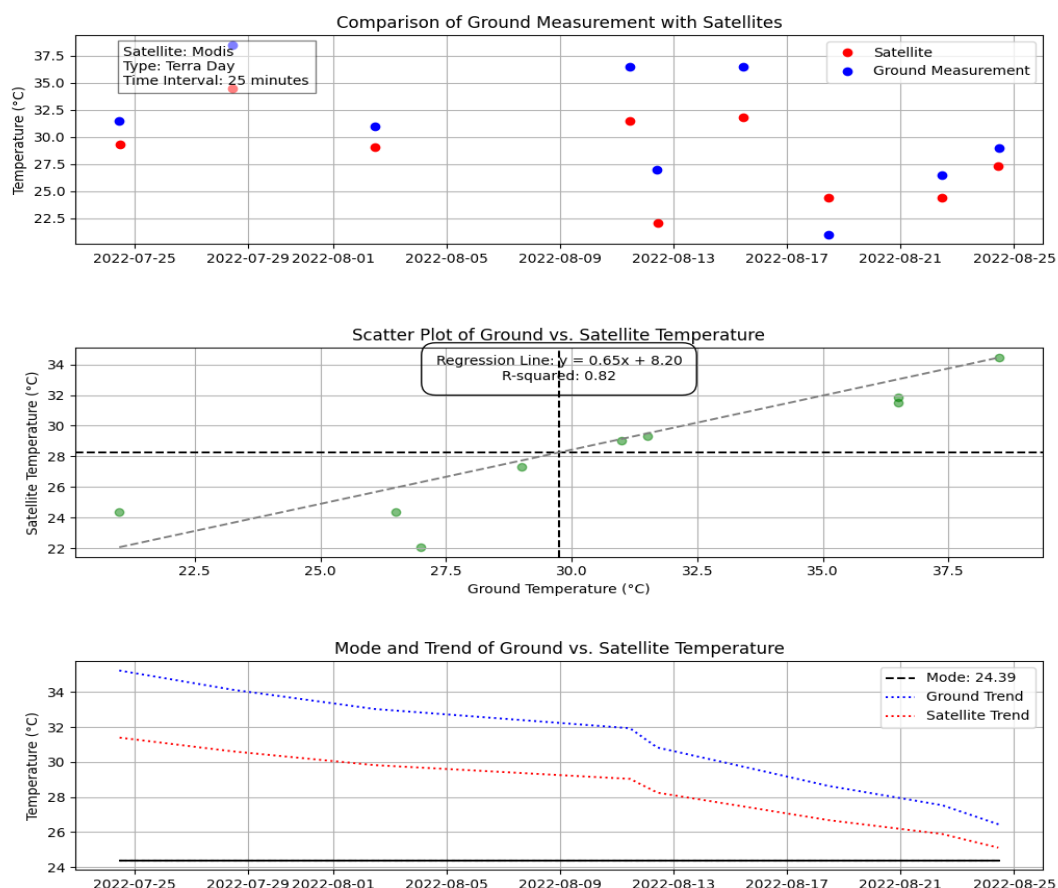


Рисунок Г7 – Графики для значений Terra/MODIS с наземных измерений в точке A1

Эти графики включают в себя три подробных графика в одном файле и еще один в формате PDF для лучшей визуализации. Первый график отображает все совпавшие точки, второй - облако рассеяния значений, а третий создан для отображения тренда для спутниковых и наземных значений, а также моды.

Рассмотрим ночные данные Aqua/MODIS в точке A3 с координатами [88.67678, 49.85173], временным интервалом 120 минут. Отчёт представлен на рисунке Г9.

Продолжение приложения Г

ID	Satellite name	Satellite value	Satellite Datetime	Ground value	Ground Datetime
1	modis Aqua Night	6.49	2022-07-23 00:12:00	11.5	2022-07-23 00:01:00
2	modis Aqua Night	7.35	2022-07-27 03:00:00	8.0	2022-07-27 03:01:00
3	modis Aqua Night	9.15	2022-07-28 02:06:00	12.5	2022-07-28 02:01:00
4	modis Aqua Night	5.69	2022-08-15 00:12:00	9.0	2022-08-15 00:01:00
5	modis Aqua Night	-1.23	2022-08-21 02:48:00	0.0	2022-08-21 02:01:00
6	modis Aqua Night	1.89	2022-08-22 01:54:00	1.0	2022-08-22 02:01:00
7	modis Aqua Night	3.05	2022-08-24 00:06:00	8.0	2022-08-24 00:01:00
8	modis Aqua Night	2.79	2022-08-25 02:24:00	5.0	2022-08-25 02:01:00
9	modis Aqua Night	4.65	2022-08-27 02:06:00	8.5	2022-08-27 02:01:00
10	modis Aqua Night	2.49	2022-08-29 01:54:00	1.0	2022-08-29 02:01:00
11	modis Aqua Night	3.83	2022-08-30 02:36:00	3.0	2022-08-30 03:01:00
12	modis Aqua Night	4.15	2022-08-31 00:06:00	7.5	2022-08-31 00:01:00
13	modis Aqua Night	5.43	2022-09-05 02:00:00	6.5	2022-09-05 02:01:00
14	modis Aqua Night	5.09	2022-09-06 02:42:00	7.0	2022-09-06 03:01:00
15	modis Aqua Night	9.59	2022-09-07 01:48:00	8.0	2022-09-07 02:01:00
16	modis Aqua Night	4.05	2022-09-10 02:18:00	1.5	2022-09-10 02:01:00
17	modis Aqua Night	-3.41	2022-09-19 02:06:00	0.0	2022-09-19 01:01:00
18	modis Aqua Night	-1.95	2022-09-24 02:24:00	2.5	2022-09-24 02:01:00
19	modis Aqua Night	-2.69	2022-09-25 03:06:00	1.0	2022-09-25 03:01:00

RMSE: 2.966631085928953
MBE: -1.8468421052631576

Рисунок Г9 – Отчёт с ночных данных Aqua/MODIS в точке А3

RMSE и MBE 2.96 и -1.84 для Aqua/MODIS в ночное время также вполне удовлетворяют статистическим показателям, что является подтверждением успешной валидации – сбору значений, сравнений времени и очистке данных. График для Aqua/MODIS представлен на рисунке Г11.

Продолжение приложения Г

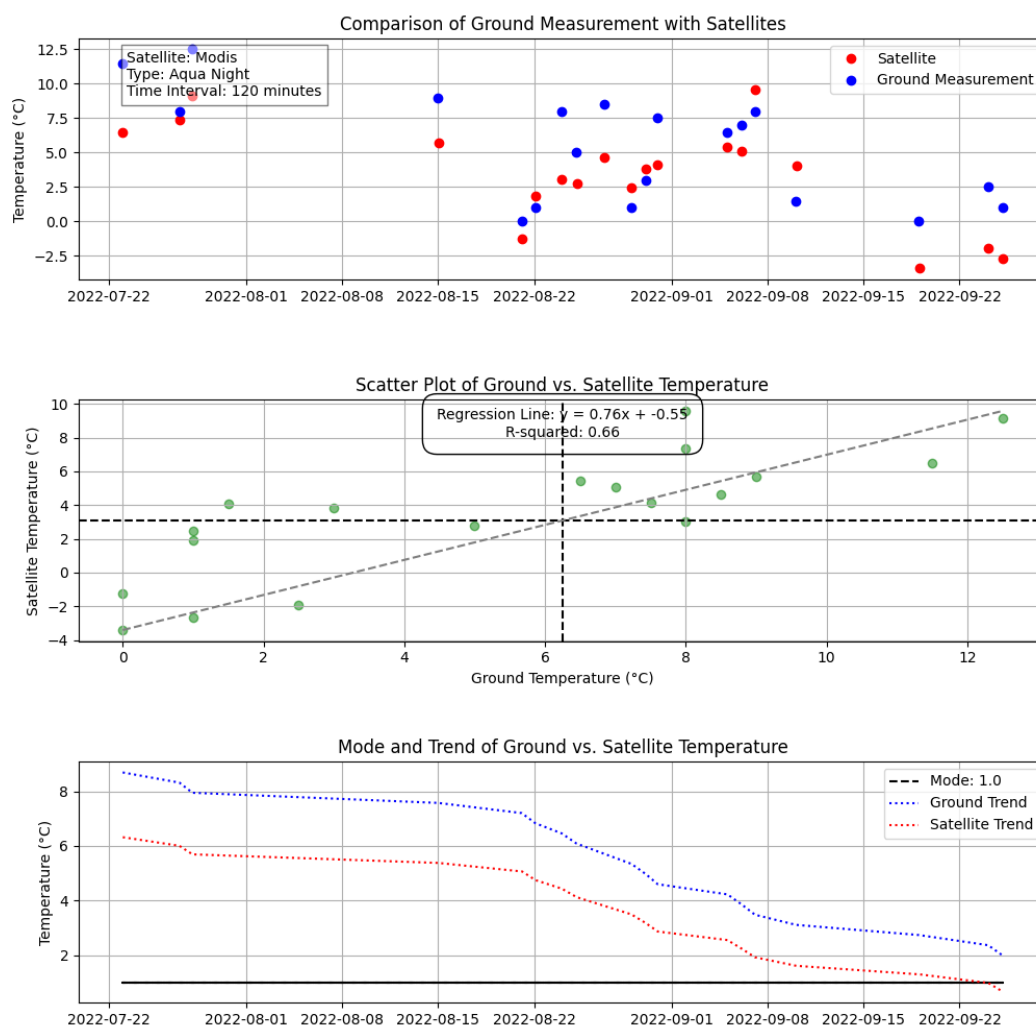


Рисунок Г11 – Графики для Aqua/MODIS в ночное время для точки А3

Landsat не предоставляет посуточное измерение температурных показателей, потому, для него не требуется вводить интервал в минутах. Отчёт для Landsat в точке А4 с координатами [88.69635, 49.96667] представлен на рисунке Г10.

ID	Satellite name	Satellite value	Satellite Datetime	Ground value	Ground Datetime
1	landsat	23.18	2022-08-21 11:54:52	21.0	2022-08-21 12:01:00
2	landsat	32.23	2022-09-06 11:54:54	31.5	2022-09-06 12:01:00
3	landsat	26.03	2022-09-22 11:54:59	25.5	2022-09-22 12:01:00

4	landsat	16.06	2022-10-08 11:54:54	14.0	2022-10-08 12:01:00
5	landsat	6.63	2022-10-24 11:54:56	6.5	2022-10-24 12:01:00
6	landsat	34.07	2022-07-27 12:00:53	34.5	2022-07-27 12:01:00
7	landsat	17.85	2022-09-13 12:01:08	18.5	2022-09-13 12:01:00
8	landsat	18.69	2022-09-29 12:01:09	17.0	2022-09-29 12:01:00
9	landsat	9.69	2022-10-15 12:01:01	10.5	2022-10-15 12:01:00

RMSE: 1.2448248426541335
MBE: 0.6033333333333332

Рисунок Г10 – Отчёт для Landsat в точке А4

RMSE для Landsat 1.24 и MBE 0.60 также показывают точность значений, например, в точке с ID [5], значения со спутника и наземных измерений практически совпали: 6.6 и 6.5 соответственно. Графики для Landsat представлены на рисунке А12.

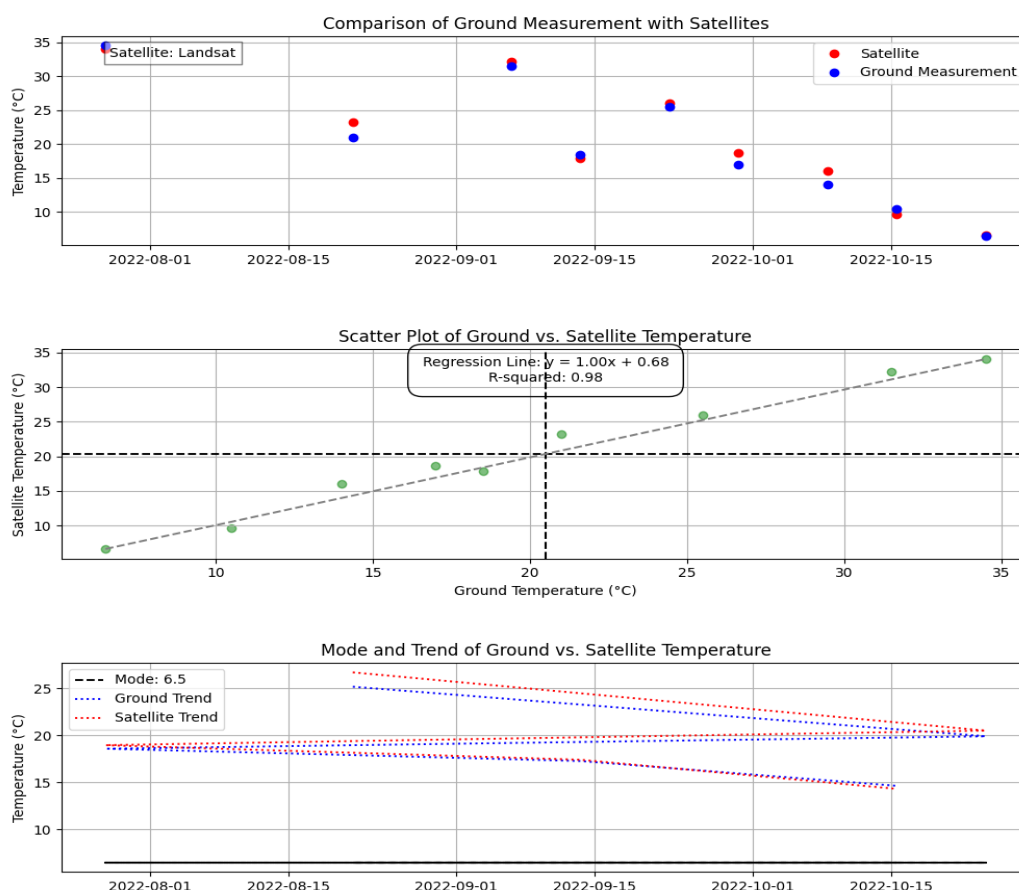


Рисунок А12 – Графики для Landsat в точке А4

Подобные показатели позволяют оценить достоверность полученных значений.

Приложение Д

Фрагмент кода представляет собой реализацию класса TerraDataManager, который взаимодействует с Google Earth Engine API для извлечения и обработки данных с Terra/MODIS.

```
import ee
import pandas as pd

class TerraDataManager:
    __instance = None

    def get_image_collection(self, coordinates, date_start, date_end):
        point = ee.Geometry.Point(coordinates)
        transformed_point = point.transform('SR-ORG:6974', 1000)

        def mask_clouds(image):
            # Извлечение QC битов
            qc = image.select('QC_Day')
            cloud_mask = qc.bitwiseAnd(1 << 0).eq(0) # Бит 0: облака
            image = image.updateMask(cloud_mask) # Применение маски облаков
            return image

        lst = ee.ImageCollection('MODIS/061/MOD11A1') \
            .filterBounds(transformed_point) \
            .filterDate(date_start, date_end) \
            .map(mask_clouds) # Применение маски облачности

        return lst

    def sample_image(self, image, point):
        image = image.addBands(image.metadata("system:time_start"))
        image =
image.addBands(image.select('LST_Day_1km').multiply(0.02).subtract(273.15),
overwrite=True)
        image =
image.addBands(image.select('LST_Night_1km').multiply(0.02).subtract(273.15),
overwrite=True)
        image = image.addBands(image.select('Night_view_time').multiply(0.1),
overwrite=True)
        image = image.addBands(image.select('Day_view_time').multiply(0.1),
overwrite=True)
        image = image.addBands(image.select('Night_view_angle').add(-65),
overwrite=True)
        image = image.addBands(image.select('Day_view_angle').add(-65),
overwrite=True)
        image =
image.addBands(image.select('Emis_31').multiply(0.002).add(0.49),
overwrite=True)
        image =
image.addBands(image.select('Emis_32').multiply(0.002).add(0.49),
overwrite=True)
        image =
image.addBands(image.select('Clear_day_cov').multiply(0.0005),
overwrite=True)
        image =
image.addBands(image.select('Clear_night_cov').multiply(0.0005),
overwrite=True)
        return image.sampleRegions(collection=point, scale=1000)
```

```

def get_feature_collection(self, lst, point):
    featureCollection = ee.FeatureCollection(lst.map(lambda image:
self.sample_image(image, point))).flatten()
    return featureCollection

def replace_numbers_by_strings(self, feature):
    def as_time_string(v):
        return ee.String(v.floor().int().format('%02d')) \
            .cat(':') \
            .cat(ee.String(v.subtract(v.floor()).multiply(60).int().format('%02d'))) \
            .cat(':00')

    d = ee.Date(feature.get('system:time_start'))
    dt = as_time_string(ee.Number(feature.get('Day_view_time')))
    nt = as_time_string(ee.Number(feature.get('Night_view_time')))
    dtemp = ee.Number(feature.get('LST_Day_1km')).format('%6.2f')
    ntemp = ee.Number(feature.get('LST_Night_1km')).format('%6.2f')
    emis31 = ee.Number(feature.get('Emis_31')).format('%6.2f')
    emis32 = ee.Number(feature.get('Emis_32')).format('%6.2f')
    day_cov = ee.Number(feature.get('Clear_day_cov')).format('%6.2f')
    night_cov = ee.Number(feature.get('Clear_night_cov')).format('%6.2f')
    day_angle = ee.Number(feature.get('Day_view_angle')).format('%6.2f')
    night_angle =
ee.Number(feature.get('Night_view_angle')).format('%6.2f')
    return feature.set({
        'Night_view_time': nt,
        'Day_view_time': dt,
        'Night_view_angle': night_angle,
        'Day_view_angle': day_angle,
        'LST_Day_1km': dtemp,
        'LST_Night_1km': ntemp,
        'Emis_31': emis31,
        'Emis_32': emis32,
        'Clear_day_cov': day_cov,
        'Clear_night_cov': night_cov,
        'date': d.format('YYYY-MM-dd')
    })

def get_datatable(self, featureCollection):
    columns = [
        "LST_Day_1km",
        "QC_Day",
        "Day_view_time",
        "Day_view_angle",
        "LST_Night_1km",
        "QC_Night",
        "Night_view_time",
        "Night_view_angle",
        "Emis_31",
        "Emis_32",
        "Clear_day_cov",
        "Clear_night_cov",
        "date",
        "name"
    ]

    modis_datatable =
featureCollection.map(self.replace_numbers_by_strings).select(columns)
    return modis_datatable

```

```

def get_image_data(self, coordinates, date_start, date_end):
    lst = self.get_image_collection(coordinates, date_start, date_end)
    return lst

def get_feature_data(self, lst, coordinates):
    point = ee.Geometry.Point(coordinates)
    featureCollection = self.get_feature_collection(lst, point)
    return featureCollection

def create_dataframe(self, featureCollection):
    datatable = self.get_datatable(featureCollection)
    if datatable is not None:
        df = pd.DataFrame([feature['properties'] for feature in
            datatable.getInfo()['features']])
        return df
    else:
        print("Feature collection is empty.")
        return None

def dataframe_to_dict(df):
    if df is not None:
        return df.to_dict(orient='records')
    else:
        return None

```

1. Метод `get_image_collection` получает коллекцию изображений MODIS для указанных координат и диапазона дат.
2. Метод `sample_image` применяет необходимые преобразования к изображению, добавляет дополнительные полосы данных и выполняет выборку регионов.
3. Метод `get_feature_collection` создает коллекцию объектов с данными для изображений.
4. Метод `replace_numbers_by_strings` заменяет числовые значения на строки и форматирует время.
5. Метод `get_datatable` формирует таблицу данных MODIS.
6. Методы `get_image_data`, `get_feature_data` и `create_dataframe` обеспечивают получение и обработку данных изображений MODIS, создание коллекции объектов и создание `DataFrame` на основе данных MODIS, соответственно. Также в коде присутствуют две вспомогательные функции: `dataframe_to_dict`, которая конвертирует `DataFrame` в словарь, и `replace_numbers_by_strings`, которая заменяет числовые значения на строки и форматирует время.

Для Aqua/Modis аналогичный код:

```

import ee
import pandas as pd

```



```

from datetime import datetime

class AquaDataManager:
    def __init__(self):
        self.modis_product = 'MODIS/006/MYD11A1'

    def get_image_collection(self, coordinates, date_start, date_end):
        point = ee.Geometry.Point(coordinates)
        lst = ee.ImageCollection(self.modis_product) \
            .filterBounds(point) \
            .filterDate(date_start, date_end)
        return lst

    def mask_clouds(self, image):
        QC_Day = image.select('QC_Day')
        QC_Night = image.select('QC_Night')
        cloud_mask_day = QC_Day.bitwiseAnd(0b11).eq(0)
        cloud_mask_night = QC_Night.bitwiseAnd(0b11).eq(0)
        return image.updateMask(cloud_mask_day).updateMask(cloud_mask_night)

    def sample_image(self, image, point):
        image = self.mask_clouds(image)
        image = image.addBands(image.metadata("system:time_start"))
        image =
image.addBands(image.select('LST_Day_1km').multiply(0.02).subtract(273.15),
overwrite=True)
        image =
image.addBands(image.select('LST_Night_1km').multiply(0.02).subtract(273.15),
overwrite=True)
        image = image.addBands(image.select('Night_view_time').multiply(0.1),
overwrite=True)
        image = image.addBands(image.select('Day_view_time').multiply(0.1),
overwrite=True)
        image = image.addBands(image.select('Night_view_angle').add(-65),
overwrite=True)
        image = image.addBands(image.select('Day_view_angle').add(-65),
overwrite=True)
        image =
image.addBands(image.select('Emis_31').multiply(0.002).add(0.49),
overwrite=True)
        image =
image.addBands(image.select('Emis_32').multiply(0.002).add(0.49),
overwrite=True)
        image = image.addBands(image.select('Clear_day_cov').multiply(0.005),
overwrite=True)
        image =
image.addBands(image.select('Clear_night_cov').multiply(0.005),
overwrite=True)
        return image.sampleRegions(collection=point, scale=1000)

    def get_feature_collection(self, lst, point):
        featureCollection = ee.FeatureCollection(lst.map(lambda image:
self.sample_image(image, point))).flatten()
        return featureCollection

    def replace_numbers_by_strings(self, feature):
        def as_time_string(v):
            return ee.String(v.floor().int().format('%02d')) \
                .cat(':') \
                .cat(ee.String(v.subtract(v.floor()).multiply(60).int().format('%02d'))) \

```

```

        .cat(':00')

    d = ee.Date(feature.get('system:time_start'))
    dt = as_time_string(ee.Number(feature.get('Day_view_time')))
    nt = as_time_string(ee.Number(feature.get('Night_view_time')))
    dtemp = ee.Number(feature.get('LST_Day_1km')).format('%6.2f')
    ntemp = ee.Number(feature.get('LST_Night_1km')).format('%6.2f')
    emis31 = ee.Number(feature.get('Emis_31')).format('%6.2f')
    emis32 = ee.Number(feature.get('Emis_32')).format('%6.2f')
    day_cov = ee.Number(feature.get('Clear_day_cov')).format('%6.2f')
    night_cov = ee.Number(feature.get('Clear_night_cov')).format('%6.2f')
    day_angle = ee.Number(feature.get('Day_view_angle')).format('%6.2f')
    night_angle =
ee.Number(feature.get('Night_view_angle')).format('%6.2f')
    return feature.set({
        'Night_view_time': nt,
        'Day_view_time': dt,
        'Night_view_angle': night_angle,
        'Day_view_angle': day_angle,
        'LST_Day_1km': dtemp,
        'LST_Night_1km': ntemp,
        'Emis_31': emis31,
        'Emis_32': emis32,
        'Clear_day_cov': day_cov,
        'Clear_night_cov': night_cov,
        'date': d.format('YYYY-MM-dd')
    })

def get_datatable(self, featureCollection):
    columns = [
        "LST_Day_1km",
        "QC_Day",
        "Day_view_time",
        "Day_view_angle",
        "LST_Night_1km",
        "QC_Night",
        "Night_view_time",
        "Night_view_angle",
        "Emis_31",
        "Emis_32",
        "Clear_day_cov",
        "Clear_night_cov",
        "date",
        "name"
    ]

    aqua_datatable =
featureCollection.map(self.replace_numbers_by_strings).select(columns)
    return aqua_datatable

def get_image_data(self, coordinates, date_start, date_end):
    lst = self.get_image_collection(coordinates, date_start, date_end)
    return lst

def get_feature_data(self, lst, coordinates):
    point = ee.Geometry.Point(coordinates)
    featureCollection = self.get_feature_collection(lst, point)
    return featureCollection

def create_dataframe(self, featureCollection):
    datatable = self.get_datatable(featureCollection)

```

```

    if datatable is not None:
        df = pd.DataFrame([feature['properties'] for feature in
            datatable.getInfo()['features']])
        return df
    else:
        print("Feature collection is empty.")
        return None

```

Для Landsat:

```

class LandsatDataManager:
    __instance = None

    def get_image_collection(self, coordinates, date_start, date_end):
        point = ee.Geometry.Point(coordinates)
        lst = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') \
            .filterBounds(point) \
            .filterDate(date_start, date_end)
        return lst

    def mask_clouds(self, image):
        qa = image.select('QA_PIXEL')
        cloud_mask = qa.bitwiseAnd(1 << 3).eq(0)
        cirrus_mask = qa.bitwiseAnd(1 << 4).eq(0)
        combined_mask = cloud_mask.And(cirrus_mask)
        return image.updateMask(combined_mask)

    def sample_image(self, image, point):
        image = self.mask_clouds(image)
        image = image.addBands(image.metadata("system:time_start"))
        image =
image.addBands(image.select('ST_B10').multiply(0.00341802).subtract(273.15).a
dd(149), overwrite=True)
        image = image.addBands(image.select('ST_ATRAN').multiply(0.0001),
overwrite=True)
        image = image.addBands(image.select('ST_CDIST').multiply(0.01),
overwrite=True)
        image = image.addBands(image.select('ST_DRAD').multiply(0.001),
overwrite=True)
        image = image.addBands(image.select('ST_EMIS').multiply(0.0001),
overwrite=True)
        image = image.addBands(image.select('ST_EMSD').multiply(0.0001),
overwrite=True)
        image = image.addBands(image.select('ST_QA').multiply(0.01),
overwrite=True)
        image = image.addBands(image.select('ST_TRAD').multiply(0.001),
overwrite=True)
        image = image.addBands(image.select('ST_URAD').multiply(0.001),
overwrite=True)
        image = image.addBands(image.select('QA_PIXEL'), overwrite=True)
        return image.sampleRegions(collection=point, scale=1000)

    def get_feature_collection(self, lst, point):
        featureCollection = ee.FeatureCollection(lst.map(lambda image:
self.sample_image(image, point))).flatten()
        return featureCollection

    def replace_numbers_by_strings(self, feature):
        d = ee.Date(feature.get('system:time_start')).advance(7,
'hour').format('YYYY-MM-dd HH:mm:ss')

```

```

st_b10 = ee.Number(feature.get('ST_B10')).format('%6.2f')
st_atran = ee.Number(feature.get('ST_ATRAN')).format('%6.2f')
st_cdists = ee.Number(feature.get('ST_CDIST')).format('%6.2f')
st_drad = ee.Number(feature.get('ST_DRAD')).format('%6.2f')
st_emis = ee.Number(feature.get('ST_EMIS')).format('%6.2f')
st_emsd = ee.Number(feature.get('ST_EMSD')).format('%6.2f')
st_qa = ee.Number(feature.get('ST_QA')).format('%6.2f')
st_trad = ee.Number(feature.get('ST_TRAD')).format('%6.2f')
st_urad = ee.Number(feature.get('ST_URAD')).format('%6.2f')
return feature.set({
    'date': d,
    'ST_B10': st_b10,
    'ST_ATRAN': st_atran,
    'ST_CDIST': st_cdists,
    'ST_DRAD': st_drad,
    'ST_EMIS': st_emis,
    'ST_EMSD': st_emsd,
    'ST_QA': st_qa,
    'ST_TRAD': st_trad,
    'ST_URAD': st_urad
})

def get_datatable(self, featureCollection):
    columns = [
        "date",
        "ST_B10",
        "ST_ATRAN",
        "ST_CDIST",
        "ST_DRAD",
        "ST_EMIS",
        "ST_EMSD",
        "ST_QA",
        "ST_TRAD",
        "ST_URAD",
        "name"
    ]

    Landsat_datatable =
featureCollection.map(self.replace_numbers_by_strings).select(columns)
    return Landsat_datatable

def get_image_data(self, coordinates, date_start, date_end):
    lst = self.get_image_collection(coordinates, date_start, date_end)
    return lst

def get_feature_data(self, lst, coordinates):
    point = ee.Geometry.Point(coordinates)
    featureCollection = self.get_feature_collection(lst, point)
    return featureCollection

def create_dataframe(self, featureCollection):
    datatable = self.get_datatable(featureCollection)
    if datatable is not None:
        df = pd.DataFrame([feature['properties'] for feature in
datatable.getInfo()['features']])
        return df
    else:
        print("Feature collection is empty.")
        return None

```

Фрагмент кода реализует класс `ExcelManager`, предназначенный для работы с данными, хранящимися в Excel-файле.

```
class ExcelManager:
    def __init__(self):
        self.file_path = None
        self.data = None
        self.date_start = None
        self.date_end = None

    def select_excel_file(self):
        root = tk.Tk()
        root.withdraw()
        self.file_path = filedialog.askopenfilename(filetypes=[("Excel
files", "*.xlsx;*.xls")])
        return self.file_path

    def read_excel(self):
        if self.file_path:
            try:
                df = pd.read_excel(self.file_path, names=['date', 'time',
'value'])

                df['date'] = df['date'].astype(str)
                df['time'] = df['time'].astype(str)

                df['datetime'] = df['date'] + ' ' + df['time']

                df['datetime'] = pd.to_datetime(df['datetime'], format='%Y-
%m-%d %H:%M:%S')

                self.date_start = df['datetime'].min()
                self.date_end = df['datetime'].max()

                df.drop(['date', 'time'], axis=1, inplace=True)

                self.data = df

                return True
            except Exception as e:
                print("Error reading Excel file:", e)
                return False
        else:
            print("No Excel file selected.")
            return False

    def print_data(self):
        if self.data is not None:
            print("Data from Excel:")
            print(self.data.to_string(index=False))
        else:
            print("No data available.")
```

В конструкторе класса инициализируются основные переменные: `file_path` для хранения пути к Excel-файлу, `data` для хранения загруженных данных, а также `date_start` и `date_end` для хранения начальной и конечной даты данных.

1. Метод `select_excel_file` открывает диалоговое окно для выбора Excel-файла и возвращает путь к выбранному файлу.
2. Метод `read_excel` считывает данные из выбранного Excel-файла. Если файл успешно загружен, то он считывается в `DataFrame` `df`, затем происходит преобразование столбцов `'date'` и `'time'` в строковый тип, объединение их в столбец `'datetime'` и преобразование в объект `datetime`. Далее определяются начальная и конечная даты данных. Удаляются столбцы `'date'` и `'time'`, после чего `DataFrame` сохраняется в переменную `self.data`.
3. Метод `print_data` печатает данные, если они доступны, или выводит сообщение о их отсутствии.

Этот фрагмент кода представляет собой реализацию класса `CommonDates`, который осуществляет сопоставление данных, полученных с различных спутниковых источников с данными, полученными с земли в определенные моменты времени.

```
class CommonDates:
    def __init__(self, df, excel_data, satellite, satellite_product=None,
time_of_day=None):
        self.excel_data = excel_data
        self.satellite_name = satellite
        self.satellite_product = satellite_product
        self.time_of_day = time_of_day

        if satellite == 'modis':
            if satellite_product == 'aqua':
                self.satellite_column = 'LST_Day_1km' if time_of_day == 'day'
            else 'LST_Night_1km'
            self.view_time_column = 'Day_view_time' if time_of_day ==
'day' else 'Night_view_time'
            elif satellite_product == 'terra':
                self.satellite_column = 'LST_Day_1km' if time_of_day == 'day'
            else 'LST_Night_1km'
            self.view_time_column = 'Day_view_time' if time_of_day ==
'day' else 'Night_view_time'
            else:
                raise ValueError("Invalid satellite_product. Please select
'aqua' or 'terra'.")
        elif satellite == 'Landsat':
            self.satellite_column = 'ST_B10'
            self.view_time_column = None
        else:
            raise ValueError("Invalid satellite_name. Please select 'modis'
or 'Landsat'.")

        # Проверка наличия необходимых колонок
        required_columns = ['date', self.satellite_column]
        if self.view_time_column:
            required_columns.append(self.view_time_column)

        missing_columns = [col for col in required_columns if col not in
df.columns]
        if missing_columns:
            raise KeyError(f"One or more columns {missing_columns} are
missing from the DataFrame")
```

```

        if self.view_time_column:
            self.selected_column = df[['date', self.satellite_column,
self.view_time_column]]
        else:
            self.selected_column = df[['date', self.satellite_column]]

    def _parse_date(self, date_string, time_string):
        if time_string:
            return datetime.strptime(date_string + ' ' + time_string, '%Y-%m-
%d %H:%M:%S')
        else:
            return datetime.strptime(date_string, '%Y-%m-%d %H:%M:%S')

    def _find_matching_rows(self, satellite_datetime):
        matching_rows = self.excel_data.iloc[(self.excel_data['datetime'] -
satellite_datetime).abs().argsort()[:1]]
        return matching_rows

    def _calculate_daily_average(self):
        daily_averages = defaultdict(list)
        for index, row in self.excel_data.iterrows():
            date = row['datetime'].date()
            daily_averages[date].append(row['value'])
        for date, values in daily_averages.items():
            daily_averages[date] = sum(values) / len(values)
        return daily_averages

    def _remove_outliers(self, data):
        mean = np.mean(data)
        std_dev = np.std(data)
        filtered_data = [x for x in data if (mean - 3 * std_dev) <= x <=
(mean + 3 * std_dev)]
        return filtered_data

    def _remove_obvious_outliers(self, data):
        filtered_data = [x for x in data if 0 <= x <= 50] # Допустимый
диапазон температур
        filtered_data = [filtered_data[i] for i in range(len(filtered_data) -
1) if
                        abs(filtered_data[i] - filtered_data[i + 1]) <= 10]
        return filtered_data

    def _remove_matches_outliers(self, matches):
        values = [float(match['value']) for match in matches]
        mean = np.mean(values)
        std_dev = np.std(values)
        cleaned_matches = [match for match in matches if
                        (mean - 3 * std_dev) <= float(match['value']) <=
(mean + 3 * std_dev)]
        return cleaned_matches

    def _remove_matches_based_on_difference(self, matches):
        differences = []
        for i in range(0, len(matches), 2):
            diff = abs(float(matches[i]['value']) - float(matches[i +
1]['value']))
            differences.append(diff)

        std_dev = np.std(differences)
        threshold = 3 * std_dev

```

```

        cleaned_matches = []
        for i in range(0, len(matches), 2):
            if abs(float(matches[i]['value']) - float(matches[i + 1]['value'])) <= threshold:
                cleaned_matches.extend([matches[i], matches[i + 1]])

        return cleaned_matches

    def match_data(self, time_interval_minutes=None):
        matches = []

        # Сначала удаляем явные выбросы
        cleaned_values =
self._remove_obvious_outliers(self.excel_data['value'].tolist())
        removed_values = set(self.excel_data['value']) - set(cleaned_values)
        self.excel_data =
self.excel_data[self.excel_data['value'].isin(cleaned_values)]

        # Затем применяем стандартное удаление выбросов через 3 сигмы
        cleaned_values =
self._remove_outliers(self.excel_data['value'].tolist())
        removed_values = set(self.excel_data['value']) - set(cleaned_values)
        self.excel_data =
self.excel_data[self.excel_data['value'].isin(cleaned_values)]

        daily_averages = self._calculate_daily_average()

        for index, row in self.selected_column.iterrows():
            satellite_date = row['date']
            satellite_time = None
            if self.view_time_column:
                satellite_time = row[self.view_time_column]
            satellite_datetime = self._parse_date(satellite_date,
satellite_time)
            matching_rows = self._find_matching_rows(satellite_datetime)

            if matching_rows.empty:
                continue

            for _, match_row in matching_rows.iterrows():
                ground_datetime = match_row['datetime']
                if time_interval_minutes is not None:
                    time_diff = abs(ground_datetime - satellite_datetime)
                    if time_diff <= timedelta(minutes=time_interval_minutes):
                        if self.satellite_product:
                            satellite_product_str =
self.satellite_product.capitalize()
                        else:
                            satellite_product_str = ''
                        if self.time_of_day:
                            time_of_day_str = self.time_of_day.capitalize()
                        else:
                            time_of_day_str = ''
                        source = f'Satellite - {self.satellite_name}
{satellite_product_str} {time_of_day_str}'
                        value = row[self.satellite_column]
                        matches.append({'source': source, 'datetime':
satellite_datetime, 'value': float(value)})
                        matches.append({'source': 'Ground', 'datetime':
ground_datetime, 'value': float(match_row['value'])})
                        else:

```



```

        if self.satellite_product:
            satellite_product_str =
self.satellite_product.capitalize()
        else:
            satellite_product_str = ''
        if self.time_of_day:
            time_of_day_str = self.time_of_day.capitalize()
        else:
            time_of_day_str = ''
        source = f'Satellite - {self.satellite_name}
{satellite_product_str} {time_of_day_str}'
        value = row[self.satellite_column]
        matches.append({'source': source, 'datetime':
satellite_datetime, 'value': float(value)})
        matches.append({'source': 'Ground', 'datetime':
ground_datetime, 'value': float(match_row['value'])})

        # Удаление выбросов из matches
        matches = self._remove_matches_outliers(matches)

        # Удаление выбросов на основе разницы значений
        matches = self._remove_matches_based_on_difference(matches)

    return matches, daily_averages

```

В конструкторе класса устанавливаются основные параметры, такие как название спутника (`satellite`), продукт спутника (`satellite_product`), время суток (`time_of_day`) и переданные данные в формате `DataFrame` (`df`).

Метод `_parse_date` используется для преобразования строкового представления даты и времени в объект `datetime`.

Метод `_find_matching_rows` находит строки из данных земной поверхности, соответствующие переданному времени спутника.

Метод `match_data` выполняет сопоставление данных спутника и данных земной поверхности на основе переданных параметров и времени суток. Если данные спутника соответствуют данным земной поверхности в пределах временного окна в минутах, они считаются сопоставленными. Полученные сопоставления возвращаются в виде списка словарей, содержащих источник данных, дату и время, а также соответствующие значения.

Также в классе реализованы методы для удаления выбросов из данных, включая методы `_remove_outliers` и `_remove_obvious_outliers` для удаления выбросов из данных земной поверхности, а также методы для очистки сопоставленных данных о выбросах.

Этот фрагмент кода представляет функцию `calculate_rmse_mbe`, которая используется для вычисления среднеквадратичной ошибки (RMSE) и средней ошибки (МБЕ) на основе совпадающих значений, полученных от спутника и с земли.

```

def calculate_rmse_mbe(matches):
    satellite_values = []

```

```

ground_values = []

idx = 0
while idx < len(matches):
    satellite_value = float(matches[idx]['value'])
    ground_value = float(matches[idx + 1]['value'])

    satellite_values.append(satellite_value)
    ground_values.append(ground_value)

    idx += 2

rmse = np.sqrt(np.mean((np.array(satellite_values) -
np.array(ground_values))**2))
mbe = np.mean(np.array(satellite_values) - np.array(ground_values))

return rmse, mbe

```

Функция `calculate_rmse_mbe` принимает список совпадающих значений (`matches`), содержащих значения как от спутника, так и с земли.

Функция `plot_data` предназначена для визуализации данных, сопоставленных между спутниковыми и земными измерениями.

```

def plot_data(satellite_name, matches, time_interval_minutes=None,
satellite_product=None, time_of_day=None):
    if matches:
        # Создание фигуры с несколькими графиками
        fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(12, 12),
gridspec_kw={'hspace': 0.5}) # Увеличенное вертикальное расстояние между
графиками

        # Первый график: наземные и спутниковые данные
        ax1 = axes[0]
        satellite_x = [] # Данные для спутниковых значений
        satellite_y = []

        ground_x = [] # Данные для наземных значений
        ground_y = []

        for match in matches:
            if match['source'].startswith('Satellite'):
                satellite_x.append(pd.to_datetime(match['datetime']))
                satellite_y.append(float(match['value']))
            else:
                ground_x.append(pd.to_datetime(match['datetime']))
                ground_y.append(float(match['value']))
            min_len = min(len(satellite_x), len(ground_x),
len(satellite_y), len(ground_y))
            satellite_x = satellite_x[:min_len]
            satellite_y = satellite_y[:min_len]
            ground_x = ground_x[:min_len]
            ground_y = ground_y[:min_len]

        # Построение точек для спутниковых значений на первом графике
        ax1.plot(satellite_x, satellite_y, marker='o', linestyle='',
color='red', label='Satellite')
        ax1.plot(ground_x, ground_y, marker='o', linestyle='', color='blue',
label='Ground Measurement')

```

```

ax1.set_ylabel('Temperature (°C)')
ax1.set_title('Comparison of Ground Measurement with Satellites')
ax1.grid(True)
ax1.legend()

# Формирование информации о спутнике и данных
satellite_info = f'Satellite: {satellite_name.capitalize()}'
if satellite_name.lower() == 'modis' and satellite_product and
time_of_day:
    satellite_info += f'\nType: {satellite_product.capitalize()}'
{time_of_day.capitalize()}'
    satellite_info += f'\nTime Interval: {time_interval_minutes}
minutes'

# Добавляем информацию о спутнике на график
ax1.text(0.05, 0.95, satellite_info, transform=ax1.transAxes,
        verticalalignment='top', bbox=dict(facecolor='white',
alpha=0.5))

# Второй график: облако рассеяния
ax2 = axes[1]

ax2.scatter(ground_y, satellite_y, color='green', alpha=0.5)
ax2.axhline((max(satellite_y) + min(satellite_y)) / 2, color='black',
linestyle='--')
ax2.axvline((max(ground_y) + min(ground_y)) / 2, color='black',
linestyle='--')

# Добавляем диагональную ось
ax2.plot([min(ground_y), max(ground_y)], [min(satellite_y),
max(satellite_y)], color='gray', linestyle='--')

# Вычисляем коэффициенты регрессии
slope, intercept, r_value, p_value, std_err = linregress(ground_y,
satellite_y)

# Создаем уравнение регрессии в виде строки
regression_eqn = f'Regression Line: y = {slope:.2f}x +
{intercept:.2f}\nR-squared: {r_value ** 2:.2f}'

# Добавляем уравнение регрессии на график
ax2.text(0.5, 0.9, regression_eqn, horizontalalignment='center',
verticalalignment='center',
        transform=ax2.transAxes, fontsize=10,
bbox=dict(facecolor='none', edgecolor='black', boxstyle='round,pad=1'))

ax2.set_xlabel('Ground Temperature (°C)')
ax2.set_ylabel('Satellite Temperature (°C)')
ax2.set_title('Scatter Plot of Ground vs. Satellite Temperature')
ax2.grid(True)

# Третий график: мода и тренд
ax3 = axes[2]

# Собираем все данные для определения моды
all_temperatures = ground_y + satellite_y

# Вычисляем моду для всех данных
mode_all = pd.Series(all_temperatures).mode()[0]

# Для всех данных

```

```

ax3.plot(ground_x + satellite_x, [mode_all] * (len(ground_x) +
len(satellite_x)), linestyle='--', color='black', label=f'Mode: {mode_all}')

# Тренд
trend_ground = np.polyfit(range(len(ground_y)), ground_y, 1)
trend_satellite = np.polyfit(range(len(satellite_y)), satellite_y, 1)
ax3.plot(ground_x, np.polyval(trend_ground, range(len(ground_y))),
linestyle=':', color='blue', label='Ground Trend')
ax3.plot(satellite_x, np.polyval(trend_satellite,
range(len(satellite_y))), linestyle=':', color='red', label='Satellite
Trend')

ax3.set_ylabel('Temperature (°C)')
ax3.set_title('Mode and Trend of Ground vs. Satellite Temperature')
ax3.grid(True)
ax3.legend()

# Сохранение графика перед показом
save_plot = input("Хотите ли сохранить график? (да/нет): ").lower()
if save_plot == "да":
    if not os.path.exists('Graphics'):
        os.makedirs('Graphics')
    current_time = pd.Timestamp.now().strftime("%Y-%m-%d_%H-%M-%S")
plt.savefig(f'Graphics/{satellite_name}_comparison_{satellite_product}_{time_
of_day}_{time_interval_minutes}_{current_time}.png')
    elif save_plot == "нет":
        print("График не сохранен.")
    else:
        print("Некорректный ответ.")

# Показ графика после сохранения
plt.show()

# Закрытие фигуры
plt.close(fig)
else:
    print("No matches found.")

```

1. Создание фигуры с несколькими графиками:

Функция создает фигуру с тремя графиками с помощью `plt.subplots`. Размер фигуры задается как 12x12 дюймов, а расстояние между графиками по вертикали увеличено для лучшей читаемости.

2. Первый график: наземные и спутниковые данные:

На первом графике отображаются точки для спутниковых и наземных значений температуры. Данные спутника и земли группируются по времени для сопоставления. Информация о спутнике и данных добавляется в виде текста на график.

3. Второй график: облако рассеяния:

На втором графике строится облако рассеяния для сопоставленных точек земной и спутниковой температуры. Добавляется диагональная ось, коэффициенты регрессии и уравнение регрессии.

4. Третий график: мода и тренд:

На третьем графике отображается горизонтальная линия, представляющая моду всех данных. Также отображаются тренды для данных земной поверхности и спутника.

5. Сохранение и отображение графика:

После построения графиков пользователю предлагается сохранить его. Если пользователь соглашается, график сохраняется в папке Graphics с именем файла на основе времени создания. После сохранения или в случае отказа от сохранения графика, функция предлагает показать график. Функция также закрывает фигуру после показа графика.

6. Обработка отсутствия сопоставленных данных:

Если сопоставленные данные отсутствуют, выводится сообщение "No matches found."

MapView — это класс, предназначенный для создания и отображения картографических данных, основанных на различных источниках спутниковой информации.

```
class MapViewer:
    def __init__(self, coordinates, date_start, date_end):
        if isinstance(coordinates, list):
            self.coordinates = coordinates
        else:
            self.coordinates = [float(coord.strip()) for coord in
coordinates.split(',')]
            self.date_start = date_start
            self.date_end = date_end
            self.map_directory = os.path.join(os.getcwd(), 'Maps') # Путь к
папке с картами
            if not os.path.exists(self.map_directory):
                os.makedirs(self.map_directory)

    def mask_modis_clouds(self, image):
        qc = image.select('QC_Day')
        cloud_mask = qc.bitwiseAnd(1 << 0).eq(0) # Бит 0: облака
        return image.updateMask(cloud_mask)

    def mask_jaxa_clouds(self, image):
        qc = image.select('QA_Flag')
        cloud_mask = qc.bitwiseAnd(1 << 0).eq(0) # Предполагаем, что бит 0 -
облака
        return image.updateMask(cloud_mask)

    def create_map(self):
        point = ee.Geometry.Point(self.coordinates)
        transformed_point = point.transform('SR-ORG:6974', 1000)

        lst_modis = ee.ImageCollection('MODIS/006/MYD11A1') \
            .filterBounds(transformed_point) \
            .filterDate(self.date_start, self.date_end) \
            .map(self.mask_modis_clouds)

        median_image = lst_modis.median()
```

```

region = transformed_point.buffer(50000)
image = median_image.clip(region)

tile_url = image.select('LST_Day_1km').getThumbUrl({
    'min': -20, 'max': 40, 'palette': ['040274', '040281', '0502a3',
'0502b8', '0502ce', '0502e6',
                                '0602ff', '235cb1', '307ef3',
'269db1', '30c8e2', '32d3ef',
                                '3be285', '3ff38f', '86e26f',
'3ae237', 'b5e22e', 'd6e21f',
                                'fff705', 'ffd611', 'ffb613',
'ff8b13', 'ff6e08', 'ff500d',
                                'ff0000', 'de0101', 'c21301',
'a71001', '911003']
})

map_center = [self.coordinates[1], self.coordinates[0]]
m = geemap.Map(center=map_center, zoom=12)

m.add_tile_layer(url=tile_url, name='MODIS MYD11A1')

modis = ee.ImageCollection('MODIS/061/MOD11A1') \
    .filterBounds(transformed_point) \
    .filterDate(self.date_start, self.date_end) \
    .map(self.mask_modis_clouds)

temperature_image = modis.select('LST_Day_1km')

m.addLayer(temperature_image, {'min': 13000.0, 'max': 16500.0,
'palette': ['040274', '040281', '0502a3', '0502b8', '0502ce', '0502e6',
'0602ff', '235cb1', '307ef3', '269db1', '30c8e2', '32d3ef',
'3be285', '3ff38f', '86e26f', '3ae237', 'b5e22e', 'd6e21f',
'fff705', 'ffd611', 'ffb613', 'ff8b13', 'ff6e08', 'ff500d',
'ff0000', 'de0101', 'c21301', 'a71001', '911003']},
    'Land Surface Temperature MODIS/061/MOD11A1')

jaxa = ee.ImageCollection('JAXA/GCOM-C/L3/LAND/LST/V2') \
    .filterBounds(transformed_point) \
    .filterDate(self.date_start, self.date_end) \
    .map(self.mask_jaxa_clouds)

if jaxa.size().getInfo() > 0:
    jaxa_image = jaxa.median().select('LST_AVE')
    m.addLayer(jaxa_image, {'min': 0, 'max': 40, 'palette': ['blue',
'green', 'red']}, 'JAXA GCOM-C LST')

oxford = ee.ImageCollection('Oxford/MAP/LST_Day_5km_Monthly') \
    .filterBounds(transformed_point) \
    .filterDate(self.date_start, self.date_end) \
    .map(self.mask_modis_clouds)

if oxford.size().getInfo() > 0:
    oxford_image = oxford.median().select('LST_Day')
    m.addLayer(oxford_image, {'min': 0, 'max': 50, 'palette':
['blue', 'green', 'red']}, 'Oxford MAP LST Day')

```

```

mod21c3 = ee.ImageCollection('MODIS/061/MOD21C3') \
    .filterBounds(transformed_point) \
    .filterDate(self.date_start, self.date_end) \
    .map(self.mask_modis_clouds)

mod21c3_image = mod21c3.median().select('LST_Day')

m.addLayer(mod21c3_image, {'min': 0, 'max': 50, 'palette': ['purple',
'blue', 'green', 'yellow', 'orange', 'red']}, 'MODIS MOD21C3 LST Day')

Landsat_tile_url =
"https://mtl.google.com/vt/lyrs=s&x={x}&y={y}&z={z}"
m.add_tile_layer(url=Landsat_tile_url, name='Landsat')

return m

def display_map(self):
    m = self.create_map()

    # Вопрос пользователю о сохранении карты
    save_map = input("Хотите ли вы сохранить карту? (да/нет): ").lower()
    if save_map == "да":
        current_time = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        map_file_name = f'map_{current_time}.html'
        map_file_path = os.path.join(self.map_directory, map_file_name)
        m.to_html(map_file_path)

webbrowser.open_new_tab(f'file://{os.path.abspath(map_file_path)}')
elif save_map == "нет":
    print("Карта не сохранена.")
else:
    print("Некорректный ответ.")

```

1. Метод `__init__`:

Конструктор класса инициализирует объект `MapView` с заданными координатами, начальной и конечной датами. Если координаты передаются в виде строки, они парсятся и преобразуются в список. Затем проверяется существование папки `Maps`, и если ее нет, она создается для сохранения карт.

2. Метод `mask_modis_clouds`:

Принимает изображение и применяет маску для удаления облаков на изображении MODIS.

3. Метод `mask_jaxa_clouds`:

Принимает изображение и применяет маску для удаления облаков на изображении JAXA.

4. Метод `create_map`:

Создает карту, используя координаты, начальную и конечную даты. Загружает данные MODIS, JAXA, Oxford и Landsat. Применяет маски облаков к изображениям. Добавляет слои изображений на карту. Возвращает объект карты.

5. Метод `display_map`:

Создает карту с помощью метода `create_map`. Пользователю предлагается сохранить карту в виде HTML-файла. Если пользователь соглашается, карта сохраняется в папке `Mars` и открывается в браузере. В противном случае выводится соответствующее сообщение.

Этот класс предоставляет простой интерфейс для создания и отображения картографических данных, включая возможность удаления облаков и сохранения карты для последующего использования.

Функция `create_pdf_report` создает PDF-отчет на основе данных и параметров, которые ей передаются.

```
def create_pdf_report(df, matches, satellite_name, start_date, end_date,
coordinates, time_interval_minutes=None, satellite_product=None,
time_of_day=None):
    # Создаем папку Reports, если она не существует
    if not os.path.exists('Reports'):
        os.makedirs('Reports')

    # Создаем папку Graphics, если она не существует
    if not os.path.exists('Graphics'):
        os.makedirs('Graphics')

    # Формируем имя файла на основе текущего времени, названия спутника и дат
    начала и конца
    current_time = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    pdf_file_name =
f"Reports/{satellite_name}_report_{start_date}_{end_date}_{current_time}.pdf"

    doc = SimpleDocTemplate(pdf_file_name, pagesize=letter)
    elements = []
    styles = getSampleStyleSheet()

    # Добавляем заголовок
    elements.append(Paragraph(f"Report from data {satellite_name}",
styles['Title']))
    elements.append(Spacer(1, 12))

    # Добавляем информацию о временном промежутке, координатах и типе
спутника
    satellite_info = f"Satellite: {satellite_name.capitalize()}"
    if satellite_product:
        satellite_info += f"\nType: {satellite_product.capitalize()}"
    if time_of_day:
        satellite_info += f"\nTime of Day: {time_of_day.capitalize()}"
    if time_interval_minutes:
        satellite_info += f"\nTime Interval: {time_interval_minutes} minutes"
    satellite_info += f"\nCoordinates: {coordinates}"
    satellite_info += f"\nDate Range: {start_date} to {end_date}"
    elements.append(Paragraph(satellite_info, styles['Normal']))
    elements.append(Spacer(1, 12))

    # Разделяем столбцы DataFrame на группы по столбцам и добавляем каждую
группу на новую строку
    columns = list(df.columns)
    for i in range(0, len(columns), 7):
```



```

subset = columns[i:i + 7]
df_table_data = [subset] + df[subset].values.tolist()
df_table = Table(df_table_data)
df_table.setStyle(TableStyle([
    ('BACKGROUND', (0, 0), (-1, 0), colors.grey),
    ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),
    ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
    ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
    ('BOTTOPPADDING', (0, 0), (-1, 0), 12),
    ('BACKGROUND', (0, 1), (-1, -1), colors.beige),
    ('GRID', (0, 0), (-1, -1), 1, colors.black)
]))
elements.append(df_table)
elements.append(Spacer(1, 12))

# Добавляем таблицу с совпадениями
if len(matches) % 2 == 0:
    matches_table_data = [['ID', 'Satellite name', 'Satellite value',
'Satellite Datetime', 'Ground value', 'Ground Datetime']]
    for idx in range(0, len(matches), 2):
        match = matches[idx]
        if idx + 1 < len(matches):
            match_next = matches[idx + 1]
            matches_table_data.append([idx // 2 + 1,
match.get('source', '').split(' -
')[1],
match.get('value', ''),
match.get('datetime', ''),
match_next.get('value', ''),
match_next.get('datetime', '')]])

    matches_table = Table(matches_table_data)
    matches_table.setStyle(TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.grey),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
        ('BOTTOPPADDING', (0, 0), (-1, 0), 12),
        ('BACKGROUND', (0, 1), (-1, -1), colors.beige),
        ('GRID', (0, 0), (-1, -1), 1, colors.black)
    ]))
    elements.append(matches_table)
    elements.append(Spacer(1, 12))
else:
    print("Matches list does not have an even number of elements. Cannot
create PDF report.")

# Вычисляем RMSE и MBE
rmse, mbe = calculate_rmse_mbe(matches)
elements.append(Paragraph(f"RMSE: {rmse}", styles['Normal']))
elements.append(Paragraph(f"MBE: {mbe}", styles['Normal']))
elements.append(Spacer(1, 12))

# Создаем и сохраняем графики
fig, ax = plt.subplots()
satellite_values = [float(matches[i]['value']) for i in range(0,
len(matches), 2)]
ground_values = [float(matches[i + 1]['value']) for i in range(0,
len(matches), 2)]
dates = [matches[i]['datetime'] for i in range(0, len(matches), 2)]

```

```

# Поворачиваем даты на 45 градусов
plt.xticks(rotation=45)

ax.plot(dates, satellite_values, label='Satellite')
ax.plot(dates, ground_values, label='Ground')
ax.set_xlabel('Date')
ax.set_ylabel('Value')
ax.set_title(f'{satellite_name} vs Ground Values')
ax.legend()
plot_file_name = f"Graphics/{satellite_name}_plot_{current_time}.png"
plt.savefig(plot_file_name)
plt.close(fig)

# Включаем графики в отчет
elements.append(Image(plot_file_name, width=640, height=480))
elements.append(Spacer(1, 12))

# Подтверждение сохранения отчета
save_report = input("Хотите ли вы сохранить отчет? (да/нет): ").lower()
if save_report == "да":
    doc.build(elements)
    print(f"PDF отчет сохранен в {pdf_file_name}")
else:
    print("Отчет не был сохранен.")

```

1. Создание папок:

Проверяется существование папки 'Reports', и если она не существует, она создается. То же самое делается и для папки 'Graphics'. Формирование имени файла:

Имя файла формируется на основе текущего времени, названия спутника и дат начала и конца. Создается объект SimpleDocTemplate для создания PDF.

2. Формирование содержания отчета:

Добавляется заголовок отчета, который включает название спутника. Добавляется информация о временном промежутке, координатах и типе спутника. Данные DataFrame разбиваются на группы по 7 столбцов и добавляются в виде таблиц. Добавляется таблица с совпадениями между данными спутника и данными с земли.

3. Вычисление RMSE и MBE:

Вычисляются значения RMSE (квадратный корень из среднеквадратической ошибки) и MBE (средняя ошибка).

4. Создание графиков:

Строится график, на котором отображаются значения спутника и земли в зависимости от даты. График сохраняется в виде файла изображения.

5. Включение графиков в отчет:

Изображение графика включается в содержание отчета.

6. Сохранение отчета:

Продолжение приложения Д

Пользователю предлагается сохранить отчет в виде PDF-файла. Если пользователь соглашается, отчет сохраняется, и выводится сообщение о месте сохранения. В противном случае выводится соответствующее сообщение.

Эта функция обеспечивает автоматическое создание и форматирование PDF-отчета на основе данных, а также включение в отчет графиков и других визуальных элементов.