Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інтелектуальна система автоматизації та моніторингу агропідприємств із використанням сучасних вебтехнологій та штучного інтелекту»

Виконав: студент 2 курсу, групи 3АКІТР-24м спеціальності 174 – Автоматизація, комп'ютеровані технології та роботехніка
(шифр і назва спеціальності)

Олександр СОКОЛОВСЬКИЙ
(ПІБ студента)

Керівник: к.т.н., доцент кафедри АІІТ Володимир КОЦЮБИНСЬКИЙ
(науковий ступінь, вчене звання / посада, ПІБ керівника)

«10» 12 2025 р.

Опонент: д.т.н., професор Марія ЮХИМЧУК
(науковий ступінь, вчене звання / посада, ПІБ опонента)

«10» 12 2025 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)

«12» 12 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти II-ий (магістерський)
Галузь знань – Електроніка, автоматизація та електронні комунікації
Спеціальність – 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітньо-професійна програма – Інформаційні системи і Інтернет речей

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«26» вересня 2025 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Соколовському Олександру Олеговичу
(ПІБ автора повністю)

1. Тема роботи: Інтелектуальна система автоматизації та моніторингу агропідприємств із використанням сучасних вебтехнологій та штучного інтелекту.
Керівник роботи: к.т.н., доцент каф. АІІТ Коцюбинський В.Ю.
Затвердженні наказом ВНТУ від «24» вересня 2025 року № 313.
2. Строк подання роботи студентом: до «12» грудня 2025 року.
3. Вихідні дані до роботи: серверна і клієнтська частини інтелектаальної системи автоматизації та моніторингу. Вимоги до техніки: ОС Ubuntu 20.04. Процесор Intel Core i7 або вище. Об'єм оперативної пам'яті 64 ГБ або більше. Жосткий диск 1 ТВ Sata2 або вище.
4. Зміст текстової частини: Вступ; Аналіз предметної області, існуючих систем автоматизації та моніторингу агропідприємств; Вибір архітектури системи та її проєктування; Огляд та вибір інструментів розробки системи; Розробка програмного забезпечення та тестування системи; Висновки; Список використаних джерел.
5. Перелік ілюстративного (або графічного) матеріалу: UML-діаграма

класів; Архітектура базової моделі в базі даних; UML-діаграма обєктів UML-діаграма розгортання; UML - діаграма варіантів використання; Вигля екранів розробленого додатку;

6. Консультанти розділів роботи

Розділ змістової частини роботи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 3	Володимир КОЦЮБИНСЬКИЙ, к.т.н., доц. каф. АІІТ	25.09.2025	10.12.2025

7. Дата видачі завдання: «25» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітк
1	Аналіз об’єкта автоматизації	25.09–05.10.2025	Виконан
2	Аналіз існуючих систем для автоматизації системи управління агропідприємств з використанням ІШ	05.10 – 25.10.2025	Виконан
3	Розробка програмного забезпечення	25.10 – 10.11.2025	Виконан
4	Тестування розробленого програмного забезпечення	05.11 – 20.11.2025	Виконан
5	Оформлення пояснювальної записки, графічного матеріалу і презентації	20.11 – 03.12.2025	Виконан
6	Попередній захист роботи	до 03.12.2025	Виконан
7	Захист роботи	до 19.12.2025	Виконано

Студент _____ (підпис) Олександр СОКОЛОВСЬКИЙ (прізвище та ініціали)

Керівник роботи _____ (підпис) Володимир КОЦЮБИНСЬКИЙ (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.8; 631.171

Соколовський О. О. Інтелектуальна система автоматизації та моніторингу агропідприємств із використанням сучасних вебтехнологій та штучного інтелекту. Магістерська кваліфікаційна робота зі спеціальності 174 - Автоматизація, комп'ютерно-інтегровані технології, та робототехніка, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2025. - 101 с.

Укр. мовою. Бібліогр.: 31 дж.; рис.:14; табл.: 3

У даній роботі розроблено інтелектуальну систему автоматизації та моніторингу агропідприємств, призначену для підвищення ефективності агровиробництва шляхом впровадження сучасних вебтехнологій, цифрового моніторингу та алгоритмів штучного інтелекту. Виконано дослідження сучасних тенденцій розвитку точного землеробства, систем агромоніторингу. Проведений аналіз наявних систем. Сформовано концепцію архітектури агросистеми. Наведено результати тестування функціональних модулів та оцінено

Ключові слова: інтелектуальна система, агромоніторинг, штучний інтелект, вебтехнології, IoT, супутникові дані, NestJS, Next.js.

ABSTRACT

Sokolovskyi O. O. Intelligent System for Automation and Monitoring of Agricultural Enterprises using Modern Web Technologies and Artificial Intelligence. Master's Thesis in Specialty 174 – Automation, Computer-Integrated Technologies, and Robotics, Educational Program – Intelligent Computer Systems. Vinnytsia: VNTU, 2025. – 101 p.

In Ukrainian, 3 chapters, 31 sources, 14 figures, 3 tables.

This Master's Thesis describes the development of an Intelligent System for Automation and Monitoring of Agricultural Enterprises, which is designed to enhance the efficiency of agricultural production through the implementation of modern web technologies, digital monitoring, and artificial intelligence algorithms. The work includes a comprehensive study of current trends in the development of precision agriculture and agro-monitoring systems, alongside an analysis of existing solutions. The conceptual architecture of the agro-system was formulated, and the results of functional module testing and system effectiveness evaluation are presented.

Keywords: Intelligent system, agro-monitoring, artificial intelligence, web technologies, IoT, satellite data, NestJS, Next.js.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах глобальних викликів, таких як зміна клімату, демографічні зрушення, дефіцит робочої сили та зростаючі вимоги до продовольчої безпеки, аграрний сектор відіграє ключову роль у забезпеченні сталого розвитку світової економіки. За даними Продовольчої та сільськогосподарської організації ООН (FAO), для задоволення глобального попиту на продовольство до 2050 року обсяги сільськогосподарського виробництва мають зрости приблизно на 50% порівняно з рівнем 2020 року [2]. Досягнення цієї мети неможливе без глибокої цифрової трансформації галузі та впровадження сучасних технологій автоматизації.

Автоматизація агропідприємств сьогодні стає не просто конкурентною перевагою, а необхідністю. Цифрові технології, такі як Інтернет речей (IoT), супутниковий моніторинг, системи точного землеробства, а також штучний інтелект (ШІ) дозволяють підвищити ефективність управління полями, зменшити витрати ресурсів, вчасно виявляти ризики (посуха, хвороби, дефіцит поживних речовин) та приймати науково обґрунтовані рішення [3].

Особливої актуальності набуває інтеграція штучного інтелекту у процеси планування, аналітики та прогнозування, що дає змогу переходити від реактивного до прогностичного управління агровиробництвом.

Зміна клімату та збільшення частоти екстремальних погодних явищ суттєво впливають на сільське господарство. Згідно з доповіддю Міжурядової групи експертів з питань зміни клімату (IPCC), за відсутності адаптаційних заходів середня врожайність основних культур у світі може знизитися на 10–25% до 2050 року [4]. У цих умовах автоматизовані системи моніторингу та управління стають ключовим інструментом для підвищення стійкості агросектору. Окрім того, автоматизація сприяє вирішенню проблеми дефіциту робочої сили в аграрному секторі. Використання автономної техніки, дронів,

роботизованих систем збирання врожаю та інтелектуальних систем прийняття рішень дозволяє зменшити залежність від людського фактору, підвищити точність виконання агротехнічних операцій та скоротити витрати [4].

Метою роботи є розробка інтелектуальної системи, яка підвищує ефективність роботи агропідприємства за рахунок цифрового моніторингу стану посівів, автоматизованого аналізу польових даних та оптимального планування агротехнічних операцій, зокрема внесення добрив. Запропонована система має забезпечити агрономів обґрунтованими рекомендаціями сформованими на базі ШІ та LLM-моделей на основі даних із сенсорів, дронів і супутникових знімків. Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної області, існуючих систем моніторингу та тенденцій розвитку агротехнологій;
- розробити архітектуру інтелектуальної системи з використанням сучасних вебтехнологій;
- реалізувати модулі збору, збереження та аналітичної обробки агроданих.
- інтегрувати штучний інтелект на основі MCP-підходу (Model–Context–Prompt) для автоматичного генерування рекомендацій;
- провести тестування системи та порівняти її ефективність із існуючими аналогами.

Об’єктом дослідження є процес автоматизації управлінськими та виробничими процесами агропідприємства.

Предметом дослідження є методи, алгоритми, архітектурні принципи, технологічні підходи та інструменти, що застосовуються для збирання, аналізау та обробки інформації, щодо стану агротехнічних обєктів.

Методи дослідження базуються на аналізі та синтезі наукових підходів до використання штучного інтелекту у збиранні, аналізі, та обробці даних, застосуванні методів машинного навчання для прогнозування та оптимізації агротехнологічних процесів, проєктуванні багатошарової архітектури інтелектуальної системи з використанням сучасних вебтехнологій.

Науково практичний результат роботи полягає в розробці комплексного підходу до побудови інтелектуальної системи агромоніторингу з використанням MCP-парадигми. Це дозволяє не лише аналізувати великі масиви агроданих, але й генерувати контекстні рекомендації, що враховують індивідуальні характеристики полів, погодні умови та історичні дані.

Практична цінність роботи полягає у створенні інтелектуальної та високомасштабованої системи, яка дозволяє агропідприємствам перейти від реактивного до проактивного управління польовими процесами. Впровадження цієї системи забезпечує кількісне підвищення ефективності за рахунок автоматизованого аналізу геопросторових та телеметричних даних, що мінімізує людський фактор і зменшує рутинне навантаження на агрономів. Зокрема, система надає обґрунтовані рекомендації (на базі ШІ) щодо оптимального внесення добрив і засобів захисту, що сприяє суттєвій економії ресурсів (до 15-20%) та зниженню негативного впливу на навколишнє середовище. Таким чином, система не лише оптимізує агротехнічні операції та підвищує прозорість процесу прийняття рішень, але й прямо впливає на збільшення врожайності та загальну фінансову стійкість агропідприємства.

Апробація та публікації матеріалів досліджень. Основні результати виконання магістерської кваліфікаційної роботи були опубліковані в матеріалах Всеукраїнської науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, ВНТУ, 2025 рр.)[1]

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ СИСТЕМ АВТОМАТИЗАЦІЇ МОНІТОРИНГУ АГРОПІДПРИЄМСТВ

1.1 Аналіз предметної області

В даній роботі аналізом об’єкту автоматизації є агропромисловий сектор, а саме процес моніторингу та збору даних[1].

Агропромисловий сектор є одним із базових елементів національної економіки, який визначає продовольчу безпеку держави, сприяє розвитку експорту та забезпечує значну частку ВВП. В умовах глобалізації, зміни клімату, дефіциту трудових ресурсів та зростання попиту на продовольство аграрні підприємства стикаються з необхідністю підвищення ефективності виробництва, оптимізації витрат і впровадження інноваційних технологій.

Цифровізація аграрного сектору, зокрема впровадження систем автоматизації та інтелектуального моніторингу, сьогодні є не лише технологічним трендом, а й стратегічною потребою. За даними FAO та World Bank, до 2050 року світове виробництво продовольства повинно зрости майже на 50%, щоб задовольнити зростаючий попит населення. Досягнення цього показника можливе лише за умови переходу до концепції «розумного землеробства» (*smart farming*), де прийняття рішень базується на даних, автоматизованих аналітичних системах та штучному інтелекті.

Системи автоматизації аграрного виробництва — це програмно-апаратні комплекси, які забезпечують моніторинг стану посівів, ґрунту, погоди та техніки в режимі реального часу, а також автоматизують планування й управління виробничими процесами. Їх основна мета — підвищення ефективності агропідприємств через зменшення витрат, підвищення врожайності та зниження ризиків.

До основних функцій таких систем належать:

- моніторинг полів: збір даних із сенсорів про вологість, температуру ґрунту, опади, вміст поживних речовин;
- супутникове та аерофотознімання: аналіз стану вегетації культур за індексами NDVI, NDRE, GNDVI;
- автоматизоване планування: складання графіків поливу, підживлення, сівозміни;
- прогнозування врожайності: на основі історичних даних, погодних умов і моделювання;
- аналіз ризиків: раннє виявлення шкідників, захворювань та нестачі елементів живлення.

Переваги впровадження таких систем:

- підвищення точності агротехнічних операцій. Рішення базуються на об’єктивних даних, а не на досвіді оператора;
- зниження витрат ресурсів. Добрива, вода та засоби захисту рослин застосовуються саме там і тоді, де це необхідно;
- збільшення врожайності. Точне планування та прогнозування дозволяє покращити результати виробництва;
- моніторинг у реальному часі. Керівництво може оперативно реагувати на будь-які зміни в полі.

З розвитком Інтернету речей (IoT), хмарних обчислень та штучного інтелекту системи автоматизації поступово трансформуються в інтелектуальні екосистеми управління агровиробництвом, здатні приймати самостійні рішення на основі багатофакторного аналізу.

1.2 Порівняння з існуючими аналогами

На ринку існує низка популярних рішень для автоматизації агровиробництва. Нижче наведено порівняльну характеристику найпоширеніших систем. Серед таких систем виділяють:

Cropwise — це комплексна платформа для моніторингу сільськогосподарських полів у режимі реального часу, яка використовує супутникові знімки, погодні дані та алгоритми аналітики для прогнозування стану культур і врожайності[5]. Система дозволяє відстежувати вегетаційні індекси (NDVI, EVI, LAI), створювати карти врожайності, аналізувати історичні дані та управляти плануванням агротехнічних робіт. Інтерфейс системи наведений на рисунку 1.1.

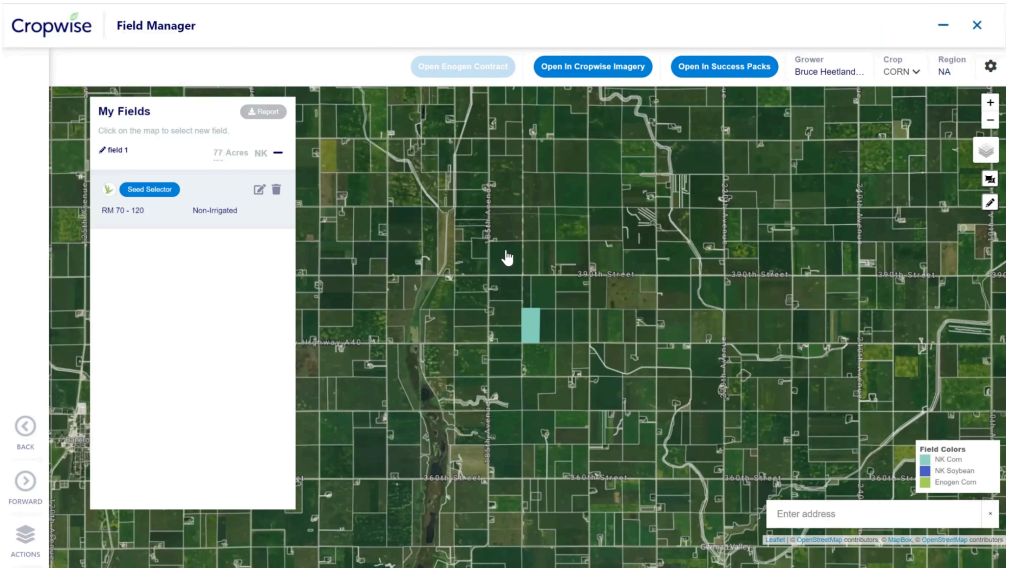

Рисунок 1.1 – Інтерфейс системи Cropwise

OneSoil — безкоштовна вебплатформа для моніторингу стану полів за допомогою супутникових знімків Sentinel-2[6]. Система автоматично визначає межі полів, розраховує вегетаційні індекси та формує рекомендації для планування робіт. Орієнтована на фермерів малого та середнього бізнесу. Інтерфейс системи наведений на рисунку 1.2.

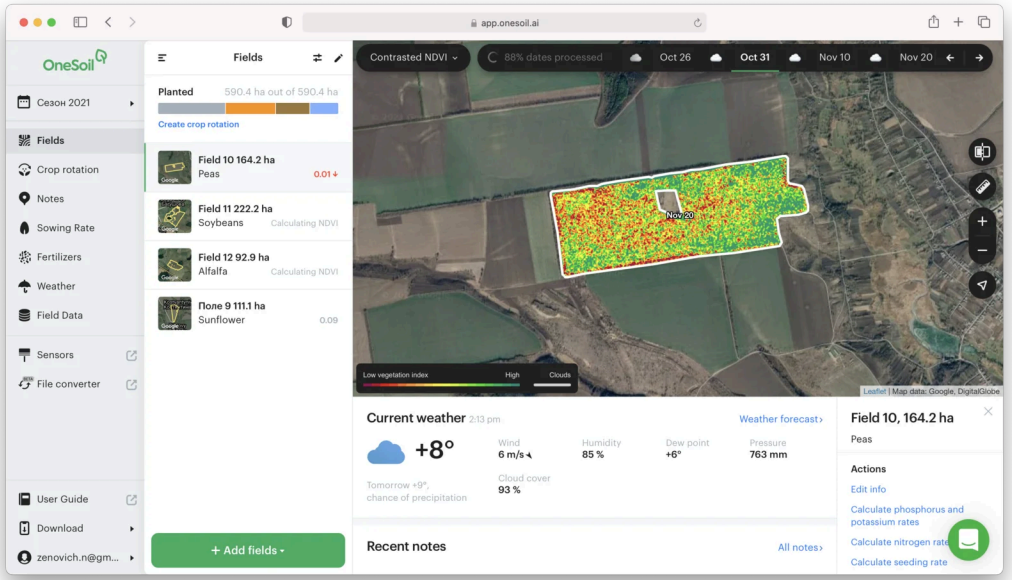

Рисунок 1.2 – Інтерфейс системи OneSoil

Climate FieldView — це професійна система аналітики та моніторингу від компанії The Climate Corporation (Bayer)[7]. Вона надає аграріям можливість аналізувати стан полів на основі даних з техніки, сенсорів, дронів і супутників. Також підтримує інтеграцію з IoT-пристроями та агромашинами для автоматичного збору даних. Інтерфейс системи наведений на рисунку 1.3.

Рисунок 1.3 – Інтерфейс системи Climate FieldView

EOSDA Crop Monitoring — це інтелектуальна онлайн-платформа для аналізу стану полів на основі супутникових знімків і алгоритмів машинного навчання. Система надає інструменти для порівняння сезонів, прогнозування розвитку культур, зонування полів та створення рекомендацій для агрономів.[8]

Платформа активно використовує спектральні індекси (наприклад, NDRE) для моніторингу посівів на різних стадіях вегетації та оцінки потреби в азотних добривах. Вона також включає функціонал для відстеження погодних умов і моделювання вологості ґрунту, що критично важливо для планування поливу. Крім того, система пропонує API для корпоративних клієнтів, забезпечуючи інтеграцію її аналітичних можливостей у сторонні ERP-системи чи власні управлінські платформи агрохолдингів. Інтерфейс системи наведений на рисунку 1.4.

Рисунок 1.4 – Інтерфейс системи EOSDA Crop Monitoring

Проведений аналіз існуючих систем автоматизації та моніторингу агропідприємств виявив значний спектр доступних на ринку рішень, що вимагало їхньої систематизації та порівняння. У таблиці 1.1 детально

розглянуто використання ними штучного інтелекту, джерела даних (супутники, IoT), основні переваги та недоліки, а також цільову аудиторію. Такий підхід дав змогу чітко ідентифікувати функціональні прогалини у наявних продуктах, зокрема у сфері комплексної інтеграції даних і повноцінної ШІ-аналітики, що, зрештою, стало обґрунтуванням необхідності розробки власної, більш гнучкої та інтелектуальної системи.

Таблиця 1.1 - Порівняння існуючих аналогів.

Назва системи	Використання AI	Джерела даних	Основні переваги	Недоліки	Цільова аудиторія
Cropio	Так	Супутники, IoT, ERP	Висока точність, історичний аналіз, планування	Висока вартість	Великі агрохолдинги
OneSoil	Частково	Супутники Sentinel-2	Простота, безкоштовність, автоматичне визначення полів	Обмежений функціонал	Малі фермери
Climate FieldView	Так	IoT, дрони, техніка	Глибока аналітика, інтеграція, прогнозування	Висока ціна, залежність від обладнання	Комерційні господарства
EOSDA Crop Monitoring	Так	Супутники, API	Машинне навчання, інтеграція, локалізація	Обмеження безкоштовної версії	Середні та великі ферми

1.3 Постановка задач дослідження

На основі проведеного аналізу сучасного стану аграрних технологій, методів моніторингу полів та тенденцій розвитку точного землеробства

встановлено, що впровадження інтелектуальних автоматизованих систем є ключовим чинником підвищення конкурентоспроможності агропідприємств[9]. Сучасні виклики — зміна клімату, нестабільність врожайності, високі витрати ресурсів, дефіцит робочої сили та потреба в оперативних управлінських рішеннях — вимагають переходу від традиційних методів до цифровізованих технологічних рішень.

Аналіз наявних систем агромоніторингу показав, що значна частина існуючих платформ не забезпечує комплексності, інтеграції супутникових, сенсорних та історичних даних, а також не має повноцінної AI-аналітики, здатної формувати персоналізовані рекомендації. Це створює потребу у розробці сучасної інтелектуальної системи, яка поєднуватиме методи машинного навчання, вебтехнології та автоматизований моніторинг агропроцесів.

Для побудови такої системи необхідно застосувати архітектурний підхід, що забезпечить масштабованість, гнучкість і надійність. Найефективнішим рішенням є використання багаторівневої (N-tier) клієнт-серверної архітектури, де фронтенд (клієнт) відокремлений від бекенду (серверу)[10]. При цьому, для забезпечення масштабованості та відмовостійкості, серверний рівень реалізовано за допомогою мікросервісного підходу із розподілом на окремі сервіси, які відповідають за обробку супутникових даних, роботу з IoT-сенсорами, аналітику та формування рекомендацій, що дозволить уникнути конфліктів функцій та підвищити стійкість системи до збоїв. Важливим також є впровадження контекстно-керованого MCP-підходу (Model–Context–Prompt), який дозволяє швидко адаптувати логіку ШІ та підвищує точність рекомендацій[11-12].

Особливу увагу необхідно приділити захисту даних агропідприємства, конфіденційності вимірювань, а також безпечній роботі з хмарними сервісами. Розділення даних телеметрії, користувачів і аналітичних модулів на окремі

сервери та застосування ролей доступу підвищує рівень безпеки і надійності системи.

Таким чином, постановка завдання дослідження полягає у створенні інтелектуальної, багатокомпонентної та надійної системи, яка забезпечить автоматизований збір, обробку та аналіз агроданих, прогнозування стану культур та формування рекомендацій для агрономів і керівників агропідприємств.

1.4 Висновки до розділу

У даному розділі були розглянуті стратегічні потреби агропромислового сектору, що вимагають переходу до концепції "розумного землеробства" для підвищення ефективності виробництва та реагування на глобальні виклики.

Проведений аналіз предметної області чітко обґрунтував необхідність впровадження інтелектуальних систем, які забезпечують моніторинг, прогнозування та автоматизоване управління агротехнічними процесами.

Порівняння існуючих аналогів (Cropwise, OneSoil, Climate FieldView та інші) виявило їхні значні переваги у використанні супутникових даних та ML, але також і суттєві недоліки, такі як висока вартість, обмежена інтеграція з IoT-пристроями та недостатня комплексність ШІ-аналітики. На основі цього було зроблено висновок про актуальність постановки завдання — розробки власної інтелектуальної системи, яка поєднує всі необхідні функціональні можливості.

2 ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ ТА МОНІТОРИНГУ АГРОПІДПРИЄМСТВ

2.1 Функціональний аналіз системи

Інтелектуальна система автоматизації та моніторингу агропідприємств проєктується як ключовий інструмент для підтримки процесів управління рослинництвом на основі цифрових даних та передових аналітичних методів. Основною метою такого високотехнологічного рішення є забезпечення агропідприємства повним комплексом інструментів, починаючи від ефективного збору інформації і закінчуючи формуванням практичних рекомендацій, що базуються на штучному інтелекті. Основні функціональні можливості включають:

- моніторинг стану полів. Ця функція передбачає не лише візуалізацію даних супутникових спостережень у динаміці, але й забезпечення контролю динаміки розвитку рослинності через аналіз спектральних індексів. Постійний аналіз даних дозволяє системі швидко виявляти аномалії у рості культур, що слугує першим сигналом для агронома;
- оцінювання ризиків. Система здатна автоматично виявляти ділянки із потенційними стресами (тепловий, водний, нутрієнтний) та оперативно проводити аналіз можливих причин стресу, включаючи погодні умови, дефіцит вологості чи пошкодження шкідниками. У разі критичних змін користувач отримує автоматичні попередження, що дозволяє миттєво реагувати на загрози.
- прогнозування продуктивності. Цей модуль використовує історичні дані врожайності та стану полів, інтегруючи їх із поточними метеоданими для оцінки майбутньої врожайності з високою точністю. Система також може аналізувати сценарії розвитку поля за різних погодних і технологічних

умов, допомагаючи керівництву приймати довгострокові стратегічні рішення;

- генерування рекомендацій за допомогою ШІ. Застосовуючи принцип контекстно-керованого підходу (MCP), система здатна до автоматичного формування рекомендацій щодо догляду за культурами — від норм і термінів внесення добрив до графіків обробки. Що важливіше, ці рекомендації є адаптованими під кожну ділянку поля, забезпечуючи принципи точного землеробства. Такий глибокий аналіз великих масивів даних значно підвищує точність прийняття рішень порівняно з традиційними методами.
- управління технологічними процесами. Це включає ведення журналу виконаних агроопераций, що забезпечує прозорість історії поля. Система також допомагає у контролі використання ресурсів та ефективному плануванні робіт із технікою та персоналом.

Для наочного відображення функціональних вимог та взаємодії користувачів із системою була розроблена UML Діаграма варіантів використання (рис. 2.1). Вона демонструє, як зовнішні учасники процесу, відомі як актори, взаємодіють з основними інтелектуальними та управлінськими можливостями системи.

Діаграма чітко розподіляє функціонал на три логічні групи: Моніторинг та Ризики, Прогнозування та Рекомендації, а також Управління та Інтеграція. Це допомагає не лише визначити ключові вимоги до програмного забезпечення, але й встановити логічний розподіл обов’язків між учасниками аграрного процесу.

Основними акторами системи є:

- агроном: Взаємодіє з модулем моніторингу, отримує персоналізовані рекомендації ШІ щодо польових робіт та веде журнал агроопераций;

- керівник: Фокусується на високорівневих завданнях, таких як оцінювання майбутньої врожайності, моделювання сценаріїв та контроль використання ресурсів;
- зовнішні системи (Сенсори, Супутники): Пасивні актори, які забезпечують безперервну інтеграцію телеметрії та геопросторових даних для роботи аналітичних модулів.

Зв'язки на діаграмі показують, що найбільш критичні інтелектуальні функції, як-от Виявлення аномалій у рості та Аналіз причин стресу, часто є включеними («include») в процес генерації персоналізованих рекомендацій ШІ. Це підкреслює, що рекомендації є науково обґрунтованими та контекстуально залежними від поточного стану поля, а не загальними.

Таким чином, діаграма варіантів використання ілюструє повний цикл роботи інтелектуальної системи: від автоматизованого збору даних до надання практичних, диференційованих рішень кінцевим користувачам.

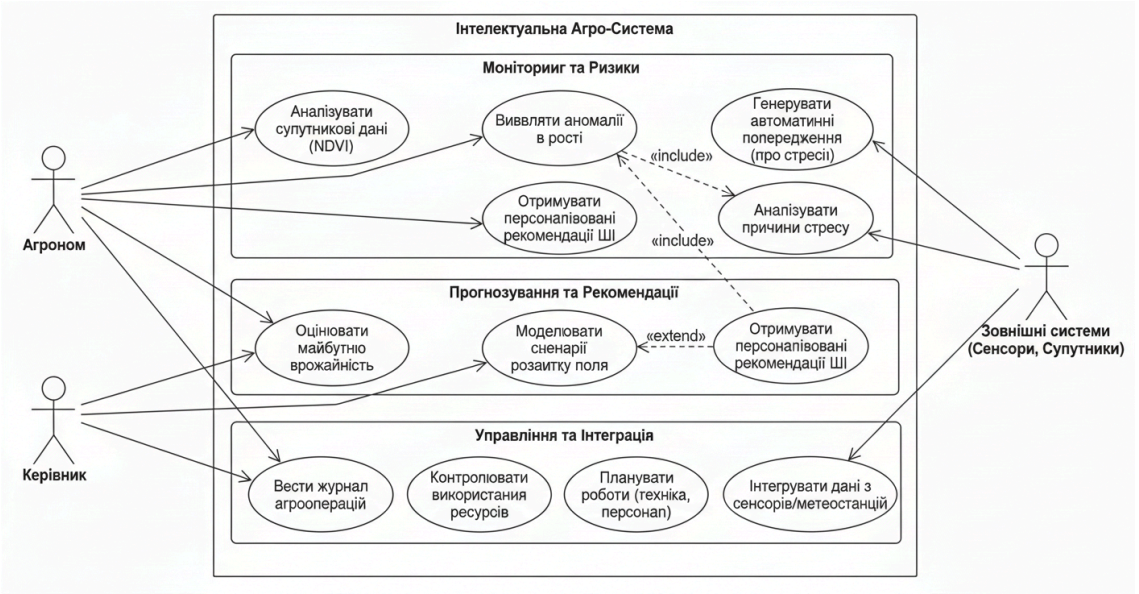

Рисунок 2.1 – UML діаграма варіантів використання

2.2 Проєктування архітектури інтелектуальної веб-системи

Проєктування архітектури інтелектуальної системи автоматизації та моніторингу агропідприємств є одним із ключових етапів розробки. Від правильності побудови архітектури залежить стабільність функціонування системи, можливість її подальшого розвитку, масштабованість під збільшення кількості користувачів та обсягів даних, а також швидкість реагування на події, пов’язані з динамічними змінами в аграрних процесах.

Система повинна обробляти великі масиви різнорідних даних: геопросторову інформацію, супутникові індекси (NDVI, EVI, NDRE), IoT-вимірювання, метеодані, історію агротехнічних операцій, дрони та зображення[13]. Крім того, система містить вбудований модуль штучного інтелекту (LLM) для підтримки прийняття рішень. Такий обсяг функціональності вимагає застосування багаторівневої клієнт-серверної архітектури, де серверний рівень реалізовано як мікросервісний, яка дозволяє адаптувати окремі компоненти під специфічні вимоги без втручання у загальний функціонал.

2.2.1 Обґрунтування вибору архітектурного підходу

Аграрні підприємства працюють у середовищі, де дані надходять із різних джерел у режимі реального часу. Це накладає на систему низку вимог:

- оперативність обробки даних (High Throughput): щогодини змінюються погодні умови, надходять нові супутникові знімки та IoT-вимірювання, що вимагає високої інтенсивності оновлення та миттєвої, оперативної обробки і збереження великих обсягів інформації;

- підтримка геопросторових даних (GIS Integration): для точного позиціювання полів, зондування та ефективного аналізу супутникових шарів критично необхідна надійна інтеграція технологій GIS (PostGIS) на рівні сховища та бізнес-логіки;
- комплексність аналітики та ШІ: наявність складних прогнозних моделей і ШІ вимагає, щоб рекомендації, які формуються, враховували актуальний контекст (історію, зовнішні фактори) і були здатні працювати з різнорідними даними;
- горизонтальна масштабованість: у періоди пікового навантаження кількість запитів може суттєво зростати, що вимагає можливості горизонтального масштабування окремих сервісів незалежно від інших;
- стійкість до відмов: система повинна забезпечувати високу відмовостійкість, при якій збої чи технічні проблеми в одному програмному модулі не призводять до зупинки всієї системи, гарантуючи її безперервну роботу.

Порівнявши можливі архітектурні варіанти, зокрема монолітну архітектуру, сервісно-орієнтовану архітектуру (SOA) та мікросервісну архітектуру, було проведено їх оцінювання щодо задоволення ключових вимог.

Таблиця 2.1 – Порівняння архітектурних підходів за ключовими вимогами

Критерій	Монолітна архітектура	SOA (Сервісно-орієнтована)	Мікросервісна архітектура
Масштабованість (Горизонтальна)	Низька (масштабується вся система)	Середня (складно розділити сервіси)	Висока (незалежне масштабування)
Відмовостійкість (Ізоляція)	Низька (збій в одному модулі зупиняє все)	Середня (велика залежність від шини)	Висока (повна ізоляція сервісів)

Продовження таблиці 2.1

Критерій	Монолітна архітектура	SOA (Сервісно-орієнтована)	Мікросервісна архітектура
Обробка різнорідних даних	Середня (вимагає єдиного стеку)	Висока (дозволяє різні стеки)	Висока (дозволяє різні технології для кожного сервісу)
Швидкість розробки/розгортання	Середня	Низька (складна інтеграція)	Висока (незалежні команди/деплойменти)

Мікросервісна архітектура була обрана, оскільки вона найкраще відповідає вимогам стійкості та гнучкості в умовах високої динаміки аграрних процесів. Цей підхід забезпечує горизонтальне масштабування ресурсомістких модулів (обробка супутникових даних та ШІ) незалежно від інших, а також гарантує високу відмовостійкість завдяки ізоляції кожного сервісу. Це є критично важливим для надійної роботи інтелектуальної системи, яка оперує даними у режимі реального часу.

2.2.2 Багаторівнева структурна модель системи

Проєктована інтелектуальна система реалізована на основі багаторівневої (N-tier) клієнт-серверної архітектури, що дозволяє досягти високої модульності, підвищеної відмовостійкості та незалежного масштабування кожного компонента. Структурна модель складається з чотирьох основних логічних рівнів, які забезпечують чітке функціональне розділення і послідовну обробку даних. Цей поділ гарантує, що збій на клієнтському рівні не вплине на обробку даних на рівні застосунків. Використання багаторівневої моделі також суттєво

підвищує безпеку, ізолюючи критичний Рівень даних від прямого доступу ззовні. Кожен рівень взаємодіє лише із сусідніми рівнями, що мінімізує залежності та спрощує подальше обслуговування системи. Такий підхід є оптимальним для систем з високим трафіком та потребою в обробці великих даних, характерних для агросектору. Крім того, чіткі межі між рівнями дозволяють незалежно оновлювати технологічний стек або мову програмування кожного рівня. Наприклад, оновлення бази даних не вимагатиме переписування фронтенду, якщо API залишається незмінним. Завдяки такій структурі, система зберігає гнучкість, необхідну для швидкої адаптації до нових типів сенсорів або супутникових даних. Це забезпечує необхідну довговічність та інвестиційну привабливість розробленого рішення.

Таблиця 2.2 – Логічні рівні структурної моделі системи

№	Назва Рівня (Tier)	Основне призначення та технології
1	Рівень подання (Presentation Tier)	Відповідає за взаємодію з кінцевим користувачем. Включає веб-інтерфейс (фронтенд) та забезпечує візуалізацію даних, інтерактивних карт та дашбордів.
2	Рівень збору та інтеграції даних (Data Collection Tier)	Приймає, валідує та нормалізує потоки даних від зовнішніх джерел: API супутників, метеостанції та IoT-сенсори. Слугує першим буфером даних.
3	Рівень застосунків (Application/Logic Tier)	Центральний рівень, де розміщені всі мікросервіси. Виконує ключову бізнес-логіку, складні аналітичні алгоритми, розрахунок індексів, ШІ-моделювання та генерацію рекомендацій.
4	Рівень даних (Data Tier)	Сховища даних. Використовує гібридний підхід: PostgreSQL + PostGIS (для геопросторових даних) та S3-сумісне сховище для зберігання великих об'єктів (сирі знімки).

2.2.3 Опис функціональних мікросервісів та їх взаємодія

Мікросервісна архітектура (обрана в 2.2.1) реалізується на Рівні застосунків (Application/Logic Tier) і передбачає повну декомпозицію системи на невеликі, незалежні та самодостатні сервіси, які комунікують між собою. Це дозволяє забезпечити необхідну гнучкість, масштабованість та відмовостійкість для роботи зі складними агротехнічними даними.

Функціональне розділення мікросервісів відбувається відповідно до ключових бізнес-функцій, визначених у Розділі 2.1:

- сервіс моніторингу та геоданих (Geo/Monitoring Service): Відповідає за обробку великих обсягів геопросторових даних. Основні функції включають завантаження та попередню обробку супутникових знімків, розрахунок індексів вегетації (NDVI, NDRE), виконання полігональних операцій (зонування, визначення меж полів) та взаємодію з базою даних PostGIS;
- сервіс телеметрії та погодних даних (Telemetry/Weather Service): Відповідає за прийом, валідацію, агрегацію та зберігання даних часових рядів, що надходять від IoT-сенсорів (датчики ґрунту, метеостанції) та зовнішніх метеорологічних API. Цей сервіс надає історичний та актуальний контекст для ШІ-аналізу;
- сервіс ШІ та рекомендацій (AI/Recommendation Service): Це інтелектуальне ядро системи. Він запускає складні ML-моделі для прогнозування врожайності, ідентифікації зон стресу, а також використовує LLM-моделі для формування фінальних, контекстуально обґрунтованих агрономічних рекомендацій (щодо добрив, іригації).
- сервіс управління та аутентифікації (Management/Auth Service): Несе відповідальність за всі адміністративні та користувацькі функції: реєстрація та авторизація, управління ролями доступу (Агроном, Керівник), ведення журналу агрооперацій та планування робіт;

Для забезпечення високої продуктивності та надійності використовується комбінація синхронної та асинхронної комунікації між сервісами.

- синхронна комунікація (API Gateway): Всі зовнішні запити від клієнтського рівня (Presentation Tier) надходять через єдиний API Gateway. Шлюз API виконує функції балансування навантаження, кешування та автентифікації, після чого маршрутизує запит до відповідного мікросервісу;
- асинхронна комунікація (Брокер повідомлень): Внутрішня взаємодія між мікросервісами реалізована за подієвою моделлю (Event-Driven) з використанням брокера повідомлень (наприклад, Apache Kafka). Це критично важливо для обробки великих потоків даних, де сервіси не повинні чекати відповіді одне від одного. Наприклад, сервіс моніторингу надсилає подію "Новий індекс NDVI готовий" у загальну чергу, а сервіс ШІ та рекомендацій, підписаний на цю подію, миттєво запускає свій аналіз. Така асинхронність гарантує високу пропускну здатність і відмовостійкість;

Ця модель забезпечує повну ізоляцію відмов: якщо, наприклад, тимчасово недоступний сервіс ШІ, інші сервіси (Моніторинг, Телеметрія) продовжують працювати та збирати дані, що зберігаються у черзі для подальшої обробки.

2.3 Вибір СУБД та проєктування моделі даних

Для реалізації інтелектуальної системи автоматизації та моніторингу агропідприємств було обрано систему керування базами даних PostgreSQL, оскільки вона поєднує високу стабільність, продуктивність і широкий спектр інструментів для обробки структурованих та аналітичних даних. PostgreSQL

ефективно працює в сервісах, побудованих за принципами мікросервісної архітектури, що було одним із ключових чинників вибору цієї СУБД[14].

PostgreSQL є сучасною реляційною базою даних із відкритим кодом, що підтримує складні типи даних, транзакційну обробку та масштабування[15]. Для даного проєкту важливою стала її здатність забезпечувати надійну роботу з великими наборами даних, що постійно оновлюються — зокрема з інформацією про агровиробництво, історію полів, телеметрію від сенсорів, прогнози погоди та рекомендації системи.

Причини вибору PostgreSQL для даної системи:

- надійність і транзакційність. PostgreSQL повністю відповідає вимогам ACID, що гарантує цілісність даних при паралельних операціях. Механізми контрольованого доступу, шифрування та налаштування ролей дозволяють безпечно працювати з конфіденційною інформацією агропідприємств;
- розвинуті засоби обробки даних. СУБД підтримує створення функцій, процедур, тригерів, що дає змогу переносити частину бізнес-логіки на рівень бази даних. Це прискорює виконання складних операцій і знижує навантаження на сервер додатків;
- висока продуктивність і масштабованість. PostgreSQL здатна ефективно опрацьовувати великі обсяги даних та підтримувати значну кількість одночасних користувачів. Завдяки індексації, паралельному виконанню запитів та оптимізованому плануванню вона забезпечує стабільно швидку реакцію системи;
- гнучкість у роботі з різними типами даних. Підтримка JSONB, масивів, географічних типів і розширень (зокрема PostGIS) робить PostgreSQL оптимальним рішенням для геоаналітики, що є критично важливим у аграрних системах — наприклад, для зберігання координат полів або аналізу зон продуктивності;

- інтеграція з хмарними платформами. PostgreSQL без проблем розгортається в AWS, Google Cloud чи Azure. У даному проєкті використовується розгортання в інфраструктурі AWS, що забезпечує резервне копіювання даних, високу доступність сервісу, автоматичне масштабування та додаткові інструменти захисту та шифрування

Використання PostgreSQL у мікросервісній архітектурі. Система побудована за мікросервісним підходом, який передбачає розділення великого застосунку на незалежні модулі. Це дає можливість окремо розвивати та розгортати компоненти, а також масштабувати їх залежно від навантаження.

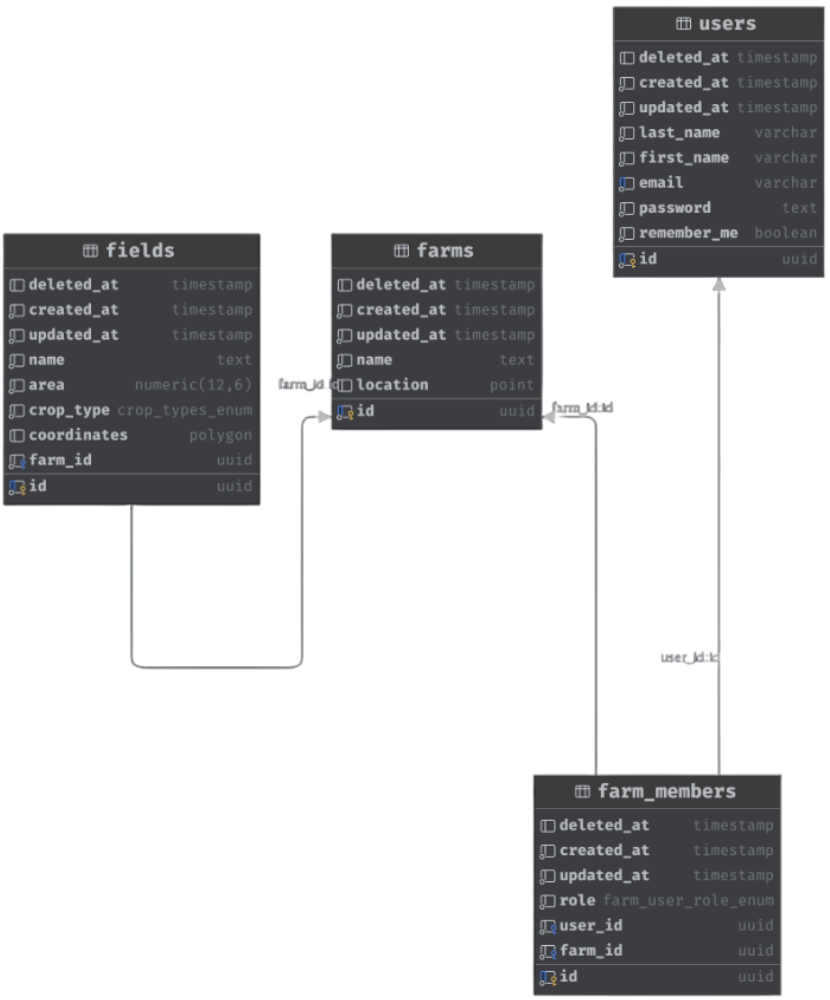

Рисунок 2.2 – Архітектура моделі агросистеми в БД

Переваги мікросервісного підходу для агросистеми:

- ізоляція логіки. Кожен сервіс відповідає за свій набір функцій (управління полями, фермою, користувачами, рекомендаціями тощо), що полегшує підтримку та оновлення системи;
- висока відмовостійкість. Помилка одного сервісу не спричиняє збоїв у всій системі, що є важливим фактором стабільності;
- можливість незалежного масштабування. Наприклад, модуль рекомендацій із великою кількістю запитів може бути масштабований окремо від модулів управління користувачами;
- гнучкість інтеграцій. Мікросервіси взаємодіють між собою та зовнішніми API (наприклад, LLM або погодними сервісами) через стандартизовані інтерфейси.

Особливості роботи PostgreSQL у мікросервісній структурі. Кожен сервіс отримує доступ до бази даних через ORM-інструменти (у проєкті — TypeORM). Це забезпечує:

- чіткий контроль за тим, які моделі доступні сервісу;
- можливість впроваджувати окремі схеми даних для різних модулів;
- ізоляцію бізнес-логіки;
- зменшення ризику конфліктів у межах загальної бази.

У PostgreSQL зберігається інформація про:

- ферми та членів команди;
- поля, їхні контури й типи культур;
- сенсорні дані й прогнози погоди;
- рекомендації, повідомлення та історію взаємодій.

Хмарна інфраструктура як основа роботи БД. Розміщення PostgreSQL у хмарі AWS забезпечує:

- автоматичне резервне копіювання та відновлення — дані захищені навіть у випадку серйозного збою;
- доступність з будь-якої точки світу — особливо важливо, якщо фермери або агрономи працюють у різних господарствах;
- горизонтальне та вертикальне масштабування — система адаптується до збільшення кількості користувачів чи обсягів аналітики;
- розширені механізми шифрування, IAM та firewall-політик.

Комбінація PostgreSQL та мікросервісної архітектури повністю відповідає вимогам до сучасних аграрних інформаційних систем. Такий підхід забезпечує:

- безпеку та цілісність даних,
- високу продуктивність під час аналітики та роботи з великими масивами інформації,
- простоту адаптації системи до нових модулів і функцій,
- можливість довгострокового масштабування.

Це робить PostgreSQL оптимальним вибором для системи, що має працювати з великим набором агроданих і підтримувати складні алгоритми штучного інтелекту.

2.4 Концептуальні підходи до побудови ШІ-агентів та інтеграція інструментів

Large Language Models (LLM) — це клас нейронних мереж, які є основою для розуміння та генерації природної людської мови[16]. У системі автоматизації агропідприємств LLM функціонує як інтелектуальний інтерфейс

та планувальник дій. Його основна функція полягає в природномовному розумінні намірів користувача, що дозволяє агрономам ставити складні запитання у вільній формі. LLM аналізує неструктурований запит (наприклад, "Який зараз рівень вологості ґрунту на полі №7 та чи потрібно поливати?") і розбиває його на послідовність кроків і необхідних для цього інструментів. LLM не виконує точних розрахунків, але відповідає за визначення наміру та маршрутизацію запиту до функціональних модулів. Після виконання всіх технічних дій, LLM також відповідає за фінальне форматування результату у вигляді змістовної та легкої для сприйняття діалогової відповіді.

Model Context Protocol (MCP) є фундаментальним компонентом архітектури, функціонуючи як стандартизований протокол-агент для зв'язку LLM із зовнішнім середовищем[17]. MCP розроблений для того, щоб дозволити LLM виходити за межі своєї внутрішньої бази знань, отримувати актуальні дані та виконувати дії в реальному застосунку. MCP реалізує механізм Tool Calling (Виклику Інструментів), приймаючи від LLM структурований JSON-об'єкт, який імітує виклик функції. У вашій системі MCP реалізований як набір Tool Servers на базі NestJS з використанням TypeScript. NestJS-сервіси, декоровані спеціальними анотаціями, перетворюються на функціональні інструменти, які можуть бути викликані LLM. При цьому TypeScript та Zod-схеми використовуються для наскрізної валідації параметрів виклику, гарантуючи, що аргументи, згенеровані LLM, завжди відповідають очікуваним типам у NestJS-функціях. Ця архітектура забезпечує чітке розмежування відповідальності: LLM думає і планує, а NestJS-мікросервіси точно виконують необхідні операції (наприклад, роботу з PostGIS або Apache Kafka).

Ключовим рішенням в архітектурі є забезпечення незалежності інтелектуального ядра від конкретного провайдера LLM, що досягається завдяки стандартизації взаємодії через MCP. Це дозволяє команді розробників гнучко обирати готові LLM-моделі з високою продуктивністю та вбудованою підтримкою механізму Function Calling (Виклик Функцій). Наявність вибору

між комерційними моделями (наприклад, від Google, OpenAI) або відкритими моделями (наприклад, на базі Llama) забезпечує відмовостійкість та можливість оптимізації витрат. Система може бути налаштована на використання різних моделей для різних завдань, наприклад, високоточну комерційну модель для критичних розрахунків і менш потужну модель для загальних діалогів. Така архітектурна гнучкість дозволяє системі швидко адаптуватися до технологічних змін та інтегрувати нові моделі, як тільки вони з'являються на ринку. Вся інтелектуальна цінність системи зосереджена у конфігурації MCP та якості функціональних інструментів NestJS, а не в самому ядрі LLM, що гарантує її довговічність та конкурентоспроможність.

2.5 Аналіз принципів використання методології IDEF4 для проєктування структури та функціонування системи

Методологія IDEF4 (Design Method for Object-Oriented Design) є специфічною методологією, розробленою для підтримки об'єктно-орієнтованого проектування та моделювання. Вона надає засоби для опису об'єктів, їхніх взаємодій, структури системи та поведінки, фокусуючись на концепціях об'єктно-орієнтованого підходу: інкапсуляції, успадкування, поліморфізму.

Основні принципи IDEF4, що використовуються для проектування структури та функціонування системи:

- об'єктна орієнтація: Центральним поняттям є об'єкт – сутність, що поєднує дані (атрибути) та поведінку (методи). Система розглядається як сукупність взаємодіючих об'єктів;
- абстракція: Можливість фокусуватися на суттєвих характеристиках об'єктів та процесів, ігноруючи другорядні деталі. IDEF4 дозволяє

створювати моделі на різних рівнях абстракції, від високорівневих до детальних;

- інкапсуляція**:** Принцип приховування внутрішньої реалізації об'єкта та надання доступу до його функціоналу лише через визначений інтерфейс. Це забезпечує модульність та спрощує модифікацію системи;
- успадкування**:** Можливість створення нових класів (нащадків), які успадковують атрибути та поведінку від існуючих класів (предків). Це сприяє повторному використанню коду та створенню ієрархій об'єктів;
- поліморфізм**:** Здатність об'єктів різних класів реагувати на один і той же виклик методу по-різному, залежно від їхнього типу. Це підвищує гнучкість та розширюваність системи;
- моделювання взаємодій**:** IDEF4 надає засоби для опису того, як об'єкти взаємодіють між собою для досягнення цілей системи, використовуючи повідомлення та виклики методів;
- ітеративний та інкрементальний підхід**:** Проектування в IDEF4 часто є ітеративним процесом, де модель поступово уточнюється та деталізується на кожній ітерації.

2.5.1 Аналіз можливостей використання основних засобів методології IDEF4

Основні засоби методології IDEF4 тісно пов'язані з концепціями UML (Unified Modeling Language), яка є стандартною мовою для візуалізації, специфікації, конструювання та документування артефактів програмних систем. Таким чином, IDEF4 може бути реалізована через використання різних діаграм UML.

- діаграми класів (Class Diagrams): Є центральним засобом IDEF4 для статичного моделювання. Вони дозволяють визначити класи системи, їхні атрибути, методи та взаємозв'язки (асоціації, агрегації, композиції, успадкування). Ці діаграми є основою для проектування структури даних та логіки системи;
- діаграми об'єктів (Object Diagrams): Показують конкретні екземпляри класів (об'єкти) в певний момент часу з їхніми поточними значеннями атрибутів. Використовуються для ілюстрації прикладів взаємодії та стану системи, допомагаючи перевірити коректність діаграм класів;
- діаграми компонентів (Component Diagrams): Дозволяють моделювати фізичну структуру системи, показуючи, як програмні компоненти (модулі, бібліотеки) взаємодіють між собою. Це важливо для організації архітектури системи.
- діаграми розгортання (Deployment Diagrams): Описують фізичне розміщення компонентів системи на апаратних вузлах (серверах, пристроях) та їхні мережеві зв'язки. Це ключовий засіб для планування інфраструктури та розгортання програмного забезпечення;
- діаграми послідовності (Sequence Diagrams) та Діаграми комунікації (Communication Diagrams): Використовуються для динамічного моделювання поведінки системи, показуючи порядок взаємодії об'єктів у часі (для діаграм послідовності) або їхні зв'язки та повідомлення (для діаграм комунікації);
- діаграми станів (State Machine Diagrams): Моделюють життєвий цикл об'єктів, показуючи їхні можливі стани та переходи між ними у відповідь на події;
- діаграми діяльності (Activity Diagrams): Відображають послідовність дій у бізнес-процесі або алгоритмі, схожі на блок-схеми, але з підтримкою паралельних потоків.

Використання цих засобів UML в рамках методології IDEF4 дозволяє створити всебічну та зрозумілу модель програмного забезпечення, що забезпечує якісне проектування та розробку.

2.6 Практична реалізація проєктування системи з використанням принципів об’єктно-орієнтованого моделювання

Практична реалізація проектування інтелектуальної системи управління агропідприємства з використанням принципів об'єктно-орієнтованого моделювання представлена за допомогою UML-діаграм.

2.6.1 UML Class Diagram: Опис основних класів системи

Діаграма класів є фундаментальним елементом об'єктно-орієнтованого проектування, оскільки вона визначає статичну структуру системи – які класи існують, які у них є атрибути та методи, і як вони взаємодіють між собою. На наданій діаграмі представлено ключові класи, що моделюють систему, призначену, судячи з назв класів, для агро-інформаційних систем або подібних до них, з функціоналом рекомендацій на основі даних сенсорів та погодних прогнозів. До основних класів системи входить:

Клас User який представляє користувача системи. Це може бути власник ферми, агроном, працівник господарства або інший учасник агропідприємства.

Атрибути:

- id: UUID – Унікальний ідентифікатор користувача.
- firstName: string – Ім’я користувача.
- lastName: string – Прізвище користувача.

- email: string – Унікальна електронна адреса користувача, яка використовується для логіну.
- password: string – Хешований пароль користувача (не повертається в запитах).
- rememberMe: boolean – Ознака того, чи бажає користувач зберігати сесію між входами в систему.

Методи (логічні, на рівні моделі):

- getFullName(): string – Повертає повне ім’я користувача.
- hasFarmRole(farmId: UUID, role: FarmUserRoleEnum): boolean – Перевіряє, чи має користувач певну роль на вказаній фермі.

Взаємозв’язки:

- Асоціація з FarmMember (1..*): Один користувач може мати кілька членств на різних фермах із різними ролями.
- Асоціація з Notification (1..*): Один користувач може отримувати багато сповіщень системи.

Клас Farm, представляє агропідприємство або окрему ферму, з якою пов’язані поля, члени команди та аналітичні дані.

Атрибути:

- id: UUID – Унікальний ідентифікатор ферми.
- name: string – Назва ферми (господарства).
- location: CoordinatesInterface | null – Географічна точка розташування ферми (широта/довгота), може бути відсутньою.

Методи:

- getFieldsCount(): number – Повертає кількість полів, пов’язаних із фермою.
- getMembersByRole(role: FarmUserRoleEnum): FarmMember[] – Повертає список членів ферми з конкретною роллю.

Взаємозв’язки:

- Асоціація з FarmMember (1..*): Одна ферма має багато членів (власник, менеджери, агрономи, працівники).
- Асоціація з Field (1..*): Одна ферма має нуль або більше полів.

Клас Field, представляє окреме сільськогосподарське поле, для якого збираються сенсорні дані, прогнозується врожайність і формуються рекомендації.

Атрибути:

- id: UUID – Унікальний ідентифікатор поля.
- name: string – Назва поля (наприклад, "Поле №3, пшениця").
- area: number – Площа поля в гектарах (зберігається в форматі numeric з потрібною точністю).
- cropType: CropTypesEnum – Тип культури, яка вирощується на полі (наприклад, WHEAT, CORN, SUNFLOWER).
- coordinates: CoordinatesInterface[] | null – Множина координат контуру поля (полігон), може бути відсутньою, якщо поле ще не оцифроване.

Методи:

- getPolygonArea(): number – Повертає розраховану площу поля за геометрією (для валідації введених даних).

- isCropSelected(): boolean – Перевіряє, чи обрана культура для цього поля.

Взаємозв’язки:

- Асоціація з Farm (1): Кожне поле належить до однієї ферми.
- Асоціація з SensorData (0..*): Для поля накопичуються історичні сенсорні дані (вологість ґрунту, температура тощо).
- Асоціація з Recommendation (0..*): Для поля можуть бути створені численні рекомендації в різні дати.

Клас Crop, представляє сільськогосподарську культуру, що вирощується на полях.

Атрибути:

- id: UUID – Унікальний ідентифікатор культури.
- name: string – Назва культури (наприклад, "Пшениця озима").
- growthStages: string[] – Перелік основних фаз росту (сівба, кущення, вихід у трубку, колосіння тощо).
- optimalTemperature: number – Оптимальна температура для росту культури.
- soilRequirements: string – Опис вимог до ґрунту (структура, pH, вологість).

Методи:

- getRecommendedFertilizationScheme(): string – Повертає загальні рекомендації щодо удобрення для цієї культури.

Взаємозв’язки:

- Асоціація з Field (1..*): Одна культура може вирощуватися на багатьох полях.

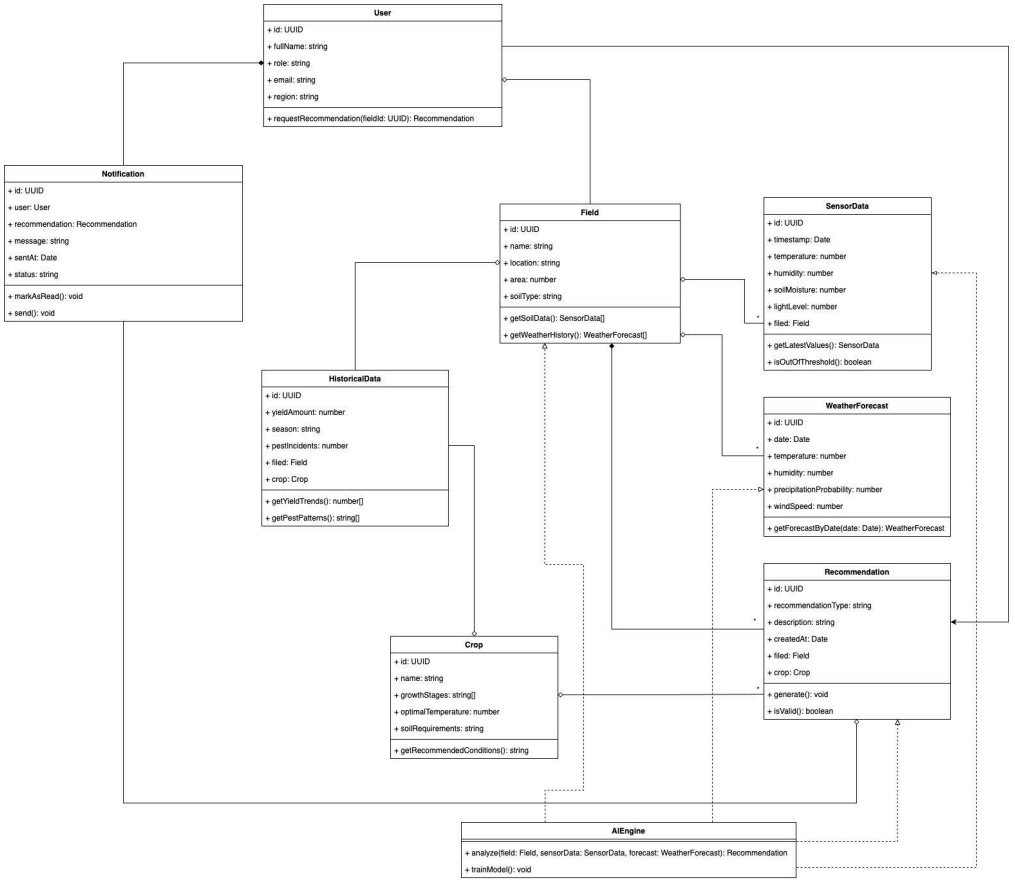

Рисунок 2.2 – UML Діаграма класів

Ця UML-діаграма відображає ключові компоненти автоматизованої системи управління процесами агропідприємства та їхню взаємодію, що дозволяє забезпечити ефективну реалізацію основних функцій.

2.6.2 UML Deployment Diagram: Опис процесу розгортання системи

UML Deployment Diagram є одним із ключових засобів моделювання у стандартизованій мові UML (Unified Modeling Language) та застосовується для відображення фізичної структури програмної системи на рівні апаратних або

хмарних обчислювальних ресурсів. Основним її призначенням є наочна демонстрація того, як саме програмні компоненти, сервіси, модулі та інші артефакти розміщуються на конкретних вузлах інфраструктури. На відміну від діаграм класів, які описують логічну модель системи, що зосереджені на динаміці взаємодії програмних сутностей, діаграма розгортання формує уявлення про реальну топологію системи та архітектурне середовище її функціонування. Вона дає змогу відобразити фізичні сервери, віртуальні машини, контейнерні середовища, хмарні сервіси, мобільні пристрої та інші елементи, на яких розгортаються програмні модулі. Такий тип діаграм дозволяє розробникам, архітекторам і DevOps-фахівцям чітко розуміти, де і як буде виконуватися кожен компонент системи, а також які мережеві з'єднання та протоколи забезпечують взаємодію між ними. Застосування Deployment Diagram є особливо важливим у випадках побудови складних багатокомпонентних систем, де програмні модулі можуть бути розподілені між різними серверами або хмарними платформами. Вона також відіграє значну роль під час проєктування систем з високими вимогами до масштабованості, відмовостійкості, інформаційної безпеки та продуктивності.

Діаграма розгортання дає змогу описати інфраструктуру як з точки зору апаратної конфігурації, так і з позиції програмних залежностей між компонентами. В ній можна визначити, які модулі працюють локально, які виконуються у контейнерах, а які — через віддалені сервіси або API хмарних провайдерів. Також вона допомагає формалізувати мережеву взаємодію між сервісами, що забезпечує можливість раннього виявлення потенційних вузьких місць або конфігураційних помилок. Архітектура системи стає прозорою та зрозумілою для всіх учасників розробки, що суттєво спрощує інтеграцію нових модулів, модернізацію існуючих компонентів і планування подальшого масштабування. Deployment Diagram також слугує важливою частиною технічної документації, яка використовується під час розгортання, супроводу, аудиту чи оцінки архітектурної надійності системи.

Логічна архітектура (Logical Deployment Tiers) системи була розроблена на основі багаторівневої (N-Tier) моделі, що забезпечує чітке функціональне розмежування, модульність та можливість незалежного масштабування кожного рівня.

Клієнтський Рівень (Client Tier): Цей рівень є точкою взаємодії з кінцевим користувачем. Його основним компонентом є Browser Application, який відповідає за візуалізацію даних, інтерактивні карти, діаграми та користувацький інтерфейс. Клієнтський рівень взаємодіє з серверною частиною системи виключно через стандартизований протокол REST API.

Основний Серверний Рівень (Main Server Tier): Цей рівень є серцем бізнес-логіки системи. Він містить два ключові функціональні шари: Business Logic Layer, що реалізує основні правила агрономічного обліку та управління, та Data Access Layer, який відповідає за безпечну та ефективну взаємодію з базами даних. Main Server Tier виконує маршрутизацію запитів, обробку аутентифікації та авторизації. Для взаємодії з іншими мікросервісами (ШІ та сповіщень) він використовує високопродуктивний протокол GRPC, а також підтримує пряме з'єднання з Data Tier.

Серверний Рівень ШІ (AI Server Tier): Цей рівень спеціалізовано на виконанні ресурсомістких обчислень та прогнозного аналізу. Його основним компонентом є Model Layer, який містить навчені моделі штучного інтелекту, логіку обробки супутникових знімків та складні агрономічні алгоритми. AI Server Tier спілкується з Main Server Tier через протокол GRPC, що мінімізує затримки, критичні для швидкого прийняття рішень.

Серверний Рівень Сповіщень (Notification Server Tier): Цей мікросервіс відповідає за формування та відправку всіх системних сповіщень (попередження про дефіцит живлення, нагадування про планові роботи). Його Business Logic Layer формує вміст сповіщень на основі подій, ініційованих Main Server Tier або AI Server Tier, і взаємодіє з ними також через GRPC.

Рівень даних (Data Tier): Цей рівень забезпечує надійне зберігання інформації. Основним компонентом є SQL Server stored procedures на базі PostgreSQL, що зберігає всі конфігурації, геопросторові дані (PostGIS), записи моніторингу та історичні дані обробки полів.

Фізичне розгортання системи націлене на консолідацію всіх критичних серверних компонентів на одному потужному хост-вузлі, що працює під керуванням стабільної операційної системи.

Хост-Вузол та операційне Середовище. Уся серверна інфраструктура розгорнута на високопродуктивному хост-вузлі (позначений як $<<device>>$), що характеризується потужним процесором Intel Core i7, значним обсягом оперативної пам'яті 64 GB RAM та дисковим простором 1 TB. Вибір таких високих характеристик обумовлений потребою в обробці великих масивів геопросторових даних та ресурсомістких обчисленнях, пов'язаних із виконанням AI-моделей. Хост-вузол працює під керуванням операційної системи Ubuntu 22.04 LTS, що є стандартом для промислового розгортання завдяки її надійності та довготривалій підтримці.

Розміщені виконавчі компоненти. На цьому єдиному фізичному вузлі розміщені та виконуються усі ключові компоненти, інтегруючись за допомогою контейнеризації:

- основний AI растосунок ($<<Server>>$ $<<AI application>>$): Це реалізація Main Server Tier, яка виконується як бекенд-сервіс на базі Node.js;
- служба ШІ для Навчання ($<<Server>>$ $<<AI service>>$): Це компонент, що виконує складні алгоритми машинного навчання (AI Server Tier), що є типовим для задач data science;
- сервіс сповіщень ($<<Server>>$ $<<Notification Service>>$): Мікросервіс, відповідальний за комунікацію з користувачем, також реалізований на Node.js;

- сервіс бази даних ($<<DB Service>>$): Це інстанс PostgreSQL, який керує безпосередньо базою даних, забезпечуючи функціональність Data Tier.

Технологічні вимоги та протоколи зв'язку. Для забезпечення належної функціональності та продуктивності встановлені чіткі вимоги до програмного стеку та протоколів зв'язку.

Використовується набір середовищ виконання, включаючи Node.js для розробки бекенду (NestJS) та Python для модулів машинного навчання. Як клієнтська платформа підтримуються всі сучасні браузери (Chrome, Safari, Opera, Mozila).

Зв'язок у системі розділений за призначенням:

- REST API: Використовується як зовнішній протокол для взаємодії між клієнтським рівнем (браузером) та Main Server Tier. Це забезпечує гнучкість та широку сумісність;
- GRPC: Використовується як внутрішній, міжсервісний протокол. Його вибір обґрунтований необхідністю досягнення високої продуктивності та низької затримки для синхронного обміну даними між Main Server Tier та AI Server/Notification Server Tiers. Це критично для швидкого повернення рекомендацій ШІ;

Вимоги щодо Hardware/Software для забезпечення процесу розгортання:

- для розгортання основних сервісів (Main Server Tier, AI Server Tier, Notification Server Tier, Data Tier) необхідна хост-машина з мінімальними характеристиками: процесор Intel Core i7, 64 ГБ оперативної пам'яті та 1 ТБ дискового простору. Ці характеристики вказують на досить потужний сервер, здатний обробляти значні обсяги даних та виконувати ресурсомісткі обчислення (особливо для AI-моделей);

- для мобільних застосунків (Mobile Application) або браузерних (Browser Application) потрібні відповідні пристрої (смартфони, планшети, настільні ПК) з підтримкою мережевого з'єднання;
- операційна система: Ubuntu 22.04 LTS (Long Term Support) на сервері. Це стабільна та поширена Linux-система, що підходить для розгортання серверних застосунків;
- середовища виконання:
    a) Для бекенду: Node.js (індексовано за файлами .js для AI application та Notification Service).
    b) Для AI-сервісів: Python (індексовано за файлом .py для AI service).
- база даних: PostgreSQL.
- протоколи зв'язку:
    a) REST API: Для взаємодії між клієнтським та основним сервером.
    b) GRPC: Для високопродуктивної міжсервісної комунікації між основними сервісами та AI/Notification сервісами.
- браузерна платформа: Chrome, Safari, Opera, Mozila

Надана діаграма розгортання ілюструє багаторівневу (N-tier) архітектуру.

- browser application або mobile application є тонким клієнтом, що взаємодіє з сервером через мережу;
- серверний рівень: Цей рівень розділений на кілька логічних "тирів" (Main Server, AI Server, Notification Server), що свідчить про мікросервісну архітектуру або, принаймні, про виділення окремих служб. Ці сервіси взаємодіють між собою за допомогою GRPC, що є оптимальним для швидкого та ефективного зв'язку в межах внутрішньої мережі;
- рівень даних: Data Tier з PostgreSQL та збереженими процедурами, до якого звертається Data Access Layer основного сервера;

Мережеві зв'язки:

- зовнішня мережа (Інтернет): Mobile Application взаємодіє з Main Server Tier через REST API, що передбачає використання HTTP/HTTPS протоколів через публічний Інтернет (для доступу з мобільних пристроїв);
- внутрішня мережа (Локальна мережа/VPN): Взаємодія між Main Server Tier, AI Server Tier, Notification Server Tier та Data Tier відбувається за допомогою GRPC та, ймовірно, з використанням приватних IP-адрес у межах локальної мережі або віртуальної приватної мережі (VPN) для підвищення безпеки та продуктивності;
- єдиний хост: Всі серверні компоненти та база даних розміщені на одній "Host machine", що може бути початковим етапом розгортання (наприклад, для тестування або невеликих навантажень). Для масштабування та високої доступності ці компоненти, ймовірно, будуть розподілені по різних фізичних або віртуальних машинах;

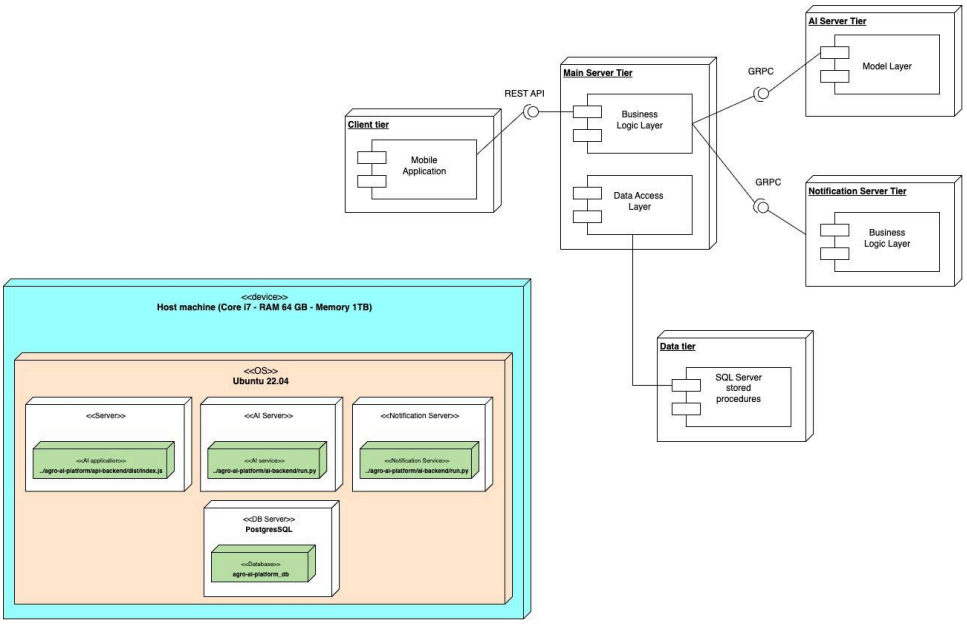

Рисунок 2.3 – UML Deployment Diagram

2.7 Вибір мови програмування та допоміжних засобів

Вибір технологічного стеку для реалізації інтелектуальної системи базується на принципах уніфікації, продуктивності та високої масштабованості, необхідних для підтримки мікросервісної архітектури та обробки великих обсягів агроданих.

Основою системи обрана мова TypeScript у поєднанні з JavaScript. Це дозволяє уніфікувати мову розробки для фронтенду та бекенду, підвищуючи ефективність команди та спрощуючи обмін знаннями[18]. TypeScript також забезпечує статичну типізацію, що значно підвищує надійність коду, необхідну для корпоративних систем, оскільки більшість помилок виявляються ще на етапі компіляції. Це мінімізує кількість багів, що потрапляють у продакшен, і забезпечує більш високу якість інтелектуальних модулів. Для забезпечення валідації даних як на фронтенді, так і на бекенді, використовується бібліотека Zod. Zod дозволяє створювати надійні схеми валідації, які гарантують, що дані, що надходять у мікросервіси та з них, відповідають очікуваній структурі. Завдяки Zod досягається повна типова безпека (End-to-End Type Safety), оскільки єдині схеми валідації використовуються і для перевірки форм на Next.js, і для валідації API-запитів на NestJS. Використання єдиної мови полегшує застосування одного підходу до автентифікації та валідації даних на обох кінцях системи. Крім того, Node.js є неблокуючим середовищем, що дозволяє досягати високої пропускної здатності при обробці великої кількості одночасних запитів від клієнтів і сенсорів. Це критично важливо для мікросервісів, які обробляють великий трафік, пов'язаний з моніторингом. Для створення бекенд-мікросервісів обрано фреймворк NestJS, що працює на платформі Node.js. NestJS ідеально підходить для побудови структурованих, легких та високопродуктивних мікросервісів, оскільки він використовує архітектурні патерни, схожі на Angular, та має вбудовану підтримку асинхронних комунікацій через *Apache Kafka*. Його модульна структура

забезпечує чистоту архітектури та легкість розширення функціоналу. Фронтенд-рівень (Presentation Tier) реалізується за допомогою Next.js, який є потужним фреймворком на базі React. Next.js підтримує рендеринг на стороні сервера (SSR) та дозволяє створювати високоефективні Single Page Applications (SPA). Використання SSR також покращує доступність системи та її первинне завантаження, що є важливим для аграріїв, які можуть мати обмежений доступ до швидкісного інтернету на місцях.

Для забезпечення консистентного та професійного вигляду користувацького інтерфейсу використовується бібліотека компонентів Shadcn UI. Ці компоненти побудовані на базі Radix UI та стилізовані за допомогою Tailwind CSS. Shadcn UI забезпечує високу якість, доступність (Accessibility) та консистентний дизайн, що особливо важливо для складних дашбордів із багатьма графіками та картами. Використання готових, але гнучких компонентів значно прискорює етап розробки інтерфейсу користувача, дозволяючи команді сфокусуватися на бізнес-логіці. Tailwind CSS, будучи утилітарним CSS-фреймворком, значно прискорює розробку та дозволяє створювати гнучкі й адаптивні стилі без необхідності залишати HTML/JSX код. Це усуває проблему "блокування" (CSS bloating) і дозволяє генерувати дуже легкі фінальні стилі. Модульність Tailwind дозволяє легко адаптувати інтерфейс під різні розміри екранів, від мобільних пристроїв до великих моніторів у диспетчерських. Це особливо важливо для відображення детальних геопросторових даних, де потрібна висока точність. Крім того, єдина система стилів полегшує підтримку та масштабування UI/UX. Використання цих інструментів забезпечує сучасний зовнішній вигляд, який викликає довіру у професійних користувачів, таких як агрономи та керівники підприємств. Загалом, цей вибір значно скорочує цикл від ідеї до впровадження нового елемента інтерфейсу. Стилі можуть бути легко змінені відповідно до фірмового стилю агропідприємства. Таким чином, досягається оптимальний баланс між швидкістю розробки та якістю кінцевого продукту.

Для ефективної роботи з даними використовується гібридний підхід. В якості основної реляційної бази даних обрано PostgreSQL з обов'язковим розширенням PostGIS. PostgreSQL є надійним, відкритим і високопродуктивним реляційним двигуном, який здатний витримувати великі транзакційні навантаження. Розширення PostGIS є галузевим стандартом для зберігання, індексації та виконання складних просторових запитів (наприклад, визначення перетину меж полів), що критично важливо для роботи з агро- та супутниковими даними. Це дозволяє здійснювати аналіз безпосередньо на рівні бази даних, зменшуючи навантаження на мікросервіси. Для забезпечення високо надійної та високопродуктивної асинхронної комунікації між мікросервісами використовується Apache Kafka. Kafka є незамінним брокером повідомлень для обробки великих потоків даних у реальному часі (телеметрія та події) від численних сенсорів. Його архітектура дозволяє сервісам "від'єднатися" від процесу обробки, підвищуючи загальну відмовостійкість системи. Це гарантує, що жодні дані від сенсорів не будуть втрачені, навіть якщо аналітичний сервіс тимчасово недоступний. Нарешті, Redis використовується як швидке сховище "ключ-значення" для кешування часто запитуваних даних, наприклад, токенів автентифікації, а також для зберігання проміжних результатів складних обчислень. Redis, завдяки роботі в оперативній пам'яті, значно підвищує загальну швидкість відгуку системи, знімаючи навантаження з основної бази даних PostgreSQL. Його також можна використовувати як брокер для простих черг завдань. Для зберігання великих неструктурованих об'єктів, таких як сирі супутникові знімки (GeoTIFF) та архіви телеметрії, обрано Amazon S3 (або S3-сумісне сховище). Amazon S3 забезпечує надзвичайну надійність, доступність та економічність для зберігання мільйонів об'єктів, що є необхідним для історії моніторингу полів. Завдяки цій комбінації сховищ, система здатна ефективно обробляти як транзакційні, так і аналітичні, а також потокові дані та великі бінарні файли.

Для забезпечення гнучкого розгортання та керування мікросервісами обрано сучасні інструменти DevOps. Docker використовується для контейнеризації кожного мікросервісу. Контейнеризація забезпечує повну ізоляцію середовища, гарантуючи, що код працюватиме однаково на машині розробника, у тестовому середовищі та в продакшені. Це значно спрощує процеси CI/CD та усуває проблеми несумісності залежностей. Крім того, Docker допомагає стандартизувати всі мікросервіси, незалежно від того, чи використовують вони NestJS, чи іншу технологію, роблячи їх легкими та портативними. Для автоматизації розгортання, масштабування та управління контейнеризованими застосунками обрано Kubernetes. Kubernetes (K8s) є промисловим стандартом оркестратором контейнерів, який автоматично керує балансуванням навантаження, забезпечує високу доступність та самовідновлення (self-healing). Якщо один з мікросервісів виходить з ладу, Kubernetes автоматично перезапустить його або перенаправить трафік на здоровий екземпляр. Це критично важливо для цілодобового моніторингу агропідприємства. Завдяки Kubernetes, система може горизонтально масштабуватися в режимі реального часу, додаючи нові екземпляри мікросервісів у відповідь на пікове навантаження (наприклад, під час масового завантаження нових супутникових знімків). Таким чином, Docker та Kubernetes забезпечують надійний, масштабований та повністю автоматизований фундамент для роботи всієї інтелектуальної системи.

2.7.1 TypeScript

TypeScript – це статично типізована надмножина JavaScript, яка компілюється у чистий JavaScript, пропонуючи розширені можливості для розробки великих застосунків[19]. Використання цієї мови дозволяє уніфікувати кодову базу для фронтенду та бекенду, підвищуючи ефективність

команди та спрощуючи обмін знаннями. Завдяки статичній типізації, більшість потенційних помилок виявляються ще на етапі компіляції, що мінімізує кількість багів у продакшені та значно підвищує надійність коду. Строга типізація є критично важливою для корпоративних систем, де необхідно гарантувати цілісність даних. Використання спільних інтерфейсів та схем валідації забезпечує наскрізну типову безпеку (End-to-End Type Safety) у всій системі.

2.7.2 NestJS

NestJS – це прогресивний фреймворк для побудови ефективних, надійних та масштабованих серверних застосунків на Node.js[20]. Він використовує архітектурні патерни, схожі на Angular (Модулі, Контролери, Сервіси), забезпечуючи чистоту архітектури, легкість розширення та підтримку. Фреймворк ідеально підходить для реалізації бекенд-мікросервісів, оскільки його вбудований механізм Dependency Injection спрощує тестування та керування залежностями. Працюючи на асинхронній платформі Node.js, NestJS досягає високої пропускної здатності, необхідної для обробки великої кількості одночасних запитів від клієнтів і сенсорів. Його модульність сприяє ефективному поділу відповідальності між мікросервісами, підвищуючи їхню ізоляцію та незалежне розгортання.

2.7.3 Next.js

Next.js – це потужний фреймворк на базі React, який розширює можливості клієнтської розробки, пропонуючи функціонал рендерингу на стороні сервера (SSR)[21]. Використання Next.js дозволяє створювати високоефективні Single Page Applications (SPA) з оптимізованим часом

завантаження, що є важливим для відображення складних графіків, інтерактивних карт та дашбордів[22]. SSR покращує доступність системи та забезпечує швидке початкове завантаження інтерфейсу. Фреймворк надає оптимізовані механізми маршрутизації та бандлінгу, що сприяє високій швидкості роботи та плавності переходу між сторінками. Next.js забезпечує оптимальний баланс між швидкістю розробки та продуктивністю, адаптуючи інтерфейс до різних пристроїв агрономів і керівників.

2.7.4 Apache Kafka

Apache Kafka – це розподілений потоковий брокер повідомлень, призначений для побудови високопродуктивних конвеєрів даних і застосунків реального часу[23]. Його використання забезпечує надійну та високопродуктивну асинхронну комунікацію між мікросервісами та є ідеальним для обробки великих потоків даних (телеметрія та події від сенсорів). Лог-орієнтована структура Kafka гарантує, що події не будуть втрачені, навіть якщо сервіс-споживач тимчасово недоступний, підвищуючи загальну відмовостійкість системи. Це дозволяє реалізовувати складні схеми обробки, де дані з одного джерела можуть бути послідовно оброблені кількома незалежними мікросервісами. Kafka є критично важливим для забезпечення безперервності та високої пропускної здатності системи моніторингу агроданих.

2.7.5 Amazon S3

Amazon S3 (Simple Storage Service) – це високомасштабоване хмарне сховище об'єктів, що забезпечує галузеві стандарти надійності, доступності та продуктивності[24]. Цей сервіс є основним сховищем для великих

неструктурованих агроданих, таких як сирі супутникові знімки (GeoTIFF), аерофотознімки з дронів та архіви телеметрії сенсорів. S3 гарантує надзвичайну довговічність даних завдяки автоматичному резервному копіюванню в різних зонах доступності. Використання S3 дозволяє мікросервісам звертатися до великих файлів безпосередньо за унікальним URL, що знижує навантаження на основні сервери застосунків. Його інтеграція з AWS-сервісами дозволяє ефективно обробляти ці великі файли за допомогою аналітичних інструментів.

2.7.6 Docker

Docker – це платформа для розробки, доставки та запуску застосунків за допомогою контейнерів, яка дозволяє інкапсулювати код, середовище виконання та всі залежності. Використання Docker для контейнеризації кожного мікросервісу та клієнтського застосунку гарантує, що код працюватиме однаково на всіх етапах: від машини розробника до продуктивного середовища[25].

Це значно спрощує процеси CI/CD та усуває проблеми несумісності залежностей, забезпечуючи високу портативність усіх компонентів системи. Docker дозволяє точно контролювати ресурси, виділені кожному мікросервісу, що є важливим для оптимізації витрат на інфраструктуру. Кожен контейнер є легкою та автономною одиницею, що спрощує розгортання та керування в умовах мікросервісної архітектури.

2.7.7 Kubernetes

Kubernetes (K8s) – це відкрита система для автоматизації розгортання, масштабування та управління контейнеризованими застосунками. Вона функціонує як промисловий стандарт оркестратор контейнерів, який

автоматично керує балансуванням навантаження, забезпечує високу доступність та самовідновлення (self-healing) системи[26]. Kubernetes дозволяє системі горизонтально масштабуватися в режимі реального часу у відповідь на пікове навантаження, наприклад, під час інтенсивного аналізу даних. Ця система забезпечує стійкість до збоїв на рівні інфраструктури, автоматично переносячи контейнери на справні вузли. Для розгортання створюються спеціалізовані конфігураційні файли (Kubernetes Deployment YAMLs), що повністю автоматизує процес інсталяції та управління всією інфраструктурою. Kubernetes є критично важливим для забезпечення безперебійної роботи системи моніторингу 24/7.

2.8 Вибір та налаштування бази даних у хмарному середовищі

Для зберігання та обробки даних у розроблюваній системі було обрано реляційну СУБД PostgreSQL, яка зарекомендувала себе як стабільна, продуктивна та функціонально розширювана платформа з відкритим кодом. PostgreSQL підтримує стандарт SQL, роботу зі складними структурами даних, транзакційність та має широкий набір інструментів для оптимізації запитів, що робить її придатною для побудови сучасних вебсистем і аналітичних рішень.

У хмарній інфраструктурі найбільш доцільним варіантом розгортання PostgreSQL є використання керованого сервісу Amazon RDS, який надає готове середовище для швидкого створення, адміністрування та масштабування баз даних. Завдяки RDS відпадає необхідність власноруч виконувати рутинні операції, такі як налаштування серверів, встановлення оновлень, резервне копіювання або моніторинг навантаження.

Однією з ключових переваг RDS є підтримка найпопулярніших СУБД, серед яких і PostgreSQL, а також можливість розгортання бази відразу у декількох зонах доступності (Multi-AZ). Це забезпечує високу відмовостійкість:

у разі збою система автоматично переключається на резервний екземпляр без втрати даних і з мінімальною затримкою.

Amazon RDS також пропонує засоби підвищеної безпеки:

- шифрування даних як у стані спокою (AES-256), так і під час передачі (SSL);
- ізоляцію інфраструктури через використання віртуальних приватних мереж (VPC);
- гнучке керування правами доступу.

Крім того, сервіс забезпечує оптимізацію продуктивності завдяки автоматичному аналізу запитів, можливості визначати вузькі місця, підтримці високопродуктивного сховища Provisioned IOPS, що є критично важливим для систем із великим навантаженням.

У контексті побудови інтелектуальної агросистеми Amazon RDS з PostgreSQL є ефективним рішенням для роботи зі структурованими даними, серед яких:

- профілі користувачів (фермери, члени господарства, агрономи);
- інформація про ферми, поля, їхні координати та агротехнічні характеристики;
- дані сенсорів і аналітичні результати обробки;
- збережені рекомендації, історія запитів та інші елементи системної логіки.

Завдяки використанню хмарної платформи AWS база даних отримує масштабованість, високу доступність, захищеність та можливість оперативного розширення ресурсів у разі збільшення кількості користувачів або обсягів даних.

2.9 Безпека та аутентифікація в мікросервісній архітектурі

Забезпечення безпеки є фундаментальною вимогою для корпоративних систем, які оперують конфіденційними даними[27]. У розподіленій мікросервісній архітектурі, де традиційні механізми сесійного управління є неефективними та непрактичними, критично важливим є впровадження стандарту JSON Web Token (JWT) для реалізації безсесійної (Stateless) аутентифікації та авторизації. Цей підхід забезпечує масштабованість, швидкість та криптографічну надійність.

2.9.1 JSON Web Token (JWT): Структура та роль у безсесійній авторизації

JWT є відкритим, компактним та URL-безпечним стандартом (RFC 7519), який визначає спосіб безпечної передачі інформації між сторонами як об'єкт JSON. Його ключова перевага полягає у здатності нести в собі всю необхідну інформацію про користувача, що дозволяє мікросервісам самостійно перевіряти справжність токена без необхідності звернення до централізованого сховища сесій. Анатомія JWT: Токен складається з трьох частин, розділених крапками[28].

Заголовок (Header): Ця частина містить метадані токена. Вона обов'язково включає тип токена (typ: JWT) та алгоритм, який використовувався для його підпису (наприклад, alg: RS256 або HS256). Цей заголовок кодується за допомогою Base64Url.

Корисне Навантаження (Payload): Центральна частина токена, що містить Клейми (Claims) — набір тверджень про користувача та його дозволи. Клейми поділяються на:

- зареєстровані клейми (Registered Claims): Стандартні поля, такі як iss (видавець), exp (час закінчення дії токена) та sub (суб'єкт, тобто ID

користувача). Використання exp є критичним для автоматичної ануляції токена;

- публічні клейми (Public Claims): Довільні імена, які повинні бути зареєстровані або мати захист від колізій;
- приватні клейми (Private Claims): Кастомні поля, які використовуються для бізнес-логіки системи. У цій архітектурі вони включають ідентифікатор користувача, його ролі (для RBAC) та ідентифікатор господарства (farm_id), забезпечуючи необхідний контекст для авторизації на рівні ресурсів. Це навантаження також кодується за допомогою Base64Url;

Підпис (Signature): Це критичний компонент, який забезпечує цілісність та справжність токена. Він генерується шляхом криптографічного хешування закодованих Заголовка та Корисного Навантаження, а також секретного ключа Identity Provider (IdP). Підпис гарантує, що дані токена не були змінені після його видачі. Кожен NestJS-мікросервіс, отримавши JWT, може перевірити цей підпис за допомогою публічного ключа IdP, підтверджуючи достовірність усіх клеймів без необхідності взаємодії з IdP на кожен запит.

2.9.2 Двокомпонентна модель токенів: Access та Refresh Token

Для забезпечення високого рівня безпеки та керованості життєвим циклом токенів у системі використовується двокомпонентна модель.

Access Token (Токен Доступу) є основним інструментом для авторизації в системі. Його характеристики та функції:

- використовується для кожного запиту до API-ендпоінтів мікросервісів. Він включається у заголовок HTTP-запиту (Authorization: Bearer <token>);

- має короткий час життя (зазвичай від 5 до 30 хвилин). Цей короткий термін дії є свідомим архітектурним рішенням. У разі компрометації токена (наприклад, перехоплення), час, протягом якого зловмисник може використовувати його, різко обмежується;
- несе в собі всі необхідні клейми (ролі, ID користувача, ID господарства) для миттєвого прийняття рішення про доступ. Мікросервіси використовують ці клейми для перевірки RBAC та обмеження доступу до ресурсів (PoLP);
- мікросервіс просто валідує підпис та термін дії токена, не звертаючись до централізованої сесії. Це забезпечує високу продуктивність та лінійну масштабованість;

Refresh Token (Токен Оновлення) призначений виключно для безпечного поновлення Access Token, термін дії якого закінчився. Його характеристики:

- має значно довший час життя (години, дні або тижні);
- використовується лише для запиту до спеціалізованого, високозахищеного ендпоінту IdP з метою отримання нового Access Token. Він не використовується для доступу до бізнес-логіки API;
- є надзвичайно цінним, оскільки дозволяє створити безліч нових Access Token. Тому він зберігається максимально безпечно — у HTTP-Only cookies або в захищеному сховищі клієнтської сторони. Використання HTTP-Only cookies запобігає доступу до токена через JavaScript, унеможливлюючи атаки типу Cross-Site Scripting (XSS);
- на відміну від Access Token, Refresh Token зазвичай зберігається у базі даних IdP. Це дозволяє адміністраторам або системою безпеки негайно відкликати скомпрометований Refresh Token, зупиняючи можливість подальшого поновлення доступу;

2.9.3 Життєвий цикл токенів та наскрізна авторизація

Двокомпонентна модель забезпечує безпечний та безперервний досвід користувача:

- після успішного входу IdP видає обидва токени. Access Token передається клієнту, а Refresh Token — зберігається безпечно;
- кожен API-запит супроводжується Access Token. Мікросервіс виконує швидку перевірку підпису та клеймів (валідація);
- коли Access Token стає недійсним (закінчується термін exp), клієнт автоматично використовує Refresh Token для запиту нового Access Token у IdP. Якщо Refresh Token не був відкликаний, IdP видає нову пару токенів, і сесія користувача продовжується без повторного входу;
- у разі виходу користувача з системи або виявлення підозрілої активності, Refresh Token негайно відкликається на стороні IdP, що гарантує припинення можливості поновлення доступу.

Такий механізм є стандартом для сучасних розподілених архітектур, забезпечуючи, що кожен мікросервіс має незалежні, але криптографічно підтверджені дані для прийняття рішень про авторизацію, підвищуючи як безпеку, так і масштабованість системи[29].

2.10 Висновки до розділу

Після детального аналізу сучасних технологій, необхідних для розробки інтелектуальної системи автоматизації та моніторингу агропідприємств, було сформовано оптимальний стек технологій, компоненти якого гармонійно доповнюють один одного та забезпечують надійну основу для реалізації складного функціоналу, що включає збір даних, інтелектуальний аналіз та

віддалений моніторинг. Для розробки серверної частини та основного ядра інтелектуального аналізу даних було обрано мову програмування, що забезпечує високу продуктивність та ефективність. Це є критичним для обробки великих масивів геопросторових даних та алгоритмів машинного навчання в режимі, близькому до реального часу. Її потужна екосистема бібліотек для Штучного Інтелекту дозволяє ефективно реалізувати модифікований алгоритм інтелектуального аналізу даних. Для створення гнучкої та масштабованої серверної архітектури обрано фреймворк, який забезпечує швидку розробку API-інтерфейсів для мобільного та веб-додатків, а також ефективне керування логікою системи. Стосовно зберігання та керування даними, була обрана система управління базами даних, яка є потужною, надійною та високопродуктивною. Ця СУБД ідеально підходить для зберігання та обробки великих обсягів часових рядів, геопросторових даних, зокрема з розширенням PostGIS, та результатів інтелектуального аналізу, що генеруються системою моніторингу полів та техніки. Для розробки клієнтської частини, а саме веб-додатка та мобільного застосунку агронома, прийнято рішення використовувати JavaScript, що дозволяє значно пришвидшити процес розробки завдяки широкій підтримці та численним бібліотекам. Для відображення інтерактивної мапи полів та треків агротехнічних робіт було обрано картографічну бібліотеку, яка є потужним засобом для інтеграції географічних даних у веб-додаток. В якості високоефективного та надійного веб-сервера для обслуговування статичних файлів та проксіювання запитів було обрано веб-сервер, що гарантує стабільну роботу системи. Сформований стек технологій забезпечує необхідний баланс між продуктивністю обчислювальних алгоритмів штучного інтелекту, надійністю зберігання даних та гнучкістю користувацького інтерфейсу, повністю відповідаючи вимогам інтелектуальної системи моніторингу агропідприємств.

3 РОЗРОБКА ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ МОНІТОРИНГУ АГРОПІДПРИЄМСТВ

У цьому розділі здійснено аналіз сучасних підходів до створення вебзастосунків у сфері агромоніторингу та систем підтримки прийняття рішень. Обґрунтовано вибір технологій для реалізації клієнтської та серверної частин, наведено опис ключових функцій, алгоритмів роботи програми, структури модулів і механізмів інтеграції зі штучним інтелектом. Представлено огляд розробленої системи, її основних можливостей і користувацьких інтерфейсів. Проведено тестування програмного забезпечення, наведено аналіз отриманих результатів та їх вплив на подальше вдосконалення системи.

3.1 Огляд сучасних підходів до реалізації функціоналу та візуальної складової

Розроблення інтелектуальної веб-системи автоматизації та моніторингу агропідприємств вимагає використання сучасних технологічних підходів, орієнтованих на високу продуктивність, масштабованість, модульність та зручність роботи користувачів. Такі системи функціонують в умовах інтенсивних потоків даних від IoT-сенсорів, метеостанцій, дронів, супутникових сервісів та внутрішніх облікових підсистем, тому архітектурні та технологічні рішення мають забезпечувати надійність, низькі затримки обробки даних і можливість інтеграції алгоритмів штучного інтелекту.

Сучасні інтелектуальні системи все частіше будуються на модульній або мікросервісній архітектурі. Такий підхід дає змогу розділити функціонал на окремі логічні блоки: модулі збору даних з полів, модулі агроаналітики, модулі управління користувачами та ролями, підсистеми інтеграції з ШІ-сервісами, підсистеми планування агротехнологічних операцій тощо. Кожен модуль може

розгортатися та масштабуватися незалежно, що особливо важливо при зростанні кількості підключених полів, сенсорів чи господарств. Для роботи з потоками даних реального часу доцільно застосовувати реактивний підхід і швидкі in-memory сховища, такі як Redis, які дозволяють передавати оновлення агропоказників із мінімальною затримкою.

Важливою особливістю розроблюваної системи є інтеграція штучного інтелекту. Використання MCP-підходу (Model–Context–Prompt) забезпечує можливість підключення готових великомовних моделей (LLM), які на основі переданого контексту — стану поля, погодних умов, результатів аналізу ґрунту — формують рекомендації для агрономів та керівників агропідприємств. Такий підхід дозволяє сконцентруватися на побудові правильної предметної моделі та контексту, не витрачаючи ресурси на навчання власних моделей на початкових етапах.

Не менш важливим є підхід до побудови візуальної складової системи. Інтерфейс користувача в інтелектуальній агросистемі має відображати велику кількість показників у наочному вигляді та при цьому не перевантажувати користувача. Для цього застосовується компонентно-орієнтований підхід до проєктування інтерфейсу, за якого система складається з незалежних UI-компонентів: панелей дашборду, карт полів, графіків динаміки, таблиць операцій, вікон з рекомендаціями ШІ тощо. Компоненти можуть повторно використовуватися в різних частинах системи, мають чітко визначені входи та виходи та легко комбінуються в складні інтерфейсні рішення.

Важливе місце у візуальній частині займають інтерактивні графіки та карти. Для агропідприємств ключовим інструментом є відображення полів на карті з накладенням тематичних шарів: вегетаційних індексів, зон продуктивності, рівнів вологості ґрунту, історії стану культур. Це дає змогу агроному оцінити ситуацію за кілька секунд, не заглиблюючись у «сирі» дані. Адаптивний дизайн інтерфейсу забезпечує коректну роботу як на робочих станціях в офісі, так і на планшетах або ноутбуках у полі.

У межах розроблюваної системи ці підходи реалізовано на базі сучасного стеку вебтехнологій. У якості основної мови програмування обрано TypeScript, що поєднує гнучкість JavaScript і переваги статичної типізації. Завдяки цьому забезпечується підвищена надійність коду, передбачуваність поведінки застосунку та можливість спільного використання типів у клієнтській і серверній частинах.

Для реалізації клієнтської частини системи використовується фреймворк Next.js, який базується на бібліотеці React і підтримує сучасний підхід до побудови вебінтерфейсів. Next.js забезпечує високу продуктивність інтерфейсу, зручну організацію маршрутизації, підтримку серверних компонентів та ефективну роботу з динамічними даними, що особливо важливо для дашбордів та аналітичних панелей. Крім того, Next.js органічно інтегрується з TypeScript, що спрощує підтримку та розвиток коду у великому проєкті.

Серверна частина системи реалізується з використанням фреймворку NestJS, який також орієнтований на TypeScript і побудований за принципами модульності та інверсії залежностей. NestJS дозволяє структуровано організувати бізнес-логіку, розділити її на окремі модулі, зручно підключати зовнішні сервіси, працювати з базами даних та чергами повідомлень. Такий підхід полегшує реалізацію мікросервісних або багатомодульних систем, де частина сервісів відповідає за обробку агроданих, частина — за синхронізацію з ШІ, інша — за авторизацію, аналітику або інтеграцію з зовнішніми агроплатформами.

Окремої уваги потребує питання валідації даних. У системі, що працює з великою кількістю форм, API-запитів, конфігурацій та аналітичних параметрів, важливо уникати дублювання правил валідації та розбіжностей між фронтендом і бекендом. Для цього обрана бібліотека Zod, яка дозволяє описувати схеми даних у єдиному вигляді та використовувати їх одночасно на клієнтській і серверній стороні. Zod-схеми застосовуються як для валідації даних у формах інтерфейсу, так і для перевірки вхідних параметрів API, що

забезпечує цілісність даних та скорочує кількість помилок, пов’язаних із некоректними вхідними значеннями.

Архітектурно програмний продукт організовано у форматі монорепозиторію, в якому зберігаються клієнтська частина (Next.js), серверна частина (NestJS) та спільні бібліотеки (типи, моделі, Zod-схеми, утиліти). Такий підхід має низку переваг: спільне використання типів і моделей між фронтендом та бекендом, відсутність дублювання коду, єдиний підхід до стилю оформлення та інженерних практик, спрощене налаштування CI/CD та контроль залежностей. Для масштабованої інтелектуальної системи, де одні й ті ж бізнес-сутності (поле, культура, сенсор, показник, рекомендація) використовуються в різних модулях, монорепозиторій є логічним і технічно виправданим рішенням.

Узагальнюючи, сучасні підходи до реалізації функціональної та візуальної складової інтелектуальної системи автоматизації та моніторингу агропідприємств базуються на поєднанні модульної архітектури, компонентного інтерфейсу, реактивної обробки даних, інтеграції ШІ та уніфікованої типобезпечної валідації. Використання TypeScript, Next.js, NestJS, Zod, Redis та монорепозиторію формує надійну технологічну основу, яка забезпечує можливість подальшого розвитку системи, її масштабування та адаптації до нових вимог агросектору.

3.2 Розробка програмного забезпечення

Розроблення інтелектуальної системи автоматизації та моніторингу агропідприємств здійснювалося із застосуванням сучасних підходів до архітектурування вебзастосунків, поетапної побудови функціональних модулів та інтеграції підсистем штучного інтелекту. У цьому розділі викладено ключові етапи реалізації клієнтської та серверної частин, побудови архітектури проєкту,

реалізації основних функцій, а також особливостей, що забезпечують масштабованість і гнучкість майбутньої системи.

3.2.1 Архітектура та організація програмного проєкту

Архітектура інтелектуальної системи автоматизації та моніторингу агропідприємств реалізована у форматі монорепозиторію, що забезпечує цілісність коду, зручність повторного використання спільних модулів та єдину систему типізації між клієнтською і серверною частинами[30]. Такий підхід дозволяє підтримувати узгодженість бізнес-логіки, пришвидшує розробку та мінімізує ризики помилок, пов’язаних із дублюванням моделей даних у різних підпроєктах. Загальна структура монорепозиторію наведена на рисунку 3.1.

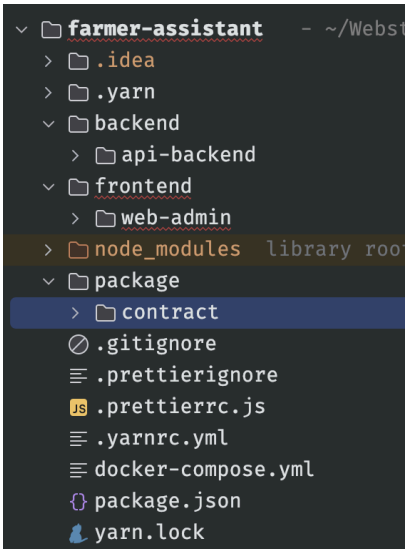

Рисунок 3.1 – Структура монорепозиторію інтелектуальної системи

Серверна частина — backend/api-backend. Цей модуль реалізовано на основі NestJS, що забезпечує:

- модульну архітектуру,
- впорядковану бізнес-логіку,

- підтримку Dependency Injection,
- інтеграцію з PostgreSQL, Redis та зовнішніми API,
- зручну реалізацію мікросервісів у майбутньому.

У серверній частині зосереджена логіка:

- авторизації та керування ролями,
- взаємодії з базою даних,
- обробки агроданих,
- генерації рекомендацій ШІ через MCP-підхід,
- формування аналітики,
- управління полями та культурами.

Клієнтська частина — frontend/web-admin. Фронтенд створено на базі Next.js (React), що забезпечує:

- швидку роботу інтерфейсу,
- можливість використання серверних компонентів,
- організовану структуру сторінок,
- адаптивний інтерфейс для планшетів і ноутбуків агрономів,
- просту інтеграцію з REST API серверної частини.

У web-admin реалізовано:

- дашборд агропідприємства,
- модуль інтерактивних карт,
- модуль рекомендацій ШІ,
- перегляд та редагування даних полів,
- управління користувачами,
- модулі агроопераций.

Спільні модулі — package/contract. Окремий пакет contract містить:

- Zod-схеми (використовуються як на фронтенді, так і на бекенді),
- спільні типи TypeScript,
- DTO та моделі,
- утиліти,
- спільні константи та перелічення.

Це дозволяє:

- уникати дублювання моделей між сервісами;
- гарантувати однакову структуру даних у фронтенді та бекенді;
- забезпечити типобезпеку системи;
- значно зменшити ризик помилок інтеграції.

Такий підхід характерний для сучасних enterprise-рішень і є найефективнішим при роботі з інтелектуальними автоматизованими системами, у яких велика кількість бізнес-сутностей використовується одночасно на клієнтській і серверній стороні.

3.2.2 Стандартизація коду та забезпечення якості

Якість програмного коду є критичним фактором для довгострокової підтримки та масштабування інтелектуальної системи, особливо в умовах мікросервісної архітектури та великого монорепозиторію. З метою забезпечення консистентності, читабельності та зменшення кількості потенційних помилок, ще на етапі ініціалізації проєкту було впроваджено комплекс інструментів для автоматизованого контролю якості. Першочерговим завданням стало налаштування лінтера ESLint, який слугує для статичного аналізу коду та виявлення програмних помилок, порушень стилю, а також застосування передових практик програмування. Правила ESLint уніфіковані для всіх підпроєктів: як для NestJS-бекенду, так і для Next.js-фронтенду, завдяки чому

вся команда дотримується єдиного набору стандартів. Додатково впроваджено жорсткі політики щодо типізації, щоб максимально використовувати переваги TypeScript, запобігаючи помилкам, пов'язаним із невірним типом даних.

Процес розробки підсилюється використанням інструменту автоматичного форматування Prettier. Основна функція Prettier полягає у забезпеченні абсолютної стилістичної одноманітності коду по всьому монорепозиторію, включаючи HTML, CSS, JavaScript та TypeScript. Завдяки інтеграції Prettier із системою контролю версій (Git hooks) та середовищем розробки (IDE), форматування коду відбувається автоматично під час збереження або коміту, повністю виключаючи суб'єктивні суперечки щодо стилю між розробниками. Ця автоматизація значно прискорює процес code review та дозволяє розробникам зосереджуватися виключно на бізнес-логіці та алгоритмах. Крім того, використовуються механізми Husky або подібні для виконання лінтерів і тестів перед кожним комітом, що є обов'язковим для підтримання "здоров'я" головної гілки коду. Інструменти типу Jest та React Testing Library інтегровані для написання unit- та інтеграційних тестів. Тести забезпечують верифікацію критичної бізнес-логіки, наприклад, коректності розрахунку індексів вегетації та функціоналу авторизації. Наявність спільних Zod-схем у пакеті *package/contract* додатково спрощує тестування, дозволяючи мокувати (mock) типобезпечні вхідні та вихідні дані для модульних тестів на бекенді. Таким чином, стандартизація коду, його автоматичне форматування та обов'язкове тестування є фундаментальними принципами, які гарантують якість та стабільність інтелектуальної системи.

3.2.3 Організація процесу розгортання та інфраструктура

Ефективне та надійне розгортання (Deployment) складної мікросервісної архітектури є ключовою вимогою для інтелектуальної системи моніторингу агропідприємств. Для забезпечення масштабованості, ізоляції та автоматичного

самовідновлення застосунків обрано використання Docker та Kubernetes. Кожен із мікросервісів (NestJS-сервіси, Next.js-фронтенд) контейнеризується за допомогою Docker, що дозволяє інкапсулювати код, середовище виконання та всі залежності в єдиний, портативний образ. Це гарантує, що застосунок працюватиме ідентично на будь-якій інфраструктурі, усуваючи проблему "працює лише на моїй машині".

Управління цими контейнеризованими компонентами здійснюється за допомогою Kubernetes (K8s), який виступає в ролі оркестратора. Налаштування Kubernetes є комплексним і вимагає створення спеціалізованих конфігураційних файлів. Для цього розробляються детальні Deployment YAML-файли або Helm charts, які описують, як саме має бути розгорнута система. Ці конфігураційні файли містять визначення кількості необхідних реплік для кожного мікросервісу, налаштування Network Policies для безпечної внутрішньої комунікації, вимоги до ресурсів (CPU/RAM) та правила автомасштабування. Використання Helm charts дозволяє створювати параметризовані конфігурації, які можуть бути легко адаптовані для різних середовищ: розробки, тестування та продуктивного середовища.

Інфраструктура даних також інтегрується через ці конфігураційні файли. Зокрема, налаштовуються з'єднання із зовнішніми сервісами, такими як PostgreSQL/PostGIS (через Secrets), Apache Kafka та Redis, гарантуючи їхню доступність для мікросервісів. Для роботи з великими бінарними файлами налаштовується інтеграція з Amazon S3 або його аналогом. Kubernetes автоматично забезпечує високу доступність (High Availability) системи. Якщо один із вузлів кластера або екземпляр мікросервісу виходить з ладу, K8s автоматично перезапускає контейнер або перенаправляє трафік на здоровий екземпляр. Також налаштовується горизонтальне автомасштабування (Horizontal Pod Autoscaler) для автоматичного збільшення кількості NestJS-сервісів у пікові моменти навантаження (наприклад, під час інтенсивного аналізу нових супутникових даних). Таким чином, процеси Docker

та Kubernetes забезпечують надійний, відтворюваний та автоматизований життєвий цикл системи, що є фундаментальним для комерційної експлуатації.

3.3 Тестування програмного забезпечення

Тестування інтелектуальної системи автоматизації та моніторингу агропідприємств є критичним етапом розробки, що гарантує функціональну коректність, стабільність роботи в умовах високого навантаження та відповідність кінцевого продукту бізнес-вимогам. Необхідність систематичного тестування обґрунтована складністю розподіленої мікросервісної архітектури, гібридним сховищем даних (PostgreSQL/PostGIS, S3, Redis, Kafka) та наявністю інтелектуального ядра, яке використовує ШІ для генерації агрономічних рекомендацій. Проведення тестування мінімізує ризики, пов'язані з невірною обробкою геопросторових даних або некоректним формуванням рекомендацій, що може призвести до значних фінансових втрат для агропідприємства.

Тестування проводиться на всіх етапах розробки, починаючи від модульного контролю окремих функцій до комплексного системного та приймального тестування кінцевими користувачами.

Написання тест-кейсів ґрунтується на детальних специфікаціях функціональних вимог і включає як позитивні, так і негативні сценарії. Кожен тест-кейс чітко документується (Ідентифікатор, Передумови, Кроки, Очікуваний результат).

Для візуалізації послідовності критичних бізнес-процесів розробляється Test Flow Diagram . Ця діаграма чітко ілюструє шлях користувача та інтеграційні точки, які необхідно перевірити.

Однією з базових можливостей системи є автентифікація користувачів, що забезпечує персоналізований доступ до даних агропідприємства. Під час тестування було підтверджено:

- коректність валідації електронної пошти та пароля під час реєстрації;
- автоматичну нормалізацію email після введення користувачем;
- захист пароля через хешування (Bcrypt);
- стабільну роботу механізму JWT-токенів для доступу та оновлення сесії;
- можливість входу в систему з різних пристроїв з коректною обробкою помилок (невалідні логін або пароль, неактивний акаунт тощо).

При успішній автентифікації система автоматично перенаправляє користувача до головної панелі керування (Dashboard), де доступні всі основні інструменти (рис 3.1).

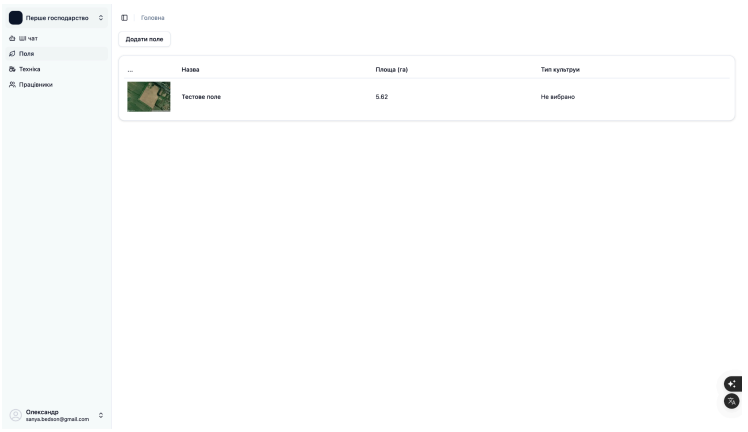

Рисунок 3.2 – Dashboard системи

Функціонал створення фермерського господарства є одним із ключових у системі, оскільки саме з цього етапу починається формування структурованих даних про аграрні підприємства, їхніх учасників та пов’язані з ними ресурси. Під час тестування даного модуля було здійснено комплекс перевірок, спрямованих на оцінку стабільності роботи інтерфейсу, коректності взаємодії з бекендом, повноти валідації та правильності запису інформації до бази даних.

Після автентифікації користувач переходить на головний екран. Якщо фермер ще не має створених господарств, система автоматично пропонує ініціювати процес створення нового підприємства. У випадку наявності

декількох господарств відображається їхній список у лівій панелі керування із можливістю перемикання між ними.

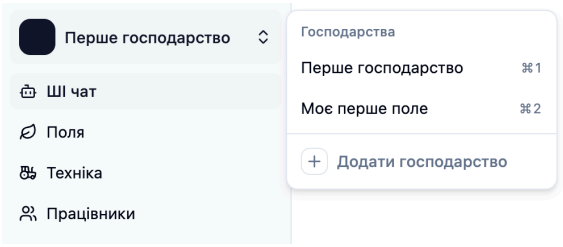

Рисунок 3.3 – Управління фермерськими господарствами

У модальном вікні створення господарства міститься форма, в якій користувач має можливість внести базові дані такі як, назва господарства, яке є обов’язковим полем та обрати географічне розташування, яке супроводжується інтерактивною картою. Це дозволяє надалі прив’язувати поля, погодні умови, супутникові дані та зони обробітку до конкретної геолокації.

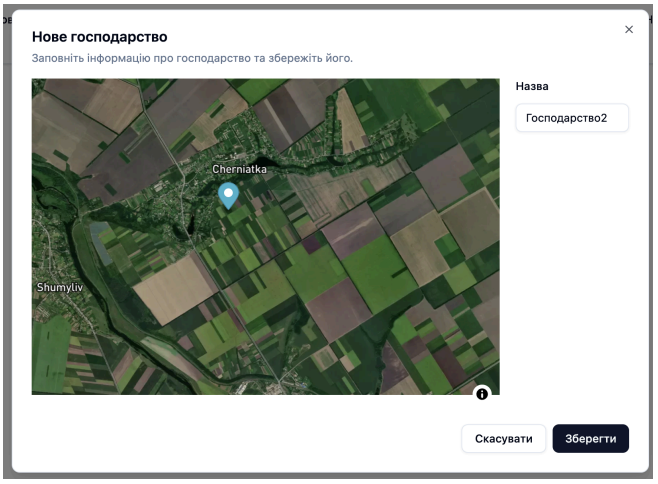

Рисунок 3.4 – Створення фермерського господарства

Після успішного створення фермерського господарства користувач отримує доступ до управління земельними ділянками, що є базовою складовою функціоналу системи. Наступним кроком є можливість додавати та конфігурувати поля, які надалі використовуються для аналітики, рекомендацій ШІ та моніторингу стану господарства.

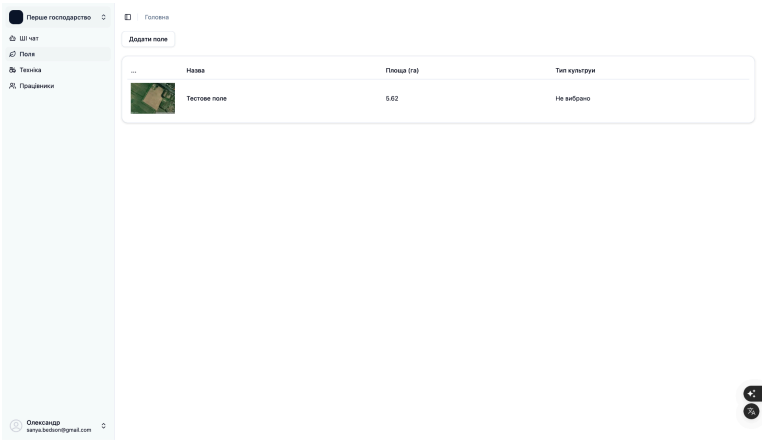

Рисунок 3.5 – Таблиця управління земельними ділянками

Процес створення поля реалізований у вигляді інтерактивної форми з картою, що дозволяє точно визначати межі земельних ділянок та зберігати їх у структурованому вигляді. Тестування цього модуля включало перевірку роботи UI-компонентів, взаємодію з картою, коректність побудови полігону та відправку даних на сервер.

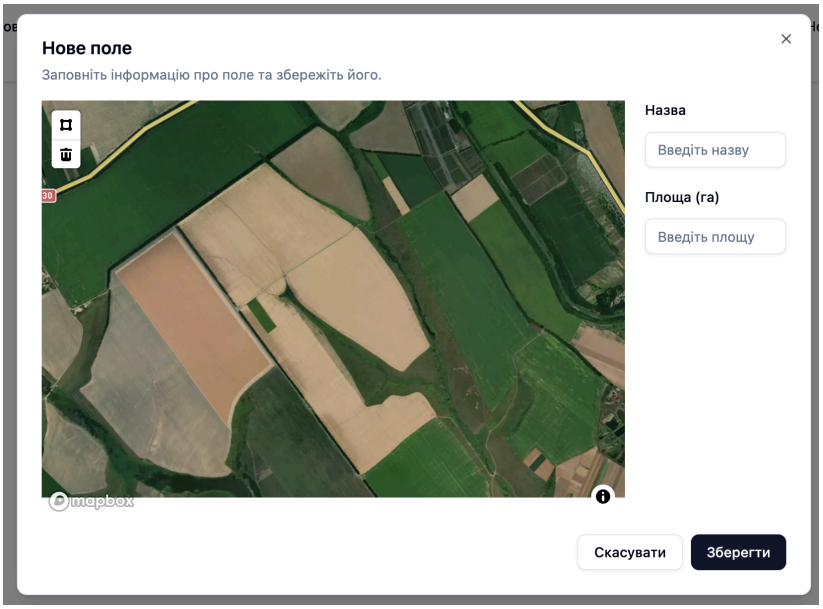

Рисунок 3.6 – Ітеративна форма створення поля

Окрему увагу під час тестування було приділено модулю ШІ-чату, який відповідає за формування рекомендацій для аграрія на основі введених ним

запитів та контексту господарства. Перевірялися як технічні аспекти роботи (швидкість формування відповіді, стабільність інтеграції з API мовної моделі, коректність обробки помилок), так і змістовна якість згенерованих рекомендацій. Для цього використовувалися типові для аграрної практики сценарії: підбір норм внесення добрив для різних культур і фаз вегетації, поради щодо поливу за умов високих температур та низької вологості, рекомендації щодо реагування на можливі прояви захворювань чи дефіциту елементів живлення. У кожному з кейсів оцінювалася логічність пояснень, відсутність суперечностей з базовими агротехнічними принципами, а також здатність ШІ адаптувати відповідь до конкретного поля чи господарства.

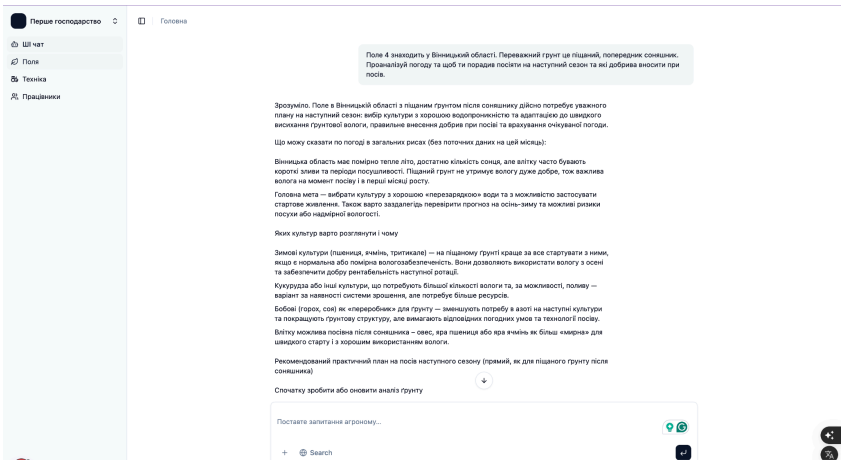

Рисунок 3.7 – Симуляція роботи ШІ помічника

Проведене тестування розробленої інтелектуальної системи управління та моніторингу агропідприємства підтвердило її коректну роботу та відповідність поставленим вимогам. Усі функціональні модулі, включаючи авторизацію користувачів, управління госопдарствами, полями, а також генерація рекомендацій, працюють належним чином. Тестування показало стабільність роботи системи за різних сценаріїв використання та її здатність ефективно обробляти великий обсяг даних. Отримані результати свідчать про готовність системи до впровадження в навчальний процес та використання у реальних умовах.

3.4 Висновки до розділу

У даному розділі було успішно здійснено практичну реалізацію інтелектуальної системи моніторингу та автоматизації агропідприємств відповідно до обраного стеку вебтехнологій. Була розроблена трирівнева архітектура системи, що включає серверну частину для обробки даних, рівень API для забезпечення комунікації та клієнтську частину для взаємодії з користувачем. Ключовим елементом реалізації став API-сервер, який слугує центральним вузлом для прийому даних від польових сенсорів та звітних модулів, інтеграції з геопросторовими сервісами, а також для виконання складного модифікованого алгоритму інтелектуального аналізу даних. Була розроблена універсальна веб-орієнтована клієнтська частина, яка забезпечує зручний та інтуїтивно зрозумілий інтерфейс для агрономів та керівництва. Ця реалізація є повністю крос-платформною, гарантуючи доступність на будь-якому пристрої, що має сучасний браузер, що напряму забезпечує віддалений моніторинг і оперативне прийняття рішень. Інтерфейс дозволяє візуалізувати стан полів на інтерактивній мапі, отримувати оперативні попередження про відхилення, а також переглядати рекомендації, згенеровані штучним інтелектом. На завершальному етапі проведено комплексне тестування розробленої системи. Таким чином, результати розділу демонструють успішну реалізацію всіх ключових програмних компонентів, що дозволило створити функціонально повну, надійну та високопродуктивну інтелектуальну систему моніторингу, доступну через веб-браузер.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було досягнуто поставленої мети — розроблено інтелектуальну систему автоматизації та моніторингу агропідприємств, яка об’єднує сучасні вебтехнології, супутникові та сенсорні дані, а також алгоритми штучного інтелекту для підтримки прийняття агротехнологічних рішень. Розроблена система забезпечує комплексний підхід до аналізу стану полів, прогнозування врожайності та оптимізації виробничих процесів у агросекторі.

У процесі роботи було проведено детальне дослідження предметної області та визначено ключові проблеми сучасних агропідприємств, пов’язані з неефективністю ручних процесів, труднощами моніторингу посівів, обмеженістю традиційних методів прогнозування та залежністю від людського фактору. Проаналізовано існуючі програмні рішення для агромоніторингу, встановлено їхні недоліки, зокрема відсутність інтеграції між різними джерелами даних, низьку гнучкість у масштабуванні та обмежені можливості застосування ШІ. На основі проведеного аналізу було розроблено архітектуру інтелектуальної системи, яка включає модулі збору та зберігання даних, обробку супутникових і сенсорних показників, блок аналітики на основі машинного навчання, а також вебінтерфейс для візуалізації результатів. Реалізовано прогнозний AI-модуль, здатний оцінювати стан культур, виявляти потенційні ризики (посуха, дефіцит азоту, ймовірність хвороб) та генерувати рекомендації для агрономів за допомогою контекстно-керованого MCP-підходу.

Створена система забезпечує:

- автоматизований збір та об’єднання багатовимірних агроданих із різних джерел (IoT-сенсори, супутники, дрони, історичні журнали обробітку);

- інтелектуальний аналіз та прогнозування стану посівів на основі алгоритмів машинного навчання;
- формування рекомендацій для агротехнологічних рішень із використанням ШІ;
- зручний та інтуїтивний вебінтерфейс з інтерактивними картами, графіками та аналітичними звітами;
- масштабовану архітектуру, що дозволяє розширювати функціонал та інтегрувати нові модулі.

Практична цінність розробленої системи полягає у підвищенні ефективності агровиробництва за рахунок автоматизації ключових процесів моніторингу та аналізу, зниженні витрат ресурсів, мінімізації ризиків, пов’язаних із помилками людського фактору, та забезпеченні оперативного доступу до аналітичної інформації. Система може бути впроваджена в агропідприємства різних масштабів, інтегрована в існуючі ERP/CRM-рішення або використана як автономна SaaS-платформа.

Інноваційність роботи полягає у поєднанні супутникових технологій, IoT-телеметрії, вебсервісів і штучного інтелекту в єдиній інтегрованій платформі, що реалізує новий підхід до агромоніторингу — від реактивного до прогнозного управління. Використання MCP-парадигми забезпечило гнучкість, контекстність і високу адаптивність рекомендацій, що є сучасним трендом у побудові інтелектуальних систем.

Результати роботи були апробовані під час тестування програмного комплексу, яке підтвердило коректність функціонування системи, точність прогнозів та доцільність застосування ШІ-підходів у агровиробництві. У перспективі розроблена система може бути розширена модулями економічного планування, оптимізації логістики та автоматизованого управління технікою, що дозволить створити повноцінну цифрову екосистему агропідприємства.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Соколовський О. О. Інтелектуальна система для автоматизації та моніторингу агропідприємств із використанням сучасних вебтехнологій і штучного інтелекту. Молодь в науці: дослідження, проблеми, перспективи (МН-2026) : Міжнар. науково-практ. інтернет-конф., м. Вінниця, 22–26 черв. 2026 р. Вінниця.
2. FAO Knowledge Repository. FAO Knowledge Repository. URL: https://openknowledge.fao.org/items/98a4c80a-b4d3-403c-8557-d8536c8316ee (дата звернення: 05.10.2025).
3. Documents & Reports - All Documents | The World Bank. URL: https://documents1.worldbank.org/curated/en/099053025063021993/pdf/P508004-f943a09b-c45f-4c93-b554-9dd1decd1e7c.pdf (дата звернення: 09.11.2025).
4. Climate Change 2022: Impacts, Adaptation and Vulnerability. IPCC – Intergovernmental Panel on Climate Change. URL: https://www.ipcc.ch/report/ar6/wg2/ (дата звернення: 30.10.2025).
5. Cropwise operations. Cropwise. URL: https://ua.cropwise.com/en/operations (дата звернення: 05.11.2025).
6. OneSoil | Free App for Precision Farming. OneSoil | Free Farming App for Precision Agriculture. URL: https://onesoil.ai/en (дата звернення: 21.11.2025).
7. Maximize Results with Our Digital Farming Solution | Climate FieldView. URL: https://climatefieldview.com (дата звернення: 13.11.2025).
8. Satellite Data Analytics And Imagery Analysis By EOSDA. EOSDA Data Analytics. URL: https://eos.com (дата звернення: 21.11.2025).

9. Перспективи та особливості точного землеробства в 2025. URL: https://agroexp.com.ua/uk/perspektivy-i-osobennosti-tochnogo-zemledeliya (дата звернення: 05.11.2025).
10. Клієнт-серверна архітектура. JavaRush. URL: https://javarush.com/ua/quests/lectures/ua.questservlets.level14.lecture00 (дата звернення: 21.10.2025).
11. Мікросервісна архітектура: значення та складові | Wezom. Wezom. URL: https://wezom.com.ua/ua/blog/scho-take-mikroservisna-arhitektura-znachenny a-skladovi-perevagi (дата звернення: 21.10.2025).
12. MCP Prompts Explained (including how to actually use them). URL: https://medium.com/@laurentkubaski/mcp-prompts-explained-including-how-t o-actually-use-them-9db13d69d7e2 (дата звернення: 22.10.2025).
13. Сервіси супутникового моніторингу для аграріїв . URL: https://www.smartfarming.ua/servisy-suputnykovoho-monitorynhu-dlya-ahrari yiv (дата звернення: 23.10.2025).
14. ПОРІВНЯННЯ РЕЛЯЦІЙНИХ БАЗ ДАНИХ ДЛЯ ВИКОРИСТАННЯ В ІНФОРМАЦІЙНИХ СИСТЕМАХ. Громадська організація "Інноваційні обрії України". URL: http://obrii.org.ua/usec/storage/article/Skrypka_2024_284.pdf (дата звернення: 28.10.2025).
15. PostgreSQL. PostgreSQL. URL: https://www.postgresql.org/ (дата звернення: 28.10.2025).
16. Large language Models, LLMs. IT-Enterprise – цифрова трансформація бізнес-процесів, ERP| it.ua. URL: https://www.it.ua/knowledge-base/technology-innovation/large-language-mode ls-llms (дата звернення: 01.11.2025).

17. What is the Model Context Protocol (MCP)? - Model Context Protocol. URL: https://modelcontextprotocol.io/docs/getting-started/intro (дата звернення: 01.11.2025).
18. TypeScript: Переваги та Використання в Україні – UA – Galaktica. *UA – Galaktica*. URL: https://galaktica.io/blog/typescript-tse/ (дата звернення: 11.11.2025).
19. JavaScript With Syntax For Types. TypeScript: JavaScript With Syntax For Types. URL: https://www.typescriptlang.org/ (дата звернення: 11.11.2025)
20. NestJS - A progressive Node.js framework. *NestJS - A progressive Node.js framework*. URL: https://nestjs.com/ (дата звернення: 11.11.2025).
21. Next.js by Vercel - The React Framework. Next.js by Vercel - The React Framework. URL: https://nextjs.org/ (дата звернення: 11.11.2025).
22. SPA (Single-page application) - Glossary | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA (дата звернення: 11.11.2025).
23. Apache Kafka. *Apache Kafka*. URL: https://kafka.apache.org/ (дата звернення: 11.11.2025).
24. Amazon S3 - Cloud Object Storage - AWS. *Amazon Web Services, Inc.* URL: https://aws.amazon.com/s3/ (дата звернення: 13.11.2025).
25. Docker: Accelerated Container Application Development. *Docker*. URL: https://www.docker.com/ (дата звернення: 14.11.2025).
26. Kubernetes Documentation. *Kubernetes*. URL: https://kubernetes.io/docs/home/ (дата звернення: 14.11.2025).
27. Проблеми безпеки та рішення в архітектурі мікросервісів. *Hostragons®*. URL: https://www.hostragons.com/uk/блог/проблеми-безпеки-в-архітектурі-мікро/ (дата звернення: 16.11.2025).
28. Як використовувати JSON Web Tokens (JWT) для автентифікації. *DevZone*. URL:

https://devzone.org.ua/post/iak-vykorystovuvaty-json-web-tokens-jwt-dlia-avtentyfikatsiyi (дата звернення: 16.11.2025).

29. OAuth 2.0: як працює авторизація через токени у веб і мобільних додатках. *WEZOM*. URL: https://wezom.com.ua/ua/blog/scho-take-oauth-20-i-yak-pratsyuje-avtorizatsiya-cherez-tokeni (дата звернення: 18.11.2025).
30. Дослідження процесу безперервної інтеграції та безперервної доставки у монорепозиторіях. *DSpace Angular*. URL: https://dspace.khadi.kharkov.ua/items/ab4c92f8-e29f-41a3-adfa-f73927e943c5 (дата звернення: 20.11.2025).
31. Що таке тестування програмного забезпечення: види, етапи, інструменти. *Sigma Software University*. URL: https://university.sigma.software/what-is-software-testing/ (дата звернення: 01.12.2025).

ДОДАТКИ

Додаток А (обов’язковий)

Технічне завдання

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«17» жовтня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«ІНТЕЛЕКТУАЛЬНА СИСТЕМА АВТОМАТИЗАЦІЇ ТА МОНІТОРИНГУ АГРОПІДПРИЄМСТВ ІЗ ВИКОРИСТАННЯМ СУЧАСНИХ ВЕБТЕХНОЛОГІЙ ТА ШТУЧНОГО ІНТЕЛЕКТУ»
08-31.МКР.010.02.000 ТЗ

Керівник роботи:
к.т.н., доц. каф. АІІТ
Володимир КОЦЮБИНСЬКИЙ
«16» жовтня 2025 р.

Виконавець:
ст. гр. ЗАКІТР-24м
Олександр СОКОЛОВСЬКИЙ
«16» жовтня 2025 р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

Інтелектуальна система автоматизації та моніторингу агропідприємств із використанням сучасних вебтехнологій та штучного інтелекту.

Галузь застосування: інформаційні системи та технології в агропромисловому комплексі, цифрове землеробство, системи моніторингу та підтримки агрономічних рішень, застосування штучного інтелекту та аналізу даних у сільському господарстві.

2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету № 313 від 24 вересня 2025 року та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій»

3. Мета та призначення розробки

Метою роботи є створення інтелектуальної системи автоматизації та моніторингу агропідприємств, яка забезпечить:

- Автоматизований збір, агрегування та попередню обробку агроданих із різних джерел: IoT-сенсорів, метеостанцій, дронів та супутникових знімків (NDVI, EVI, карти вологи, температурні карти тощо);
- Інтелектуальний аналіз стану посівів із використанням алгоритмів машинного навчання та LLM-моделей, включно з визначенням ознак стресу рослин, прогнозуванням розвитку культур і виявленням аномалій;
- Генерацію рекомендацій щодо агротехнічних операцій, зокрема оптимізації норм та строків внесення добрив, зрошення, обробки від шкідників і планування польових робіт;
- Формування аналітичних дашбордів у реальному часі, що відображають ключові показники стану полів, прогноз врожайності, ризики та рекомендації;
- Підтримку прийняття рішень агрономами та керівниками господарств за рахунок інтеграції MCP-підходу (Model–Context–Prompt), що забезпечує адаптивну генерацію текстових та структурованих порад;

- Можливість інтеграції з зовнішніми аграрними сервісами (метеодані, геоплатформи, постачальники супутникових знімків) через REST API;
- Підтримку хмарної інфраструктури та мікросервісної архітектури для масштабування системи залежно від розмірів агропідприємства та обсягів даних.

Призначення системи: забезпечення підвищення ефективності агровиробництва за рахунок точного землеробства, оптимізації внесення добрив і ресурсів, мінімізації людського фактору, своєчасного виявлення ризиків та формування обґрунтованих агрономічних рішень на основі комплексного аналізу польових даних і алгоритмів штучного інтелекту.

4. Джерела розробки

1. FAO Knowledge Repository. FAO Knowledge Repository. URL: https://openknowledge.fao.org/items/98a4c80a-b4d3-403c-8557-d8536c8316ee (дата звернення: 05.10.2025).
2. Documents & Reports - All Documents | The World Bank. URL: https://documents1.worldbank.org/curated/en/099053025063021993/pdf/P508004-f943a09b-c45f-4c93-b554-9dd1decd1e7c.pdf (дата звернення: 09.11.2025).
3. Climate Change 2022: Impacts, Adaptation and Vulnerability. IPCC – Intergovernmental Panel on Climate Change. URL: https://www.ipcc.ch/report/ar6/wg2/ (дата звернення: 30.10.2025).

5. Показники призначення

5.1. Основні технічні характеристики системи

Функціональні можливості:

5.1.1 Модуль збору та підготовки агроданих

Система забезпечує автоматизоване збирання, обробку та нормалізацію даних з різних джерел, включаючи:

- IoT-сенсори (вологість ґрунту, температура, рівень освітленості, pH);
- метеостанції (опади, швидкість вітру, температура повітря);
- супутникові знімки NDVI/EVI, температурні карти та карти вологи (формати GeoTIFF/JPEG);
- дрон-знімки високої роздільності;
- історичні агрономічні показники у вигляді CSV/JSON.

Підтримується:

- автоматична конвертація геоданих у єдиний формат;
- очищення даних від шумів, артефактів та пропусків;
- геоприв’язка контурів полів.

5.1.2 Модуль аналітики та оцінки стану посівів

Система використовує алгоритми машинного навчання та LLM-моделі для:

- виявлення стресових зон (нестача вологи, хвороби, дефіцит азоту);
- класифікації типів покращення/погіршення стану рослин;
- прогнозування врожайності на основі часових рядів NDVI та погодних даних;
- визначення ключових агрономічних ризиків (заморозки, бурі, суховії);
- оцінки динаміки росту культур.

5.1.3 Модуль формування рекомендацій (AI Decision Support)

Система генерує рекомендації на основі MCP-підходу (Model–Context–Prompt):

- норми внесення добрив за зонами поля;
- визначення оптимальних дат підживлення;
- поради щодо зрошення;
- попередження про можливі хвороби або шкідників;
- рекомендації щодо планування обробітку.

Підтримуються:

- адаптація рекомендацій до типу культури, стадії росту, погодних умов;
- пояснення рішень (explainable AI);
- можливість ручного коригування агрономом.

5.1.4 Модуль візуалізації та аналітичних дашбордів

Система забезпечує:

- інтерактивні карти полів із шаром NDVI/EVI;
- карти вологи ґрунту та температури;
- 3D-моделі рельєфу ділянок;
- графіки динаміки стану посівів;
- агрономічні KPI (Field Health Index, Fertilizer Efficiency Index, Water Stress Index, прогнозована врожайність);

Дані відображаються у реальному часі завдяки використанню WebSockets та кешування в Redis.

5.1.5 Модуль інтеграцій та API

Підтримуються:

- REST API для взаємодії з іншими агроплатформами;
- імпорт/експорт даних у форматах CSV, Shapefile, GeoJSON;
- підключення до зовнішніх джерел погодних даних (OpenWeather, NOAA, MeteoBlue);
- інтеграція з LLM-провайдерами (OpenAI, Gemini, Claude).

5.2. Мінімальні системні вимоги

5.2.1 Апаратне забезпечення

Серверна частина:

- Процесор: 4-8 ядер (рекомендовано для ML-аналітики);
- Оперативна пам’ять: 16 GB RAM;
- Місце на диску: 50–200 GB (залежно від обсягів супутникових даних);
- Мережа: 50 Mbps+ для роботи з великими супутниковими файлами;

Клієнтська частина:

- Будь-який сучасний браузер (Chrome, Firefox, Edge);
- 4 GB RAM.

5.2.2 Програмне забезпечення

- Операційні системи: Linux (Ubuntu 20.04+), Windows Server 2019+, macOS 12+;
- Node.js 22+ для серверної частини NestJS;
- PostgreSQL 14+ з розширенням PostGIS;
- Redis 6+ для кешування та черг;
- Docker / Docker Compose;
- Хмарні сервіси (AWS, GCP або Hetzner Cloud).

5.3. Вхідні дані

- Геопросторові дані полів: координати контурів у форматі GeoJSON/CSV;
- Супутникові NDVI/EVI знімки (Sentinel-2, Landsat-8);
- Дані IoT-сенсорів (JSON, MQTT потоки);
- Метеодані: історичні та прогнозні (API);
- Дані дронів у вигляді ортофотопланів;
- Історія агротехнічних операцій (внесення добрив, полив, обробка);
- Дані користувачів та структур господарства.

5.4. Результати роботи системи

5.4.1 Аналітичні результати та прогнозні моделі

Система формує:

- AI-моделі оцінки стану культур;
- прогнозні моделі врожайності;
- карти ризиків та стресу рослин;
- рекомендації щодо добрив, зрошення, обробітку.

Результати можуть експортуватися у:

- PDF-звіти,

- GeoTIFF,
- CSV,
- інтерактивні карти.

5.4.2 Аналітичні звіти та візуалізації

Система генерує:

- графіки динаміки NDVI,
- карти стану поля з поділом на зони,
- графік Water Stress Index,
- дашборд KPI агропідприємства,
- історію виконання рекомендацій,
- порівняння ефективності застосованих добрив.

Формати експорту:

- PNG (карти та графіки),
- PDF (зведений звіт),
- JSON/CSV для передачі даних іншим системам.

6. Стадії розробки:

1. Розділ 1 «Дослідження предметної області та існуючих методів» має бути виконаний до 05.10.2025 р.
2. Розділ 2 «Розробка математичного та алгоритмічного забезпечення системи прийняття торгових рішень» має бути виконаний до 25.10.2025 р.
3. Розділ 3 «Розробка програмного забезпечення системи прийняття торгових рішень» має бути виконаний до 20.11.2025 р.

7. Порядок контролю та приймання

1. Рубіжний контроль провести до 14.11.2025.
2. Попередній захист магістерської кваліфікаційної роботи провести до 02.12.2025.
3. Захист магістерської кваліфікаційної роботи провести в період з 15.12.2025 р. до 19.12.2025

Додаток Б (обов’язковий)
Ілюстративна частина

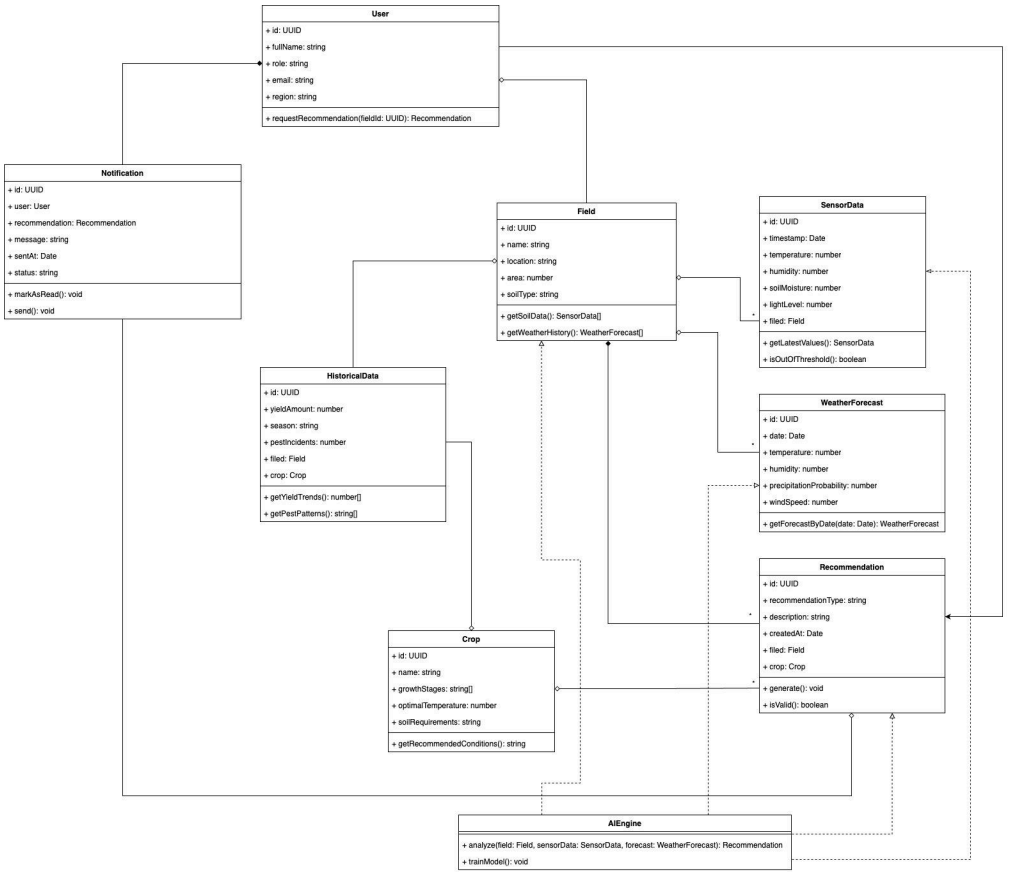

Рисунок Б.1 - UML-діаграма класів

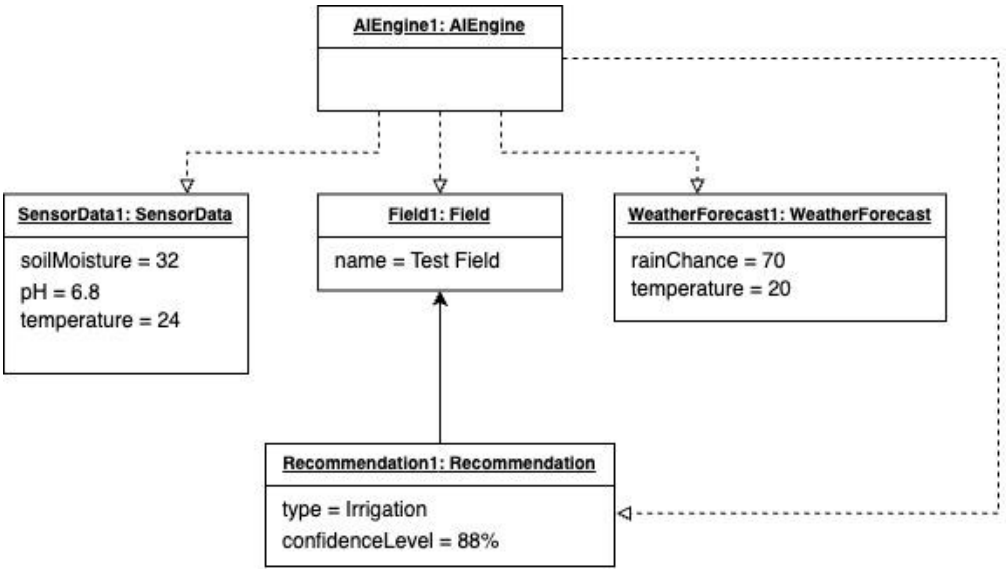

Рисунок Б.2 - UML-діаграма обєктів

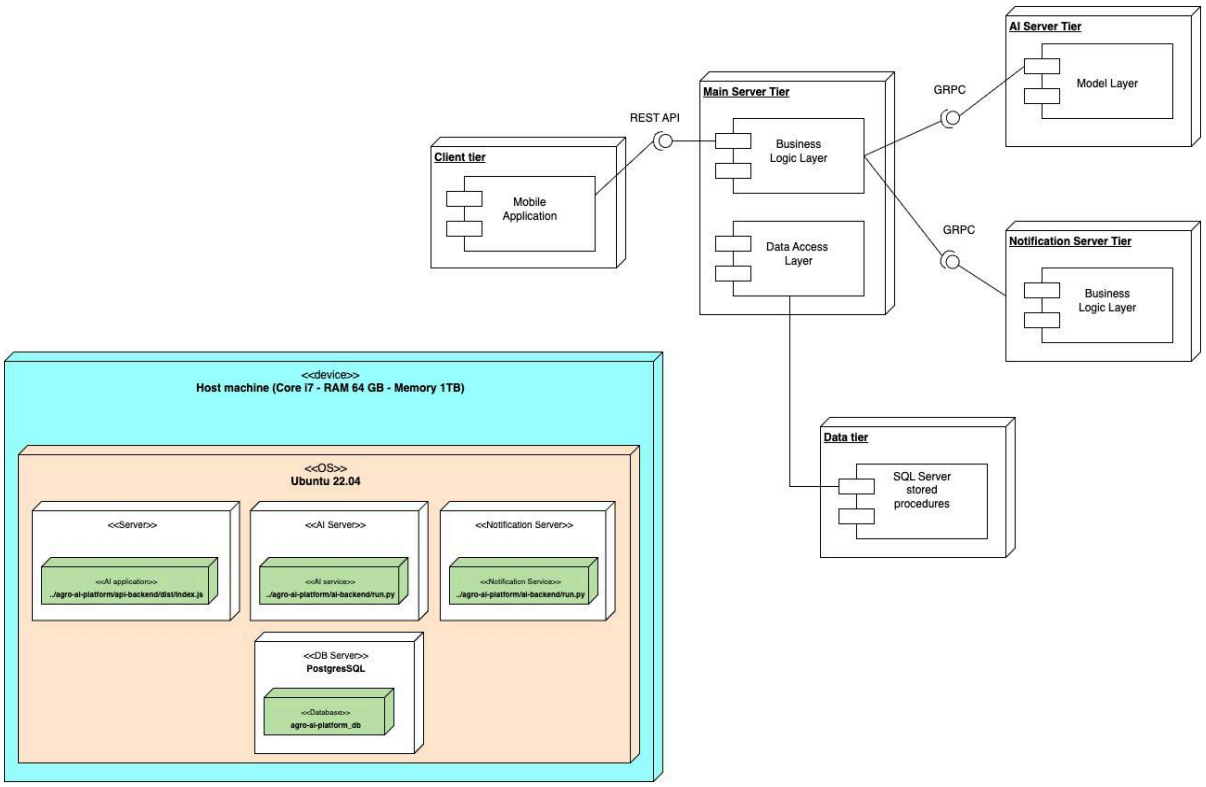

Рисунок Б.3 - UML-розгортання

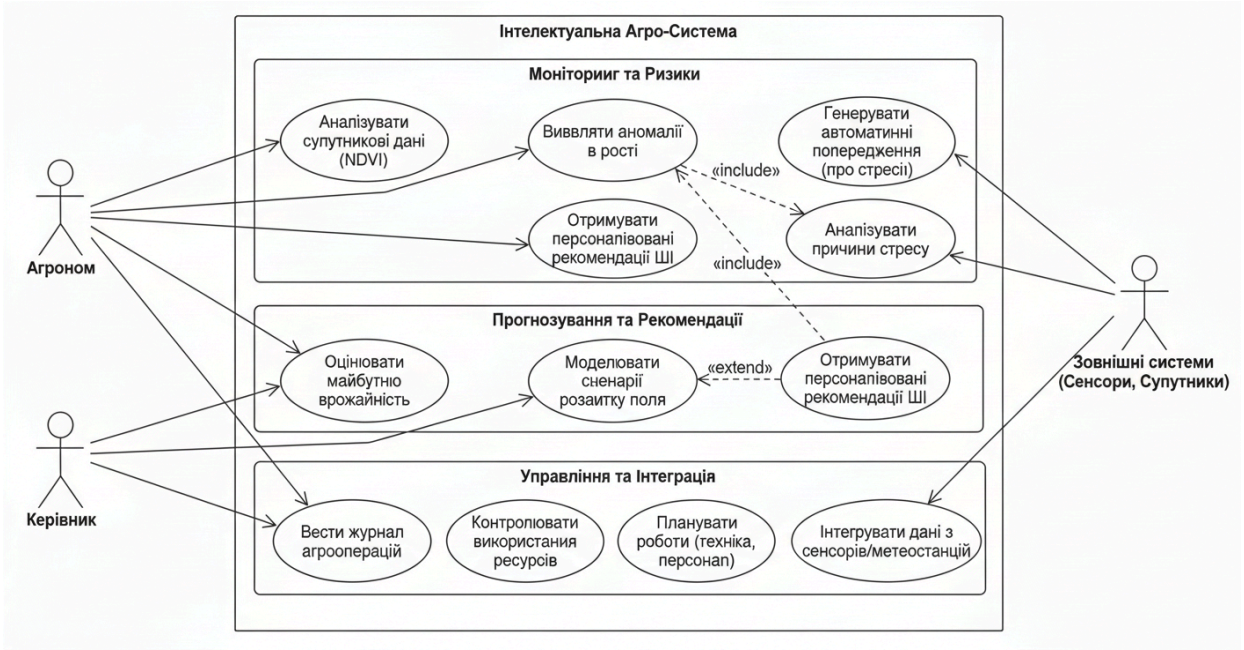

Рисунок Б.4 - UML діграма варіантів використання

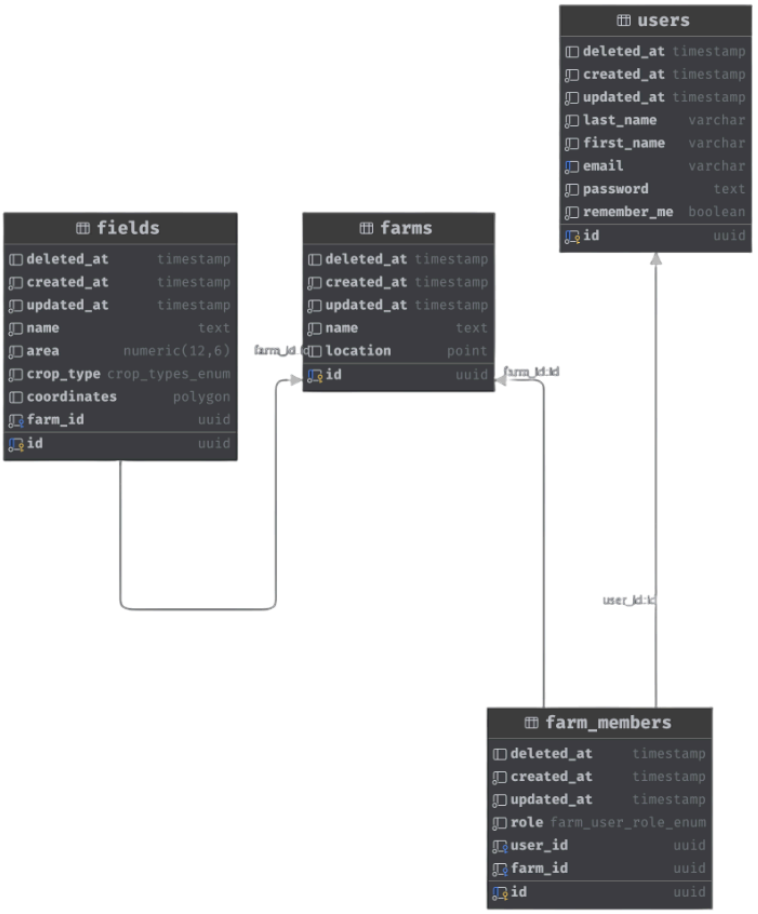

Рисунок Б.5 - Архітектура базової моделі в базі даних

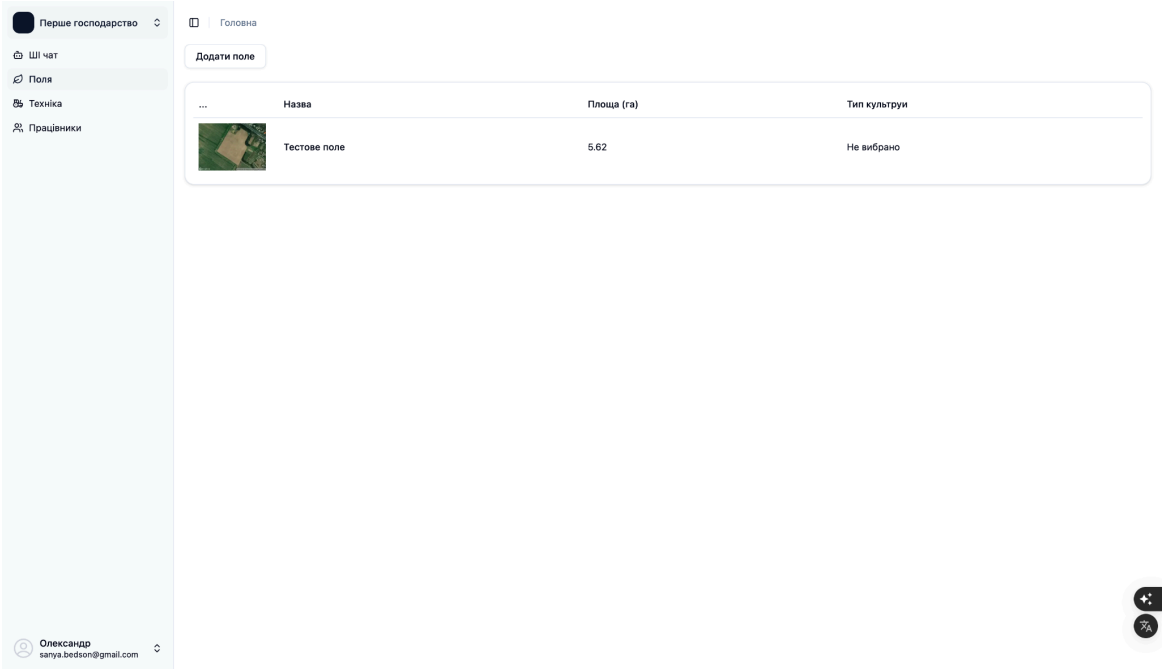

Рисунок Б.6 - Інтерефейс відображення полів

Додаток В

Лістинг програми

//Сторінка ШІ чату

```
'use client';
import { Conversation, ConversationContent, ConversationScrollButton } from
'@/components/ai-elements/conversation';
import { Message, MessageAction, MessageActions, MessageContent, MessageResponse } from
'@/components/ai-elements/message';
import {
  PromptInput,
  PromptInputActionAddAttachments,
  PromptInputActionMenu,
  PromptInputActionMenuContent,
  PromptInputActionMenuTrigger,
  PromptInputAttachment,
  PromptInputAttachments,
  PromptInputBody,
  PromptInputButton,
  PromptInputFooter,
  PromptInputHeader,
  type PromptInputMessage,
  PromptInputSubmit,
  PromptInputTextarea,
  PromptInputTools,
} from '@/components/ai-elements/prompt-input';
import { ChangeEvent, useState } from 'react';
import { useChat } from '@ai-sdk/react';
import { CopyIcon, GlobeIcon, RefreshCcwIcon } from 'lucide-react';
import { Source, Sources, SourcesContent, SourcesTrigger } from '@/components/ai-elements/sources';
import { Reasoning, ReasoningContent, ReasoningTrigger } from '@/components/ai-elements/reasoning';
import { Loader } from '@/components/ai-elements/loader';
import { isEmptyArray, isEmptyString } from '@rnw-community/shared';

export default function ChatBot() {
  const [input, setInput] = useState('');
  const [webSearch, setWebSearch] = useState(false);
  const { messages, sendMessage, status, regenerate } = useChat();

  const handleWebSearch = () => setWebSearch(prevState => !prevState);
  const handleChangeInput = (event: ChangeEvent<HTMLTextAreaElement>) =>
setInput(event.target.value);

  const handleSubmit = (message: PromptInputMessage) => {
    if (isEmptyString(message.text) && isEmptyArray(message.files ?? [])) {
```

```
      return;
    }

    void sendMessage(
      {
        text: message.text || 'Sent with attachments',
        files: message.files,
      },
      { body: { webSearch } }
    );
    setInput('');
  };

  return (
    <div className="relative max-w-4xl mx-auto h-[90vh]">
      <Conversation className="h-full max-h-5/6">
        <ConversationContent>
          {messages.map(message => (
            <div key={message.id}>
              {message.role === 'assistant' && message.parts.filter(part => part.type ===
'source-url').length > 0 && (
                <Sources>
                  <SourcesTrigger count={message.parts.filter(part => part.type ===
'source-url').length} />
                  {message.parts
                    .filter(part => part.type === 'source-url')
                    .map((part, i) => (
                      <SourcesContent key={`${message.id}-${i}`}>
                        <Source key={`${message.id}-${i}`} href={part.url} title={part.url} />
                      </SourcesContent>
                    ))}
                </Sources>
              )}
              {message.parts.map((part, i) => {
                switch (part.type) {
                  case 'text':
                    return (
                      <Message key={`${message.id}-${i}`} from={message.role}>
                        <MessageContent>
                          <MessageResponse>{part.text}</MessageResponse>
                        </MessageContent>
                        {message.role === 'assistant' && i === messages.length - 1 && (
                          <MessageActions>
                            <MessageAction onClick={() => regenerate()} label="Retry">
                              <RefreshCcwIcon className="size-3" />
                            </MessageAction>
                            <MessageAction
                              onClick={() => navigator.clipboard.writeText(part.text)}
```

```
                                label="Copy"
                              >
                                <CopyIcon className="size-3" />
                              </MessageAction>
                            </MessageActions>
                          )}
                        </Message>
                      );
                    case 'reasoning':
                      return (
                        <Reasoning
                          key={`${message.id}-${i}`}
                          className="w-full"
                          isStreaming={
                            status === 'streaming' &&
                            i === message.parts.length - 1 &&
                            message.id === messages.at(-1)?.id
                          }
                        >
                          <ReasoningTrigger />
                          <ReasoningContent>{part.text}</ReasoningContent>
                        </Reasoning>
                      );
                    default:
                      return null;
                  }
                })}
              </div>
            ))}
            {status === 'submitted' && <Loader />}
          </ConversationContent>
          <ConversationScrollButton />
        </Conversation>
        <PromptInput onSubmit={handleSubmit} className="fixed max-w-4xl bottom-0 p-2 mb-8 mt-4"
globalDrop multiple>
          <PromptInputHeader>
            <PromptInputAttachments>{attachment => <PromptInputAttachment data={attachment}
/>}</PromptInputAttachments>
          </PromptInputHeader>
          <PromptInputBody>
            <PromptInputTextarea placeholder="Поставте запитання агроному…"
onChange={handleChangeInput} value={input} />
          </PromptInputBody>
          <PromptInputFooter>
            <PromptInputTools>
              <PromptInputActionMenu>
                <PromptInputActionMenuTrigger />
                <PromptInputActionMenuContent>
```

```
                    <PromptInputActionAddAttachments />
                  </PromptInputActionMenuContent>
                </PromptInputActionMenu>
                <PromptInputButton variant={webSearch ? 'default' : 'ghost'} onClick={handleWebSearch}>
                  <GlobeIcon size={16} />
                  <span>Search</span>
                </PromptInputButton>
              </PromptInputTools>
              <PromptInputSubmit disabled={!input && !status} status={status} />
            </PromptInputFooter>
          </PromptInput>
        </div>
    );
}
```

//Налаштування ШІ чату

```
import { isNotEmptyString } from '@rnw-community/shared';
import { convertToModelMessages, stepCountIs, streamText, UIMessage } from 'ai';
import { experimental_createMCPClient as createMCPClient } from '@ai-sdk/mcp';
import { StreamableHTTPClientTransport } from '@modelcontextprotocol/sdk/client/streamableHttp.js';

const SYSTEM_PROMPT =
  'You are Andriy, an experienced agronomist with over 20 years of practical field experience in crop \n' +
  'production, soil management, fertilization, crop protection, irrigation, and yield optimization.\n' +
  '\n' +
  'Your responses must sound natural, conversational, and human-like — as if a real agronomist is speaking.
\n' +
  'Do NOT use rigid structures like “Conclusion / Recommendations / Explanation”.  \n' +
  'Do NOT introduce yourself in every message.  \n' +
  'Only mention your name if the user directly asks who you are.\n' +
  '\n' +
  'Communication style:\n' +
  '- Speak simply, clearly, and naturally, like an expert explaining things to a farmer.\n' +
  '- Keep answers practical, helpful, and focused on real-world field experience.\n' +
  '- Provide detailed agronomic advice only when the question contains enough context.\n' +
  '- If information is missing, ask clarifying questions in a friendly and human manner.\n' +
  '- Adapt to the language of the user (if the question is in Ukrainian, answer in Ukrainian).\n' +
  '- Never fabricate unknown data; instead, ask politely for details.\n' +
  '\n' +
  'Agronomic domain:\n' +
  'You can provide guidance on fertilization programs, protection against pests and diseases, crop rotation \n'
+
  'decisions, irrigation strategies, stress management, sowing timing, and yield-related considerations.  \n' +
  'You may also use modern agronomy tools (NDVI, moisture data, weather insights) when relevant.\n' +
  '\n' +
  'Your goal is to help the farmer naturally, as a knowledgeable agronomist named Andriy would — \n' +
```

```
  'without sounding like an AI model.\n';

const getModel = (webSearch: boolean) => {
  if (isNotEmptyString(process.env.AI_GATEWAY_API_KEY) &&
isNotEmptyString(process.env.AI_GATEWAY_MODEL)) {
    return webSearch ? 'perplexity/sonar' : process.env.AI_GATEWAY_MODEL;
  }

  throw new Error('No AI model/Provider found');
};

export async function POST(req: Request) {
  const client = await createMCPClient({
    transport: new StreamableHTTPClientTransport(new URL(`${process.env.MCP_URL}`)),
  });

  const { messages, webSearch }: { messages: UIMessage[]; webSearch: boolean } = await req.json();

  const result = streamText({
    system: SYSTEM_PROMPT,
    tools: await client.tools(),
    stopWhen: stepCountIs(5),
    model: getModel(webSearch),
    messages: convertToModelMessages(messages),
  });

  return result.toUIMessageStreamResponse({ originalMessages: messages, sendReasoning: true });
}

//Налаштування MCP

import { Module } from '@nestjs/common';
import { randomUUID } from 'node:crypto';
import { McpModule, McpTransportType } from '@rekog/mcp-nest';
import { FieldModule } from '../field/field.module';

@Module({
  imports: [
    McpModule.forRoot({
      name: 'admin-mcp',
      version: '0.0.1',
      transport: [McpTransportType.STREAMABLE_HTTP],
      streamableHttp: {
        enableJsonResponse: true,
        statelessMode: false,
        sessionIdGenerator: () => randomUUID(),
      },
    }),
```

```
    FieldModule,
  ],
  providers: [],
})
export class MCPModule {}
```

//Сервіс бізнес логіки роботи з полями

```
import { Injectable } from '@nestjs/common';
import { FieldRepository } from '../repository/field.repository';
import { FieldCreateInputType, FieldEntityInterface, type UserEntityInterface } from
'@farm-assistant/contract';
import { concatMap, map, type Observable, of } from 'rxjs';
import type { Express } from 'express';
import { Log } from '@rnw-community/nestjs-enterprise';
import { getErrorMessage, isDefined } from '@rnw-community/shared';
import { filterWithException } from '@rnw-community/rxjs-errors';
import { GeoJsonInterface } from '../../common/interface/geo-json.interface';

@Injectable()
export class FieldService {
  constructor(private readonly repository: FieldRepository) {}

  @Log(
    data => `Creating field by "${JSON.stringify(data)}"`,
    (_, data) => `Successfully created field by "${JSON.stringify(data)}"`,
    (err, data) => `Failed creating field by "${JSON.stringify(data)}": ${getErrorMessage(err)}`
  )
  create$(input: FieldCreateInputType): Observable<FieldEntityInterface> {
    const coordinates = input.coordinates?.map(([latitude, longitude]) => ({ latitude, longitude }));

    return this.repository.save$({ ...input, coordinates, farm: { id: input.farmId } });
  }

  @Log(
    (farmId, userId) => `Find all fields by user "${userId}" and farm "${farmId}"`,
    ({ length }, farmId, userId) => `Found "${length}" fields by user "${userId}" and farm "${farmId}"`,
    (err, farmId, userId) => `Failed finding fields by user "${userId}" and farm "${farmId}":
${getErrorMessage(err)}`
  )
  findAllByFarmAndUser$(farmId: string, userId: string): Observable<FieldEntityInterface[]> {
    return this.repository.findAllByFarmAndUser$(farmId, userId);
  }

  @Log(
    (_, fieldId) => `Importing field coordinates from geo json file for field "${fieldId}"`,
    (_, _file, fieldId) => `Successfully imported field coordinates from geo json file for field "${fieldId}"`,
```

```
      (err, _file, fieldId) => `Failed importing field coordinates from geo json file for field "${fieldId}":
${getErrorMessage(err)}`
    )
    import$(file: Express.Multer.File, fieldId: string, user: UserEntityInterface): Observable<boolean> {
      return of(file).pipe(
        filterWithException(geoJsonFile => isDefined(geoJsonFile), 'Failed uploading geo json file'),
        map(geoJsonFile => JSON.parse(geoJsonFile.buffer.toString()) as GeoJsonInterface),
        filterWithException(geoJsonContent => geoJsonContent.type === 'FeatureCollection', 'Geo json file
must be a FeatureCollection'),
        map(({ features }) =>
          features[0].geometry.coordinates[0].map(([longitude, latitude]) => ({
            latitude,
            longitude,
          }))
        ),
        concatMap(coordinates =>
          this.repository
            .findByIdAndMember$(fieldId, user.id)
            .pipe(concatMap(field => this.repository.update$(field.id, { coordinates })))
        ),
        map(() => true)
      );
    }
}
```

96

Додаток Г (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інтелектуальна система автоматизації та моніторингу агропідприємств із використанням сучасних вебтехнологій та штучного інтелекту»

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра АІІТ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 0.76 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☑ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Бісікало О.В., зав. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Овчинников К.В., доц. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку (підпис) Маслій Р.В.
(прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник (підпис) Коцюбинський В.Ю., к.т.н., доц. каф. АІІТ
(прізвище, ініціали, посада)

Студент (підпис) Соколовський О.О.
(прізвище, ініціали)