ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНІКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Магістр»

«Розробка прикладного програмного інтерфейса (API) для систематизації супутникових даних»

(тема кваліфікаційної роботи українською мовою)

«Development of an application program interface (API) for systematization of satellite data»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання

спеціальності 122 Комп'ютерні науки

(код, назва спеціальності)

Освітня програма Комп'ютерні науки

(назва)

Коваленко Владислав Сергійович

(прізвище, ім'я, по-батькові здобувача)

Керівник к.т.н., доцент Перелигін Б.В. _____

(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент кафедри інженерії ПЗ національного університету "Одеська політехніка", Зіноватна Світлана Леонідівна

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій
№ ___ від ___ _____ 2024 р.

Завідувачка кафедри

_____ _____

(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № _____
протокол № ___ від _____ 2024 р.

Оцінка _____ / _____ / _____

(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ _____

(підпис) (прізвище,ім'я)

Одеса 2024

АНОТАЦІЯ

У кваліфікаційній роботі розробляється тема «Розробка прикладного програмного інтерфейса (API) для систематизації супутникових даних»

Актуальність теми магістерської кваліфікаційної роботи визначається стрімким розвиток технологій дистанційного зондування Землі, що супроводжується постійним зростанням обсягу даних, які потребують обробки та зберігання. Наявність стандартизованого та зручного API дозволяє полегшити доступ до супутникових даних, забезпечуючи інтеграцію цих даних у програмні застосунки різного призначення. Крім того, наявність ефективного API сприяє оптимізації використання ресурсів і підвищує точність прийняття рішень у різних сферах.

Мета роботи – розробка прикладного програмного інтерфейсу (API) для систематизації супутникових даних, що дозволяє автоматизувати їх обробку, структурувати та забезпечувати доступ до них для широкого кола користувачів.

Результатом є розроблений та протестований прикладний програмний інтерфейс (API), який дозволяє ефективно працювати з супутниковими даними. API забезпечує їх структурування, зручну обробку, інтеграцію з базами даних та надає доступ до них для користувачів і розробників програмного забезпечення. Розроблене рішення відповідає сучасним вимогам масштабованості, продуктивності та зручності використання.

ABSTRACT

The qualification work explores the topic «Development of an application program interface (API) for systematization of satellite data»

The relevance of the master's qualification thesis is determined by the rapid development of Earth remote sensing technologies, which is accompanied by a constant increase in the volume of data requiring processing and storage. The availability of a standardized and user-friendly API facilitates access to satellite data, enabling the integration of this data into software applications for various purposes. Moreover, an effective API contributes to the optimization of resource utilization and enhances decision-making accuracy across various fields.

The result of the research is a developed and tested application programming interface (API) that enables efficient management of satellite data. The API ensures data structuring, seamless processing, integration with databases, and accessibility for users and software developers. The proposed solution adheres to modern standards of scalability, performance, and usability.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

База даних – організована структура для зберігання, управління та обробки даних, що забезпечує зручний доступ та оновлення інформації.

ГІС (геоінформаційні системи) – комп’ютерні системи для зберігання, обробки, аналізу та візуалізації географічних та просторових даних, використовуються для картографії, просторового аналізу та прийняття рішень.

Дистанційне зондування Землі – це процес збору інформації про об'єкти та явища на поверхні Землі без фізичного контакту з ними, за допомогою супутникових систем, безпілотних літальних апаратів (дронів) та інших космічних технологій.

Супутникові дані – інформація, отримана через супутники, яка використовується для моніторингу стану Землі, її поверхні, кліматичних умов, а також для дослідження природних і антропогенних явищ.

API (application programming interfaces) – система інструментів і ресурсів в застосунку, яка дозволяє розробникам створювати програмні продукти, які взаємодіють з іншими службами.

SQL-запит – запит, що використовується для взаємодії з базами даних на мові SQL (Structured Query Language), яка дозволяє здійснювати операції пошуку, вставки, оновлення та видалення даних.

Synthetic Aperture Radar – синтетична апертурна рада, радарна система, яка використовується в супутниковому зондуванні для отримання зображень Землі, незважаючи на погодні умови або час доби.

ГІС – геоінформаційні системи.

ДЗЗ – дистанційного зондування Землі.

API – application programming interface.

GDAL – Geospatial Data Abstraction Library.

OGC – Open Geospatial Consortium.

SAR – Synthetic Aperture Radar.

ВСТУП

Сучасні технології дистанційного зондування Землі (ДЗЗ) відіграють ключову роль у вирішенні важливих задач у таких сферах, як екологічний моніторинг, сільське господарство, урбаністичне планування, наукові дослідження, управління природними ресурсами та забезпечення національної безпеки. Завдяки розвитку супутникових систем, обсяг даних, отримуваних через дистанційне зондування, постійно зростає, що зумовлює потребу в їхній ефективній обробці та інтеграції. Використання цих даних дозволяє здійснювати точне моделювання процесів, прогнозування та прийняття обґрунтованих рішень у різних галузях.

Однак велика кількість супутникових даних, їх різнорідність за форматом і структурою створюють серйозні виклики для обробки та зберігання інформації. Це зумовлює необхідність розробки інструментів, які забезпечують зручний доступ до цих даних, їх систематизацію та інтеграцію у програмні рішення. Прикладний програмний інтерфейс (API) у цьому контексті стає важливим інструментом, що забезпечує автоматизацію роботи з супутниковими даними та інтеграцію їх у складні інформаційні системи.

API дозволяє спростити процеси доступу до супутникових даних, стандартизувати їх обробку та забезпечити можливість масштабування рішень відповідно до потреб кінцевих користувачів. Це особливо актуально в умовах сучасних викликів, таких як необхідність швидкого реагування на екологічні кризи, оптимізація ресурсів у сільському господарстві чи побудова ефективних систем моніторингу у сфері безпеки.

Актуальність теми полягає у необхідності створення зручних і надійних інструментів для систематизації супутникових даних з урахуванням сучасних вимог до обробки великих даних, автоматизації та інтеграції з іншими системами.

Мета роботи – розробка прикладного програмного інтерфейсу (API) для систематизації супутникових даних, що дозволяє автоматизувати їх обробку,

структурувати та забезпечувати доступ до них для широкого кола користувачів.

Об’єкт дослідження є процеси систематизації та інтеграції супутникових даних для застосування у програмних системах.

Предмет дослідження – методи, інструменти та технології розробки API для роботи з супутниковими даними.

Методи дослідження – комбінація системного аналізу, об’єктно-орієнтованого програмування, проєктування програмних інтерфейсів, а також методи тестування програмного забезпечення для оцінки ефективності розробленого рішення.

Розроблений API стане основою для побудови ефективних інформаційних рішень, що дозволяють працювати з супутниковими даними максимально продуктивно, забезпечуючи їхню доступність і високу точність у різних прикладних сферах.

Дана магістерська кваліфікаційна робота складається з 50 сторінок, 14 рисунків, 3 таблиць та 17 джерел посилання.

1. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ВИКОРИСТАННЯ СУПУТНИКОВИХ ДАНИХ

1.1. Огляд сучасних технологій дистанційного зондування Землі

Дистанційне зондування Землі (ДЗЗ) – це процес збору інформації про об’єкти та явища на поверхні Землі без фізичного контакту з ними, за допомогою супутникових систем, безпілотних літальних апаратів (дронів) та інших космічних технологій. ДЗЗ активно використовується в численних галузях, таких як екологія, сільське господарство, картографія, управління природними ресурсами, а також в оборонній та безпековій сферах. На рис. 1 наведено основні типи супутникових систем і сенсорів, що використовуються для збору супутникових даних.

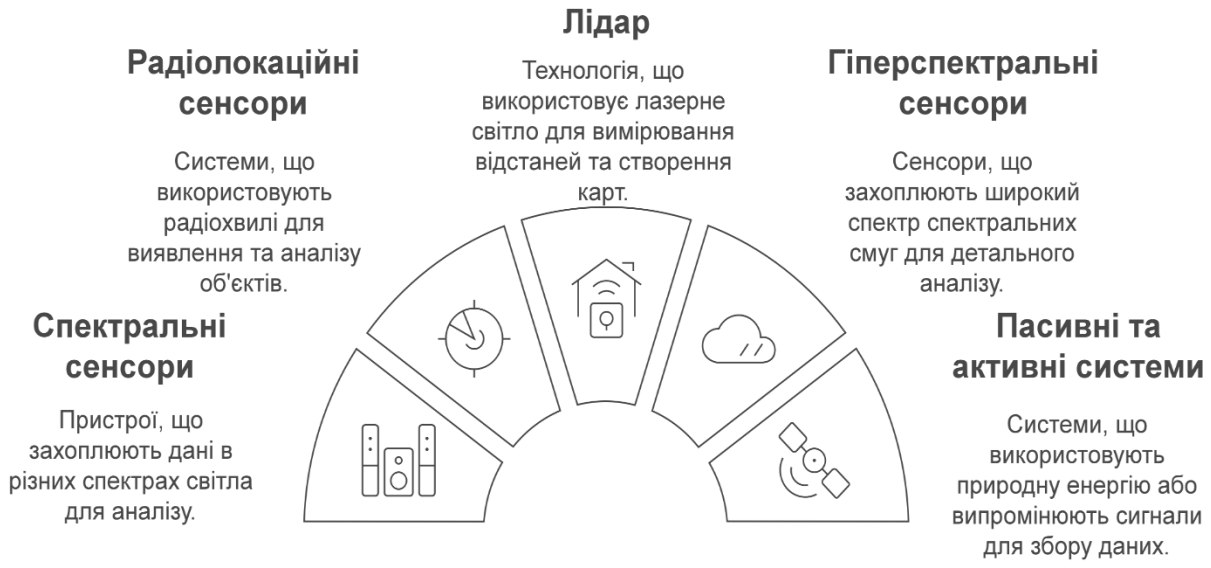

Рисунок 1 – Супутникові системи та сенсори

Сенсори, що використовують спектри світла для отримання інформації про поверхню Землі, поділяються на:

- оптичні сенсори: вони працюють у видимому та ближньому інфрачервоному спектрах, здатні забезпечувати високу роздільну здатність зображень, що використовуються для картографії, оцінки сільськогосподарських культур, лісового господарства та моніторингу змін на поверхні Землі;
- інфрачервоні сенсори: використовуються для моніторингу температури поверхні та температурних аномалій, а також для виявлення лісових пожеж і гідрологічних змін.

Радіолокаційні сенсори використовують радіохвилі для збору даних про поверхню Землі, навіть за умов поганої видимості (хмари, темрява). Найпоширенішим типом є системи Synthetic Aperture Radar (SAR). SAR (Радар з синтетичною апертурою) забезпечує високу просторову та тимчасову роздільну здатність і може використовуватись для моніторингу змін у земних покриттях, таких як зсуви ґрунту, зміни в ландшафті, аналіз прибережних зон, сільське господарство та гідрологія.

Лідар (Light Detection and Ranging) – це технологія, яка використовує лазерні імпульси для визначення відстані до об’єкта. Лідар є корисним інструментом для створення тривимірних моделей поверхні та вимірювання висоти об’єктів. Лідарні дані часто використовуються для створення цифрових моделей рельєфу (DEM), для лісового та водного моніторингу, а також для планування інфраструктури та містобудування.

Гіперспектральні сенсори здатні фіксувати значно більше спектральних каналів, ніж стандартні оптичні сенсори. Це дозволяє отримати більш детальну інформацію про властивості різних матеріалів. Гіперспектральне зображення може забезпечити точнішу класифікацію об’єктів на поверхні Землі, наприклад, для визначення типів рослин, виявлення забруднень або моніторингу зміни водних ресурсів.

Пасивні системи (наприклад: оптичні та інфрачервоні сенсори) збирають інформацію, відображену або випромінювану об’єктами на поверхні Землі. Активні системи (як, наприклад: SAR та лідар) використовують власні

джерела випромінювання для отримання даних, що дозволяє їм працювати в будь-яких умовах освітлення та погодних умовах.

1.1.1. Застосування технологій ДЗЗ у різних галузях діяльності

ДЗЗ для напрямку «Екологія та природокористування» може застосовуватися для виконання наступних цілей, а саме:

- моніторинг лісових пожеж, змін клімату та рівня забруднення атмосфери;
- оцінка стану водних ресурсів, контроль за рівнем води в річках і озерах;
- вивчення процесів деградації земель, зокрема дефляції та ерозії ґрунтів.

Якщо розглядати напрямок «Сільське господарство», слід звернути увагу на:

- оцінку здоров’я сільськогосподарських культур (моніторинг стану рослин);
- визначення оптимальних умов для поливання, збирання врожаю та планування сільськогосподарських робіт;
- створення карт урожайності, моніторинг стресу на рослинах через температуру та вологість.

З точки зору «Урбаністика та містобудування»:

- вивчення урбаністичних змін, планування розвитку міст, моніторинг інфраструктури;
- оцінка якості повітря та рівня забруднення, а також розробка політик щодо розвитку сталих міст.

Напрямок «Моніторинг природних катастроф» забезпечить:

- виявлення та відстеження природних катастроф, таких як землетруси, повені, урагани та цунамі;

- оцінка наслідків катастроф і допомога в плануванні заходів з відновлення.

З розрізу «Національна безпека», ДЗЗ можна застосувати для:

- моніторингу ситуації в умовах військових конфліктів;
- аналізу змін на території, контроль за переміщенням об’єктів, моніторинг можливих загроз.

Напрямок «Наукові дослідження»:

- вивчення змін клімату, біорізноманіття, а також геологічні та географічні дослідження;
- проведення досліджень в умовах, недоступних для традиційних методів.

Сучасні технології ДЗЗ надають безліч можливостей для збору, обробки та аналізу даних, що відкриває нові горизонти для досліджень і практичних застосувань у різних галузях. Кожен тип сенсора має свої переваги та обмеження, тому для досягнення максимального ефекту часто використовуються комбінації різних технологій. ДЗЗ є важливим інструментом для моніторингу стану навколишнього середовища та прийняття обґрунтованих рішень на рівні національних і міжнародних організацій.

1.2. Існуючі методи обробки та зберігання супутникових даних

Обробка та зберігання супутникових даних є важливими етапами в управлінні великими масивами інформації, що виникають в результаті використання технологій ДЗЗ. Супутникові зображення та інші види даних мають високий обсяг і потребують спеціалізованих методів для їх збереження, обробки та інтеграції в різні системи. Огляд існуючих методів обробки та зберігання даних дозволяє виявити ключові аспекти, що забезпечують ефективне використання супутникової інформації.

1.2.1. Методи зберігання супутникових даних

Оскільки супутникові дані часто займають великі обсяги пам’яті (від кількох гігабайтів до терабайтів на день), зберігання та управління такими даними вимагає використання спеціалізованих технологій і методів (див.рис.2).

Рисунок 2 – Зберігання та управління супутникових даних

Розподілені файлові системи (Distributed File Systems) використовуються для:

- для зберігання великих обсягів супутникових даних активно використовуються розподілені файлові системи, такі як Hadoop Distributed File System (HDFS) і Google File System (GFS). Ці системи дозволяють ефективно зберігати дані, що розподілені на багатьох серверах, забезпечуючи високу доступність і масштабованість;
- використання розподілених систем дозволяє обробляти дані паралельно на кількох вузлах, що прискорює процеси зберігання та доступу до інформації.

Хмарні технології:

- хмарні платформи, такі як Amazon Web Services (AWS), Google Cloud, та Microsoft Azure, пропонують спеціалізовані сервіси для зберігання великих обсягів даних (наприклад, Amazon S3, Google Cloud Storage). Вони забезпечують високу масштабованість і гнучкість для зберігання супутникових зображень і супутніх даних;
- хмарні платформи також дозволяють ефективно розподіляти навантаження зберігання між різними сервісами і оптимізувати витрати на інфраструктуру.

Реляційні та нереляційні бази даних:

- реляційні бази даних (RDBMS), такі як PostgreSQL з розширеннями для роботи з геопросторовими даними (PostGIS), часто використовуються для зберігання метаданих супутникових зображень та їхніх атрибутів;
- нереляційні бази даних (NoSQL) на кшталт MongoDB або Cassandra застосовуються для зберігання великих масивів супутникових зображень у форматах, таких як GeoTIFF або NetCDF.

1.2.2. Індексування супутникових даних

Індексування є важливим етапом, що дозволяє швидко знаходити та отримувати необхідну інформацію з великих обсягів супутникових даних. Основні підходи до індексації супутникових даних включають:

- просторові індекси;
- індексування на основі метаданих;
- індексування за спектральними характеристиками.

Просторове індексування застосовується для впорядкування та структурування просторових даних таким чином, щоб забезпечити максимально ефективний і гнучкий доступ до окремих фрагментів зображень,

які відповідають певним географічним районам чи регіонам. Це особливо важливо для роботи з великими масивами супутникових знімків або топографічних мап, оскільки просторове індексування дозволяє суттєво скоротити час пошуку та вибірки даних. Одним із найпоширеніших і високооптимізованих методів організації просторової інформації є R-дерева. Вони забезпечують швидкий пошук об’єктів у багатовимірних просторах, включаючи географічні координати, часові проміжки, спектральні характеристики та інші виміри, що мають значення для галузей, де детальний аналіз просторових об’єктів є критично важливим. Крім R-дерев, у практиці широко застосовуються Quadtrees і KD-дерева. Quadtrees забезпечують ієрархічний поділ простору на чотири рівні квадрантів, спрощуючи пошук об’єктів у межах певних площ. KD-дерева, у свою чергу, дозволяють розбивати простір на прямокутні області, що значно полегшує роботу з неоднорідними та нерівномірно розподіленими даними, коли необхідно швидко і точно ідентифікувати зону інтересу серед великого масиву інформації.

Для досягнення максимальної ефективності у пошуку супутникових зображень недостатньо лише просторового індексування. Також створюються спеціалізовані індекси метаданих – структуровані таблиці або каталоги, що зберігають ключову інформацію про кожне зображення. Серед такої інформації можуть бути дата і час зйомки, географічне місцезнаходження об’єкта спостереження, тип сенсора, просторовий розділ, спектральний діапазон, кут нахилу супутника, а також інші унікальні характеристики, важливі для аналізу та інтерпретації знімків. Наявність індексів метаданих дає змогу швидко звузити пошук, враховуючи лише ті матеріали, що відповідають вказаним критеріям. Це суттєво прискорює процес обробки, оскільки немає потреби аналізувати кожне доступне зображення в реальному часі. Такий підхід підвищує ефективність виконання складних геопросторових завдань,

оптимізує використання обчислювальних ресурсів, знижує навантаження на системи зберігання даних та надає можливість опрацьовувати суттєво більші обсяги інформації без втрати точності чи швидкості реагування на запити. Таким чином, поєднання просторового індексування з індексами метаданих утворює багаторівневу систему організації даних, здатну задовольнити потреби сучасних користувачів у швидкому та надійному доступі до потрібних фрагментів геопросторової інформації.Зазначені методи використовуються для пошуку та аналізу супутникових даних за спектральними властивостями, наприклад, для виявлення аномалій, зміни стану рослин або виявлення катастроф (наприклад, лісових пожеж). Для таких цілей застосовуються техніки кластеризації та аналізу схожості за допомогою спектральних індексів (NDVI, EVI, NDSI).

1.2.3. Методи обробки супутникових даних

Обробка супутникових даних є необхідним етапом для їх аналізу і видобутку корисної інформації. Ось деякі сучасні методи, які використовуються для обробки великих обсягів супутникових зображень:

- геопросторова обробка зображень;
- машинне навчання та штучний інтелект;
- паралельна обробка даних.

Сучасні програмні рішення, такі як GDAL (Geospatial Data Abstraction Library), дозволяють виконувати різноманітні операції з геопросторовими даними, включаючи переведення з одного формату в інший, геопросторове злиття, обтинання та аналіз. Використання таких інструментів дозволяє виконувати автоматизовану обробку великої кількості супутникових зображень, що зменшує час, необхідний для аналізу.

Для аналізу супутникових даних все більше використовуються алгоритми машинного навчання, зокрема глибоке навчання, для автоматичної класифікації та виявлення об’єктів на зображеннях (наприклад:

сільськогосподарські культури, лісові покриви, будівлі). Такі методи дозволяють автоматизувати процеси обробки та аналізу, що раніше вимагали ручного втручання.

Для роботи з великими обсягами супутникових даних використовуються паралельні обчислення. Це дозволяє зменшити час обробки даних за рахунок розподілу завдань між кількома обчислювальними вузлами. Використання таких платформ, як Apache Hadoop і Apache Spark, дозволяє обробляти дані на тисячах серверів, виконуючи різні обчислювальні завдання одночасно.

1.2.4. Інтеграція даних з різних джерел

Зберігання та обробка супутникових даних не обмежується тільки роботою з одним типом даних. Важливим аспектом є інтеграція супутникових зображень з іншими джерелами інформації, такими як:

- інформація з наземних датчиків: поєднання даних, отриманих із супутників, з даними від земних датчиків, таких як метеорологічні станції або сенсори на сільськогосподарських машинах;
- відкриті дані: використання відкритих даних з інших джерел, таких як бази даних картографічних систем, для поліпшення точності та контексту супутникових зображень.

Сучасні методи зберігання, індексації та обробки супутникових даних стають все більш ефективними завдяки розвитку технологій, таких як хмарні обчислення, машинне навчання та розподілені системи зберігання. Водночас зростаючі обсяги супутникових даних ставлять нові вимоги до обробки і зберігання таких даних, що потребує постійного вдосконалення існуючих методів і впровадження новітніх технологій для їх ефективної обробки та інтеграції.

1.3. Аналіз проблем інтеграції супутникових даних у програмні системи

Інтеграція супутникових даних у програмні системи є складним і багатоетапним процесом, який передбачає вирішення низки технічних та організаційних проблем. Супутникові дані за своєю природою є великими обсягами інформації, що можуть бути представлені в різних форматах, містити велику кількість метаданих і потребують відповідних інструментів для їх ефективної обробки, зберігання та аналізу. Тому для забезпечення безперешкодної інтеграції таких даних у програмні системи необхідно вирішити кілька важливих проблем. Однією з основних проблем є різноманітність форматів, в яких ці дані можуть зберігатися та передаватися. Супутникові зображення можуть бути представлені в таких форматах, як GeoTIFF, NetCDF, HDF5, JPEG2000 та інші, кожен з яких має свої особливості. Деякі з цих форматів підтримують геопросторову інформацію (наприклад: GeoTIFF), інші – ні. Крім того, дані можуть включати метадані, які описують географічні координати, часові мітки, параметри сенсорів та інші характеристики, що додають додаткову складність при їх обробці та інтеграції. Різноманітність форматів вимагає застосування спеціалізованих бібліотек та інструментів для їх конвертації і обробки. Наприклад, бібліотеки GDAL (Geospatial Data Abstraction Library) і Rasterio дозволяють працювати з різними форматами геопросторових даних, однак для кожного формату можуть бути свої особливості, які потребують додаткових налаштувань та адаптацій. Це створює додаткову складність, оскільки необхідно враховувати всі можливі варіанти форматів даних та забезпечувати їх коректну інтеграцію.

Іншою суттєвою проблемою є відсутність єдиного стандарту для супутникових даних. Хоча існують певні загальноприйняті формати, такі як GeoTIFF або NetCDF, кожен постачальник супутникових даних може використовувати власні варіанти форматів, що ускладнює інтеграцію різних джерел даних. Наприклад, різні супутники можуть мати різні характеристики

сенсорів, тому дані з одного супутника можуть відрізнятися за точністю, роздільною здатністю та іншими параметрами, що створює додаткові труднощі в процесі інтеграції. Також для кожного зображення або набору даних необхідно вказувати метадані, які пояснюють, коли і де були отримані ці дані, які параметри використовувалися для їх зйомки, та інші характеристики. Відсутність уніфікації цих метаданих, а також різноманітність їх форматів, створює труднощі для розробників програмних систем, які повинні обробляти ці метадані і коректно їх використовувати для аналізу. Для вирішення цієї проблеми активно розвиваються міжнародні стандарти, такі як ISO 19115 (метадані для географічної інформації) та OGC (Open Geospatial Consortium), що допомагають стандартизувати метадані та інтерфейси доступу до геопросторових даних. Вони полегшують інтеграцію різних джерел даних у програмні системи, забезпечуючи більш уніфікований підхід до роботи з супутниковими даними.

Ще однією важливою проблемою є забезпечення доступу до супутникових даних. Оскільки ці дані часто містять велику кількість інформації та мають обмежену доступність через великі обсяги, необхідно використовувати ефективні інтерфейси для запиту та доступу до даних. Одним з основних методів доступу до супутникових даних є використання API, які дозволяють отримати дані в реальному часі або за запитом. Однак для забезпечення зручного і ефективного доступу до супутникових даних необхідно враховувати декілька факторів: швидкість відповіді API, його надійність, зручність для користувача та підтримка різних форматів даних. Найбільш поширеними інтерфейсами доступу є REST API та WMS (Web Map Service), що дозволяють отримувати супутникові зображення або карти через Інтернет. Вони використовуються для інтеграції супутникових даних у геоінформаційні системи (ГІС) або в інші програмні системи для аналізу та візуалізації даних. Проте інтерфейси доступу можуть відрізнятися за рівнем складності, можливістю роботи з великими обсягами даних і необхідністю забезпечення високої швидкості обробки запитів. Для зберігання та доступу

до супутникових даних використовуються як традиційні бази даних, так і спеціалізовані системи для геопросторових даних, такі як PostGIS або MongoDB для геопросторових даних. Ці інструменти дозволяють зберігати та ефективно запитувати дані, що містять просторову та тимчасову інформацію.

Інтеграція супутникових даних у програмні системи вимагає не лише коректного форматування, чіткої стандартизації та легкого доступу до них, а й забезпечення високого рівня масштабованості та загальної продуктивності всієї системи. Оскільки такі дані зазвичай містять величезні масиви інформації, процес інтеграції потребує використання потужних обчислювальних ресурсів, продуктивних сховищ для довготривалого зберігання та ефективних каналів передачі даних. Необхідно також оптимізувати алгоритми обробки для скорочення часу доступу до даних та прискорення аналізу, що особливо важливо при роботі в режимі реального часу або під час опрацювання запитів у динамічних середовищах. Це включає розробку інтелектуальних алгоритмів, здатних автоматично визначати оптимальні підходи до зберігання, попереднього опрацювання та аналізу даних, а також динамічно масштабувати обчислювальні потужності залежно від обсягів інформації та складності аналітичних завдань.

Успішна інтеграція супутникових даних у програмні системи є комплексним процесом, що охоплює подолання труднощів із різноманітними форматами даних, адаптацію до загальноприйнятих стандартів галузі, забезпечення безперервного доступу та налагодження ефективних підходів до обробки великих обсягів інформації з урахуванням можливих перешкод. Для досягнення цієї мети слід використовувати стандартизовані формати та інтерфейси доступу, які полегшують взаємодію між різними компонентами системи та роблять інтеграцію прозорою й передбачуваною. Потрібно запроваджувати гнучкі інструменти для завантаження, обробки, аналізу та зберігання даних, що дозволяють ефективно масштабувати ресурси за потреби.

Таким чином, забезпечення ефективної інтеграції супутникових даних у програмні системи полягає не тільки в технічній стандартизації та зручних форматах, але й у розвитку високопродуктивних інструментів, здатних підтримувати аналіз величезних потоків інформації з максимальною оперативністю. Це, своєю чергою, уможливлює прийняття швидких і обґрунтованих рішень у галузях, де точний і своєчасний аналіз даних є критично важливим, таких як метеорологія, агросектор, екологічний моніторинг, картографія, безпека та логістика. Завдяки такому підходу, використання супутникових даних стає більш доступним, масштабованим та результативним, що дає змогу максимально розкрити потенціал цих безцінних джерел інформації для задоволення потреб різноманітних користувачів і застосувань.

1.4. Огляд існуючих API для роботи з супутниковими даними

Існує ряд рішень для роботи з супутниковими даними, які надають API для інтеграції, доступу та обробки цих даних. Вони дозволяють розробникам та кінцевим користувачам отримувати супутникові зображення, дані про земну поверхню та інші види інформації, що містять географічні координати, тимчасові мітки, а також дані, отримані від різних сенсорів супутників. Існуючі API на ринку використовуються в різних сферах, таких як екологія, сільське господарство, урбаністика, картографія та наукові дослідження.

Google Earth Engine (GEE) – одна з найпоширеніших платформ для роботи з гігантськими масивами супутникових даних, яка надає дослідникам, екологам, інженерам та іншим фахівцям інструменти для ефективної обробки інформації про Землю. Вона пропонує API для доступу до архівів багатьох супутникових місій, включно з Landsat, Sentinel та MODIS, дозволяючи здійснювати аналіз часових рядів, застосовувати алгоритми машинного навчання, проводити спектральні та просторові дослідження. Завдяки хмарній інфраструктурі, GEE позбавляє необхідності утримувати локальні

обчислювальні кластери, оскільки складні аналітичні операції та обчислення виконуються на серверах Google. Це дає змогу виявляти зміни в земному покриві, моніторити рослинність, ґрунти, водні ресурси або інфраструктуру без надмірних витрат часу й ресурсів. Зручна інтеграція з мовами програмування (JavaScript, Python) та існуючими аналітичними інструментами забезпечує широкі можливості для автоматизації завдань, прискорюючи обробку даних та отримання результатів, а також стимулює розвиток інновацій у сфері дистанційного зондування та геоінформатики.

Earth Observing System Data and Information System (EOSDIS) – масштабна ініціатива NASA, метою якої є надання легкого та прозорого доступу до різноманітних масивів супутникових даних з орбітальних місій агентства. Платформа пропонує API для отримання інформації з таких супутників, як Terra, Aqua, Suomi NPP, а також інших апаратів, що досліджують різні аспекти Землі. Завдяки цьому користувачі можуть швидко знаходити, фільтрувати та завантажувати історичні дані, які охоплюють великі часові проміжки та просторові області, спрощуючи розв’язання складних наукових та інженерних завдань. Інтеграція з інструментами обробки даних, можливість роботи з довготривалими архівами спостережень, а також підтримка різних форматів забезпечують гнучкість у застосуванні цих даних у наукових дослідженнях, ресурсному менеджменті, моделюванні кліматичних процесів та моніторингу навколишнього середовища.

Sentinel Hub – це платформа, створена для ефективної обробки та аналізу супутникових даних, отриманих у межах місії Copernicus Sentinel, яка охоплює спектрально різноманітні й високоточні супутникові знімки. Використовуючи пропоновані API, розробники та аналітики можуть легко інтегрувати ці дані у власні застосунки, системи ГІС або хмарні сервіси, поєднуючи супутникові зображення з іншими джерелами геопросторової інформації. Sentinel Hub дозволяє проводити спектральний аналіз, визначати індекси рослинності, оцінювати стан лісових і сільськогосподарських угідь, вивчати зміни у водних ресурсах чи урбанізованих територіях. Така інтеграція суттєво полегшує

процес прийняття рішень у сферах екології, агрономії, лісництва, природоохоронної діяльності та міського планування, забезпечуючи оперативний доступ до актуальних та високоякісних даних.

USGS Earth Explorer – онлайн-сервіс Геологічної служби США, що надає API для доступу до супутникових зображень і додаткових геопросторових матеріалів, включно з даними Landsat, ASTER, MODIS та інших орбітальних місій. Цей сервіс дозволяє користувачам гнучко налаштовувати пошук за географічними, часовими та якісними параметрами, швидко знаходячи релевантні дані для подальшої обробки та аналізу. Інтеграція з іншими інструментами та можливість завантаження у різних форматах дають змогу без зайвих витрат отримувати доступ до архівів спостережень, прискорюючи наукові дослідження, екологічний моніторинг, оцінку природних ресурсів, дослідження впливу кліматичних змін та інші завдання. Таким чином, USGS Earth Explorer стає універсальним інструментом для широкого кола фахівців, які працюють із геопросторовими даними.

OpenStreetMap (OSM) – хоча сама ця платформа не є джерелом супутникових даних, її API забезпечує доступ до відкритих геопросторових даних, які можна поєднувати з супутниковими знімками для формування детальних і функціональних картографічних продуктів. Інтеграція інформації OSM із супутниковими даними дозволяє додавати дорожню інфраструктуру, адмінкордони, гідрографію, будівлі та інші об’єкти на карту, підвищуючи рівень деталізації та точності. Це створює нові можливості для дослідників, розробників та аналітиків, які прагнуть поєднати реалістичні зображення земної поверхні з семантичною інформацією про об’єкти, що на ній розташовані. Результатом такої інтеграції може стати багатий набір інструментів для навігації, планування маршрутів, оцінки інфраструктури, моніторингу змін у навколишньому середовищі та створення інноваційних геоінформаційних сервісів.Існуючі API для роботи з супутниковими даними мають різноманітні можливості та обмеження, залежно від платформи та її функціональних можливостей. Платформи, такі як Google Earth Engine, NASA

EOSDIS, Sentinel Hub і USGS Earth Explorer, надають великі обсяги супутникових даних і потужні інструменти для їх обробки та аналізу (табл.1). Однак для вибору найкращого рішення важливо враховувати специфіку задач, вимоги до доступу, а також бюджету користувачів.

Таблиця 1 – Порівняльна таблиця основних API для роботи з супутниковими даними.

API	Тип даних	Платформи/ Супутники	Сильні сторони	Слабкі сторони
Google Earth Engine	Супутникові зображення, геопросторові дані	Landsat, MODIS, Sentinel, і інші	- широкий доступ до супутникових даних; - масштабованість; - легке використання	- залежність від інтернет-з'єднання; - обмежений контроль над даними
NASA EOSDIS	Супутникові зображення, екологічні дані	Terra, Aqua, Suomi NPP, і інші	- широкий вибір даних NASA; - висока точність даних	- складність використання; - обмежені можливості обробки даних
Sentinel Hub	Супутникові зображення, індекси поверхні	Sentinel 1, Sentinel 2	- швидка обробка даних; - інтеграція з іншими системами; - миттєва обробка	- обмежена кількість супутників; - платні послуги для деяких функцій
USGS Earth Explorer	Супутникові зображення, дані гідрології, зміни земної поверхні	Landsat, ASTER, MODIS, і інші	- великий вибір супутникових даних; - безкоштовний доступ до деяких даних	- складність інтеграції з іншими платформами; - обмежена функціональність обробки
OpenStreet Map (OSM)	Геопросторові дані, картографія	-	- відкритий доступ; - легко інтегрується з іншими даними, зокрема супутниковими	- не надає високоякісних супутникових зображень; - менша точність даних

2. ФОРМУЛЮВАННЯ ВИМОГ ДО РОЗРОБКИ API

2.1. Огляд функціональних вимог до API

Функціональні вимоги до API спрямовані на забезпечення швидкого, зручного та безпечного доступу до супутникових даних, їх ефективну обробку та інтеграцію з іншими системами (див.рис.3). Це дозволить користувачам із різних сфер, таких як екологія, сільське господарство, наукові дослідження або національна безпека, ефективно працювати з супутниковими даними, отримуючи точні результати та підтримку прийняття рішень.

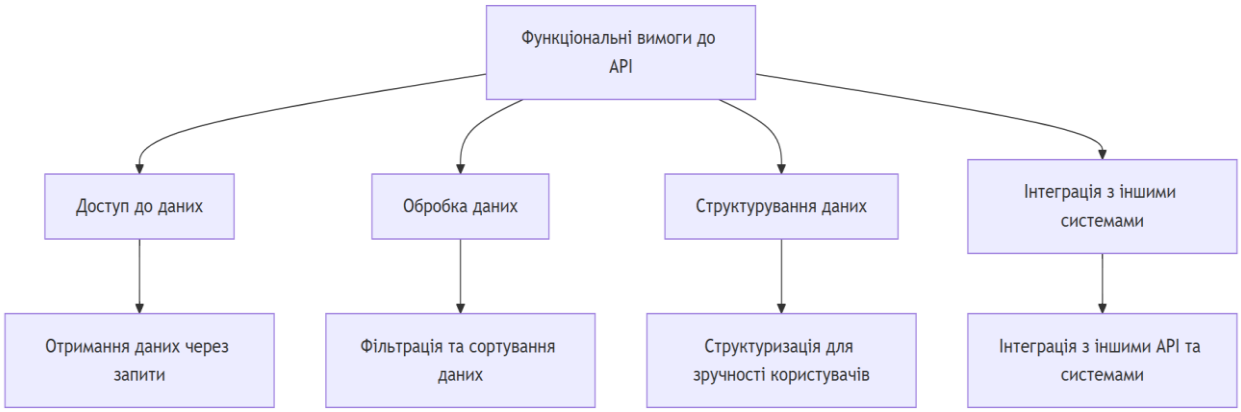

Рисунок 3 – Функціональні вимоги до створення API

API для роботи з супутниковими даними має забезпечувати наступні функціональні можливості для ефективного доступу, обробки, структуризації та інтеграції даних.

API має підтримувати запити, що дозволяють отримати доступ до супутникових даних за різними критеріями, такими як рік, місяць, географічні координати, типи супутників або типи вимірів. API повинен надавати дані у зручному форматі (JSON або XML), який буде легко оброблятись іншими системами або користувачами. Для зручності користувачів API має

підтримувати можливість фільтрації та пошуку даних за різними параметрами (за часом, місцем або типом супутникових сенсорів).

API має обробляти отримані дані, перетворюючи їх у зручний для користувача формат, що може включати агрегування даних, розподіл за категоріями або переведення даних у зручні одиниці виміру. Деякі API можуть містити функції для моделювання або виконання простих аналітичних операцій над супутниковими даними (побудова графіків, статистичний аналіз або перерахунок даних).

API має можливість структуризувати дані за допомогою різних категорій, що можуть включати типи даних, рівень деталізації, географічне положення тощо. Для ефективного пошуку та швидкого доступу до великих обсягів супутникових даних, API повинно підтримувати індексацію і можливість зберігання даних у базах даних або на сервері. Зокрема, це дозволить швидко отримувати потрібну інформацію за запитом користувача.

API повинно бути здатним інтегруватися з іншими програмними системами, такими як системи управління даними, наукові платформи, інструменти для обробки географічних даних (GIS-системи), а також платформи для аналізу великих даних. Для забезпечення сумісності з іншими системами, API має підтримувати стандарти обміну даними, такі як REST або GraphQL, а також використовувати загальноприйняті формати даних, такі як JSON або XML. API має бути спроєктовано так, щоб у разі необхідності можна було легко додавати нові функції або інтегрувати нові джерела даних без значних змін у основній структурі API.

API повинно підтримувати механізми автентифікації користувачів для обмеження доступу до конфіденційних або захищених даних. Це може включати використання ключів API, OAuth або інших методів безпеки. Всі дані, що передаються через API, повинні бути захищені за допомогою протоколів безпеки, таких як HTTPS, для забезпечення конфіденційності та цілісності переданої інформації.

API повинно мати здатність працювати з великими обсягами запитів і даних, зберігаючи високу продуктивність, навіть при значних навантаженнях. Це досягається через механізми кешування, балансування навантаження та горизонтальне масштабування. Час відповіді на запити користувачів має бути мінімальним, особливо для великих наборів даних. Для цього можуть бути використані оптимізації в архітектурі API та застосуванні індексів в базах даних.

2.2. Огляд нефункціональних вимог до API

Нефункціональні вимоги описують критерії, що визначають якість роботи API та його здатність ефективно виконувати завдання в умовах реального використання. Вони охоплюють продуктивність, надійність, масштабованість, безпеку та зручність використання, які є критично важливими для успішної інтеграції API в програмні системи та забезпечення високого рівня користувацького досвіду (рис.4).

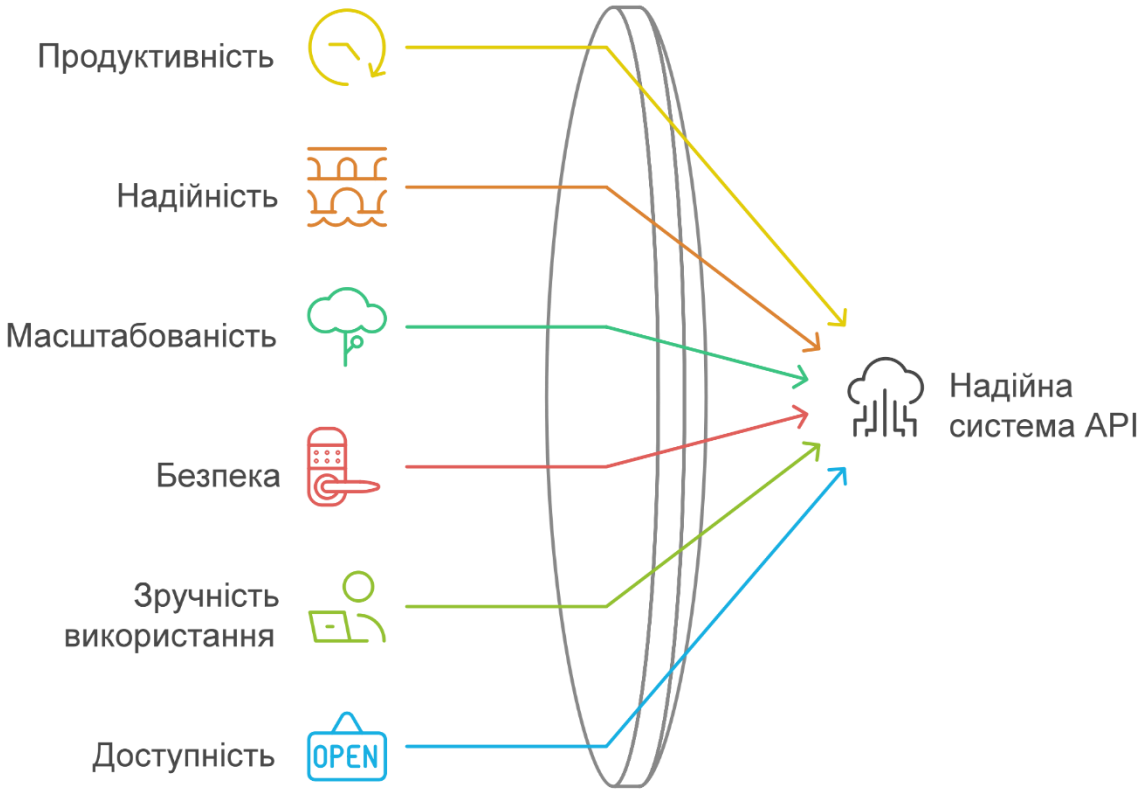

Рисунок 4 – Нефункціональні вимоги до створення високопродуктивного API

API повинно забезпечувати мінімальний час відповіді на запити, особливо при роботі з великими обсягами супутникових даних. Час відповіді на запит не повинен перевищувати декількох секунд, навіть при високих навантаженнях. API повинно мати здатність ефективно обробляти великий обсяг одночасних запитів, особливо в умовах високих навантажень, що можуть виникати під час піків використання.

API повинно бути здатне коректно обробляти помилки, що виникають під час виконання запитів (помилки з'єднання або неправильні параметри запиту), і надавати чіткі повідомлення про помилки користувачу або розробнику. У разі виникнення критичних збоїв API повинно мати механізми резервного копіювання та відновлення даних для мінімізації втрат та забезпечення безперервності роботи. API повинно включати механізми для моніторингу та логування запитів, що дозволяє оперативно виявляти несправності та визначати причини проблем.

API повинно підтримувати горизонтальне масштабування, що дозволяє додавати нові сервери для обробки збільшеного навантаження, зберігаючи стабільну роботу навіть при рості кількості запитів. API повинно автоматично адаптувати свої ресурси відповідно до зміни навантаження, збільшуючи або зменшуючи кількість оброблюваних запитів залежно від поточного попиту. Для забезпечення рівномірного розподілу запитів між серверами необхідно використовувати балансувальники навантаження, що забезпечують високу доступність і продуктивність API.

API повинно забезпечувати захист даних під час передачі через мережу за допомогою сучасних протоколів шифрування (HTTPS, SSL/TLS), щоб уникнути можливих атак на конфіденційну інформацію. Також, API повинно включати надійні механізми аутентифікації (API-ключі, OAuth) та авторизації, щоб забезпечити доступ тільки авторизованим користувачам і обмежити доступ до конфіденційних даних. API повинно бути захищене від різноманітних атак, таких як SQL-ін'єкції, атаки типу «відмова в

обслуговуванні» (DoS), фальсифікація запитів тощо. Використання міжмережевих екранів (firewall) та систем запобігання вторгненням (IDS) допоможе знизити ймовірність таких атак.

API повинно мати зручну документацію і простий інтерфейс, що дозволяє розробникам швидко і без зусиль інтегрувати API в інші системи та використовувати його функціональні можливості. Документація до API має бути детальною та доступною, включаючи приклади запитів і відповідей, пояснення функцій, а також можливі помилки і способи їх виправлення. Для забезпечення високої зручності використання, API повинно постійно оновлюватися, і розробники повинні мати доступ до служби підтримки для вирішення можливих проблем.

Доступність до API 24/7, забезпечить користувачам безперебійний доступ до супутникових даних. Це досягається через використання технологій резервування, реплікації та балансування навантаження. У разі збоїв у роботі, API повинно мати вбудовані механізми відновлення, щоб забезпечити повернення до нормальної роботи в мінімальні терміни.

Нефункціональні вимоги є важливими для забезпечення високої якості API і його безперебійної роботи в умовах реального використання. Забезпечення високої продуктивності, надійності, масштабованості, безпеки та зручності використання дозволить зробити API ефективним інструментом для роботи з великими обсягами супутникових даних та інтеграції в різноманітні програмні системи.

2.3. Специфікації даних та їх обробки в рамках API

Супутникові дані характеризуються великою кількістю форматів і структур, що обумовлено різноманітністю сенсорів і типів супутників. Для ефективної роботи API необхідно визначити стандартизовані формати даних, підходи до їх обробки та методи представлення для кінцевих користувачів (рис.5).

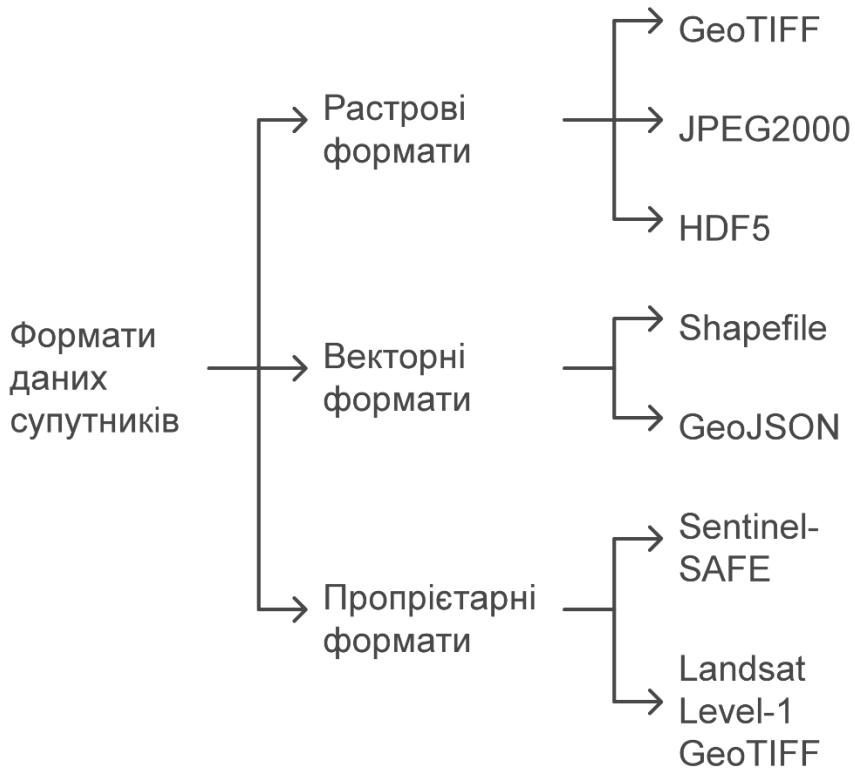

Рисунок 5 – Формати даних супутників

Растрові формати супутникових даних:

- GeoTIFF: широко використовується для геопросторових даних а також забезпечує збереження координат і метаданих;
- JPEG2000: підтримує стискання без втрат і геоприв’язку, що робить його популярним у космічній галузі;
- HDF5: використовується для зберігання великих обсягів наукових даних із супутників, таких як кліматичні або екологічні дані.

Векторні формати супутникових даних:

- Shapefile: популярний у ГІС-системах для збереження просторових об’єктів;
- GeoJSON: використовується для веб-застосунків і надає можливість легко інтегрувати геопросторові дані.

Пропрієтарні формати супутникових даних:

- Sentinel-SAFE: формат супутників програми Sentinel, який містить як метадані, так і дані спостережень;
- Landsat Level-1 GeoTIFF: спеціалізований формат для знімків із серії супутників Landsat.

Попередня обробка супутникових даних передбачає виконання радіометричної та геометричної корекції зображень для усунення шумів і спотворень, а також калібрування даних для приведення їх до стандартизованих одиниць вимірювання. Систематизація та індексація здійснюються за допомогою використання метаданих, таких як дата, місце і тип сенсора, що дозволяє організувати зручний пошук даних, а також створення просторових індексів для швидкого доступу до географічної інформації. Конвертація та трансформація включає підтримку переходу між різними форматами, наприклад, GeoTIFF у JPEG2000 або Sentinel-SAFE у HDF5, а також виконання проекції даних для приведення їх до різних географічних систем координат. Зберігання даних може здійснюватися локально, наприклад, у файлових системах, що підтримують великі обсяги інформації, таких як Hadoop HDFS або NAS-сервери, або ж у хмарних рішеннях, таких як Amazon S3, Google Cloud Storage чи Azure Blob, які забезпечують масштабованість і зручність доступу. Для представлення даних користувачам використовуються API-інтерфейси, які надають можливість робити запити до даних за параметрами, такими як координати, дата чи тип сенсора. Крім того, реалізуються функції фільтрування, попереднього перегляду знімків (Thumbnails) і статистичного аналізу даних. Також підтримується асинхронний доступ для обробки великих обсягів даних із можливістю повернення результатів у реальному часі. Специфікації супутникових даних та їх обробки в контексті розробки API визначають стандарти форматів, підходи до обробки й методи інтеграції. Це забезпечує сумісність API з сучасними програмними системами і гарантує зручність для кінцевих користувачів, незалежно від їхнього технічного рівня.

2.4. Забезпечення доступності та інтеграції API

Доступність та інтеграція API є одним із ключових чинників, що впливають на його успішне впровадження, стабільну експлуатацію та широку адаптацію серед користувачів. Здатність API надавати безперебійний, надійний і швидкий доступ до супутникових даних відіграє вирішальну роль у формуванні позитивного досвіду взаємодії з технологією, а також у підвищенні ефективності процесів аналізу, обробки та візуалізації інформації. Забезпечення сумісності API з різноманітними зовнішніми системами, платформами й сервісами дозволяє розробникам інтегрувати його у власні робочі процеси та програмні рішення без суттєвих перешкод. Така сумісність створює сприятливі умови для розширення можливостей користувачів, стимулюючи впровадження інноваційних підходів до роботи із супутниковими даними, а також покращуючи рівень взаємодії між різними компонентами інформаційної інфраструктури.

Однією з основних вимог до API є забезпечення максимальної доступності та безперервності роботи системи. Це передбачає масштабований підхід до управління ресурсами: резервування серверів, використання віртуальних машин та контейнеризованих середовищ, а також балансування навантаження для уникнення збоїв і перевантажень. Запровадження механізмів гарячого резервування, своєчасного оновлення компонентів та регулярного тестування на стійкість дозволяє ефективно реагувати на можливі несправності. Резервне копіювання даних та дублювання критичних служб прискорюють відновлення після помилок або втрат інформації, знижуючи ризик довготривалих простоїв. Важливим аспектом є моніторинг продуктивності, пропускної здатності та доступності API, що включає автоматизовані системи сповіщень, аналіз логів та метрик у реальному часі. Завдяки цьому адміністратори зможуть своєчасно виявляти проблеми, знижувати затримки та оптимізувати обробку запитів до баз даних. Застосування кешування, механізмів масштабування горизонтального та

вертикального типу, а також використання CDN для географічно розподіленого доставляння контенту сприяють збереженню високої швидкості доступу до API в умовах динамічного зростання навантажень. Інженери можуть інтегрувати автоматизовані інструменти для стрес-тестування й динамічної адаптації конфігурацій, що дозволить підтримувати стабільну роботу системи навіть за екстремальних умов.

Інтеграція API з іншими системами вимагає гнучкості та універсальності. Підтримка реляційних і нереляційних баз даних, можливість підключення до сторонніх сервісів через REST або GraphQL, а також розширений набір форматів даних (GeoJSON, CSV, HDF5 та інші) дозволяють забезпечити безпроблемну взаємодію з різноманітними інструментами й середовищами розробки. Завдяки цьому розробники можуть легко інтегрувати API у свої проекти, застосовувати аналітичні бібліотеки, об’єднувати дані з різних джерел або імпортувати їх для подальшої обробки. Така гнучкість створює сприятливі умови для колаборації між командами, обміну досвідом, розповсюдження кращих практик та прискорення процесів прийняття рішень.

Безпека відіграє не менш важливу роль. Захист даних і контроль доступу до них є критичними параметрами, які визначають довіру користувачів до API. Реалізація стандартів аутентифікації та авторизації (OAuth2.0, JWT), впровадження шифрування на рівні передачі (HTTPS/TLS), а також систем виявлення аномальної активності допомагають захистити інформацію від несанкціонованого доступу, підробок або викрадення. Додаткові механізми безпеки, такі як рольові моделі доступу, журналювання дій користувачів, аналітика поведінки запитів, дозволяють точніше контролювати використання даних та оперативно реагувати на потенційні загрози.

Отже, збільшення доступності, інтеграції та безпеки API є комплексним завданням, яке поєднує технічні аспекти, організаційні підходи та сучасні інструменти управління. Ретельно спроєктована інфраструктура, динамічне масштабування, кешування, моніторинг, дотримання відкритих стандартів, підтримка популярних форматів даних і протоколів взаємодії гарантують

гнучкість та адаптивність системи. Усе це допомагає створити екосистему, в якій користувачі отримують швидкий, стабільний і безпечний доступ до супутникових даних, легко інтегруючи їх у свої робочі процеси й аналітичні сценарії. Таким чином, збільшення доступності та сумісності API з іншими системами стимулює розвиток екосистеми геопросторових додатків та сервісів, відкриває нові перспективи для досліджень і впровадження технологічних інновацій у різних галузях.Зворотна сумісність є важливим аспектом, оскільки вона дозволяє старим версіям API працювати навіть після оновлень. Важливим елементом інтеграції також є детальна документація API, яка полегшує роботу розробникам, надаючи приклади запитів і відповіді для різних сценаріїв.

Отже, доступність і інтеграція API є ключовими характеристиками, які забезпечують його ефективне функціонування, безпеку та зручність використання для різних категорій користувачів і систем.

3. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ API

3.1. Розробка архітектури API

Архітектура API є ключовим елементом його дизайну, що визначає спосіб організації та взаємодії компонентів для досягнення високої продуктивності, масштабованості та зручності використання. Архітектура створюється з урахуванням модульності, що дозволяє легко додавати або оновлювати функції, та дотримання принципів розподіленої обробки для ефективної роботи з великими обсягами супутникових даних (рис.6).

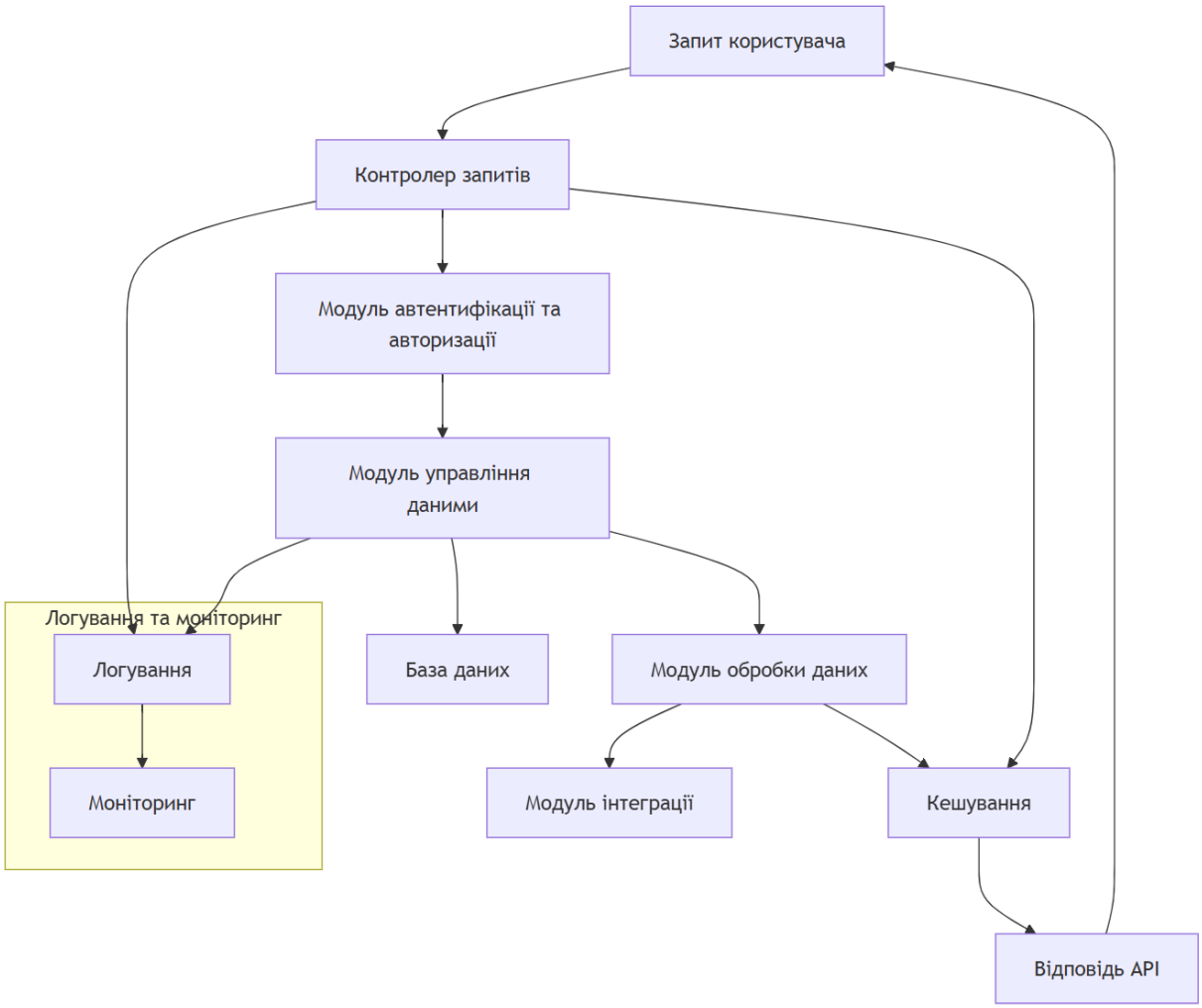

Рисунок 6 – Блок-схема архітектури API

Основні компоненти архітектури API:

- контролер запитів (Request Controller): обробляє вхідні запити від користувачів або систем, відповідає за маршрутизацію запитів до відповідних модулів API та включає валідацію параметрів запитів;
- модуль управління даними (Data Management Module): відповідає за взаємодію з базами даних для зберігання супутникових даних та реалізує функції читання, запису, індексації та пошуку даних; підтримує різні формати даних, зокрема GeoJSON, HDF5 і CSV.
- модуль обробки даних (Data Processing Module): виконує попередню обробку супутникових даних, таку як фільтрація, нормалізація або агрегація та включає механізми для роботи з великими обсягами даних (паралельну обробку або використання черг);
- модуль інтеграції (Integration Module): забезпечує можливість підключення API до зовнішніх систем і сервісів, таких як геоінформаційні системи (GIS), хмарні платформи та сторонні API і підтримує REST, GraphQL та інші стандарти інтеграції;
- модуль аутентифікації та авторизації (Authentication and Authorization Module): реалізує механізми безпечного доступу до API, зокрема OAuth2.0 і JWT та контролює рівень доступу користувачів до різних функцій API;
- кешування (Caching Layer): оптимізує продуктивність API, зберігаючи результати часто використовуваних запитів та використовує технології Redis або Memcached;
- моніторинг та логування (Monitoring and Logging): забезпечує відстеження стану API та запис дій користувачів для аналізу й діагностики і реалізує механізми оповіщення у разі виникнення помилок або збоїв;
- інтерфейс документації (Documentation Interface): надає користувачам детальну документацію API через інтерактивний

інтерфейс Swagger UI або Postman Collections та включає приклади запитів і відповіді.

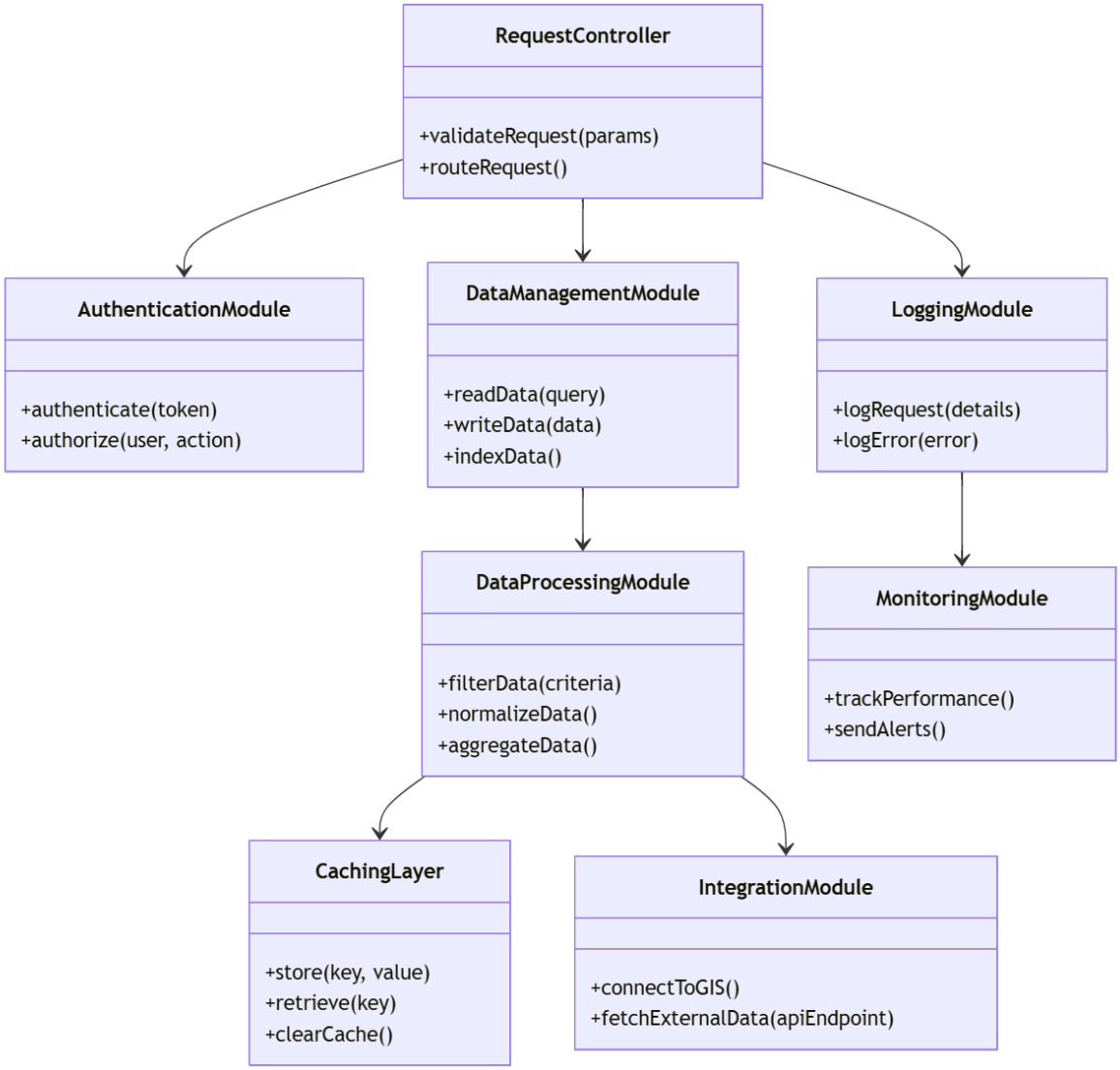

Рисунок 7 – Діаграма компонентів API

Користувацький запит (рис.7) спершу обробляється Контролером запитів, який перевіряє коректність параметрів і передає запит до відповідного модуля. Далі, Модуль управління даними витягує або зберігає інформацію, за необхідності взаємодіючи з базою даних. Якщо потрібно, Модуль обробки даних виконує необхідні трансформації. Після виконання запиту результати проходять через Кешування, щоб оптимізувати повторне використання даних.

Моніторинг та логування записують усі дії для забезпечення прозорості та діагностики.

На рис. 8 представлена діаграма взаємодії між компонентами, яка необхідна для візуалізації та аналізу того, як різні елементи системи взаємодіють один з одним.

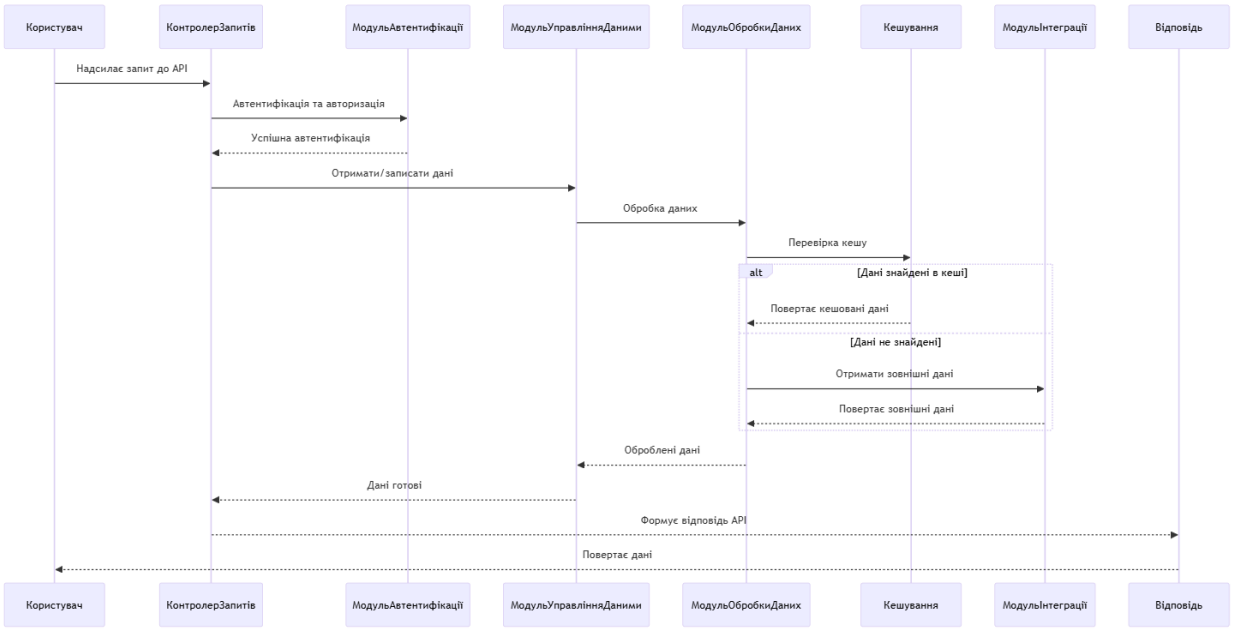

Рисунок 8 – Діаграма взаємодії між компонентами

Переваги використання даної архітектури:

- модульність: окремі частини API можуть оновлюватися або масштабуватися незалежно;
- гнучкість: підтримка різних форматів і стандартів;
- продуктивність: використання кешування та паралельної обробки;
- безпека: механізми аутентифікації та захисту даних.

Така архітектура дозволяє API ефективно працювати з великими обсягами супутникових даних, забезпечуючи надійну та швидку обробку і доступність інформації для кінцевих користувачів.

Побудова діаграми послідовності (Sequence Diagram) взаємодії користувача з API є важливим етапом в процесі проектування та розробки

програмного забезпечення. Вона допомагає наочно продемонструвати, як користувач або інша система взаємодіє з API в контексті обміну повідомленнями та даними (див.рис.9).

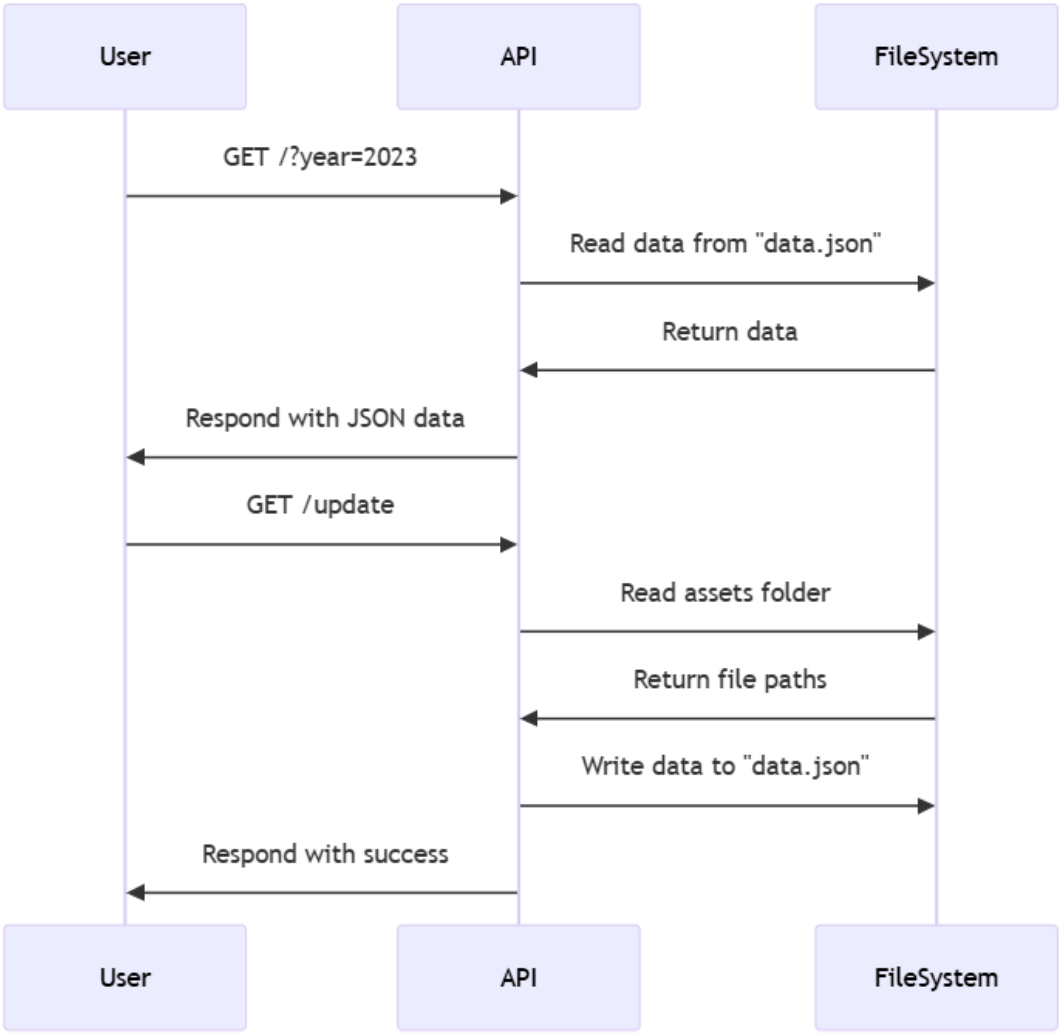

Рисунок 9 – Sequence Diagram (діаграма послідовності) взаємодії користувача з API

3.2. Вибір технологій для розробки API

Розробка API для роботи з супутниковими даними вимагає ретельного підходу до вибору технологій та інструментів, щоб забезпечити високу продуктивність, масштабованість, зручність у підтримці та відповідність

сучасним стандартам розробки програмного забезпечення. Оскільки супутникові дані можуть бути великими за обсягом і вимагати обробки в реальному часі, важливо використовувати технології, які дозволяють швидко обробляти запити, ефективно зберігати та обробляти дані, а також забезпечувати зручний доступ для користувачів.

В табл.2 наведено обґрунтування вибору технологій та інструментів, які були використані для реалізації API.

Таблиця 2 – Вибір технологій для розробки API

Категорія	Технологія/ Інструмент	Причина вибору
Мова програмування	Node.js (JavaScript/ TypeScript)	асинхронна модель виконання для високої продуктивності, легка інтеграція фронтенд- і бекенд-розробки
Фреймворк	Express.js	простота у використанні, висока продуктивність, підтримка маршрутизації та інтеграції з іншими модулями
Обробка файлів	Модуль fs	зручний для зчитування та запису JSON-файлів, що підходить для локальних даних
Робота з шляхами	Модуль path	стандартизоване управління шляхами у різних операційних системах
Допоміжні функції	Власний модуль utils.js	спрощує повторювані завдання, такі як зчитування директорій і запис даних
Зберігання даних	JSON	простий для обробки формат, легко інтегрується з веб-технологіями
	PostgreSQL (планується)	надійна реляційна база для структурованих даних (майбутня інтеграція)
	MongoDB (планується)	гнучкість для зберігання нереляційних даних (майбутня інтеграція)
Кешування	Redis (планується)	забезпечує швидкий доступ до часто запитуваних даних
Масштабування	Кластери Node.js	легке горизонтальне масштабування, підтримка багатопоточності
	Docker (планується)	контейнеризація для спрощення розгортання та масштабування
Безпека	HTTPS (SSL/TLS)	шифрує трафік між клієнтом та сервером
	JWT (JSON Web Token)	забезпечує надійну автентифікацію користувачів і захист даних

3.3. Розробка основних функціональних можливостей API

Перша ключова функція – доступ до даних. API надає можливість отримати доступ до супутникових даних у стандартизованому форматі. Для цього використовується HTTP-запит методу GET, де через параметри запиту, такі як year (рік) та month (місяць), можна отримати потрібний обсяг даних. У результаті користувач отримує відповідь у форматі JSON, яка містить дані, що відповідають зазначеним критеріям.

Наступною важливою функцією є обробка даних. Вона автоматизує процес структурування та перевірки даних перед їх поданням користувачеві. Для цього використовуються вбудовані функції модуля fs, які забезпечують обробку великих обсягів файлів, перевірку коректності даних і їх перетворення у формат JSON. Цей процес гарантує, що лише валідні дані будуть надані в результаті запиту.

Формати вихідних даних – ще одна суттєва функція. API підтримує стандартний формат JSON для передачі даних. Це дозволяє спростити інтеграцію з іншими системами, оскільки JSON є широко використовуваним і підтримується більшістю мов програмування. Також є можливість у майбутньому додавати нові формати даних, такі як XML або CSV, в залежності від потреб користувачів.

Функція фільтрації даних дозволяє користувачам здійснювати вибірку за певними параметрами. Наприклад, можна вибрати дані за конкретним роком або місяцем. Це здійснюється за допомогою запитів із параметрами year і month. Такий підхід дає змогу зменшити обсяг даних, які необхідно обробити, і підвищує ефективність роботи з API.

Остання основна функція – обробка запитів. API підтримує асинхронну обробку запитів, що дозволяє забезпечити швидку і стабільну відповідь на кожен запит. Для цього використовується маршрутизація запитів за допомогою фреймворку Express.js, що дає можливість обробляти запити від клієнта з максимальною ефективністю.

Ці основні функції API взаємодіють одна з одною, що дозволяє досягти високої ефективності при роботі з супутниковими даними та забезпечити зручний і гнучкий доступ до них для користувачів (див.рис.10 та табл.3).

Рисунок 10 – Функції API

Таблиця 3 – Функції API

Функція	Опис	Метод	Формат
Доступ до даних	отримання супутникових даних за рік або місяць	GET	JSON
Обробка даних	перевірка, структура та очищення даних перед поданням	Внутрішній	JSON
Формати вихідних даних	стандартизовані формати даних для інтеграції з іншими системами	GET	JSON
Фільтрація даних	запити за параметрами, такими як рік та місяць	GET	JSON
Обробка запитів	швидка маршрутизація та асинхронна обробка запитів	GET	JSON

3.4. Інтеграція API з базами даних

Процес інтеграції розробленого API з базами даних є важливою частиною для забезпечення зберігання, обробки та ефективного доступу до супутникових даних. Для цієї мети необхідно вибрати правильний тип бази даних, а також встановити взаємодію між API та базою для досягнення максимальної ефективності роботи системи.

Першим кроком у процесі інтеграції є вибір типу бази даних, який найбільш підходить для зберігання супутникових даних. Враховуючи обсяг і структуру даних, для цієї задачі найкраще підходять реляційні бази даних, такі як PostgreSQL, або нереляційні бази даних, такі як MongoDB, залежно від вимог щодо гнучкості і швидкості обробки. Реляційні бази даних надають можливість організувати сувору структуру даних з таблицями та зв’язками між ними, тоді як нереляційні бази даних можуть забезпечити більшу гнучкість при зберіганні даних у форматі JSON, що часто використовуються для супутникових даних.

API взаємодіє з базою даних через запити, що здійснюються за допомогою різних технологій, таких як SQL-запити для реляційних баз або запити за допомогою MongoDB для NoSQL рішень. API виконує запити для вставки, оновлення, видалення або отримання даних з бази даних у відповідь на запити користувачів. Наприклад, коли користувач запитує дані за певний рік або місяць, API звертається до бази даних, де зберігаються ці дані, і повертає результат у форматі JSON.

Супутникові дані часто складаються з великих обсягів інформації, тому важливо організувати базу даних таким чином, щоб вона могла ефективно зберігати та обробляти ці дані. Для цього використовуються індексація та оптимізація запитів, які значно прискорюють доступ до даних. Наприклад, використання індексів для полів, за якими часто здійснюються запити (такі як рік, місяць, координати), дозволяє значно покращити швидкість пошуку необхідної інформації.

Система зберігання повинна бути масштабованою, щоб мати можливість підтримувати великий обсяг даних, який буде збільшуватися з часом. Враховуючи обсяг супутникових даних, система зберігання повинна підтримувати горизонтальне масштабування, що дозволить ефективно обробляти додаткові запити і зберігати нові дані без зниження продуктивності. Також важливим є забезпечення високої надійності даних, тому система повинна мати механізми резервного копіювання та відновлення даних, щоб запобігти втраті інформації.

Кожен запит до API передбачає перевірку вхідних даних і їх подальшу обробку в базі даних. Наприклад, коли користувач запитує супутникові дані за певний рік і місяць, API здійснює запит до бази даних, щоб отримати ці дані. Залежно від структури даних, API може виконати складні фільтрації, агрегації або обчислення, щоб обробити дані перед їх відправкою користувачу.

Оскільки супутникові дані можуть містити чутливу інформацію, особливо у випадках, коли вони використовуються для цілей національної безпеки або для комерційних цілей, необхідно забезпечити високий рівень безпеки при інтеграції API з базами даних. Це включає в себе використання шифрування даних, аутентифікацію і авторизацію користувачів, а також захист від несанкціонованого доступу до бази даних через API.

У підсумку слід зазначити, що процес інтеграції API з базами даних є важливим кроком для забезпечення ефективного зберігання, обробки і доступу до супутникових даних, а також для створення масштабованої, безпечної та надійної системи для роботи з цими даними.

3.5. Реалізація обробки супутникових даних за допомогою API

Процес реалізації обробки супутникових даних за допомогою розробленого API охоплює кілька етапів, зокрема збір, індексаціяї, зберігання та обробка різноманітних типів даних. Кожен етап забезпечує ефективне

управління даними, дозволяючи розширювати базу даних та здійснювати доступ до них через API (див.рис.11).

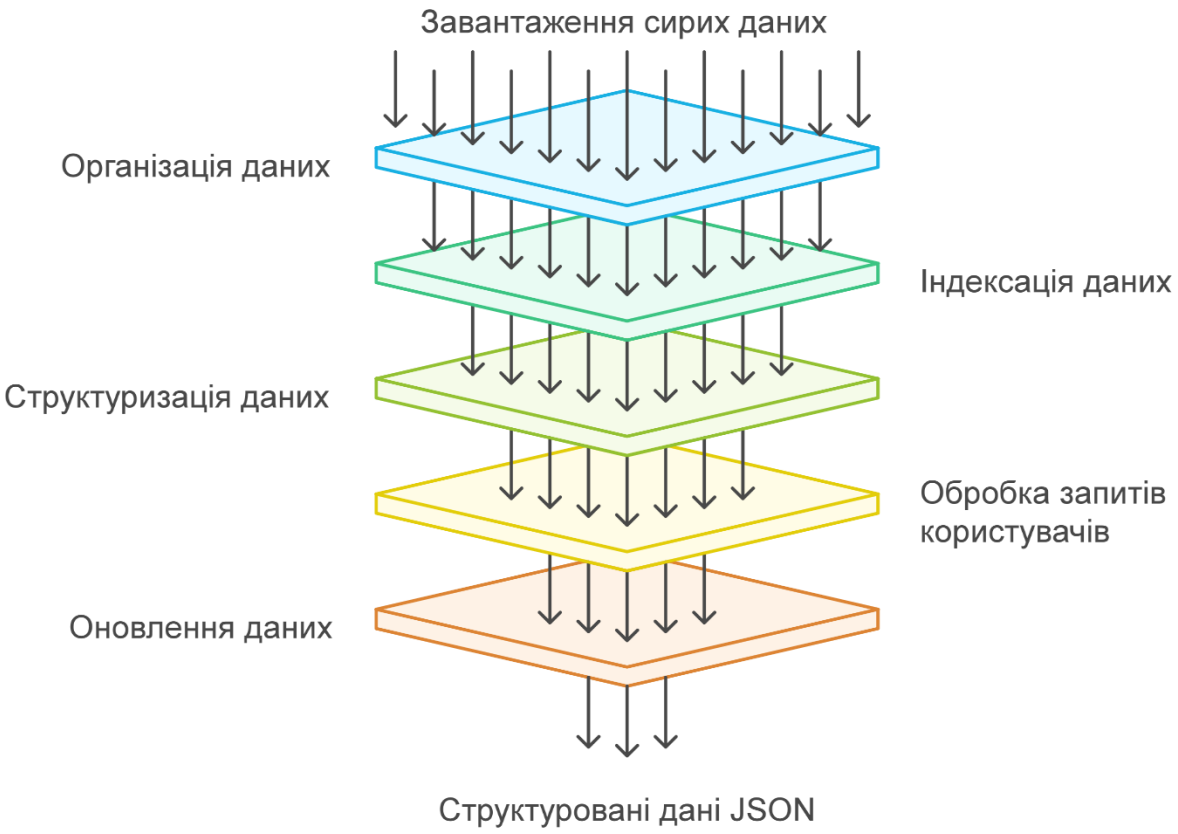

Рисунок 11 – Процес управління даними

На першому етапі дані зберігаються в папці assets, де кожен набір даних організовано за датами. Кожен набір даних містить різні формати файлів (наприклад, зображення (jpg), аудіо (wav) та текстові файли). Ці дані можуть бути додані в систему просто шляхом завантаження нових файлів у відповідні папки, де вони автоматично структуруються і організовуються за допомогою логіки, визначеної в API (див.рис. 12).

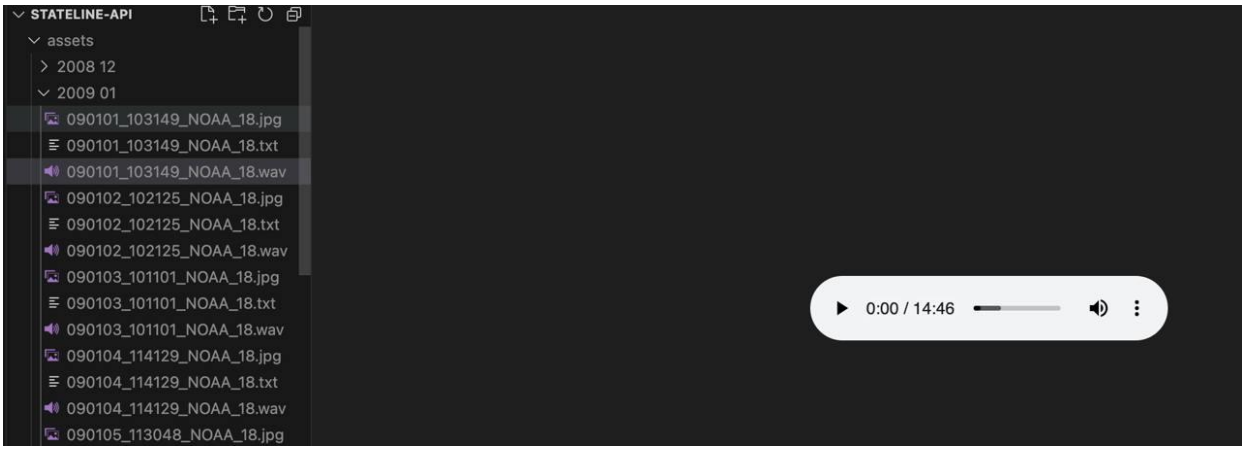

Рисунок 12 – Структурне відображення даних в папці assets

Після завантаження даних у систему виконується індексація через API, що аналізує вміст папок і організовує їх за допомогою ієрархічної структури. Вся інформація щодо файлів збирається у вигляді JSON-об’єкта, де кожен рік, місяць і тип даних (зображення, аудіо, текст) зберігаються окремо. Структурована інформація містить шляхи до файлів у файловій системі.

Після індексації API забезпечує обробку даних. Файли з однаковою датою, незалежно від їх формату, об’єднуються в один масив або структуру, що дозволяє ефективно працювати з ними через запити. Наприклад, можна отримати всі дані за певний рік і місяць, включаючи зображення, звукові записи та текстову інформацію, в єдиному форматі, який підготовлений для подальшої обробки.

Запити від користувачів API дозволяють отримати потрібну інформацію по певному року та місяцю. Користувач може надіслати запит до API, вказавши рік і/або місяць, і отримати відповідь з відповідними даними в форматі JSON. Запити можуть бути адаптовані для фільтрації або пошуку конкретних типів даних (наприклад, тільки зображення, тільки аудіо тощо).

Оскільки нові дані можуть бути легко додані до системи, API включає механізм для оновлення індексації. Коли нові файли додаються до папки assets, API виконує перевірку нових файлів і оновлює відповідну структуру даних, забезпечуючи постійну актуальність інформації. Це дозволяє користувачам

отримувати доступ до найсвіжішої інформації без необхідності вручну перезапускати процеси.

Усі оброблені дані зберігаються в JSON-файлі, що дозволяє зберігати інформацію про всі доступні набори даних в компактному і зручному для обробки форматі. API надає можливість зберігати та швидко отримувати інформацію про будь-які дані, включаючи доступ до звукових записів, зображень або текстів, які можуть бути збережені в одному наборі для кожної конкретної дати.

API використовує функції для читання і запису JSON-файлів для зберігання індексованої інформації про дані. Для цього використовуються методи fs.readFileSync для зчитування даних з файлів і fs.writeFile для запису нової інформації після оновлення даних.

```
// Читання вмісту файлу
export const read = (from) => {
    try {
        return fs.readFileSync(from, 'utf8');
    } catch (e) {
        // Якщо файл не знайдений або помилка, повертаємо пустий
об'єкт
        return JSON.stringify({});
    }
};

// Запис вмісту в файл
export const write = (to, content) => {
    fs.writeFile(to, content, err => {
        if (err) {
            console.error('Помилка запису файлу:', err);
        } else {
            console.log('Файл успішно записаний');
        }
    });
};
```

Для індексації використовуються функції, що рекурсивно проходять по папках і обробляють їх вміст. Функція readDir визначає структуру даних, розбиваючи її за роками, місяцями та типами файлів.

```
// Функція для рекурсивного читання директорій
export const readDir = (folderPath, struct = {}) => {
    for (const fileName of fs.readdirSync(folderPath)) {
        if (bannedFolderNames.includes(fileName)) {
            continue; // Пропускаємо заборонені папки
        }

        const nextPath = path.join(folderPath, fileName);
        const stat = fs.lstatSync(nextPath);

        if (stat.isDirectory()) {
            const [year, month] = fileName.split(' ');
            struct[year] = {
                ...struct?.[year],
                [month]: {
                    path: nextPath,
                    data: readDir(nextPath) // Рекурсивно обробляємо
вміст директорії
                }
            };
        } else {
            const dotIndex = fileName.lastIndexOf('.');
            const file = fileName.slice(0, dotIndex);
            const format = fileName.slice(dotIndex + 1); //
Визначаємо формат файлу

            if (!struct[file]) {
                struct[file] = { [format]: nextPath };
            } else {
                struct[file][format] = nextPath;
            }
        }
    }
    return struct;
};
```

Запити від користувачів API обробляються за допомогою маршрута app.get('/'), де в залежності від параметрів року та місяця, API повертає відповідні дані в JSON-форматі.

```
// Основний маршрут для отримання даних
app.get('/', (req, res) => {
    const { year, month } = req.query;

    const data = fs.readFileSync(path.join(DATASET_FOLDER,
DATASET_FILES.data));
    let responseData = JSON.parse(data);

    if (year) {
        responseData = responseData[year] ?? {};  // Повертаємо дані
для конкретного року
    }

    if (year && month) {
        responseData = responseData[month] ?? {};  // Повертаємо
дані для конкретного місяця
    }

    res.setHeader('Content-Type', 'application/json');
    res.json(responseData);  // Відправка даних у форматі JSON
});

// Оновлення індексації даних
app.get('/update', (_, res) => {
    const filePaths = readDir(path.join(ASSETS_FOLDER));    //
Індексація папки assets
    const stringData = JSON.stringify(filePaths);
    write(path.join(DATASET_FOLDER, DATASET_FILES.data),
stringData);  // Запис індексованих даних
    res.send();
});
```

Функція /update відповідає за індексацію нових даних у папці assets і оновлення структури індексованих даних у файлі data.json.

```
app.get('/update', (_, res) => {
    const filePaths = readDir(path.join(ASSETS_FOLDER));    //
Читання нових файлів з папки assets
    const stringData = JSON.stringify(filePaths);  // Перетворення
даних у формат JSON
```

```
    write(path.join(DATASET_FOLDER,          DATASET_FILES.data),
stringData);  // Запис оновлених даних у файл
    res.send('Дані оновлені');
});
```

Таким чином, весь процес збору, індексації, обробки та доступу до супутникових даних значно автоматизований, що спрощує процес розширення бази даних і забезпечує високий рівень доступності та ефективності при роботі з даними (рис.13).

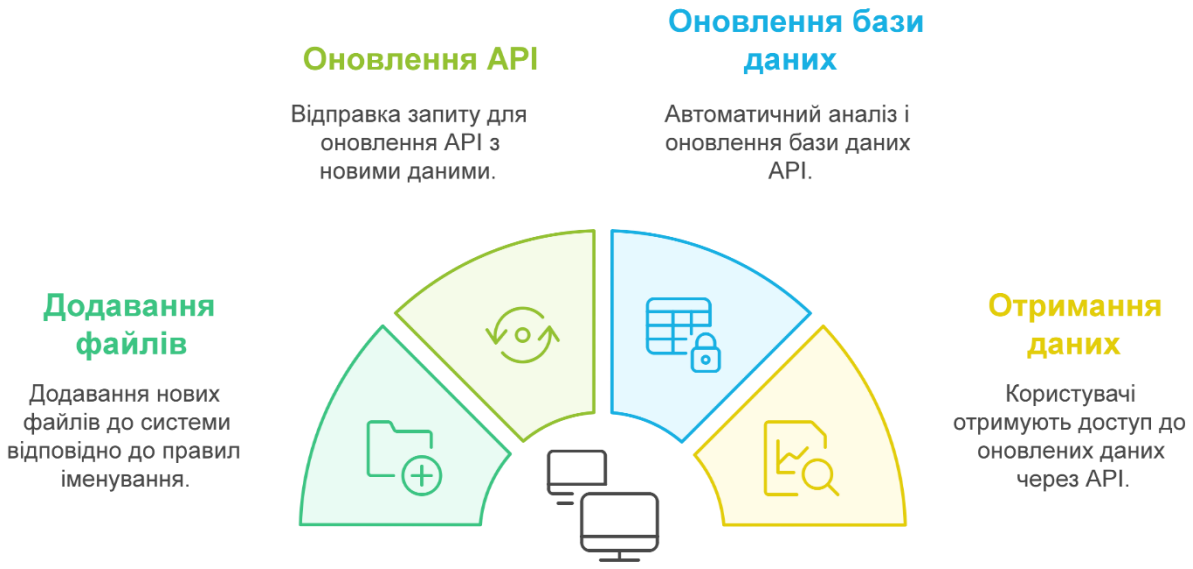

Рисунок 13 – Процес оновлення API

Створення користувацького інтерфейсу для інтеграції з API (рис.14), дозволяє користувачам взаємодіяти з системою через веб-інтерфейс. Взаємодія між клієнтом та сервером реалізується через API-запити. Важливим аспектом є обробка даних та їх відображення в зручному для користувача форматі (таблиці, графіки, форми).

Створення користувацького інтерфейсу для інтеграції з API є важливим етапом у розробці сучасних веб-застосунків, оскільки він дозволяє користувачам ефективно взаємодіяти з системою через інтуїтивно зрозумілий інтерфейс. Це передбачає кілька ключових етапів, починаючи з проєктування структури інтерфейсу, що дозволяє зручно взаємодіяти з даними. На цьому

етапі розробляються елементи управління, такі як кнопки, поля вводу, випадаючі списки, що забезпечують користувачам зручність під час взаємодії з системою.

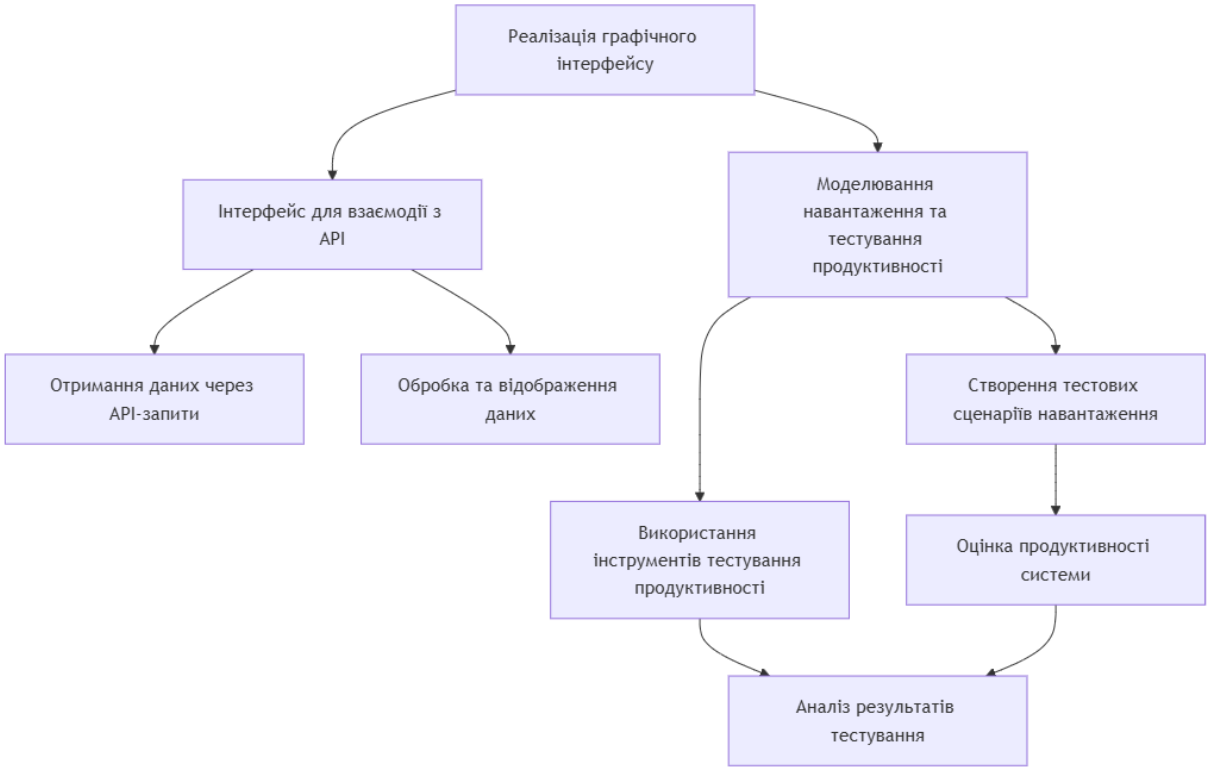

Рисунок 14 – Блок-схема реалізації графічного інтерфейсу

Також важливо створити адаптивний дизайн інтерфейсу, який дозволяє підтримувати доступність на різних пристроях, включаючи комп'ютери, планшети та мобільні телефони. Інтеграція з API є наступним важливим етапом. Користувацький інтерфейс взаємодіє з сервером через API-запити, що дозволяють передавати дані між клієнтською та серверною частинами системи. Зазвичай для цього використовуються RESTful API або GraphQL, що дозволяють організувати обмін даними, де запити можуть включати отримання, оновлення, створення або видалення даних, залежно від потреб користувача. Однією з важливих переваг є використання асинхронних запитів, таких як AJAX або fetch API, що дозволяють уникати блокування інтерфейсу під час очікування відповіді від сервера. Отримані дані потребують обробки

та подальшого відображення у зручному для користувача форматі. Це може бути реалізовано шляхом виведення даних у вигляді таблиць, що дозволяють сортувати та фільтрувати інформацію за необхідністю. Крім того, для візуалізації тенденцій та полегшення сприйняття інформації можуть використовуватися графіки або діаграми. У разі необхідності також розробляються форми для введення нових даних або редагування наявних. Важливо, щоб відображення даних було зручним і наочним, що забезпечує користувачам швидке сприйняття і взаємодію з інформацією. Одним з важливих аспектів створення інтерфейсу є забезпечення зручності та доступності. Для цього передбачаються механізми підказок, повідомлень про помилки та підтверджень перед виконанням критичних операцій, щоб користувач міг з легкістю орієнтуватися в інтерфейсі та уникати помилок. Також необхідно забезпечити оптимізацію швидкості завантаження та відповіді інтерфейсу, щоб зменшити час очікування користувача і підвищити загальну продуктивність системи. Важливо також враховувати потреби людей з обмеженими можливостями, надаючи підтримку екранним рідер (reader/считувачі) та використання клавіатурних скорочень. Тестування та налагодження є важливими етапами в процесі розробки користувацького інтерфейсу. Перевірка інтеграції інтерфейсу з API за допомогою тестів на різних пристроях і браузерах забезпечує коректну роботу інтерфейсу. Для тестування правильності відповіді від сервера та часу обробки API-запитів можуть використовуватися інструменти, такі як Postman, що дозволяють проводити глибоке тестування і виявляти можливі проблеми. Завдяки такому підходу користувачі отримують можливість працювати з даними через інтерактивний веб-інтерфейс, що забезпечує простоту використання, високу ефективність і гнучкість. Інтерфейс дозволяє зручно отримувати оновлену інформацію в реальному часі, взаємодіяти з даними та здійснювати необхідні дії без перезавантаження сторінки, що значно підвищує комфорт і швидкість роботи з системою.

ВИСНОВКИ

Сучасні технології ДЗЗ займають важливе місце в різних сферах, таких як екологічний моніторинг, сільське господарство, урбаністичне планування, наукові дослідження та управління природними ресурсами. Враховуючи постійне зростання обсягу супутникових даних, виникає потреба в ефективних інструментах для їх обробки, зберігання та інтеграції. У цьому контексті важливою є розробка інтерфейсів, таких як API, які забезпечують зручний доступ до цих даних і їх автоматизацію.

API є ключовим інструментом для систематизації та інтеграції супутникових даних. Його застосування дозволяє стандартизувати процеси обробки, зберігання та доступу до інформації, що є важливим для забезпечення масштабованості та інтеграції з іншими системами. API забезпечує ефективне взаємодію між різними програмними компонентами та полегшує доступ до даних для кінцевих користувачів, що особливо актуально в умовах сучасних екологічних та технологічних викликів.

Метою роботи було розробити API, який дозволяє систематизувати супутникові дані та автоматизувати їх обробку. У результаті було створено ефективний інструмент, який відповідає вимогам сучасних стандартів програмного забезпечення, забезпечує зручний доступ до даних і дозволяє інтегрувати їх у інші програмні рішення. API підтримує масштабування, що робить його ефективним інструментом для подальшого розвитку та адаптації до змінюваних умов.

Розроблене API дозволяє значно спростити доступ до супутникових даних і забезпечити їх ефективну обробку. Користувачі можуть отримувати необхідну інформацію в зручному форматі, що сприяє підвищенню продуктивності в різних галузях, таких як екологія, сільське господарство та безпека. Це дозволяє оперативно реагувати на різні виклики, аналізувати ситуації в реальному часі та приймати обґрунтовані рішення.

Розроблене API має великий потенціал для подальшого розвитку та вдосконалення. В майбутньому можна розширити функціональність, додаючи нові методи обробки даних, підтримку різних типів даних та інтеграцію з іншими інформаційними системами. Це дозволить значно розширити можливості використання супутникових даних у різних прикладних сферах, підвищуючи ефективність прийняття рішень та управління ресурсами.

Розробка такого API відкриває нові можливості для досліджень у галузі дистанційного зондування Землі, автоматизації обробки великих даних та інтеграції їх у складні інформаційні системи. Подальші вдосконалення та використання API дозволять підвищити точність та ефективність аналізу супутникових даних, що може мати великий вплив на багато важливих галузей, включаючи моніторинг навколишнього середовища, управління природними ресурсами та національну безпеку.

Розроблений API є потужним інструментом для ефективної обробки та інтеграції супутникових даних, що відповідає сучасним вимогам до обробки великих даних та автоматизації процесів. Це рішення сприятиме розвитку прикладних технологій у галузі дистанційного зондування Землі та забезпечить зручний доступ до даних для широкого кола користувачів, що сприятиме прийняттю більш обґрунтованих рішень у різних галузях діяльності.

ПЕРЕЛІК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. What is API? [Електронний ресурс] // RedHat. – Режим доступу: https://www.redhat.com/en/topics/api/what-is-an-api.
2. Satellite Remote Sensing Technology [Електронний ресурс] // Earth Observatory. – Режим доступу: https://earthobservatory.nasa.gov/features/RemoteSensing.
3. GDAL - Geospatial Data Abstraction Library [Електронний ресурс] // GDAL. – Режим доступу: https://gdal.org/.
4. Open Geospatial Consortium (OGC) [Електронний ресурс] // OGC. – Режим доступу: https://www.ogc.org/.
5. What is SAR (Synthetic Aperture Radar)? [Електронний ресурс] // GEOINT. – Режим доступу: https://www.geoint.com/what-is-sar/.
6. GIS for Environmental Management [Електронний ресурс] // ESRI. – Режим доступу: https://www.esri.com/en-us/industries/environment.
7. Remote Sensing for Agriculture [Електронний ресурс] // NASA. – Режим доступу: https://www.nasa.gov/feature/remote-sensing-for-agriculture.
8. Earth Observing Satellite Systems [Електронний ресурс] // ESA. – Режим доступу: https://www.esa.int/Applications/Observing_the_Earth.
9. Data Storage in Big Data Systems [Електронний ресурс] // IBM. – Режим доступу: https://www.ibm.com/topics/data-storage.
10. Big Data and Data Processing [Електронний ресурс] // Forbes. – Режим доступу: https://www.forbes.com/sites/forbestechcouncil/2021/04/08/how-big-data-is-driving-innovation-and-transformation-in-2021/.
11. Satellite Data for Disaster Response [Електронний ресурс] // Copernicus. – Режим доступу: https://www.copernicus.eu/en/applications/emergency-management.
12. API Design Best Practices [Електронний ресурс] // ProgrammableWeb. – Режим доступу: https://www.programmableweb.com/news/api-design-best-practices.

13. Introduction to Remote Sensing [Електронний ресурс] // University of California, Berkeley. – Режим доступу: https://geography.berkeley.edu/introduction-remote-sensing.
14. Role of APIs in Modern Development [Електронний ресурс] // Twilio. – Режим доступу: https://www.twilio.com/docs/usage/api.
15. Space-based Remote Sensing Technology [Електронний ресурс] // NASA Earth Science. – Режим доступу: https://earthscience.nasa.gov/learners/space-based-remote-sensing.
16. Understanding Geospatial Data Formats [Електронний ресурс] // Data.gov. – Режим доступу: https://www.data.gov/metadata/geospatial-data.
17. GIS Data Visualization Techniques [Електронний ресурс] // GIS Lounge. – Режим доступу: https://www.gislounge.com/gis-visualization-techniques/.