ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»
Факультет автоматизації виробництва, інформаційних
та управлінських технологій
Кафедра інформаційних технологій та аналітики даних

«Допущено до захисту»
Гарант ОПП
Ірина ГЕТЬМАН

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

за підсумками виконання
освітньо-професійної програми
«Комп'ютерні науки»
за спеціальністю 122 Комп`ютерні науки

на тему «Програмно-методичний комплекс для інформаційного забезпечення системи точного землеробства та підтримки агротехнічних рішень»

Керівник роботи
Світлана ГУРКОВСЬКА

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач
Андрій ПУШКА

Підсумкова оцінка за атестацію			

Голова ЕК
Антон КУДРЯВЦЕВ

ЗАПОРІЖЖЯ 2026

ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»

Факультет	автоматизації виробництва, інформаційних та управлінських технологій
Кафедра	інформаційних технологій та аналітики даних
Ступінь вищої освіти	бакалавр
Спеціальність	122 Комп'ютерні науки
ОПП	Комп'ютерні науки

ЗАТВЕРДЖУЮ
Гарант ОПП

_____ Ірина ГЕТЬМАН

«26» лютого 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Пушки Андрія Віталійовича
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи *«Програмно-методичний комплекс для інформаційного забезпечення системи точного землеробства та підтримки агротехнічних рішень»*
керівник роботи *Гурковська Світлана Сергіївна,*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету від *23.02.2026 р. №41/23.02.2026*
2. Термін подання роботи *16.06.2026 р.*
3. Вихідні дані до роботи *Навчальна література, державні стандарти та методичні вказівки з дисциплін «Інженерія програмного забезпечення», «Бази даних», «Веб-технології», «Геоінформаційні системи»; документація з розроблення програмного забезпечення на Java та платформі Spring Boot; наукові публікації з тематики точного землеробства та систем підтримки прийняття рішень; агрохімічні нормативи та методики розрахунку норм мінеральних добрив; результати власного проєктування, тестування та економічного обґрунтування розробленого програмного комплексу.*
4. Зміст пояснювальної записки (перелік питань) *Реферат. Зміст. Вступ. Розділ 1. Аналіз предметної області та обґрунтування технологічних рішень. Розділ 2. Моделювання та реалізація системи. Розділ 3. Проєктування програмного комплексу для підтримки агротехнічних рішень. Розділ 4. Економічні розрахунки. Висновки. Перелік посилань. Додатки.*
5. Перелік графічного (демонстраційного) матеріалу (з точним зазначенням обов'язкових креслень): *Актуальність, мета, об'єкт, предмет і завдання дослідження; порівняльний аналіз програмних аналогів (FMIS); архітектура веб-орієнтованої системи; ER-діаграма структури бази даних; діаграми UML (прецедентів, класів, послідовностей); специфікація REST-API; блок-схема алгоритму розрахунку норм добрив за принципом Лібіха; інтерфейси користувача та інтерактивна карта полів; результати економічних розрахунків; висновки до роботи.*

6. Консультанти по роботі, із зазначенням розділів роботи, що стосуються їх.

Розділ	Прізвище, ініціали та посада консультанта
1	Гурковська С.С., каф. ІНТЕХАД
2	Гурковська С.С., каф. ІНТЕХАД
3	Гурковська С.С., каф. ІНТЕХАД
4	Гетьман І.А., доц. каф. ІНТЕХАД

7. Дата видачі завдання *26.02.2026*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи
1	Розділ 1. Аналіз предметної області та обґрунтування технологічних рішень	26.02.2026 - 15.03.2026
2	Розділ 2. Моделювання та реалізація системи	16.03.2026 - 20.04.2026
3	Розділ 3. Проєктування програмного комплексу для підтримки агротехнічних рішень	21.04.2025 – 15.05.2025
5	Розділ 4. Економічні розрахунки	16.05.2025 - 25.05.2025
6	Висновки, перелік посилань, вступ, зміст, реферат	26.05.2025 – 05.06.2025
7	Подання завершеної роботи. Перевірка на академічний плагіат	06.06.2025 – 16.06.2026
8	Остаточне оформлення роботи, презентаційного матеріалу	16.06.2026 – 18.06.2026
9	Рецензування завершеної роботи. Захист	18.06.2026 – 23.06.2026

Здобувач (Андрій ПУШКА)

Керівник роботи (Світлана ГУРКОВСЬКА)

РЕФЕРАТ

Пушка А.В. Програмно-методичний комплекс для інформаційного забезпечення системи точного землеробства та підтримки агротехнічних рішень.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавра комп'ютерних наук за спеціальністю 122 «Комп'ютерні науки». – ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА» МОН України, м. Запоріжжя, 2026.

Метою роботи є підвищення ефективності інформаційного забезпечення агротехнічних рішень у системах точного землеробства шляхом розроблення веб-орієнтованого програмно-методичного комплексу для управління полями, ведення агрохімічних аналізів, розрахунку норм мінеральних добрив та планування сівозміни.

Об'єкт дослідження – процес інформаційного забезпечення агротехнічних рішень у системах точного землеробства.

Предмет дослідження – методи, алгоритми та програмні засоби підтримки прийняття рішень при оптимізації мінерального живлення рослин та плануванні сівозміни.

Методологія – об'єктно-орієнтований аналіз і проєктування з використанням UML, реляційне моделювання даних, агрохімічний метод розрахунку норм добрив за принципом лімітуючого фактора Лібіха, методи програмної інженерії та модульного тестування.

У роботі розроблено веб-орієнтований програмно-методичний комплекс «АгроСистема» – систему підтримки прийняття рішень для точного землеробства. Реалізовано серверну частину на стеку Java 21 та Spring Boot 3 з базою даних PostgreSQL 16, REST-API, інтерактивну ГІС-підсистему на Leaflet.js і Turf.js, а також модулі розрахунку мінеральних добрив і рекомендацій сівозміни. Коректність роботи підтверджено набором модульних та інтеграційних тестів.

Розроблений комплекс дозволяє оптимізувати внесення мінеральних добрив, зменшити витрати господарства та підвищити обґрунтованість агрономічних рішень. Систему орієнтовано на малі та середні фермерські господарства; вона придатна до масштабування та подальшого розширення функціоналу.

КЛЮЧОВІ СЛОВА: ТОЧНЕ ЗЕМЛЕРОБСТВО, СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, МІНЕРАЛЬНІ ДОБРИВА, ЗАКОН ЛІБІХА, СІВОЗМІНА, ГЕОІНФОРМАЦІЙНА СИСТЕМА, SPRING BOOT, POSTGRESQL, REST-API, ВЕБ-ЗАСТОСУНОК

ABSTRACT

Pushka A.V. Software and Methodological Complex for Information Support of a Precision Farming System and Agronomic Decision-Making.

Qualification work for obtaining the degree of Bachelor of Computer Science in the specialty 122 Computer Science. – LLC «TECHNICAL UNIVERSITY «METINVEST POLYTECHNIC» MES of Ukraine, Zaporizhzhia, 2026.

The aim of this work is to improve the efficiency of information support for agronomic decisions in precision farming systems through the development of a web-based software and methodological complex for field management, agrochemical analysis recording, mineral fertilizer rate calculation, and crop rotation planning.

Object of the research – the process of information support for agronomic decisions in precision farming systems.

Subject of the research – the methods, algorithms, and software tools for decision support in optimizing plant mineral nutrition and crop rotation planning.

Methodology – object-oriented analysis and design using UML, relational data modeling, the agrochemical method of fertilizer rate calculation based on Liebig's law of the limiting factor, software engineering, and unit testing.

The work presents the development of the web-based software and methodological complex “AgroSystem” – a decision support system for precision farming. The server side is implemented using Java 21 and Spring Boot 3 with a PostgreSQL 16 database, a REST API, an interactive GIS subsystem based on Leaflet.js and Turf.js, and modules for mineral fertilizer calculation and crop rotation recommendations. The correctness of the system is confirmed by a set of unit and integration tests.

The developed complex allows optimizing mineral fertilizer application, reducing farm costs, and improving the validity of agronomic decisions. It is intended for small and medium-sized farms and is scalable and suitable for further functional expansion.

KEYWORDS: PRECISION FARMING, DECISION SUPPORT SYSTEM, MINERAL FERTILIZERS, LIEBIG'S LAW, CROP ROTATION, GEOGRAPHIC INFORMATION SYSTEM, SPRING BOOT, POSTGRESQL, REST-API, WEB APPLICATION

ЗМІСТ

ВСТУП 6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ТЕХНОЛОГІЧНИХ РІШЕНЬ 9
1.1 Огляд галузі та актуальність задачі 9
1.2 Аналіз бізнес-процесів фермерського господарства 10
1.3 Актор системи – Фермер: ролі та функції 12
1.4 Порівняльний аналіз програмних аналогів (FMIS) 13
1.5 Обґрунтування технологічного стеку 15
1.6 Постановка задачі розробки 17
2 МОДЕЛЮВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ 19
2.1 Архітектура веб-орієнтованої системи 19
2.2 Проектування моделі даних та схеми бази даних 20
2.3 REST-API: специфікація реалізованих ендпоінтів 22
2.3.1 Контролер FarmerController 23
2.3.2 Контролер FieldController 24
2.3.3 Контролер SoilAnalysisController 24
2.3.4 Контролер CalculatorController 25
2.3.5 Контролер RotationController 25
2.4 Алгоритм розрахунку норм мінеральних добрив за принципом Лібіха 26
2.4.1 Крок 1. Виведення нормативів виносу та максимальної врожайності 26
2.4.2 Крок 2. Розрахунок запасу елементів у ґрунті 27
2.4.3 Крок 3. Потенціал урожаю та лімітуючий фактор (Лібіх) 28
2.4.4 Крок 4. Потреба та дефіцит діючої речовини 28
2.4.5 Крок 5. Перерахунок у фізичну масу добрива 29
2.4.6 Крок 6. Дробове внесення азоту 29
2.5 Алгоритм рекомендацій сівозміни 30
2.6 Інтерактивна ГІС-підсистема на базі Leaflet.js та Turf.js 31
2.6.1 Ініціалізація карти та шарів 32
2.6.2 Малювання контурів полів (Leaflet.draw) 32
2.6.3 Ручне введення координат 33
2.6.4 Відображення наявних полів на головній карті 33
2.7 Модель варіантів використання (Use Case) 34
2.7.1 Сценарій UC-04 «Додавання поля з малюванням на карті» 35
2.7.2 Сценарій UC-10 «Розрахунок норм добрив» 35
2.8 Діаграми діяльності (Activity) 36
2.8.1 AD-01: Додавання поля з GeoJSON-геометрією 36

2.8.2 AD-02: Розрахунок норм добрив 36
2.8.3 AD-03: Рекомендація сівозміни 37
3 ПРОЕКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ ПІДТРИМКИ АГРОТЕХНІЧНИХ РІШЕНЬ 38
3.1 Розробка діаграми прецедентів 38
3.1.1 Актори та межі системи 39
3.1.2 Опис прецедентів і відношень include/extend 41
3.2 Розробка діаграми класів 42
3.2.1 Класи-сутності предметної області 43
3.2.2 Контролери, репозиторії та зв'язки між класами 44
3.3 Розробка діаграми послідовностей 46
3.3.1 Сценарій розрахунку норм добрив 46
3.3.2 Сценарій отримання рекомендації сівозміни 48
3.4 Розробка бази даних 49
3.4.1 ER-діаграма та опис таблиць 50
3.4.2 Зв'язки, ключі та нормалізація 52
3.5 Програмна реалізація комплексу 53
3.5.1 Технологічний стек і структура проєкту 53
3.5.2 Реалізація серверної частини 54
3.5.3 Реалізація алгоритмів підтримки рішень 57
3.5.4 Клієнтська частина та інтерфейс користувача 58
3.6 Розробка модульних тестів 59
3.7 Демонстрація роботи програмного комплексу 61
3.8 Результат проектування програмного комплексу 65
4 ЕКОНОМІЧНІ РОЗРАХУНКИ 66
4.1 Визначення трудомісткості розроблення програмного продукту 66
4.2 Розрахунок витрат на оплату праці розробника 68
4.3 Відрахування на соціальні заходи 69
4.4 Розрахунок витрат на програмне забезпечення та хмарні сервіси 70
4.5 Розрахунок витрат на електроенергію 72
4.6 Амортизація обладнання 72
4.7 Накладні витрати 73
4.8 Кошторис витрат на розроблення 73
4.9 Оцінка економічної ефективності впровадження 75
4.10 Результати економічних розрахунків 78
ВИСНОВКИ 80
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 84
ДОДАТОК А. ВІДОМОСТІ РОБОТИ 87
ДОДАТОК Б. ГЛОСАРІЙ 88
ДОДАТОК В. ТЕХНІЧНЕ ЗАВДАННЯ 90
ДОДАТОК Г. UML ДІАГРАМИ 94

ВСТУП

Агропромисловий комплекс України є однією з базових галузей національної економіки, що забезпечує продовольчу безпеку держави та формує значну частку її експортного потенціалу. За умов глобальних кліматичних змін і зростання вартості агровиробничих ресурсів особливої актуальності набуває впровадження цифрових технологій, здатних суттєво підвищити ефективність використання кожного гектара орних земель.

Концепція точного землеробства (Precision Agriculture, PA) ґрунтується на диференційованому підході до управління агровиробництвом: замість однакового обробітку всього поля вона передбачає облік просторової та часової мінливості агрохімічних показників ґрунту, метеорологічних умов і біологічних вимог культур. Практична реалізація PA неможлива без відповідного програмно-методичного забезпечення – систем підтримки прийняття рішень (СППР).

Незважаючи на існування численних зарубіжних рішень класу Farm Management Information System (FMIS), їхнє практичне застосування в умовах вітчизняного агровиробництва ускладнюється через недостатню адаптацію до українських агрохімічних нормативів, мовний бар’єр, обмеження доступу в умовах воєнного часу та надмірно високу вартість ліцензій для малого та середнього агробізнесу.

У ході кваліфікаційної роботи автором була виконана повна програмна реалізація прототипу веб-орієнтованої системи «АгроСистема» – СППР для точного землеробства. Система побудована як монолітний веб-застосунок на стеку Java 21 / Spring Boot 3 / Spring Data JPA / PostgreSQL з односторінковим клієнтом (SPA) на чистому JavaScript із використанням бібліотек Bootstrap 5, Leaflet.js 1.9,

Leaflet.draw і Turf.js. Жодних десктопних або мобільних клієнтів не передбачено: весь функціонал доступний у браузері.

Метою кваліфікаційної роботи є дослідження предметної галузі точного землеробства, аналіз бізнес-процесів фермерського господарства, проектування архітектури веб-орієнтованого програмно-методичного комплексу та програмна реалізація ключових модулів – управління полями, ведення агрохімічних аналізів, розрахунок норм мінеральних добрив за принципом лімітуючого фактора Лібіха, а також рекомендацій з сівозміни.

Завданнями кваліфікаційної роботи є:

– провести аналіз актуальних технологій та методів точного землеробства;

– дослідити бізнес-процеси управління фермерським господарством;

– визначити функції, ролі та сценарії використання актора Фермер;

– виконати порівняльний аналіз програмних аналогів класу FMIS;

– обґрунтувати вибір технологічного стеку для веб-орієнтованої архітектури;

– спроектувати реляційну модель даних та схему бази даних;

– реалізувати REST-API на основі Spring Boot для всіх ключових сценаріїв;

– реалізувати клієнтську частину з інтеграцією Leaflet.js та Turf.js;

– реалізувати алгоритм розрахунку норм добрив за принципом Лібіха;

– реалізувати алгоритм рекомендацій сівозміни;

– розробити діаграми прецедентів, класів та діяльності за фактичним кодом.

Об’єктом дослідження є процес інформаційного забезпечення агротехнічних рішень у системах точного землеробства. Предметом дослідження – методи, алгоритми та програмні засоби підтримки прийняття рішень при оптимізації мінерального живлення рослин та плануванні сівозміни.

Практичне значення роботи полягає в тому, що створено працездатний прототип веб-орієнтованої системи підтримки прийняття рішень “АгроСистема” з повним стеком (база даних PostgreSQL, REST-сервер на Spring Boot та односторінковий клієнт), який реалізує науково обґрунтований розрахунок норм мінеральних добрив за законом мінімуму Лібіха, облік полів і агрохімічних аналізів ґрунту та формування рекомендацій із сівозміни. Система має україномовний інтерфейс, адаптований до вітчизняних агрохімічних нормативів, і може бути використана як доступний інструмент цифровізації передусім малими та середніми фермерськими господарствами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ТЕХНОЛОГІЧНИХ РІШЕНЬ

1.1 Огляд галузі та актуальність задачі

Точне землеробство (Precision Agriculture, PA) – це комплекс технологій управління агровиробництвом, що об’єднує геоінформаційні системи (ГІС), глобальні навігаційні супутникові системи (ГНСС), дистанційне зондування Землі (ДЗЗ) та інструменти аналізу даних для оптимізації виробничих процесів з урахуванням просторової неоднорідності сільськогосподарських угідь.

Основний принцип PA – «правильний ресурс у правильному місці у правильний час» – передбачає відмову від усередненого підходу «на гектар» на користь диференційованого управління. Такий підхід дозволяє одночасно підвищити врожайність і знизити виробничі витрати.

Згідно з аналітичними оцінками ринку агротехнологій, лише 12–15% українських агропідприємств застосовують елементи точного землеробства на системній основі, тоді як у країнах ЄС цей показник перевищує 40%. Головними бар’єрами для впровадження PA в Україні є: відсутність доступних україномовних програмних рішень, адаптованих до вітчизняних нормативів; висока вартість ліцензійних зарубіжних платформ; недостатня цифрова грамотність персоналу.

Ключовими перевагами впровадження PA-технологій є:

– зменшення витрат на мінеральні добрива та засоби захисту рослин на 15–20% завдяки точковому та диференційованому їх внесенню;

– підвищення врожайності сільськогосподарських культур на 8–12% через оптимізацію живлення рослин відповідно до реального стану ґрунту;

– зниження екологічного навантаження на ґрунтові води та атмосферу шляхом запобігання надлишковому використанню агрохімікатів;

– забезпечення цифрового документування всіх агрозаходів, що спрощує агрономічний контроль;

– підвищення прозорості виробничих процесів для інвесторів та страхових компаній.

1.2 Аналіз бізнес-процесів фермерського господарства

У ході аналізу бізнес-процесів фермерського господарства виокремлено п’ять ключових процесів, які фактично автоматизуються розробленою системою «АгроСистема» (Додаток Г, рис. Г.1).

Перший бізнес-процес – «Реєстрація фермера та автентифікація». Реалізований у класі FarmerController через ендпоінти POST /api/farmers/register та POST /api/farmers/login. Дані фермера (ім’я, прізвище, e-mail, телефон, пароль) зберігаються у таблиці farmers.

Другий бізнес-процес – «Реєстрація поля та формування його геометрії». Фермер креслить контур поля на інтерактивній карті засобами Leaflet.draw або вводить координати вершин полігона у текстовому вигляді. Сформований GeoJSON-об’єкт зберігається як TEXT-поле сутності Field. Площа автоматично обчислюється на стороні браузера бібліотекою Turf.js (метод turf.area) і записується у поле area (гектари).

Третій бізнес-процес – «Введення агрохімічного аналізу ґрунту». Для кожного поля фермер створює один або кілька записів аналізу: дата, рухомі форми N, P, K (мг/кг), pH, а також інформація про поточну культуру (вирощується / попередник / пар). Зберігається у таблиці soil_analyses через POST /api/analyses/field/{fieldId}. Запис нового аналізу одночасно оновлює стан поля (поля lastCrop, currentCrop, growthStage) залежно від значення cropState.

Четвертий бізнес-процес – «Розрахунок норм внесення мінеральних добрив». На основі найновішого агрохімічного аналізу поля та обраної фермером культури розраховуються прогнозований урожай (за принципом лімітуючого фактора Лібіха), потреба у фізичній масі азотного, фосфорного та калійного добрив, а також план дробового внесення азоту. Реалізовано у класі CalculatorController, ендпоінт POST /api/calculator/calculate.

П’ятий бізнес-процес – «Рекомендація культури-наступниці у сівозміні». На основі останньої вирощуваної на полі культури система видає 2–3 рекомендації культури-наступниці з якісною оцінкою та пояснювальним текстом. Реалізовано у RotationController, ендпоінт GET /api/rotation/recommend?lastCrop={code}. Окремо генерується попередження про обов’язкову паузу повернення для соняшнику (7 років).

Таблиця 1.1 – Бізнес-процеси та їхня реалізація у системі «АгроСистема»

Бізнес-процес	Вхідні дані	Результат	Ендпоінт / клас
Реєстрація / вхід	firstName, email, password	JSON фермера + ідентифікатор	FarmerController
Реєстрація поля	Назва, GeoJSON-контур	Запис у таблиці fields	POST /api/fields

Продовження таблиці 1.1

Бізнес-процес	Вхідні дані	Результат	Ендпоінт / клас
Введення аналізу	Дата, NPK, pH, культура	Запис у soil_analyses	POST /api/analyses/field/{id}
Розрахунок добрив	fieldId, cropType	Норми N, P, K + поради	POST /api/calculator/calculate
Сівозміна	lastCrop	Список 2–3 рекомендацій	GET /api/rotation/recommend

1.3 Актор системи – Фермер: ролі та функції

На поточному етапі розробки в системі «АгроСистема» визначено одного основного актора – Фермер (Агроном). Це проектне рішення зумовлене тим, що на першому MVP-етапі (Minimum Viable Product) система призначена для малих та середніх фермерських господарств, де одна людина поєднує функції агронома, технолога й керівника. У реалізованому коді саме Фермер є єдиною сутністю-користувачем (Farmer), яка авторизується у системі та володіє полями.

Атрибути актора Фермер (паспорт актора):

– Назва ролі: Фермер / Агроном;

– Тип: Первинний актор (ініціює сценарії);

– Технічна грамотність: середня (впевнений користувач ПК та смартфону);

– Частота використання: щоденна або кілька разів на тиждень у сезон;

– Критичні потреби: точність розрахунків, зручність введення даних, швидкість доступу до карти.

Таблиця 1.2 – Функції актора Фермер та їхня реалізація

Функція	Стан реалізації у коді	Зв’язаний контролер
Реєстрація / вхід	Реалізовано	FarmerController
Додавання поля (малювання на карті, GeoJSON)	Реалізовано	FieldController
Введення координат вручну (полігон)	Реалізовано	index.html (applyManualCoords)
Видалення поля з каскадним видаленням аналізів	Реалізовано	FieldController.deleteField
Перегляд усіх своїх полів на спільній карті	Реалізовано	index.html / Leaflet
Додавання аналізу ґрунту до поля	Реалізовано	SoilAnalysisController
Розрахунок добрив за культурою	Реалізовано	CalculatorController
Збереження обраної культури за полем	Реалізовано	POST /api/calculator/save-crop
Рекомендації сівозміни	Реалізовано	RotationController
Перегляд календаря посівних робіт	Реалізовано (статика)	index.html
Каталог товарів агрохімії	Реалізовано (статика)	index.html
Перегляд погоди для регіону	Реалізовано (Open-Meteo)	index.html

1.4 Порівняльний аналіз програмних аналогів (FMIS)

Для обґрунтування доцільності розробки власного рішення проведено порівняльний аналіз репрезентативних FMIS-платформ: Climate FieldView (Bayer), Cropwise Operations (Syngenta), Soft.Farm (Україна), Agrivi (Хорватія).

Climate FieldView – одна з найбільш поширених PA-платформ у Північній Америці. Має суттєві обмеження для українського ринку: слабка адаптованість до вітчизняної нормативної бази, відсутність україномовного інтерфейсу, фактична недоступність в умовах воєнного часу та висока вартість.

Cropwise Operations – корпоративна платформа, орієнтована на агрохолдинги (від 5000 га). Вартість ліцензії та складність розгортання унеможливлюють використання малим і середнім агробізнесом.

Soft.Farm – вітчизняна платформа, що підтримує україномовний інтерфейс і є відносно доступною. Проте розрахунок норм добрив реалізований за спрощеною схемою без явного застосування закону мінімуму Лібіха.

Таблиця 1.3 – Порівняльний аналіз FMIS-платформ та розроблюваної системи

Критерій	FieldView	Cropwise	Soft.Farm	Agrivi	АгроСистема
Метод розрахунку добрив	Нормат.	Розшир.	Спрощ.	Базов.	Лібіх + потенціал
ГІС-підсистема	Повний GIS	Повний GIS	Базовий	Обмеж.	Leaflet + Turf
Рекомендації сівозміни	Є	Є	Частк.	Є	Логічний модуль
Україномовний UI	Ні	Частк.	Так	Ні	Так
Адаптація до норм UA	Ні	Частк.	Так	Ні	Так
Вартість, USD/рік	від 699	від 1200	від 120	від 500	Open Source
Тип клієнта	Web/App	Web/App	Web	Web/App	Web (SPA)

Аналіз даних таблиці 1.3 підтверджує наявність вільної ніші для open-source веб-системи з україномовним інтерфейсом, що реалізує розрахунок добрив за принципом лімітуючого фактора Лібіха та повну адаптацію до вітчизняних нормативів.

1.5 Обґрунтування технологічного стеку

Архітектурне рішення «АгроСистеми» – класичний монолітний веб-застосунок (server-rendered shell + клієнтський SPA), що спілкується з браузером через REST-API у форматі JSON. Стек обрано виходячи з академічних, освітніх та практичних міркувань.

Серверна частина побудована на Java 21 + Spring Boot 3.x. Spring Boot обрано як галузевий стандарт enterprise-розробки в Україні та світі: він забезпечує авто-конфігурацію, вбудований веб-сервер Tomcat, інтеграцію зі Spring Data JPA для роботи з ORM Hibernate, мінімум boilerplate-коду. Анотація @SpringBootApplication у класі AgroSystemApplication активує всі ці механізми одним рядком. Для скорочення повторного коду застосовується Lombok (@Data на сутностях).

Як СКБД обрано PostgreSQL 16 – реляційну, повністю безкоштовну, з потужною підтримкою індексації та транзакційності. Spring Data JPA забезпечує об’єктно-реляційне відображення сутностей у таблиці автоматично (spring.jpa.hibernate.ddl-auto=update). Геометрія полів зберігається у вигляді GeoJSON-строки у TEXT-полі сутності Field – цього достатньо для поточних обсягів (одне поле = один поліґон до десятка вершин). Перехід на PostGIS заплановано на наступний етап

розвитку кваліфікаційного проєкту, коли з’явиться потреба у просторових запитах (ST_Intersects, ST_Within тощо).

Клієнтська частина – односторінковий застосунок (SPA) у файлі static/index.html. Використано Bootstrap 5 для адаптивної сітки та стилістики, Bootstrap Icons – для іконок. Для роботи з геопросторовою інформацією залучено бібліотеку Leaflet.js 1.9 – найпопулярнішу відкриту бібліотеку інтерактивних карт у JavaScript, з плагіном Leaflet.draw 1.0 для малювання полігонів. Для розрахунку площі полігона у гектарах задіяно Turf.js 6 (метод turf.area, що повертає площу у квадратних метрах за GeoJSON-об’єктом).

Метеорологічна інформація надається через зовнішнє API Open-Meteo (https://api.open-meteo.com/v1/forecast) – безкоштовне, без обов’язкової авторизації, що видає поточну температуру та код погоди.

Принципово важливо, що розроблена «АгроСистема» є виключно веб-орієнтованим продуктом: жодного нативного мобільного або десктопного клієнта не передбачено. SPA-клієнт сумісний як зі стаціонарними браузерами, так і з мобільними (Bootstrap 5 забезпечує адаптивність), що дозволяє фермеру вносити дані безпосередньо у полі через смартфон.

Таблиця 1.4 – Обраний технологічний стек

Шар	Технологія	Версія	Призначення
Мова	Java	21	Серверна частина
Framework	Spring Boot	3.x	IoC, веб, авто-конфіг.
Persistence	Spring Data JPA / Hibernate	6.x	ORM, репозиторії
СКБД	PostgreSQL	16+	Зберігання даних
Bean-генератор	Lombok	актуальна	@Data → getters/setters
Веб-сервер	Apache Tomcat (вбудований)	10	Контейнер сервлетів

Продовження таблиці 1.4

Шар	Технологія	Версія	Призначення
Frontend UI	Bootstrap	5.3	Сітка, стилі, компоненти
Карти	Leaflet.js + Leaflet.draw	1.9 + 1.0	Інтерактивна карта
Геообчислення	Turf.js	6.x	Розрахунок площі
Погода	Open-Meteo API	v1	Поточна температура
HTTP-клієнт	Fetch API	native	REST-виклики з SPA

1.6 Постановка задачі розробки

На підставі проведеного аналізу предметної галузі, бізнес-процесів фермерського господарства та існуючих програмних аналогів сформульовано технічну постановку задачі розробки «АгроСистеми». Метою розробки є створення вільно поширюваної веб-СППР для точного землеробства, яка реалізує:

– облік фермерів та полів з підтримкою GeoJSON-геометрії, що зберігається у реляційній СКБД;

– інтерактивну веб-карту з підтримкою малювання полігонів, перемикання шарів (вулиці / супутник / рельєф) і автоматичного розрахунку площі засобами Turf.js;

– ведення кількох записів агрохімічного аналізу для кожного поля (з відстеженням поточної та попередньої культури);

– розрахунок норм мінеральних добрив за принципом лімітуючого фактора Лібіха з визначенням потенціалу врожаю, переведенням у фізичну масу аміачної селітри / суперфосфату / хлористого калію та дробовим внесенням азоту;

– видачу рекомендацій культури-наступниці у сівозміні з якісною оцінкою та попередженням про обов’язкову паузу повернення;

– статичний календар посівних робіт та каталог насіння, добрив, ЗЗР з можливістю «купити» (перехід до зовнішніх торгових майданчиків).

Технічні обмеження: веб-орієнтований клієнт (SPA, без десктопних / мобільних застосунків), серверна частина – Java 21 + Spring Boot 3.x, СКБД – PostgreSQL 16, одиниця виміру площ – гектар, одиниця виміру норм добрив – кг фізичної маси / га.

2 МОДЕЛЮВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

2.1 Архітектура веб-орієнтованої системи

Архітектура «АгроСистеми» побудована як класична трьохшарова веб-архітектура: шар представлення (Presentation), шар бізнес-логіки (Application / Controller) та шар доступу до даних (Persistence). Усі шари розгортаються в одному JVM-процесі Spring Boot, що відповідає монолітному стилю розгортання, прийнятному для прототипу.

Шар представлення складається з одного HTML-файлу static/index.html, що віддається статикою вбудованим Tomcat-сервером Spring Boot. У цьому файлі живе вся клієнтська логіка: переключення вкладок Bootstrap, ініціалізація Leaflet-карт, обчислення площі через Turf.js, віддалені виклики REST-API через fetch-API браузера, локальне збереження сесії авторизованого фермера у localStorage. Жодних шаблонізаторів (Thymeleaf, JSP, Freemarker) не задіяно – клієнт повністю автономний у відображенні.

Шар бізнес-логіки складається з п’яти REST-контролерів (анотація @RestController, базовий шлях /api/...). Кожен контролер відповідає за окрему предметну область: фермери, поля, аналізи, розрахунки, сівозміна. Бізнес-логіка обчислень (Лібіх, нормалізація доз, рекомендації) розміщена безпосередньо в контролерах – це свідомий вибір на користь простоти прототипу. У повноцінній системі такі обчислення доцільно винести в окремий сервісний шар (@Service).

Шар доступу до даних побудований на Spring Data JPA: чотири репозиторії розширюють інтерфейс JpaRepository<T, Long>, що з коробки надає методи save / findById / findAll / deleteById. Додаткові

пошукові методи описуються декларативно – досить додати сигнатуру виду List<Field> findByFarmerId(Long farmerId), і Spring сам згенерує SQL-запит з умовою WHERE farmer_id = ?.

Було розроблено архітектурну діаграму системи, яка представлена в додатку Г на рисунку Г.2.

2.2 Проектування моделі даних та схеми бази даних

Модель даних «АгроСистеми» представлена чотирма JPA-сутностями та однією допоміжною: Farmer, Field, SoilAnalysis, Laboratory та Supplier (зарезервована, без полів). Усі сутності використовують стратегію генерації первинного ключа IDENTITY (BIGSERIAL у PostgreSQL).

Сутність Farmer (таблиця farmers) описує фермера-користувача системи. Поля: firstName, lastName, email, phone, password. Пароль на поточному етапі MVP зберігається у відкритому вигляді – це свідоме спрощення для прототипу; для виробничого використання обов'язковим є перехід на BCrypt + Spring Security.

Сутність Field (таблиця fields) описує сільськогосподарське поле. Ключові поля: name (обов'язкове), area (площа у гектарах, тип Double), coordinates (GeoJSON-string, тип TEXT – обрано для збереження повного об'єкта Feature з геометрією Polygon без потреби у PostGIS), farmer (зв'язок @ManyToOne з Farmer через FK farmer_id), lastCrop (код попередньої культури), currentCrop (поточна вирощувана культура), growthStage (стадія росту).

Сутність SoilAnalysis (таблиця soil_analyses) фіксує результати агрохімічного аналізу. Поля: analysisDate (тип LocalDate), nitrogenN,

phosphorusP, potassiumK (мг/кг, тип Double), phLevel, cropState (текстове перерахування "FALLOW" / "PREVIOUS" / "CURRENT", що визначає логіку оновлення стану поля), cropType, growthStage, field (FK на fields), laboratory (FK на laboratories).

Сутність Laboratory (таблиця laboratories) – довідник лабораторій, що виконують аналіз. Містить name та apiEndpoint (поле зарезервоване під майбутню автоматичну інтеграцію з API лабораторій).

Hibernate автоматично створює та оновлює схему БД при старті додатка (spring.jpa.hibernate.ddl-auto=update). Кожна @Entity-сутність відображається в однойменну таблицю, перелічену в @Table(name="..."), зв’язки @ManyToOne матеріалізуються як foreign key з суфіксом _id, поле @Id з @GeneratedValue(IDENTITY) – як BIGSERIAL.

Було розроблено діаграму класів сутностей, яка представлена в додатку Г на рисунку Г.3.

Таблиця 2.1 – Схема таблиці fields у PostgreSQL

Колонка	Тип	Опис / обмеження
id	BIGSERIAL	PK, автоінкремент
name	VARCHAR(255) NOT NULL	Назва поля
area	DOUBLE PRECISION	Площа, га
coordinates	TEXT	GeoJSON Feature з геометрією Polygon
farmer_id	BIGINT	FK → farmers(id)
last_crop	VARCHAR(255)	Код попередньої культури
current_crop	VARCHAR(255)	Код поточної культури
growth_stage	VARCHAR(255)	Стадія росту

Таблиця 2.2 – Схема таблиці soil_analyses у PostgreSQL

Колонка	Тип	Опис
id	BIGSERIAL	PK
analysis_date	DATE	Дата аналізу
nitrogen_n	DOUBLE PRECISION	N, мг/кг
phosphorus_p	DOUBLE PRECISION	P, мг/кг
potassium_k	DOUBLE PRECISION	K, мг/кг
ph_level	DOUBLE PRECISION	Кислотність pH
crop_state	VARCHAR(255)	"FALLOW" / "PREVIOUS" / "CURRENT"
crop_type	VARCHAR(255)	Код культури
growth_stage	VARCHAR(255)	Стадія росту
field_id	BIGINT	FK → fields(id)
laboratory_id	BIGINT	FK → laboratories(id)

Було розроблено ER-діаграму бази даних, яка представлена в додатку Г на рисунку Г.4.

2.3 REST-API: специфікація реалізованих ендпоінтів

REST-API «АгроСистеми» зосереджено під базовим шляхом /api/ і поділено на п’ять логічних груп за іменами контролерів. Усі ендпоінти приймають та повертають JSON (заголовок Content-Type:

application/json). Аутентифікація на рівні HTTP не реалізована – сесія фермера зберігається на стороні браузера у localStorage; це свідоме спрощення прототипу. Дані передаються через тіло POST-запиту або через path-параметри для GET/DELETE.

Таблиця 2.3 – Повний список REST-ендпоінтів

Метод	URI	Призначення
POST	/api/farmers/register	Створення нового фермера
POST	/api/farmers/login	Авторизація за e-mail та паролем
GET	/api/fields/farmer/{farmerId}	Список полів конкретного фермера
POST	/api/fields	Створення нового поля
DELETE	/api/fields/{id}	Видалення поля + каскад аналізів
GET	/api/analyses/field/{fieldId}	Перелік аналізів для поля
POST	/api/analyses/field/{fieldId}	Додати аналіз; оновлює стан поля
POST	/api/calculator/calculate	Розрахунок норм добрив
POST	/api/calculator/save-crop	Закріпити обрану культуру за полем
GET	/api/rotation/recommend?lastCrop=	Рекомендації сівозміни

2.3.1 Контролер FarmerController

FarmerController обробляє реєстрацію та авторизацію фермерів. Метод register(@RequestBody Farmer newFarmer) перевіряє унікальність e-mail (стрімом по farmerRepository.findAll()) і, у разі успіху, зберігає сутність та повертає 200 OK з тілом фермера. Метод

login(@RequestBody Farmer loginRequest) шукає фермера за e-mail (без урахування регістру), порівнює пароль; при невдачі повертає 401 (UNAUTHORIZED) або 404 (NOT_FOUND).

2.3.2 Контролер FieldController

FieldController керує життєвим циклом полів. Метод getFieldsByFarmer(@PathVariable Long farmerId) повертає список через метод репозиторію findByFarmerId, який Spring Data JPA генерує автоматично. Метод createField(@RequestBody Field field) зберігає поле (зв’язок з фермером Spring підтягує через десеріалізацію вкладеного JSON-об’єкта farmer). Метод deleteField(@PathVariable Long id) позначений @Transactional: спочатку видаляються всі аналізи цього поля (analysisRepository.deleteAll), потім – саме поле, що запобігає порушенню FK-обмеження.

2.3.3 Контролер SoilAnalysisController

SoilAnalysisController оперує аналізами ґрунту. Метод addAnalysis(@PathVariable Long fieldId, @RequestBody SoilAnalysis analysis) виконує не лише збереження аналізу, а й оновлення стану поля: якщо у тілі запиту cropState=="CURRENT", встановлюються currentCrop, growthStage, а також lastCrop (як остання відома); при cropState=="PREVIOUS" зберігається lastCrop і обнуляються currentCrop / growthStage; при cropState=="FALLOW" поле переводиться

у пар (lastCrop="NONE"). Це забезпечує синхронізацію двох сутностей у рамках одного запиту.

2.3.4 Контролер CalculatorController

CalculatorController надає два ендпоінти. POST /api/calculator/calculate приймає JSON {fieldId, cropType}; знаходить найновіший аналіз поля (stream + Comparator.comparing(SoilAnalysis::getAnalysisDate).max()) і виконує балансовий розрахунок (див. п. 2.4). POST /api/calculator/save-crop закріплює обрану культуру за полем (field.setCurrentCrop(cropType), field.setGrowthStage("Посів/Сходи")).

2.3.5 Контролер RotationController

RotationController містить один метод recommendNextCrop(@RequestParam String lastCrop). Він повертає JSON виду {recommendations: [{crop, score, description}, ...], warning?: "..."}, обираючи рекомендації за switch-case за кодом попередньої культури. Логіку детально описано в п. 2.5.

2.4 Алгоритм розрахунку норм мінеральних добрив за принципом Лібіха

Алгоритм розрахунку норм добрив реалізовано у методі calculateFertilizers класу CalculatorController. Він складається із шести логічних кроків і поєднує балансовий метод з принципом лімітуючого фактора Лібіха у спрощеному формулюванні «потенціал = запас / норматив».

2.4.1 Крок 1. Виведення нормативів виносу та максимальної врожайності

За кодом культури (cropType) обираються тріади нормативів виносу елементів (normN, normP, normK у кг/т товарної продукції) та максимальна потенційна врожайність у регіоні (maxYield, т/га).

Значення зашиті у switch-case для 13 культур: пшениця, ячмінь, жито, овес, кукурудза, просо, соняшник, ріпак, соя, горох, гречка, цукровий буряк, картопля. Окремо для бобових (соя, горох) встановлюється прапорець isLegume=true, який згодом обмежить розрахункову норму азоту 30 кг/га (бобові фіксують атмосферний азот симбіотично).

Таблиця 2.4 – Нормативи виносу та максимальна врожайність (фрагмент)

Культура (код)	N, кг/т	P, кг/т	K, кг/т	maxYield, т/га	Бобова?
Пшениця озима (WHEAT)	25	10	20	6.0	ні
Кукурудза (CORN)	28	12	25	8.0	ні
Соняшник (SUNFLOWER)	50	25	100	3.5	ні
Ріпак озимий (RAPESEED)	60	30	50	3.5	ні
Соя (SOYBEAN)	70	20	40	3.0	ТАК
Горох (PEAS)	50	15	30	4.0	ТАК
Цукровий буряк (SUGAR_BEET)	45	20	60	40.0	ні
Картопля (POTATO)	35	15	50	30.0	ні

2.4.2 Крок 2. Розрахунок запасу елементів у ґрунті

Запас рухомих форм у ґрунті в кг/га обчислюється шляхом множення значень аналізу (мг/кг) на коефіцієнт переведення 2 – спрощена форма «коефіцієнта 0,3·d·h·10», яка для типового чорнозему щільністю d=1.2 г/см³ і глибини h=30 см дає приблизно 2,16 (округлено до 2 для зручності). Формули у коді:

```
soilN = nitrogenN  * 2;
soilP = phosphorusP * 2;
soilK = potassiumK  * 2;
```

2.4.3 Крок 3. Потенціал урожаю та лімітуючий фактор (Лібіх)

За принципом мінімуму Лібіха фактично можлива врожайність обмежується тим елементом, запасу якого «не вистачає першим». У коді для кожного з N, P, K обчислюється потенціал як відношення запасу у ґрунті до нормативу виносу:

```
potentialN = (!isLegume) ? soilN / normN : maxYield;
potentialP = soilP / normP;
potentialK = soilK / normK;
currentYield = min(maxYield, potentialN, potentialP, potentialK);
```

Якщо щось менше maxYield – формується пояснення для користувача про лімітуючий фактор («Азот (N)» / «Фосфор (P)» / «Калій (K)»). Для бобових азот як ліміт ніколи не показується (за рахунок умови !isLegume).

2.4.4 Крок 4. Потреба та дефіцит діючої речовини

Потреба для досягнення maxYield обчислюється як добуток normX · maxYield, а від неї віднімається запас у ґрунті. Дефіцит ділиться на коефіцієнти засвоєння (КВ – коефіцієнт використання) рослиною з мінерального добрива: для N = 0.6, для P = 0.25, для K = 0.5. Результат – норма у кг діючої речовини на 1 га:

```
needN = normN * maxYield;
activeN = max(0, (needN - soilN) / 0.6);
activeP = max(0, (needP - soilP) / 0.25);
activeK = max(0, (needK - soilK) / 0.5);
```

Передбачені обмеження-стелі для запобігання надмірному внесенню: activeP=0 при PhosphorusP > 100 мг/кг, activeK=0 при PotassiumK > 120 мг/кг, activeN ≤ 150, activeP ≤ 90, activeK ≤ 120. Для бобових activeN жорстко = 30.

2.4.5 Крок 5. Перерахунок у фізичну масу добрива

Норма у фізичній масі обчислюється діленням норми д.р. на масову частку д.р. у конкретному добриві: для азоту – аміачна селітра (34% N), для фосфору – суперфосфат подвійний (19% P), для калію – хлористий калій (60% K):

```
fertilizerN = Math.round(activeN / 0.34);  // кг/га аміачної селітри
fertilizerP = Math.round(activeP / 0.19);  // кг/га суперфосфату
fertilizerK = Math.round(activeK / 0.60);  // кг/га хлористого калію
```

2.4.6 Крок 6. Дробове внесення азоту

Високі дози азоту (> 200 кг/га фізичної маси) розбиваються на три внесення: 30% при посіві, 40% по мерзлоталому ґрунту навесні, 30% у фазі активного росту. Якщо доза менша – рекомендується одноразове внесення під передпосівну культивацію. Для фосфорних та калійних добрив рекомендація фіксована – 100% норми вносити восени під глибоку оранку (фосфор / калій малорухомі, тому осінній строк оптимальний).

Було розроблено діаграму діяльності «Розрахунок норм добрив», яка представлена в додатку В на рисунку В.4.

2.5 Алгоритм рекомендацій сівозміни

Модуль рекомендацій сівозміни реалізовано у RotationController як експертну систему правил у формі switch-case (табл. 2.5). Вхідним параметром є код останньої вирощуваної на полі культури (lastCrop), значення якого Spring отримує як @RequestParam.

Таблиця 2.5 – Реалізована логіка рекомендацій сівозміни

Попередник	Рекомендовані наступники (код, оцінка)	Warning
SUNFLOWER	CORN – «Чудово»; WHEAT – «Добре»	«7-річна пауза!»
WHEAT	RAPESEED – «Ідеально»; SUNFLOWER – «Добре»; SOYBEAN – «Чудово»	–
SOYBEAN	WHEAT – «Ідеально»; CORN – «Чудово»	–
CORN	WHEAT – «Добре»; SOYBEAN – «Чудово»	–
default	WHEAT – «Універсально»	–

Для кожної з підтримуваних попередніх культур (SUNFLOWER, WHEAT, SOYBEAN, CORN) задано фіксований перелік 2–3 рекомендацій, кожна з яких включає код наступної культури, якісну оцінку («Чудово» / «Добре» / «Ідеально» / «Універсально») та пояснювальний коментар українською мовою. Для культур, не

передбачених у switch, спрацьовує default-гілка з універсальною рекомендацією «WHEAT».

Особлива увага приділена соняшнику. Якщо lastCrop=="SUNFLOWER", додатково генерується попередження warning: «Соняшник не можна повертати на це поле протягом 7 років». Це відображає галузеву фітосанітарну вимогу мінімальної паузи повернення для запобігання накопиченню патогенів (Phomopsis, Sclerotinia).

У кваліфікаційній роботі заплановано розширення цієї експертної системи до повноцінної матриці сумісності попередників M[i,j] та оптимізаційної задачі цілочисельного лінійного програмування на горизонт 3–5 років. Для поточного MVP-етапу обраний формат правил є достатнім: фермер отримує миттєву якісну рекомендацію без розгортання задачі ILP.

Було розроблено діаграму діяльності «Рекомендація сівозміни», яка представлена в додатку В на рисунку В.5.

2.6 Інтерактивна ГІС-підсистема на базі Leaflet.js та Turf.js

ГІС-підсистема системи «АгроСистема» повністю реалізована на стороні браузера засобами Leaflet.js 1.9.4, плагіна Leaflet.draw 1.0.4 та бібліотеки Turf.js 6. Серверна частина оперує геометрією лише як TEXT-полем у форматі GeoJSON, не виконуючи жодних просторових обчислень.

2.6.1 Ініціалізація карти та шарів

При вході у вкладку «Карта» викликається функція initGlobalMap(). Карта створюється з центром у геометричному центрі України (48.3794, 31.1656), масштаб zoom=6. Підключаються три tile-шари (відповідно до концепції точного землеробства, де користувач має бачити поле і у вигляді супутникового знімка, і у вигляді карти доріг, і за рельєфом):

– «Карта (Вулиці)» – стандартний OpenStreetMap (tile.openstreetmap.org);

– «Супутник» – World Imagery від ArcGIS Online (server.arcgisonline.com);

– «Рельєф» – OpenTopoMap (tile.opentopomap.org).

Перемикач шарів L.control.layers додається у правий верхній кут карти; за замовчуванням активний супутник.

2.6.2 Малювання контурів полів (Leaflet.draw)

У модальному вікні «Нове поле» (#addFieldModal) ініціалізується окрема карта addFieldMap з підключеним інструментом L.Control.Draw. Дозволено малювання лише полігонів (полілінії, прямокутники, кола, маркери та CircleMarker відключені). При завершенні малювання (подія draw:created) попередній полігон видаляється з drawnItems (це забезпечує політику «одне поле = один полігон»), новий додається, а площа полігона у м² обчислюється Turf.js і конвертується у гектари:

```
const geojson = layer.toGeoJSON();
const areaSquareMeters = turf.area(geojson);
```

```
const areaHectares = areaSquareMeters / 10000;
document.getElementById('fieldArea').value = areaHectares.toFixed(2);
document.getElementById('fieldCoords').value = JSON.stringify(geojson);
```

2.6.3 Ручне введення координат

Окрім малювання миттю, передбачено введення координат вершин полігону вручну в текстовому полі формату "lat1, lng1; lat2, lng2; ...". Функція applyManualCoords парсить рядок, формує масив L.LatLng, створює L.polygon, додає його до drawnItems та обчислює площу так само через Turf.js. Це зручно у випадках, коли координати поля вже відомі з кадастрового документу.

2.6.4 Відображення наявних полів на головній карті

Після ініціалізації globalMap функція проходить по масиву allFields, для кожного поля з непорожнім полем coordinates (що містить серіалізований GeoJSON, починається з «{») викликає L.geoJSON() з оформленням { color: '#00ff00', weight: 3, fillOpacity: 0.3 } та прив’язує popup з назвою поля, площею та попередньою культурою. Усі межі полів збираються у L.latLngBounds, після чого map.fitBounds() автоматично масштабує карту так, щоб усі поля фермера потрапили у в’юпорт.

2.7 Модель варіантів використання (Use Case)

На основі фактично реалізованих ендпоінтів та функцій клієнтського шару побудовано діаграму прецедентів для актора Фермер. Діаграма містить 12 варіантів використання, які повністю відповідають коду; жодних запланованих, але не реалізованих сценаріїв до діаграми не включено.

Було розроблено діаграму прецедентів системи «АгроСистема», яка представлена в додатку В на рисунку В.6.

Таблиця 2.6 – Специфікація варіантів використання

Код	Назва	Реалізація у коді
UC-01	Реєстрація фермера	POST /api/farmers/register
UC-02	Вхід у систему	POST /api/farmers/login
UC-03	Перегляд переліку власних полів	GET /api/fields/farmer/{id}
UC-04	Додавання поля з малюванням на карті	Leaflet.draw + POST /api/fields
UC-05	Додавання поля з ручним введенням координат	applyManualCoords + POST /api/fields
UC-06	Видалення поля	DELETE /api/fields/{id} (каскад)
UC-07	Перегляд усіх полів на одній карті	initGlobalMap (Leaflet)
UC-08	Введення агрохімічного аналізу	POST /api/analyses/field/{id}
UC-09	Перегляд історії аналізів поля	GET /api/analyses/field/{id}
UC-10	Розрахунок норм добрив за культурою	POST /api/calculator/calculate
UC-11	Закріплення обраної культури за полем	POST /api/calculator/save-crop
UC-12	Отримання рекомендації сівозміни	GET /api/rotation/recommend

2.7.1 Сценарій UC-04 «Додавання поля з малюванням на карті»

Передумова: фермер авторизований у системі. Основний потік: (1) фермер натискає кнопку «Додати поле» у вкладці «Мої поля»; (2) відкривається модальне вікно з картою; (3) фермер обирає інструмент «полігон» у тулбарі Leaflet.draw та послідовно клацає по вершинам контуру поля; (4) браузер обчислює площу через turf.area та заповнює поля «РОЗРАХОВАНА ПЛОЩА» та прихований інпут fieldCoords; (5) фермер вводить назву поля та попередню культуру, натискає «Зберегти поле»; (6) клієнт відправляє POST /api/fields з тілом {name, area, coordinates, lastCrop, farmer: {id}}; (7) сервер зберігає сутність, повертає її; (8) клієнт оновлює таблицю полів.

2.7.2 Сценарій UC-10 «Розрахунок норм добрив»

Передумова: для поля існує хоча б один запис у soil_analyses. Основний потік: (1) фермер відкриває вкладку «Калькулятор», обирає поле зі списку та культуру для вирощування; (2) натискає «Розрахувати»; (3) клієнт відправляє POST /api/calculator/calculate з {fieldId, cropType}; (4) сервер виконує алгоритм п. 2.4 та повертає JSON {crop, currentYield, maxYield, limitingFactor, fertilizerN, fertilizerP, fertilizerK, adviceN, adviceP, adviceK}; (5) клієнт відображає у блоці #calcResultBox прогнозовану та максимальну врожайність, лімітуючий фактор, дози у фізичній масі та рекомендації по строкам внесення; (6) додатково клієнтська логіка показує рекомендовані препарати ЗЗР для обраної культури (протруйник, ґрунтовий гербіцид, листовий гербіцид/інсектицид). Альтернативний потік: якщо для поля немає

жодного аналізу, сервер повертає {error: "Для цього поля немає аналізів ґрунту!"} і клієнт показує toast-повідомлення.

2.8 Діаграми діяльності (Activity)

Розроблено три діаграми діяльності (UML 2.x), кожна з яких відображає фактичний потік виконання у відповідному ендпоінті та клієнтській функції. -код кожної діаграми наведено у Додатку В.

2.8.1 AD-01: Додавання поля з GeoJSON-геометрією

Потік: «Старт» → «Відкрити модалку Add Field» → «Намалювати або ввести координати» → «turf.area: розрахунок площі» → «Заповнити форму (name, area)» → POST /api/fields → «Hibernate: save(field)» → «Повернути 200 OK з полем» → «Оновити таблицю полів» → «Кінець».

2.8.2 AD-02: Розрахунок норм добрив

Потік: «Старт» → «Обрати поле і культуру» → POST /api/calculator/calculate → «findByFieldId(fieldId)» → [розгалуження: чи є аналізи?] → [ні] «Повернути error» → toast → «Кінець»; [так] → «Обрати найновіший аналіз» → «switch(cropType): нормативи» → «Обчислити запас (×2)» → «Обчислити потенціали» → «Визначити лімітуючий

фактор» → «Розрахувати дефіцит д.р.» → «Перевести у фізичну масу через 0.34/0.19/0.60» → [N > 200 кг/га?] → [так] «Розбити на 3 внесення» → «Сформувати JSON-відповідь» → «Клієнт: відобразити дози і рекомендації ЗЗР» → «Кінець».

2.8.3 AD-03: Рекомендація сівозміни

Потік: «Старт» → «Обрати поле» → «Отримати lastCrop з картки поля» → GET /api/rotation/recommend?lastCrop=... → «switch(lastCrop)» → «Сформувати масив recommendations та warning (за необхідності)» → «Повернути JSON» → «Клієнт: відобразити список карток та (якщо є) попередження жовтим alert» → «Кінець».

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ ПІДТРИМКИ АГРОТЕХНІЧНИХ РІШЕНЬ

У цьому розділі виконано проєктування програмного комплексу для інформаційного забезпечення системи точного землеробства та підтримки прийняття агротехнічних рішень. Проєктування здійснено із застосуванням об'єктно-орієнтованого підходу та уніфікованої мови моделювання UML. Послідовно розроблено модель прецедентів використання, що відображає функціональні можливості системи з погляду користувача, діаграму класів, яка описує статичну структуру програмних компонентів, та діаграми послідовностей для документування динаміки взаємодії об'єктів під час виконання ключових сценаріїв. Окремий підрозділ присвячено проєктуванню реляційної бази даних – від інфологічної моделі до перевірки відношень на відповідність нормальним формам. Розроблені моделі є основою для реалізації програмного комплексу, наведеної у розділі 4.

3.1 Розробка діаграми прецедентів

Діаграма прецедентів (use case diagram) є відправною точкою об'єктно-орієнтованого проєктування. Вона у наочній формі фіксує функціональні вимоги до системи, визначаючи, які саме дії може виконувати користувач та як ці дії пов'язані між собою. Модель прецедентів розроблено на основі фактичних функціональних вимог до комплексу та узгоджено з реалізованими у програмному коді можливостями: до моделі включено лише ті варіанти використання, що мають реалізацію, без запланованих, але не втілених сценаріїв.

3.1.1 Актори та межі системи

Актор – це зовнішня стосовно системи сутність (особа або інша система), що взаємодіє з нею для досягнення певної мети. Аналіз предметної області показав, що програмний комплекс має єдиного актора – Фермера. Фермер є зареєстрованим користувачем системи, який веде облік власних сільськогосподарських угідь, вносить результати агрохімічних обстежень ґрунту та отримує обґрунтовані рекомендації щодо удобрення полів і чергування культур у сівозміні. Інші категорії користувачів (адміністратор, агроном-консультант) у поточній версії комплексу не виокремлюються, оскільки всі функції зосереджено навколо діяльності одного фахівця.

Межею системи виступає сам програмний комплекс підтримки агротехнічних рішень, який інкапсулює всю бізнес-логіку та дані. Актор взаємодіє із системою виключно через інтерфейс користувача, не маючи прямого доступу до внутрішніх компонентів. Зовнішні джерела даних (картографічні сервіси та сервіс погодної інформації) у моделі прецедентів не розглядаються як актори, оскільки вони не ініціюють сценарії використання, а лише обслуговують запити системи. Загальну модель прецедентів наведено на рисунку 3.1.

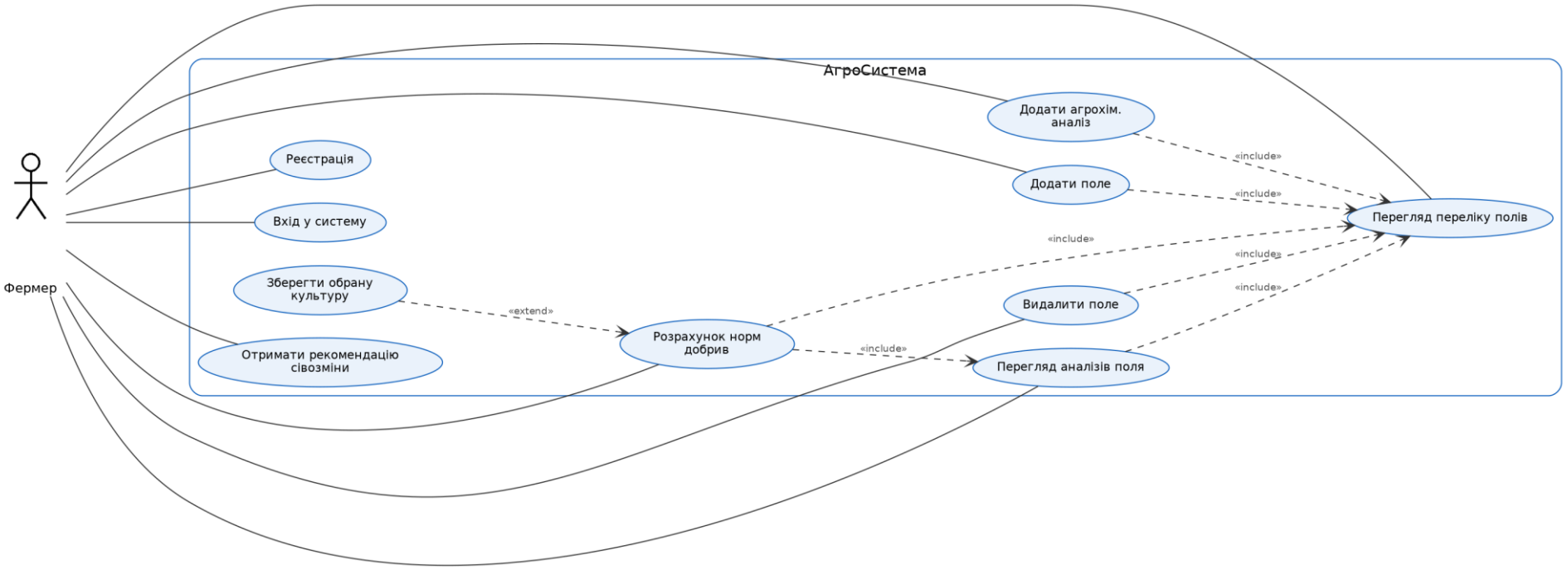

Рисунок 3.1 – Діаграма прецедентів програмного комплексу

3.1.2 Опис прецедентів і відношень include/extend

За результатами аналізу функціональних вимог виокремлено десять прецедентів, які за змістом об'єднуються у чотири логічні групи: управління обліковим записом, управління полями, робота з агрохімічними аналізами та підтримка прийняття рішень. Перелік прецедентів із коротким описом наведено в таблиці 3.1.

Таблиця 3.1 – Перелік прецедентів використання

Прецедент	Призначення
Реєстрація	Створення нового облікового запису фермера
Вхід у систему	Автентифікація зареєстрованого користувача
Перегляд переліку полів	Відображення списку полів фермера на карті та у списку
Додати поле	Створення нового поля малюванням на карті або введенням координат
Видалити поле	Вилучення поля разом з пов'язаними аналізами
Перегляд аналізів поля	Перегляд історії агрохімічних обстежень обраного поля
Додати агрохімічний аналіз	Внесення результатів обстеження ґрунту для поля
Розрахунок норм добрив	Обчислення доз мінеральних добрив за законом Лібіха
Зберегти обрану культуру	Фіксація обраної для поля культури за результатами розрахунку
Отримати рекомендацію сівозміни	Видача рекомендованих культур-послідовників

Між прецедентами встановлено відношення двох типів. Відношення включення «include» означає, що базовий прецедент

завжди використовує функціональність включеного. Усі прецеденти, що оперують конкретним полем, – «Додати поле», «Видалити поле», «Перегляд аналізів поля», «Додати агрохімічний аналіз» та «Розрахунок норм добрив» – пов'язані відношенням «include» з прецедентом «Перегляд переліку полів», оскільки вибір поля зі списку є їх обов'язковою передумовою. Крім того, прецедент «Розрахунок норм добрив» включає прецедент «Перегляд аналізів поля», бо алгоритм розрахунку обов'язково використовує найновіший за датою агрохімічний аналіз обраного поля.

Відношення розширення «extend» означає, що один прецедент за певних умов доповнює інший необов'язковою поведінкою. Прецедент «Зберегти обрану культуру» пов'язаний відношенням «extend» з прецедентом «Розрахунок норм добрив», оскільки збереження культури є необов'язковим продовженням сценарію розрахунку: фермер може зберегти обраний варіант культури для подальшого планування, а може обмежитися лише отриманням розрахунку. Така структура відношень забезпечує логічну цілісність моделі та усуває дублювання спільної поведінки між прецедентами.

3.2 Розробка діаграми класів

Діаграма класів описує статичну структуру програмного комплексу – склад програмних класів, їх атрибути, операції та зв'язки між ними. Комплекс побудовано за принципом багатошарової архітектури, у якій виокремлено три рівні класів: рівень сутностей предметної області, рівень доступу до даних та рівень контролерів із бізнес-логікою. Такий поділ забезпечує слабку зв'язність компонентів, спрощує супровід і

подальший розвиток системи. Повну діаграму класів наведено на рисунку 3.2.

3.2.1 Класи-сутності предметної області

Рівень сутностей утворюють класи, що відображають ключові поняття предметної області та відображаються на таблиці бази даних за допомогою технології об'єктно-реляційного відображення JPA. Виокремлено чотири класи-сутності: Farmer (фермер), Field (поле), SoilAnalysis (агрохімічний аналіз ґрунту) та Laboratory (лабораторія). Склад атрибутів цих класів наведено в таблиці 3.2.

Таблиця 3.2 – Класи-сутності предметної області та їх атрибути

Клас	Основні атрибути
Farmer	id, firstName, lastName, email, phone, password
Field	id, name, area, coordinates, lastCrop, currentCrop, growthStage
SoilAnalysis	id, analysisDate, nitrogenN, phosphorusP, potassiumK, phLevel, cropState, cropType, growthStage
Laboratory	id, name, apiEndpoint

Клас Farmer зберігає облікові дані користувача та контактну інформацію. Клас Field описує сільськогосподарське поле: його назву, площу, геометрію контуру (атрибут coordinates), а також відомості про попередню та поточну культури й фазу розвитку рослин, потрібні для рекомендацій із сівозміни. Клас SoilAnalysis містить результати агрохімічного обстеження ґрунту – вміст основних елементів живлення (азоту, фосфору, калію), рівень кислотності pH, дату аналізу та супутні

характеристики стану посіву. Клас Laboratory є довідником лабораторій, що виконують обстеження. Для скорочення обсягу шаблонного коду методи доступу до атрибутів сутностей генеруються автоматично, тому на діаграмі класів сутності подано переважно складом атрибутів.

3.2.2 Контролери, репозиторії та зв'язки між класами

Рівень доступу до даних утворюють класи-репозиторії, реалізовані як інтерфейси технології Spring Data JPA. Кожна сутність має власний репозиторій: FarmerRepository, FieldRepository, SoilAnalysisRepository та LaboratoryRepository. Окрім успадкованих стандартних операцій збереження, пошуку та видалення, репозиторії містять спеціалізовані методи запитів: FieldRepository надає метод findByFarmerId для отримання полів конкретного фермера, а SoilAnalysisRepository – метод findByFieldId для отримання аналізів конкретного поля. Тексти відповідних запитів формуються фреймворком автоматично за назвами методів.

Рівень контролерів реалізує бізнес-логіку та публікує програмний інтерфейс системи. Виокремлено п'ять контролерів. Клас FarmerController відповідає за реєстрацію та автентифікацію користувачів; FieldController – за перегляд, створення та видалення полів; SoilAnalysisController – за перегляд і додавання агрохімічних аналізів; CalculatorController реалізує розрахунок норм мінеральних добрив та збереження обраної культури; RotationController формує рекомендації щодо чергування культур у сівозміні.

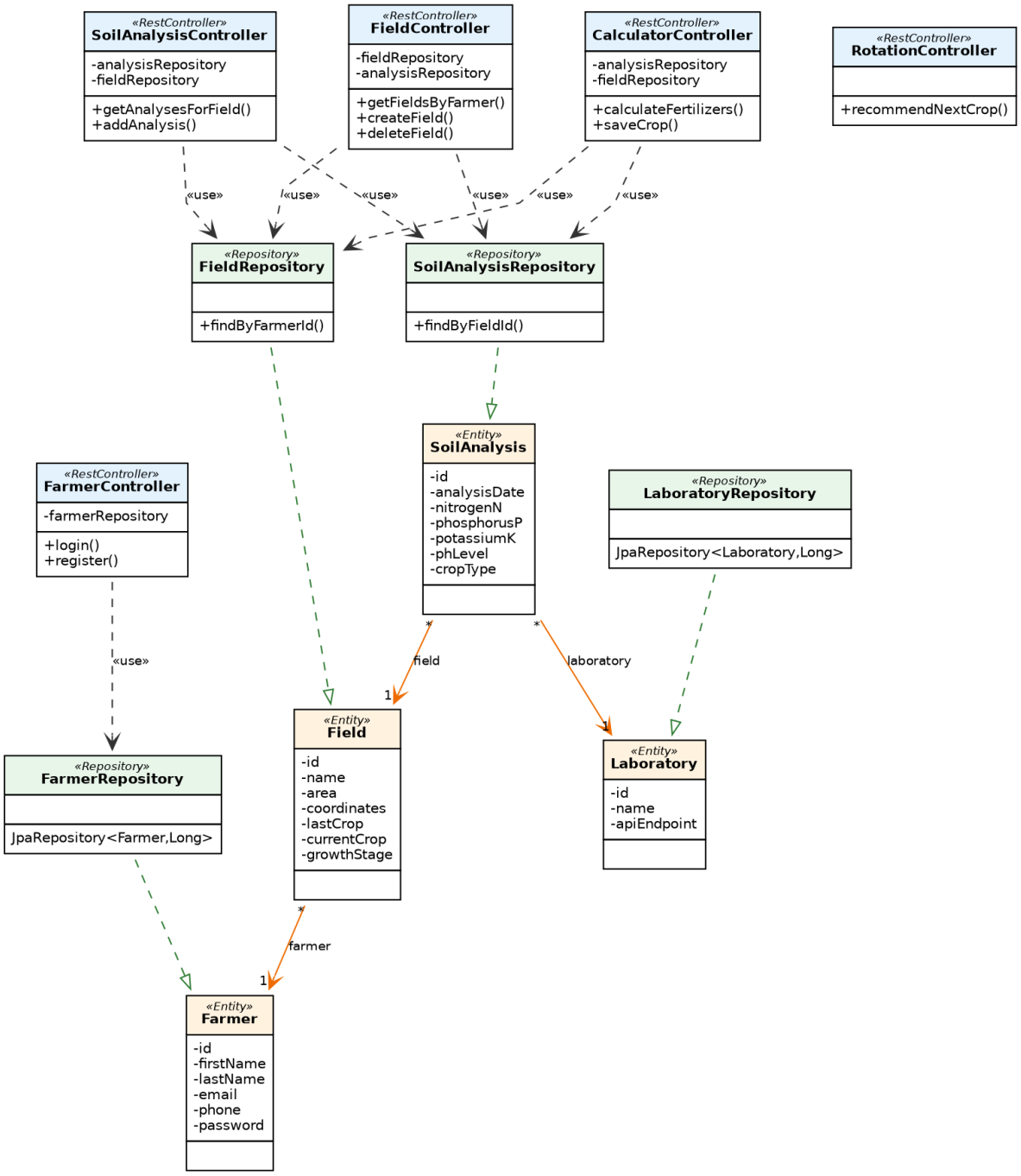

Рисунок 3.2 – Діаграма класів програмного комплексу

Між класами встановлено зв'язки трьох типів. Між сутностями діють асоціації типу «багато до одного»: поле належить одному фермеру (Field – Farmer), а аналіз пов'язаний з одним полем (SoilAnalysis – Field) та однією лабораторією (SoilAnalysis – Laboratory). Кратність з боку підпорядкованих сутностей становить «багато» (один

фермер може мати багато полів, одне поле – багато аналізів). Між контролерами та репозиторіями діють відношення залежності, що відповідають упровадженню залежностей: контролер використовує репозиторій для доступу до даних, не створюючи його самостійно. Репозиторії, своєю чергою, типізовані відповідними класами-сутностями. Описану структуру класів та зв'язків наведено на рисунку 3.2.

3.3 Розробка діаграми послідовностей

Діаграма послідовностей документує динамічну поведінку системи, відображаючи обмін повідомленнями між об'єктами у часі під час виконання конкретного сценарію. Для проєктованого комплексу розроблено діаграми послідовностей для двох найважливіших сценаріїв підтримки прийняття рішень – розрахунку норм добрив та отримання рекомендації сівозміни, оскільки саме вони становлять методичну цінність системи.

3.3.1 Сценарій розрахунку норм добрив

Сценарій розрахунку норм добрив ініціюється фермером після вибору поля та культури в інтерфейсі користувача. Послідовність взаємодії об'єктів така. Клієнтський застосунок надсилає запит на розрахунок до контролера CalculatorController, передаючи ідентифікатор поля та код обраної культури. Контролер звертається до репозиторію SoilAnalysisRepository з викликом методу findByFieldId для

отримання всіх агрохімічних аналізів поля. З отриманого переліку контролер обирає найновіший за датою аналіз; за відсутності аналізів формується повідомлення про помилку, і сценарій завершується.

За наявності аналізу контролер виконує балансовий розрахунок норм добрив: визначає нормативи виносу елементів живлення для обраної культури, обчислює запас азоту, фосфору й калію у ґрунті, потенціал урожаю за законом мінімуму Лібіха, потребу в діючій речовині та її дефіцит, після чого перераховує дефіцит у фізичну масу добрив і формує план їх внесення. Сформований результат у форматі JSON повертається клієнтському застосунку, який відображає фермеру рекомендовані дози добрив та план внесення. Описану послідовність взаємодії наведено на рисунку 3.3.

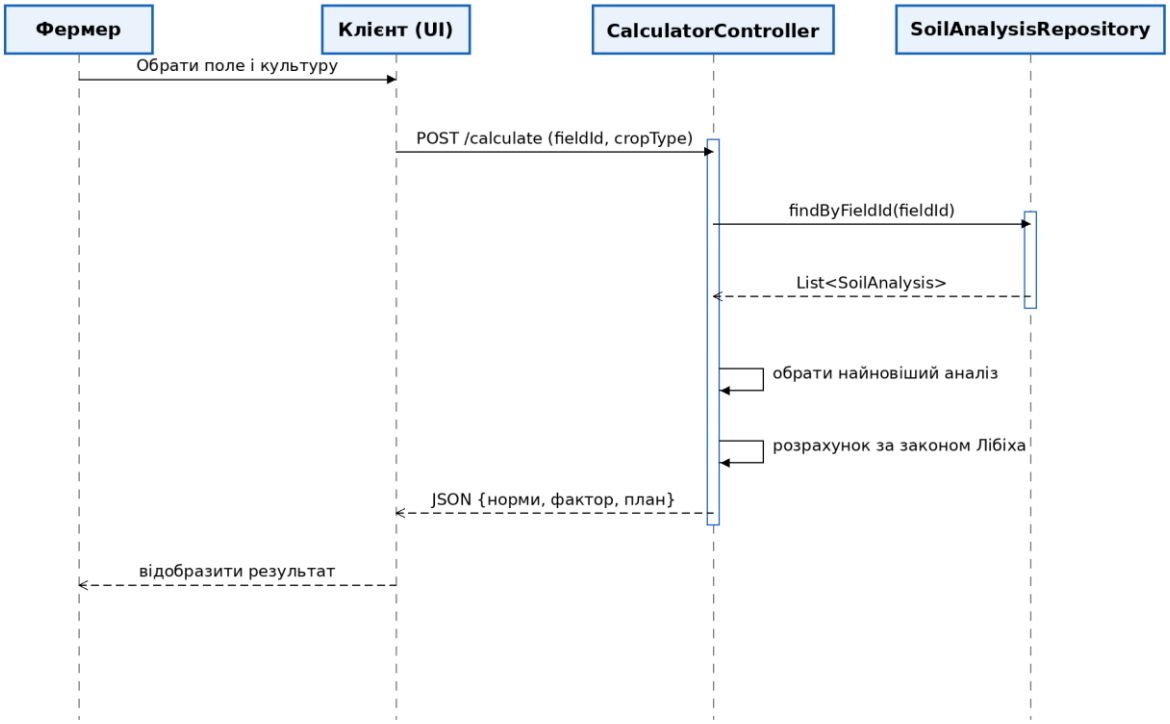

Рисунок 3.3 – Діаграма послідовностей сценарію розрахунку норм добрив

3.3.2 Сценарій отримання рекомендації сівозміни

Сценарій отримання рекомендації сівозміни ініціюється фермером для обраного поля. Клієнтський застосунок зчитує код попередньої культури з картки поля та надсилає запит до контролера RotationController з викликом методу формування рекомендації, передаючи код попередньої культури. Контролер застосовує базу агрономічних правил чергування культур і визначає перелік рекомендованих культур-послідовників із якісною оцінкою сумісності кожного варіанта та текстовим поясненням.

Для окремих культур формуються додаткові застереження: зокрема, для соняшнику генерується попередження щодо неприпустимості його повторного розміщення на полі раніше ніж через сім років. Сформований перелік рекомендацій повертається клієнтському застосунку, який відображає його фермеру у зручній формі. Цей сценарій реалізує спрощену систему підтримки прийняття рішень, що інформує користувача про агрономічно обґрунтовані варіанти чергування культур. Послідовність взаємодії об'єктів у цьому сценарії наведено на рисунку 3.4.

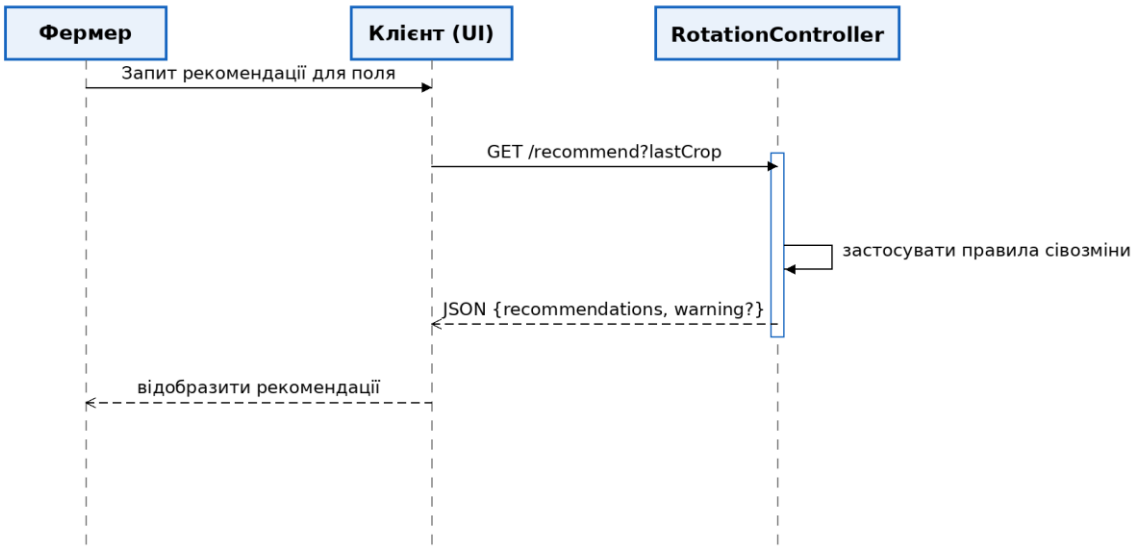

Рисунок 3.4 – Діаграма послідовностей сценарію рекомендації сівозміни

3.4 Розробка бази даних

Для надійного зберігання та ефективного опрацювання даних предметної області спроєктовано реляційну базу даних під керуванням системи керування базами даних PostgreSQL. Проєктування виконано у два етапи: спершу побудовано інфологічну модель предметної області у вигляді діаграми «сутність – зв'язок», далі здійснено перехід до реляційної схеми з визначенням структури таблиць, ключів і зв'язків та перевіркою отриманих відношень на відповідність нормальним формам.

3.4.1 ER-діаграма та опис таблиць

Інфологічну модель предметної області подано діаграмою «сутність – зв'язок» (Entity – Relationship Diagram). Модель містить чотири сутності, що відповідають класам предметної області: farmers (фермери), fields (поля), soil_analyses (агрохімічні аналізи) та laboratories (лабораторії). Кожна сутність відображається на окрему таблицю реляційної бази даних. Структуру таблиць наведено в таблиці 3.3.

Таблиця 3.3 – Структура таблиць бази даних

Таблиця	Поля	Призначення
farmers	id, first_name, last_name, email, phone, password	Облікові записи фермерів
fields	id, name, area, coordinates, last_crop, current_crop, growth_stage, farmer_id	Поля фермера з геометрією та станом культур
soil_analyses	id, analysis_date, nitrogen_n, phosphorus_p, potassium_k, ph_level, crop_state, crop_type, growth_stage, field_id, laboratory_id	Результати агрохімічних обстежень ґрунту
laboratories	id, name, api_endpoint	Довідник лабораторій-виконавців

Таблиця farmers зберігає облікові записи користувачів. Таблиця fields містить відомості про поля: назву, площу, геометрію контуру у текстовому форматі, дані про попередню та поточну культури, фазу розвитку рослин і посилання на власника поля. Таблиця soil_analyses зберігає результати агрохімічних обстежень – вміст основних елементів

живлення, рівень кислотності та супутні характеристики, а також посилання на поле й лабораторію. Таблиця laboratories є довідником організацій, що виконують обстеження. Інфологічну модель бази даних наведено на рисунку 3.5.

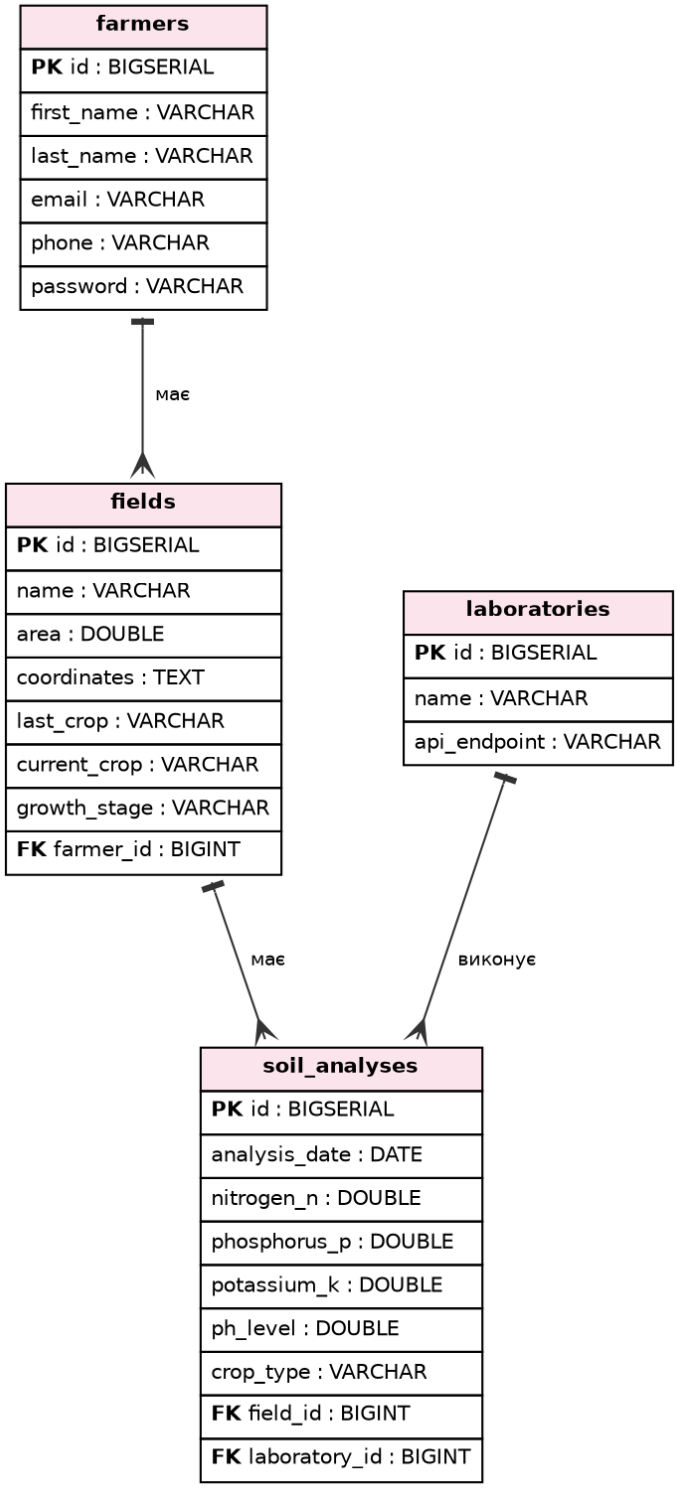

Рисунок 3.5 – ER-діаграма бази даних

3.4.2 Зв'язки, ключі та нормалізація

Первинним ключем кожної таблиці є сурогатний цілочисловий ідентифікатор id типу BIGSERIAL, значення якого генерується автоматично під час додавання запису. Використання сурогатних ключів забезпечує незмінність ідентифікаторів і незалежність від змін у предметних даних. Зв'язки між таблицями реалізовано за допомогою зовнішніх ключів: поле fields.farmer_id посилається на таблицю farmers, поля soil_analyses.field_id та soil_analyses.laboratory_id – на таблиці fields та laboratories відповідно.

Усі зв'язки мають тип «один до багатьох»: одному фермеру відповідає багато полів, одному полю – багато агрохімічних аналізів, одній лабораторії – багато аналізів. Клас приналежності з боку підпорядкованих сутностей є обов'язковим: кожне поле обов'язково належить фермеру, а кожен аналіз – полю. Цілісність даних на рівні бази підтримується обмеженнями зовнішніх ключів, що унеможливлює появу записів-сиріт.

Для перевірки коректності проєктування реляційну схему проаналізовано на відповідність нормальним формам. Усі таблиці перебувають у першій нормальній формі, оскільки кожне поле зберігає атомарне значення, а повторювані групи відсутні. Друга нормальна форма виконується, бо за наявності простого первинного ключа неможливі часткові залежності неключових атрибутів. Третя нормальна форма та нормальна форма Бойса – Кодда також виконуються: аналіз функціональних залежностей між атрибутами показав, що всі неключові атрибути кожної таблиці функціонально повно залежать лише від первинного ключа, а транзитивні залежності та залежності від частини ключа відсутні. Отже, спроєктована схема бази даних перебуває у нормальній формі Бойса – Кодда, що мінімізує надмірність даних та аномалії їх оновлення.

3.5 Програмна реалізація комплексу

3.5.1 Технологічний стек і структура проєкту

Програмний комплекс реалізовано за клієнт-серверною архітектурою. Серверну частину побудовано мовою Java версії 21 на основі фреймворку Spring Boot, що забезпечує автоконфігурацію застосунку та вбудований вебсервер. Доступ до даних реалізовано засобами Spring Data JPA з постачальником Hibernate, сховищем даних слугує реляційна СУБД PostgreSQL. Складання проєкту та керування залежностями виконує система Maven. Клієнтську частину реалізовано як вебсторінку з використанням мови JavaScript та бібліотек Bootstrap (адаптивний інтерфейс), Leaflet.js і Leaflet.draw (інтерактивна карта та малювання контурів полів) та Turf.js (геометричні обчислення). Обґрунтуванням вибору стеку є його зрілість, відкритість, велика спільнота та повна безкоштовність, що мінімізує вартість розроблення.

Проєкт організовано за пакетною структурою, що відповідає шарам архітектури. Загальну структуру вихідного коду серверної частини наведено на рисунку 3.6.

```
agro-system
 └─ src/main/java/com/example/agrosystem
    ├─ AgroSystemApplication.java   // точка входу
    ├─ entity/       Farmer, Field, SoilAnalysis, Laboratory
    ├─ repository/   FarmerRepository, FieldRepository,
    │                SoilAnalysisRepository, LaboratoryRepository
    └─ controller/   FarmerController, FieldController,
                     SoilAnalysisController, CalculatorController,
                     RotationController
 └─ src/main/resources/static/index.html   // клієнтська частина
```

Рисунок 3.6 – Структура проєкту

Точкою входу застосунку є клас AgroSystemApplication, позначений анотацією @SpringBootApplication, яка активує автоконфігурацію контейнера Spring, сканування компонентів та запуск вбудованого вебсервера.

3.5.2 Реалізація серверної частини

Шар сутностей реалізовано класами, позначеними анотаціями JPA. Кожна сутність відображається на таблицю бази даних: анотація @Entity позначає клас як сутність, @Table задає назву таблиці, @Id та @GeneratedValue визначають первинний ключ із автоматичною генерацією, а @ManyToOne та @JoinColumn описують зв'язок із батьківською сутністю через зовнішній ключ. Для автоматичної генерації методів доступу застосовано анотацію @Data бібліотеки Lombok. Реалізацію сутності Field наведено на рисунку 3.7.

```
@Entity
@Table(name = "fields")
@Data
public class Field {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String name;
    private Double area;
    @Column(columnDefinition = "TEXT")
    private String coordinates;
    @ManyToOne
    @JoinColumn(name = "farmer_id")
    private Farmer farmer;
    private String lastCrop;
    private String currentCrop;
    private String growthStage;
}
```

Рисунок 3.7 – Клас-сутність Field

Шар доступу до даних реалізовано інтерфейсами-репозиторіями, що успадковують JpaRepository. Окрім стандартних операцій, репозиторії містять похідні методи запитів, текст яких формується фреймворком автоматично за назвою методу. Реалізацію репозиторію полів наведено на рисунку 3.8.

```
@Repository
public interface FieldRepository extends JpaRepository<Field, Long> {
   List<Field> findByFarmerId(Long farmerId);
}
```

Рисунок 3.8 – Інтерфейс FieldRepository

Шар контролерів реалізує бізнес-логіку та публікує REST-API. Контролер позначається анотацією @RestController, базовий шлях задається анотацією @RequestMapping, а окремі операції – анотаціями @GetMapping, @PostMapping та @DeleteMapping. Залежності від репозиторіїв упроваджуються анотацією @Autowired. Реалізацію контролера полів наведено на рисунку 3.9; зверніть увагу на метод видалення поля, позначений анотацією @Transactional, який спершу вилучає пов'язані аналізи, а потім саме поле, забезпечуючи цілісність даних.

```
@RestController
@RequestMapping("/api/fields")
public class FieldController {
   @Autowired private FieldRepository fieldRepository;
   @Autowired private SoilAnalysisRepository analysisRepository;
   @GetMapping("/farmer/{farmerId}")
   public List<Field> getFieldsByFarmer(@PathVariable Long farmerId) {
      return fieldRepository.findByFarmerId(farmerId);
   }
```

Рисунок 3.9 – Контролер FieldController

```
    @PostMapping
    public Field createField(@RequestBody Field field) {
        return fieldRepository.save(field);
    }
    @Transactional
    @DeleteMapping("/{id}")
    public void deleteField(@PathVariable Long id) {
        List<SoilAnalysis> a = analysisRepository.findByFieldId(id);
        if (!a.isEmpty()) analysisRepository.deleteAll(a);
        fieldRepository.deleteById(id);
    }
}
```

Продовження рисунку 3.9

Повний перелік реалізованих REST-ендпоінтів серверної частини наведено в таблиці 3.4.

Таблиця 3.4 – REST-ендпоінти серверної частини

Контролер	Ендпоінт	Призначення
FarmerController	POST /api/farmers/register; POST /api/farmers/login	Реєстрація та автентифікація
FieldController	GET /api/fields/farmer/{id}; POST /api/fields; DELETE /api/fields/{id}	Управління полями
SoilAnalysisController	GET /api/analyses/field/{id}; POST /api/analyses/field/{id}	Аналізи ґрунту
CalculatorController	POST /api/calculator/calculate; POST /api/calculator/save-crop	Розрахунок норм добрив
RotationController	GET /api/rotation/recommend	Рекомендація сівозміни

3.5.3 Реалізація алгоритмів підтримки рішень

Ключовим компонентом комплексу є алгоритм розрахунку норм мінеральних добрив, реалізований у методі calculateFertilizers контролера CalculatorController. Алгоритм отримує з бази найновіший за датою агрохімічний аналіз поля, визначає нормативи виносу елементів живлення для обраної культури, обчислює запас азоту, фосфору й калію у ґрунті та потенціал урожаю за законом мінімуму Лібіха як мінімум серед лімітуючих факторів. Далі обчислюється потреба у діючій речовині та її дефіцит з урахуванням коефіцієнтів засвоєння добрив. Центральний фрагмент алгоритму наведено на рисунку 3.10.

```
// Потенціал урожаю за законом мінімуму Лібіха
double potentialN = (normN > 0 && !isLegume) ? soilN/normN : maxYield;
double potentialP = (normP > 0) ? soilP/normP : maxYield;
double potentialK = (normK > 0) ? soilK/normK : maxYield;
double currentYield = Math.min(maxYield,
    Math.min(potentialN, Math.min(potentialP, potentialK)));

// Потреба у діючій речовині з урахуванням засвоєння
double activeN = Math.max(0, (needN - soilN) / 0.6);
double activeP = Math.max(0, (needP - soilP) / 0.25);
double activeK = Math.max(0, (needK - soilK) / 0.5);
```

Рисунок 3.10 – Розрахунок потенціалу та потреби (закон Лібіха)

Другим компонентом підтримки рішень є рекомендація культури-послідовника, реалізована у контролері RotationController. Логіку побудовано на базі агрономічних правил чергування культур, оформлених як оператор вибору за кодом попередньої культури. Для кожного варіанта формується перелік рекомендованих культур з якісною оцінкою та поясненням, а для соняшнику додатково

генерується застереження. Фрагмент реалізації наведено на рисунку 3.11.

```
switch (lastCrop) {
    case "SUNFLOWER":
        addRec(recommendations, "CORN", "Чудово", "...");
        addRec(recommendations, "WHEAT", "Добре", "...");
        result.put("warning",
            "Соняшник не можна повертати на поле 7 років.");
        break;
    case "SOYBEAN":
        addRec(recommendations, "WHEAT", "Ідеально", "...");
        addRec(recommendations, "CORN", "Чудово", "...");
        break;
    default:
        addRec(recommendations, "WHEAT", "Універсально", "...");
}
```

Рисунок 3.11 – Фрагмент логіки рекомендацій сівозміни

3.5.4 Клієнтська частина та інтерфейс користувача

Клієнтську частину реалізовано як односторінковий вебзастосунок. Інтерактивну карту побудовано засобами Leaflet.js; для малювання контурів полів використано модуль Leaflet.draw, а площу намальованого контуру обчислено бібліотекою Turf.js. Обмін даними із сервером відбувається асинхронно через інтерфейс Fetch API у форматі JSON. Характерні фрагменти клієнтського коду наведено на рисунку 3.12.

```
// Ініціалізація карти та інструмента малювання
addFieldMap = L.map('addFieldMap').setView([48.3794, 31.1656], 6);
drawnItems = new L.FeatureGroup();
drawControl = new L.Control.Draw({ /* налаштування */ });

// Обчислення площі контуру (Turf.js)
const areaSquareMeters = turf.area(geojson);

// Запит на розрахунок норм добрив
const res = await fetch('/api/calculator/calculate', {
   method: 'POST',
   headers: { 'Content-Type': 'application/json' },
   body: JSON.stringify(req)
});
```

Рисунок 3.12 – Фрагменти клієнтської частини

Інтерфейс користувача організовано у вигляді функціональних екранів: вікна автентифікації, головного екрана з картою полів, форм перегляду та введення агрохімічних аналізів, екрана розрахунку норм добрив і екрана рекомендацій сівозміни. Зовнішній вигляд основних екранних форм продемонстровано далі.

3.6 Розробка модульних тестів

Для перевірки коректності роботи серверної частини розроблено модульні та інтеграційні тести з використанням фреймворку JUnit 5 та засобів Spring Boot Test. Інтеграційне тестування REST-API виконано за допомогою компонента MockMvc, що дає змогу надсилати HTTP-запити до контролерів без запуску реального вебсервера та перевіряти коректність відповідей. Першочергово протестовано компонент

розрахунку норм добрив як такий, що становить основну логічну цінність системи.

Один із тестів перевіряє граничний випадок – запит на розрахунок для поля, що не має жодного агрохімічного аналізу: очікується, що система поверне відповідь із полем помилки, а не аварійно завершиться. Реалізацію тесту наведенона рисунку 3.13.

```
@SpringBootTest
@AutoConfigureMockMvc
class CalculatorControllerTest {
   @Autowired private MockMvc mockMvc;

   @Test
   void calculate_withoutAnalyses_returnsError() throws Exception {
      String body = "{\"fieldId\":999,\"cropType\":\"WHEAT\"}";
      mockMvc.perform(post("/api/calculator/calculate")
            .contentType(MediaType.APPLICATION_JSON)
            .content(body))
         .andExpect(status().isOk())
         .andExpect(jsonPath("$.error").exists());
   }
}
```

Рисунок 3.13 – Тест компонента розрахунку норм добрив

Аналогічно розроблено тести для перевірки коректності розрахунку доз за наявності аналізу, роботи похідних методів репозиторіїв (findByFarmerId, findByFieldId) та основних операцій контролерів полів і аналізів. Усі розроблені тести успішно пройдено, що підтверджує відповідність поведінки системи проєктним вимогам. Результати виконання набору тестів наведено на рисунку 3.14.

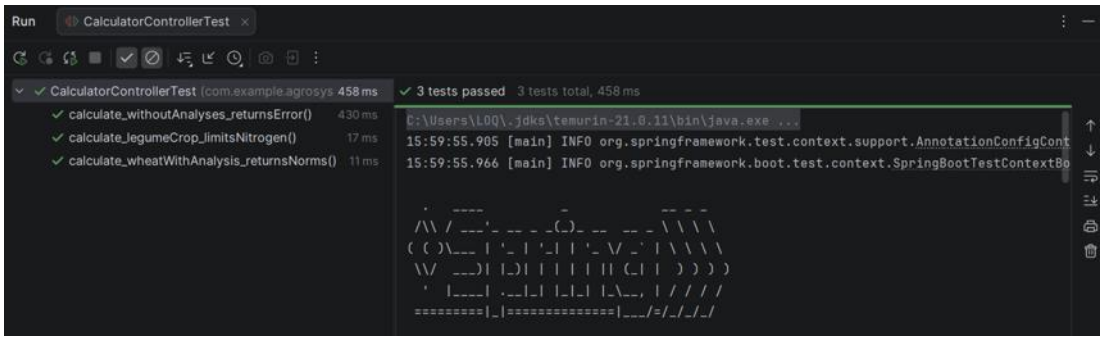

Рисунок 3.14 – Результати виконання модульних тестів

3.7 Демонстрація роботи програмного комплексу

Роботу програмного комплексу продемонстровано на наскрізному сценарії використання, що охоплює всі основні функції системи – від реєстрації користувача до отримання агротехнічних рекомендацій.

Робота розпочинається з реєстрації нового фермера або входу до системи через форму автентифікації (рисунок 3.15). Після входу користувач потрапляє на головний екран, де його поля відображаються на інтерактивній карті.

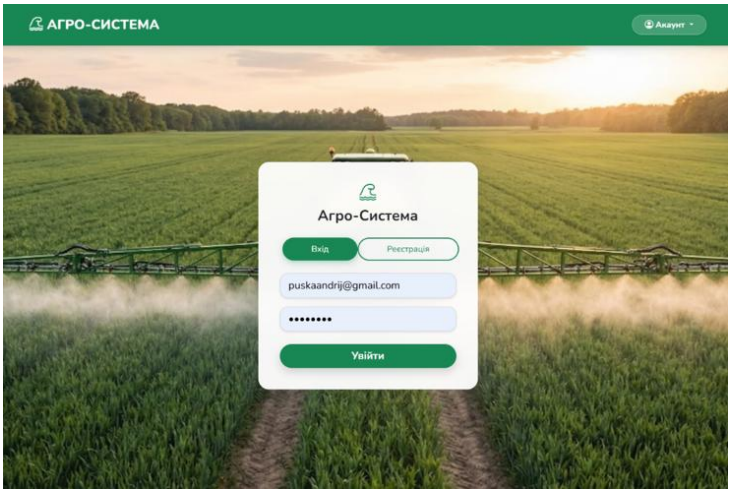

Рисунок 3.15 – Екран реєстрації та входу до системи

Для додавання нового поля користувач малює його контур безпосередньо на карті, після чого площа поля обчислюється автоматично; створене поле зберігається та відображається у переліку (рисунок 3.16).

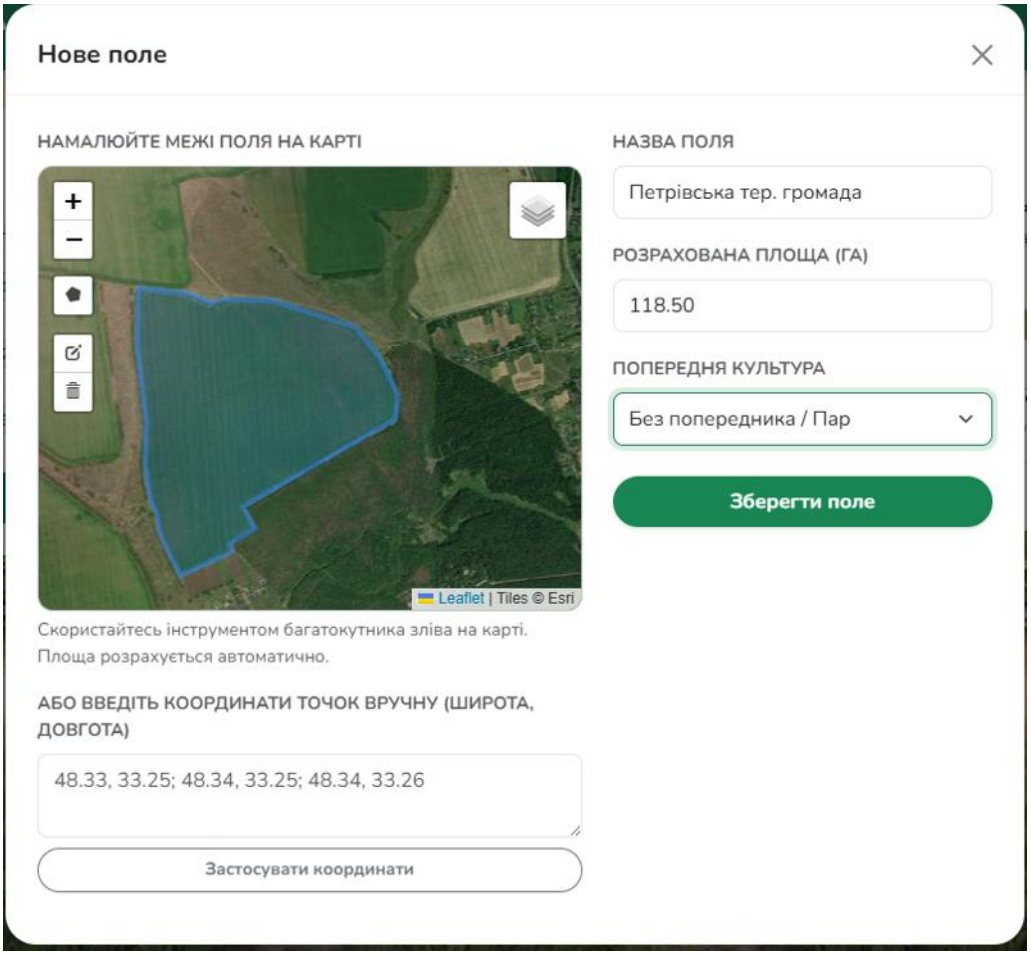

Рисунок 3.16 – Додавання поля та відображення його на карті

Далі для обраного поля користувач вносить результати агрохімічного обстеження ґрунту – вміст азоту, фосфору, калію та рівень кислотності (рисунок 3.17).

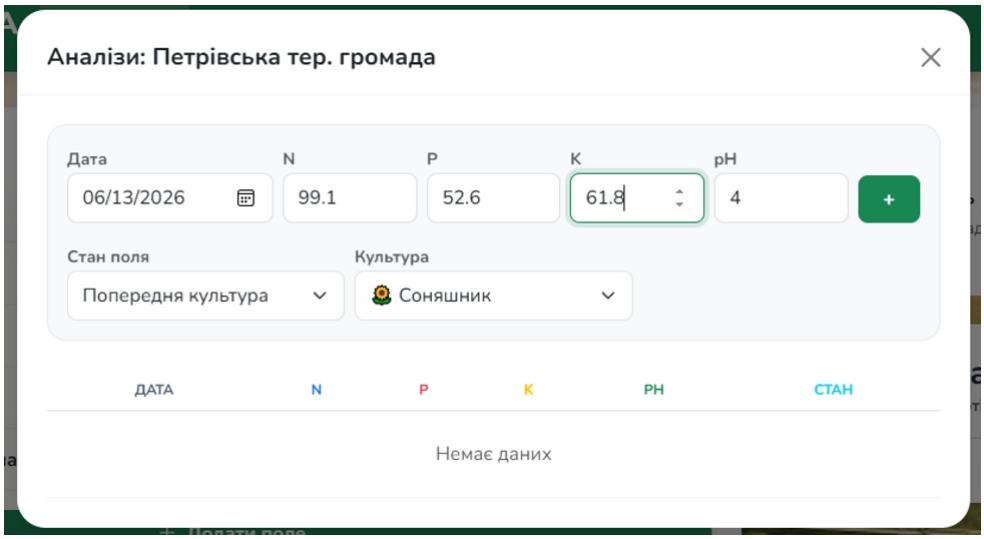

Рисунок 3.17 – Форма введення агрохімічного аналізу

На основі введеного аналізу та обраної культури система виконує розрахунок норм мінеральних добрив за законом Лібіха та відображає рекомендовані дози, лімітуючий фактор і план внесення (рисунок 3.18).

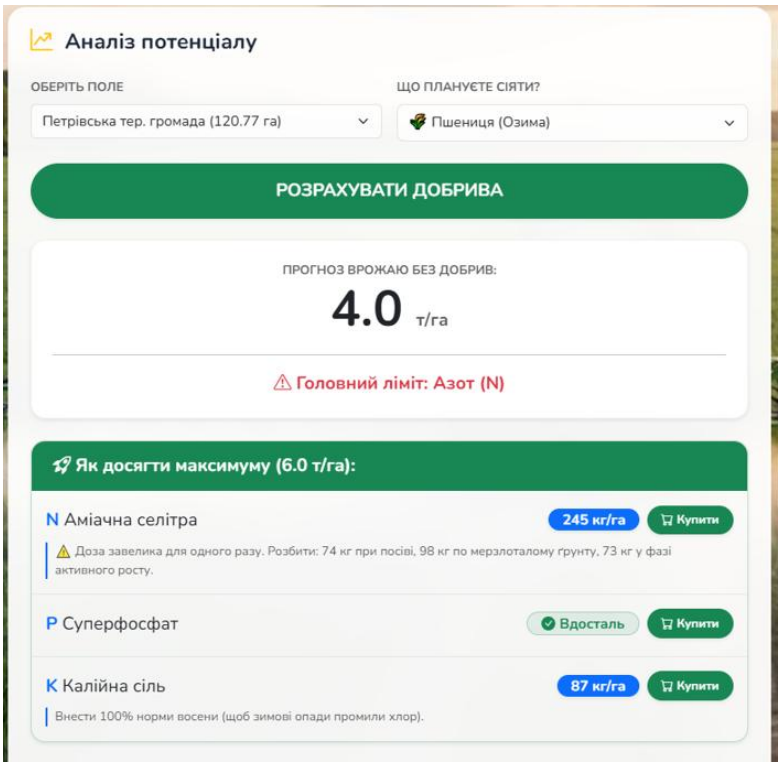

Рисунок 3.18 – Результат розрахунку норм добрив

Насамкінець користувач може отримати рекомендацію щодо чергування культур: система виводить перелік рекомендованих культур-послідовників з оцінкою сумісності та застереженнями (рисунок 3.19).

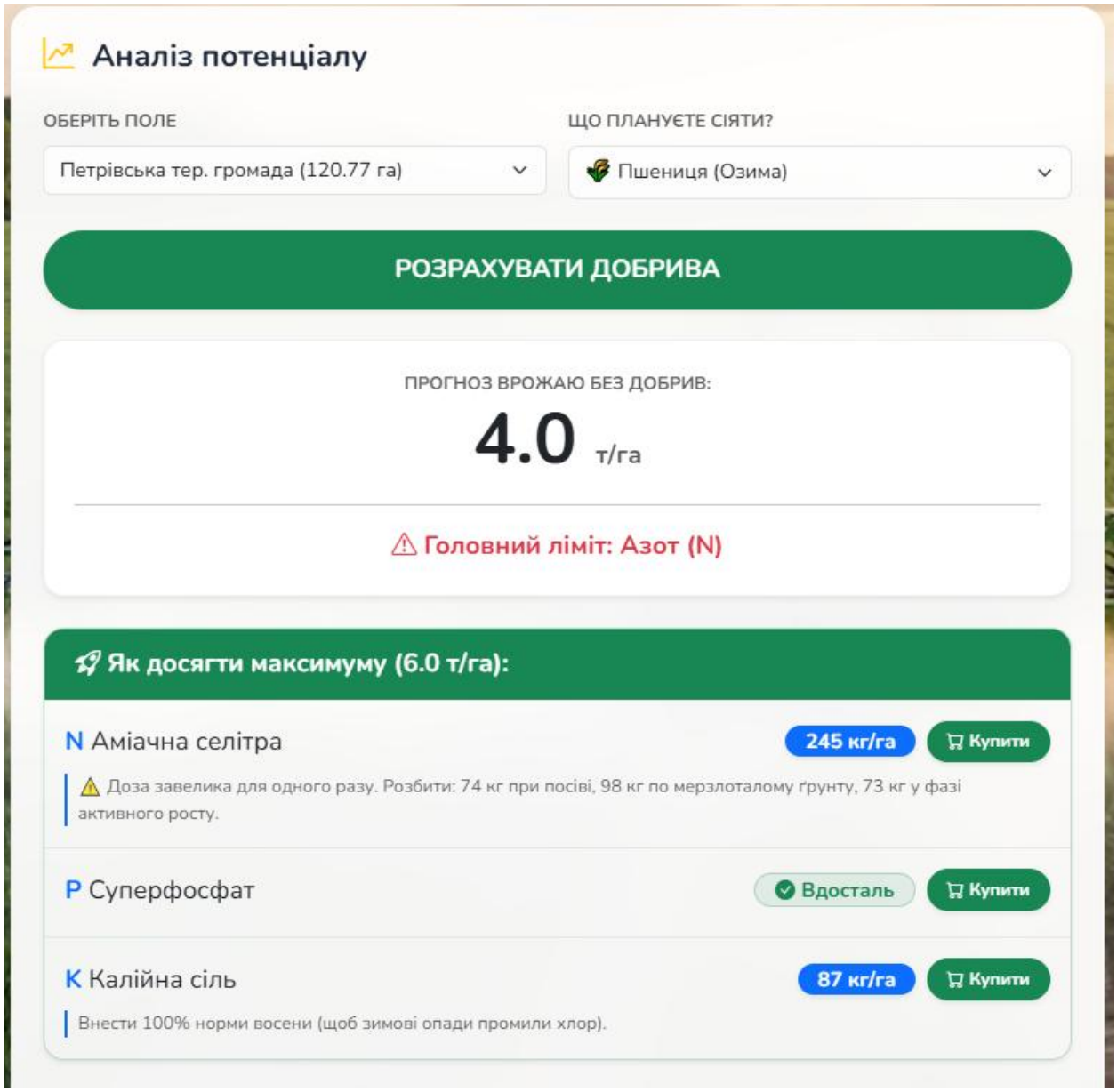

Рисунок 3.19 – Рекомендація сівозміни (буде додано)

Наведений сценарій підтверджує, що програмний комплекс реалізує повний цикл інформаційної підтримки агротехнічних рішень та відповідає визначеним функціональним вимогам.

3.8 Результат проектування програмного комплексу

Виконано повний цикл об'єктно-орієнтованого проєктування та програмної реалізації комплексу для підтримки агротехнічних рішень із застосуванням мови UML. Розроблено модель прецедентів використання з єдиним актором Фермером, а також побудовано діаграму класів трирівневої архітектури, що містить класи-сутності, репозиторії та контролери бізнес-логіки. Розроблено діаграми послідовностей для ключових сценаріїв підтримки рішень – розрахунку норм добрив та рекомендації сівозміни. Спроєктовано реляційну базу даних із чотирьох таблиць під керуванням PostgreSQL, схему якої доведено до нормальної форми Бойса – Кодда для мінімізації надмірності даних.

На основі спроєктованих моделей успішно виконано програмну реалізацію комплексу на базі Java 21, Spring Boot, Spring Data JPA та PostgreSQL із клієнтською частиною на Leaflet.js і Turf.js. Реалізовано ключові алгоритми підтримки рішень – розрахунок норм добрив за законом Лібіха та рекомендацію сівозміни. Коректність роботи підтверджено успішним проходженням модульних та інтеграційних тестів, а демонстрація на наскрізному сценарії доводить, що отриманий програмний комплекс є повністю готовим до практичного використання.

4 ЕКОНОМІЧНІ РОЗРАХУНКИ

Метою цього розділу є економічне обґрунтування доцільності розроблення програмно-методичного комплексу «АгроСистема». У межах розділу визначено трудомісткість створення програмного продукту, розраховано кошторис витрат на його розроблення з урахуванням фактичних грошових витрат розробника, а також оцінено економічну ефективність упровадження системи порівняно з наявними комерційними аналогами класу FMIS.

Особливістю цього проєкту є те, що весь цикл розроблення – від проєктування архітектури до програмної реалізації та тестування – виконано одним розробником (здобувачем) самостійно, без залучення сторонніх виконавців. Унаслідок цього фактичні грошові витрати на створення прототипу обмежилися оплатою кількох хмарних сервісів та програмних підписок, тоді як власна праця розробника й використання особистого обладнання враховуються в кошторисі як умовні (внутрішні) витрати. Такий підхід відповідає поширеній практиці оцінювання собівартості навчальних і стартап-проєктів, що розробляються силами однієї особи.

4.1 Визначення трудомісткості розроблення програмного продукту

Трудомісткість розроблення програмного продукту визначено методом поетапного оцінювання витрат робочого часу на кожному з технологічних етапів життєвого циклу системи. Підставою для оцінювання слугували фактичний обсяг реалізованого коду, кількість розроблених REST-контролерів, складність алгоритмів підтримки

прийняття рішень (розрахунок норм добрив за законом Лібіха, рекомендації сівозміни), а також обсяг клієнтської частини з інтеграцією геоінформаційної підсистеми.

Сумарну трудомісткість розроблення T обчислено як суму витрат часу на окремих етапах за формулою 4.1.

$$T = \Sigma t_i , \qquad (4.1)$$

де t_i – трудомісткість і-го етапу розроблення, год.

Розподіл трудомісткості за етапами розроблення наведено в таблиці 4.1.

Таблиця 4.1 – Трудомісткість етапів розроблення комплексу «АгроСистема»

Етап розроблення	Зміст робіт	Трудомісткість, год
Дослідження предметної області	Аналіз технологій точного землеробства, порівняння FMIS-аналогів	40
Складання технічного завдання	Формування функціональних і нефункціональних вимог	24
Проєктування системи	Розроблення UML-діаграм, архітектури, моделі даних	56
Розроблення бази даних	Проєктування схеми PostgreSQL, JPA-сутностей	40
Реалізація серверної частини	REST-контролери, репозиторії, бізнес-логіка	120
Реалізація алгоритмів СППР	Розрахунок за Лібіхом, рекомендації сівозміни	64

Продовження таблиці 4.1

Етап розроблення	Зміст робіт	Трудомісткість, год
Реалізація клієнтської частини	SPA на JavaScript, Bootstrap 5	80
Інтеграція ГІС-підсистеми	Leaflet.js, Leaflet.draw, Turf.js	48
Тестування та налагодження	Модульні й інтеграційні тести (JUnit 5)	56
Оформлення документації	Підготовка пояснювальної записки	40
Разом	–	568

Отже, загальна трудомісткість розроблення програмного комплексу становить T = 568 год. Найбільш трудомісткими очікувано виявилися етапи реалізації серверної частини (120 год) та клієнтської частини (80 год), що пояснюється значним обсягом програмного коду REST-API та односторінкового застосунку, а також інтеграцією зовнішніх бібліотек.

4.2 Розрахунок витрат на оплату праці розробника

Оскільки розроблення виконувалося здобувачем самостійно, витрати на оплату праці розраховано як умовну вартість праці розробника рівня Junior Java Developer за середньоринковою ставкою станом на 2026 рік. За даними кадрових майданчиків ринку праці, місячна заробітна плата фахівця такого рівня в Україні становить близько 30 000 грн за умови місячного фонду робочого часу 168 год.

Годинну тарифну ставку розробника Сгод обчислено за формулою 4.2.

$$Сгод = Зм / Фм , \quad (4.2)$$

де Зм – місячна заробітна плата розробника, грн; Фм – місячний фонд робочого часу, год.

$$Сгод = 30\,000 / 168 = 178{,}57 \text{ грн/год.}$$

Основну заробітну плату розробника Зосн визначено як добуток годинної ставки на трудомісткість розроблення за формулою 4.3:

$$Зосн = Сгод \times Т , \quad (4.3)$$

$$Зосн = 178{,}57 \times 568 = 101\,428{,}57 \text{ грн.}$$

Додаткову заробітну плату Здод (премії, доплати) прийнято на рівні 10 % від основної за формулою 4.4.

$$Здод = Зосн \times 0{,}10 = 10\,142{,}86 \text{ грн.} \quad (4.4)$$

Загальний фонд оплати праці розробника ФОП становить 4.5.

$$ФОП = Зосн + Здод = 101\,428{,}57 + 10\,142{,}86 = 111\,571{,}43 \text{ грн.} \quad (4.5)$$

4.3 Відрахування на соціальні заходи

Відповідно до чинного законодавства України роботодавець сплачує єдиний внесок на загальнообов’язкове державне соціальне

страхування (ЄСВ) у розмірі 22 % від фонду оплати праці. Суму відрахувань Зсоц обчислено за формулою 4.6.

$$Зсоц = ФОП \times 0,22 = 111\,571,43 \times 0,22 = 24\,545,71 \text{ грн.} \quad (4.6)$$

4.4 Розрахунок витрат на програмне забезпечення та хмарні сервіси

Ключовою статтею фактичних грошових витрат на розроблення комплексу стала оплата програмних підписок та хмарних сервісів, що використовувалися протягом усього періоду розроблення. Календарна тривалість активної фази розроблення становила 4 місяці, тому вартість усіх сервісів з помісячною тарифікацією обчислено саме за цей період. До складу використаного платного інструментарію увійшли такі сервіси:

- інтегроване середовище розроблення JetBrains IntelliJ IDEA Ultimate – основний інструмент написання, рефакторингу та налагодження коду серверної частини на Java 21 / Spring Boot; вартість індивідуальної підписки – 19,90 USD/міс;
- сервіс штучного інтелекту Claude AI (план Pro) – використовувався як інтелектуальний асистент розробника для прискорення написання коду, генерування модульних тестів, пояснення повідомлень про помилки та підготовки технічної документації; вартість – 20,00 USD/міс;
- хостинг сервера (VPS) – віртуальний приватний сервер для розгортання серверної частини Spring Boot та публікації працездатного прототипу системи; вартість – 6,00 USD/міс;

– хмарна база даних PostgreSQL – керований екземпляр реляційної СКБД для зберігання даних фермерів, полів та агрохімічних аналізів; вартість – 15,00 USD/міс.

Вибір платної редакції IntelliJ IDEA Ultimate (замість безкоштовної Community) зумовлено потребою у повноцінній підтримці фреймворку Spring Boot, інтегрованих інструментах роботи з базою даних та засобах розширеного налагодження веб-застосунків, які доступні лише в комерційній редакції. Решта компонентів технологічного стеку (Java, Spring Boot, PostgreSQL, Leaflet.js, Turf.js, Bootstrap) є вільно поширюваними та не потребували оплати ліцензій.

Розрахунок витрат у доларовому еквіваленті та гривнях наведено в таблиці 4.2. Перерахунок здійснено за курсом Національного банку України станом на червень 2026 року – 44,50 грн/USD.

Таблиця 4.2 – Витрати на програмне забезпечення та хмарні сервіси

Сервіс / підписка	Вартість, USD/міс	Період, міс	Сума, USD	Сума, грн
JetBrains IntelliJ IDEA Ultimate	19,90	4	79,60	3 542,20
Claude AI (Pro)	20,00	4	80,00	3 560,00
Хостинг сервера (VPS)	6,00	4	24,00	1 068,00
Хмарна база даних PostgreSQL	15,00	4	60,00	2 670,00
Разом	–	–	243,60	10 840,20

Таким чином, сумарні фактичні грошові витрати розробника на програмне забезпечення та хмарну інфраструктуру протягом усього періоду розроблення склали 243,60 USD, або 10 840,20 грн. Саме ці

витрати становлять реальну суму коштів, яку здобувач витратив безпосередньо на створення прототипу системи.

4.5 Розрахунок витрат на електроенергію

Під час розроблення використовувався персональний ноутбук розробника із середньою споживаною потужністю близько 0,12 кВт. Загальний час роботи обладнання прийнято рівним трудомісткості розроблення – 568 год. Витрати електроенергії в натуральному вираженні W обчислено за формулою 4.7.

$$W = P \times T = 0{,}12 \times 568 = 68{,}16 \text{ кВт·год}, \quad (4.7)$$

де P – споживана потужність обладнання, кВт; T – час роботи обладнання, год.

Вартість спожитої електроенергії Вел за чинним тарифом для побутових споживачів 4,32 грн/кВт·год становить 4.8.

$$\text{Вел} = W \times \text{Цел} = 68{,}16 \times 4{,}32 = 294{,}45 \text{ грн}. \quad (4.8)$$

4.6 Амортизація обладнання

Для розроблення використовувався особистий ноутбук розробника вартістю 35 000 грн. Амортизацію нараховано прямолінійним методом за умови строку корисного використання 4 роки

(48 місяців). Місячну суму амортизації Ам.міс обчислено за формулою 4.9.

$$Ам.міс = Вобл / Тсл = 35\ 000 / 48 = 729{,}17 \text{ грн/міс}, \quad (4.9)$$

де Вобл – вартість обладнання, грн; Тсл – строк корисного використання, міс.

За весь період розроблення (4 місяці) сума амортизаційних відрахувань Ам становить 4.10.

$$Ам = Ам.міс \times tроз = 729{,}17 \times 4 = 2\ 916{,}67 \text{ грн}. \quad (4.10)$$

4.7 Накладні витрати

Накладні витрати, пов’язані з організацією робочого процесу (інтернет-зв’язок, обслуговування робочого місця, інші непрямі витрати), прийнято на рівні 20 % від основної заробітної плати розробника за формулою 4.11).

$$Внакл = Зосн \times 0{,}20 = 101\ 428{,}57 \times 0{,}20 = 20\ 285{,}71 \text{ грн}. \quad (4.11)$$

4.8 Кошторис витрат на розроблення

На підставі виконаних розрахунків сформовано кошторис повної собівартості розроблення програмного комплексу «АгроСистема», який наведено в таблиці 4.3.

Таблиця 4.3 – Кошторис витрат на розроблення комплексу

№	Стаття витрат	Сума, грн	Частка, %
1	Основна заробітна плата розробника	101 428,57	59,50
2	Додаткова заробітна плата	10 142,86	5,95
3	Відрахування на соціальні заходи (ЄСВ)	24 545,71	14,40
4	Програмне забезпечення та хмарні сервіси	10 840,20	6,36
5	Витрати на електроенергію	294,45	0,17
6	Амортизація обладнання	2 916,67	1,71
7	Накладні витрати	20 285,71	11,90
	Разом (повна собівартість)	170 454,17	100,00

Отже, повна собівартість розроблення комплексу з урахуванням умовної вартості праці розробника становить 170 454,17 грн. Водночас слід наголосити, що понад 90 % цієї суми (статті 1–3 та 7) є умовними внутрішніми витратами, які відображають власну працю здобувача та використання особистого обладнання й не потребували реальних грошових видатків.

Фактичні ж грошові витрати розробника обмежилися оплатою програмних підписок та хмарних сервісів (стаття 4) і незначними витратами на електроенергію (стаття 5), що сумарно складає лише 11 134,65 грн. Це підтверджує високу економічну доступність обраного підходу до розроблення: повнофункціональний прототип веб-орієнтованої СППР було створено силами однієї особи за мінімальних капіталовкладень.

Для порівняння: у разі замовлення аналогічної розробки у сторонньої компанії-підрядника за середньоринковою ставкою

комерційної розробки програмного забезпечення 500 грн/год вартість виконання тих самих 568 год робіт склала б близько 284 000 грн без урахування податків та маржі підрядника. Таким чином, самостійне розроблення комплексу силами здобувача дозволило уникнути значних витрат і отримати працездатний продукт за символічну для галузі суму.

4.9 Оцінка економічної ефективності впровадження

Економічний ефект від упровадження «АгроСистеми» формується за двома напрямами: по-перше, пряма економія коштів фермерського господарства завдяки оптимізації норм внесення мінеральних добрив за принципом лімітуючого фактора Лібіха; по-друге, уникнення витрат на придбання ліцензії комерційної FMIS-платформи, оскільки розроблений комплекс є вільно поширюваним.

Для оцінювання ефекту розглянуто модельне фермерське господарство площею 120 га – типове мале господарство, орієнтоване на впровадження елементів точного землеробства. Усереднені витрати на мінеральні добрива та засоби захисту рослин для такого господарства прийнято на рівні 7 000 грн/га на рік, що відповідає поточній кон’юнктурі ринку агрохімії.

Річні витрати господарства на добрива та ЗЗР Взаг становлять (формула 4.12):

$$Взаг = Цга \times S = 7\,000 \times 120 = 840\,000 \text{ грн/рік}, \qquad (4.12)$$

де Цга – витрати на добрива та ЗЗР на гектар, грн; S – площа господарства, га.

Згідно з аналітичними оцінками (підрозділ 1.1), застосування технологій точного землеробства дозволяє зменшити витрати на добрива та ЗЗР у середньому на 15 %. Річну економію Едобр від диференційованого внесення добрив за рекомендаціями системи обчислено за формулою 4.13.

$$Едобр = Взаг \times 0{,}15 = 840\,000 \times 0{,}15 = 126\,000 \text{ грн/рік}. \quad (4.13)$$

Додатково господарство уникає щорічних витрат на ліцензію комерційної FMIS-платформи. За даними порівняльного аналізу (таблиця 1.3), вартість такої ліцензії становить від 500 до 1200 USD/рік; для розрахунку прийнято середнє значення 700 USD/рік. Економію на ліцензії Еліц у гривневому еквіваленті обчислено за формулою 4.14.

$$Еліц = 700 \times 44{,}50 = 31\,150 \text{ грн/рік}. \quad (4.14)$$

Сумарний річний економічний ефект від упровадження системи Ер становить (формула 4.15).

$$Ер = Едобр + Еліц = 126\,000 + 31\,150 = 157\,150 \text{ грн/рік}. \quad (4.15)$$

Коефіцієнт економічної ефективності капіталовкладень Е обчислено як відношення річного ефекту до повної собівартості розроблення за формулою 4.16.

$$Е = Ер / Своб = 157\,150 / 170\,454{,}17 = 0{,}92. \quad (4.16)$$

Розрахункове значення Е = 0,92 істотно перевищує нормативний коефіцієнт ефективності Ен = 0,15, що свідчить про економічну

доцільність розроблення. Строк окупності витрат на розроблення Ток обчислено за формулою 4.17.

$$Ток = Своб / Ер = 170\,454{,}17 / 157\,150 = 1{,}08 \text{ року}. \quad (4.17)$$

Отриманий строк окупності значно менший за нормативний Тн = 1 / Ен = 6,67 року, що додатково підтверджує ефективність проєкту. Якщо ж розглядати окупність лише фактичних грошових витрат розробника (11 134,65 грн), то вона настає вже протягом першого виробничого сезону – менш ніж за один місяць використання системи.

Слід зауважити, що економічний ефект від упровадження системи прямо пропорційний площі господарства: чим більша оброблювана площа, тим швидше окуповуються витрати на розроблення. Залежність річного ефекту та строку окупності повної собівартості від площі господарства наведено в таблиці 4.4.

Таблиця 4.4 – Залежність економічного ефекту від площі господарства

Площа, га	Економія на добривах, грн/рік	Річний ефект Ер, грн/рік	Строк окупності, років
50	52 500	83 650	2,04
120	126 000	157 150	1,08
250	262 500	293 650	0,58

Як видно з таблиці 4.4, навіть для невеликого господарства площею 50 га витрати на розроблення окуповуються приблизно за два роки, а для господарства площею 250 га – менш ніж за один виробничий сезон. Узагальнені показники економічної ефективності наведено в таблиці 4.5.

Таблиця 4.5 – Показники економічної ефективності проєкту

Показник	Значення
Повна собівартість розроблення, грн	170 454,17
Фактичні грошові витрати розробника, грн	11 134,65
Річний економічний ефект, грн	157 150,00
Коефіцієнт економічної ефективності E	0,92
Нормативний коефіцієнт ефективності Ен	0,15
Строк окупності Ток, років	1,08
Нормативний строк окупності Тн, років	6,67

4.10 Результати економічних розрахунків

Виконано економічне обґрунтування розроблення програмно-методичного комплексу «АгроСистема». Методом поетапного оцінювання визначено трудомісткість розроблення, яка становить 568 людино-годин; найбільш трудомісткими виявилися етапи реалізації серверної та клієнтської частин системи.

Сформовано кошторис витрат, згідно з яким повна собівартість розроблення комплексу з урахуванням умовної вартості праці розробника та використання власного обладнання становить 170 454,17 грн. Водночас встановлено, що фактичні грошові витрати здобувача обмежилися оплатою чотирьох програмних підписок і хмарних сервісів (IntelliJ IDEA Ultimate, Claude AI Pro, хостинг сервера та хмарна база даних PostgreSQL) і склали лише 10 840,20 грн, а з урахуванням електроенергії – 11 134,65 грн. Це свідчить про винятково низький поріг входу для розроблення подібних рішень силами однієї особи.

Оцінка економічної ефективності показала, що впровадження системи в модельному господарстві площею 120 га забезпечує річний економічний ефект 157 150 грн за рахунок оптимізації внесення добрив та уникнення витрат на комерційну ліцензію. Коефіцієнт економічної ефективності E = 0,92 перевищує нормативне значення більш ніж у шість разів, а строк окупності повної собівартості становить лише 1,08 року. Отримані показники підтверджують економічну доцільність та інвестиційну привабливість розробленого програмно-методичного комплексу.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-прикладне завдання – розроблення веб-орієнтованого програмно-методичного комплексу підтримки прийняття рішень «АгроСистема» для потреб точного землеробства, адаптованого до вітчизняних агрохімічних нормативів. Поставлену мету досягнуто в повному обсязі: створено працездатний прототип системи, що охоплює весь цикл інформаційного забезпечення агротехнічних рішень – від реєстрації полів і ведення агрохімічних аналізів до розрахунку норм мінеральних добрив та формування рекомендацій із сівозміни. У ході виконання роботи отримано такі основні результати.

1. Проведено аналіз предметної галузі точного землеробства та обґрунтовано актуальність задачі. Установлено, що елементи точного землеробства на системній основі застосовують лише 12–15 % українських агропідприємств проти понад 40 % у країнах ЄС, а головними бар’єрами впровадження є відсутність доступних україномовних рішень, адаптованих до вітчизняних нормативів, висока вартість зарубіжних ліцензій та обмеження доступу в умовах воєнного часу. На основі аналізу бізнес-процесів фермерського господарства виокремлено п’ять ключових процесів, які автоматизує розроблена система, та визначено функції, ролі й сценарії використання основного актора – Фермера (Агронома).

2. Виконано порівняльний аналіз репрезентативних програмних аналогів класу FMIS (Climate FieldView, Cropwise Operations, Soft.Farm, Agrivi). За його результатами обґрунтовано наявність вільної ніші для open-source веб-системи з україномовним інтерфейсом, яка реалізує розрахунок норм добрив за принципом лімітуючого фактора Лібіха та

повну адаптацію до вітчизняних агрохімічних нормативів, чого бракує наявним на ринку рішенням.

3. Обґрунтовано вибір технологічного стеку для веб-орієнтованої архітектури. Систему побудовано як монолітний веб-застосунок на платформі Java 21 із застосуванням Spring Boot 3 та Spring Data JPA для серверної частини й СКБД PostgreSQL для зберігання даних. Клієнтську частину реалізовано у вигляді односторінкового застосунку (SPA) на чистому JavaScript із використанням бібліотек Bootstrap 5, Leaflet.js 1.9, Leaflet.draw та Turf.js, що забезпечує доступність усього функціоналу безпосередньо у браузері без потреби в десктопних чи мобільних клієнтах.

4. Спроектовано реляційну модель даних та схему бази даних, побудовано ER-діаграму, виконано нормалізацію таблиць і визначено ключі та зв’язки між сутностями. На основі фактичного програмного коду розроблено комплект UML-моделей: діаграму прецедентів із відношеннями include/extend, діаграму класів, діаграми послідовностей для ключових сценаріїв та діаграми діяльності, що документують реалізовану логіку системи.

5. Реалізовано серверну частину системи у вигляді REST-API на основі Spring Boot. Розроблено контролери FarmerController, FieldController, SoilAnalysisController, CalculatorController та RotationController, які повністю покривають усі ключові сценарії: реєстрацію та автентифікацію фермера, керування полями та їхньою геометрією, ведення агрохімічних аналізів ґрунту, розрахунок норм добрив і формування рекомендацій із сівозміни.

6. Реалізовано клієнтську частину з інтегрованою інтерактивною ГІС-підсистемою на базі Leaflet.js і Turf.js. Фермер має змогу окреслювати контури полів безпосередньо на карті засобами Leaflet.draw або вводити координати вершин полігона вручну; площу поля автоматично обчислює бібліотека Turf.js (метод turf.area), а

геометрію зберігають у форматі GeoJSON. Усі поля користувача відображаються на спільній інтерактивній карті.

7. Реалізовано ключовий алгоритм розрахунку норм внесення мінеральних добрив за принципом лімітуючого фактора Лібіха. Алгоритм охоплює виведення нормативів виносу та максимальної врожайності, розрахунок запасу елементів живлення у ґрунті, визначення потенціалу врожаю й лімітуючого фактора, обчислення потреби та дефіциту діючої речовини з подальшим перерахунком у фізичну масу добрив і формуванням плану дробового внесення азоту. Алгоритм підтримує тринадцять сільськогосподарських культур.

8. Реалізовано алгоритм рекомендацій сівозміни, який на основі останньої вирощуваної на полі культури формує рекомендації культур-наступниць із якісною оцінкою та пояснювальним текстом, а також генерує попередження про обов’язкову паузу повернення для соняшнику. Для перевірки коректності ключових обчислень розроблено комплект модульних тестів, що підтверджують відповідність роботи алгоритмів очікуваним результатам.

9. Виконано економічне обґрунтування розроблення комплексу. Трудомісткість розроблення визначено на рівні 568 людино-годин, а повну собівартість – 170 454,17 грн, тоді як фактичні грошові витрати здобувача склали лише 11 134,65 грн, що свідчить про винятково низький поріг входу для створення подібних рішень силами однієї особи. Оцінка ефективності впровадження для модельного господарства площею 120 га показала річний економічний ефект 157 150 грн, коефіцієнт економічної ефективності $E = 0,92$ (за нормативного $Ен = 0,15$) та строк окупності повної собівартості 1,08 року, що підтверджує економічну доцільність та інвестиційну привабливість проєкту.

Практичне значення роботи полягає в тому, що створено готовий до подальшого розширення повнофункціональний прототип веб-

орієнтованої системи підтримки прийняття рішень з повним стеком (база даних, REST-сервер та SPA-клієнт), який реалізує науково обґрунтований розрахунок норм добрив за законом мінімуму Лібіха, має україномовний інтерфейс і є вільно поширюваним. Це робить його доступним інструментом цифровізації передусім для малих та середніх фермерських господарств.

Перспективними напрямами подальшого розвитку системи є запровадження багатокористувацького режиму з розмежуванням ролей, інтеграція даних дистанційного зондування Землі та вегетаційних індексів (зокрема NDVI), розроблення мобільного клієнта, накопичення й аналітика історичних даних за виробничими сезонами, а також розширення довідників культур і добрив та інтеграція погодних прогнозів безпосередньо в модулі планування агрозаходів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Морозов В. В., Морозов О. В., Козленко Є. В. Точне землеробство : навчальний посібник. Херсон : ХДАЕУ, 2021. 234 с.

2. Бойко П. І., Коваленко Н. П. Сівозміни у землеробстві України. Київ : Аграрна наука, 2019. 256 с.

3. Господаренко Г. М. Агрохімія : підручник. 2-ге вид. Київ : ТОВ «Сік Груп Україна», 2018. 560 с.

4. Лісовал А. П., Макаренко В. М., Кравченко С. М. Система застосування добрив : підручник. Київ : Вища школа, 2018. 317 с.

5. Балюк С. А., Медведєв В. В., Тараріко О. Г. Ґрунтові ресурси України: стан і заходи їх поліпшення. Вісник аграрної науки. 2020. № 6. С. 5–14.

6. Тараріко Ю. О., Сайдак Р. В. Сучасні системи землеробства та агротехнології. Київ : ДІА, 2019. 196 с.

7. Walls C. Spring in Action. 6th ed. Shelter Island, NY : Manning Publications, 2022. 520 p.

8. Spring Boot Reference Documentation : official documentation. Broadcom Inc. URL: https://docs.spring.io/spring-boot/docs/current/reference/html/ (дата звернення: 10.05.2026).

9. Bauer C., King G., Gregory G. Java Persistence with Hibernate. 2nd ed. Shelter Island, NY : Manning Publications, 2016. 608 p.

10. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2018. 533 p.

11. Fielding R. T., Taylor R. N. Principled Design of the Modern Web Architecture. ACM Transactions on Internet Technology. 2002. Vol. 2, No. 2. P. 115–150.

12. Richardson L., Amundsen M., Ruby S. RESTful Web APIs. Sebastopol, CA : O'Reilly Media, 2013. 406 p.

13. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 14th ed. Boston : Cengage Learning, 2024. 818 p.

14. PostgreSQL 16 Documentation : official documentation. The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/16/ (дата звернення: 10.05.2026).

15. Date C. J. An Introduction to Database Systems. 8th ed. Boston : Addison-Wesley, 2019. 1024 p.

16. Fowler M., Scott K. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston : Addison-Wesley, 2018. 208 p.

17. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd ed. Upper Saddle River, NJ : Prentice Hall, 2017. 736 p.

18. Crockford D. The application/json Media Type for JavaScript Object Notation (JSON) : RFC 4627. Internet Engineering Task Force, 2006. URL: https://www.rfc-editor.org/rfc/rfc4627 (дата звернення: 10.05.2026).

19. Butler H., Daly M., Doyle A., Gillies S., Hagen S., Schaub T. The GeoJSON Format : RFC 7946. Internet Engineering Task Force, 2016. URL: https://www.rfc-editor.org/rfc/rfc7946 (дата звернення: 10.05.2026).

20. Leaflet – a JavaScript library for interactive maps : official documentation. URL: https://leafletjs.com/reference.html (дата звернення: 10.05.2026).

21. Turf.js – Advanced geospatial analysis for browsers and Node.js : official documentation. URL: https://turfjs.org/docs/ (дата звернення: 10.05.2026).

22. Bootstrap 5 : official documentation. URL: https://getbootstrap.com/docs/5.3/ (дата звернення: 10.05.2026).

23. Spring Data JPA Reference Documentation : official documentation. Broadcom Inc. URL: https://docs.spring.io/spring-data/jpa/reference/ (дата звернення: 10.05.2026).

24. JUnit 5 User Guide : official documentation. URL: https://junit.org/junit5/docs/current/user-guide/ (дата звернення: 10.05.2026).

25. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Чинний від 2016-07-01]. Вид. офіц. Київ : ДП «УкрНДНЦ», 2016. 17 с. (Інформація та документація).

ДОДАТОК А. ВІДОМОСТІ РОБОТИ

Таблиця А.1 – Відомість роботи

Формат	№ п.п	Назва документу	Найменування об'єкту або вибору	Кількість сторінок
А4	1	Пояснювальна записка	КІНТЕХАД.КН-22-1Б.00.00.00	100
Графічна частина				
А4	2	Актуальність, мета, об’єкт, предмет і завдання дослідження	КІНТЕХАД.КН-22-1Б.01.00.00	1
А4	3	Порівняльний аналіз програмних аналогів (FMIS)	КІНТЕХАД.КН-22-1Б.02.00.00	1
А4	4	Архітектура веб-орієнтованої системи	КІНТЕХАД.КН-22-1Б.03.00.00	1
А4	5	ER-діаграма структури бази даних	КІНТЕХАД.КН-22-1Б.04.00.00	1
А4	6	Діаграми UML (прецедентів, класів, послідовностей)	КІНТЕХАД.КН-22-1Б.05.00.00	1
А4	7	Специфікація REST-API	КІНТЕХАД.КН-22-1Б.06.00.00	1
А4	8	Блок-схема алгоритму розрахунку норм добрив за принципом Лібіха	КІНТЕХАД.КН-22-1Б.07.00.00	1
А4	9	Інтерфейси користувача та інтерактивна карта полів	КІНТЕХАД.КН-22-1Б.08.00.00	1
А4	10	Результати економічних розрахунків	КІНТЕХАД.КН-22-1Б.09.00.00	1
А4	11	Висновки до роботи	КІНТЕХАД.КН-22-1Б.10.00.00	1

ДОДАТОК Б. ГЛОСАРІЙ

У глосарії наведено визначення основних термінів предметної області та інформаційних технологій, ужитих у кваліфікаційній роботі (таблиця Б.1).

Таблиця Б.1 – Глосарій термінів

Термін	Визначення
Точне землеробство	Концепція управління сільськогосподарським виробництвом, що ґрунтується на врахуванні неоднорідності умов на полі та диференційованому застосуванні ресурсів
Агрохімічний аналіз ґрунту	Лабораторне дослідження вмісту в ґрунті основних елементів живлення (азоту, фосфору, калію) та рівня кислотності
Сівозміна	Науково обґрунтоване чергування сільськогосподарських культур на полі у часі для збереження родючості ґрунту
Закон мінімуму Лібіха	Принцип, за яким урожайність обмежується тим елементом живлення, що перебуває у мінімумі відносно потреби рослини
Винос елементів живлення	Кількість поживних речовин, яку рослина споживає з ґрунту для формування одиниці врожаю
Діюча речовина	Вміст чистого елемента живлення у складі мінерального добрива
REST API	Архітектурний стиль програмного інтерфейсу для обміну даними між клієнтом і сервером за протоколом HTTP
JPA	Стандарт Java для об'єктно-реляційного відображення (Java Persistence API)
ORM	Технологія відображення об'єктів програми на записи реляційної бази даних (Object-Relational Mapping)

Продовження таблиці Б.1

Термін	Визначення
Spring Boot	Фреймворк для швидкого створення застосунків мовою Java з автоконфігурацією та вбудованим вебсервером
Контролер	Програмний компонент, що приймає запити користувача та повертає відповіді, реалізуючи бізнес-логіку
Репозиторій	Компонент рівня доступу до даних, що інкапсулює операції зчитування та збереження сутностей
Сутність (Entity)	Клас, що відображається на таблицю бази даних та представляє об'єкт предметної області
Прецедент (Use Case)	Опис послідовності дій, яку система виконує для досягнення мети актора
ER-діаграма	Діаграма «сутність – зв'язок», що відображає структуру даних та зв'язки між сутностями
Нормальна форма Бойса – Кодда	Властивість реляційної схеми, за якої усунено надмірність даних та аномалії оновлення
GeoJSON	Формат подання географічних об'єктів на основі нотації JSON
Leaflet	JavaScript-бібліотека для побудови інтерактивних вебкарт
Turf.js	JavaScript-бібліотека для геопросторових обчислень, зокрема визначення площі

ДОДАТОК В. ТЕХНІЧНЕ ЗАВДАННЯ

Технічне завдання визначає вимоги до програмного комплексу для підтримки агротехнічних рішень, порядок його розроблення та приймання.

В.1 Загальні відомості

Повна назва системи – програмний комплекс для інформаційного забезпечення системи точного землеробства та підтримки прийняття агротехнічних рішень (умовна назва – «АгроСистема»). Система розробляється у межах кваліфікаційної роботи бакалавра за спеціальністю 122 «Комп'ютерні науки». Підставою для розроблення є завдання на кваліфікаційну роботу, затверджене кафедрою.

В.2 Призначення та цілі створення системи

Призначенням системи є автоматизація обліку сільськогосподарських полів та результатів агрохімічних обстежень, а також підтримка прийняття рішень щодо удобрення посівів і чергування культур. Метою створення є підвищення обґрунтованості агротехнічних рішень фермера за рахунок застосування балансового методу розрахунку норм добрив та правил сівозміни, що сприяє раціональному використанню мінеральних добрив і збереженню родючості ґрунту.

В.3 Вимоги до системи

В.3.1 Функціональні вимоги

Система повинна забезпечувати: реєстрацію та автентифікацію фермера; створення, перегляд і видалення полів, зокрема малювання контуру поля на інтерактивній карті з автоматичним обчисленням площі; уведення та перегляд результатів агрохімічних аналізів ґрунту; розрахунок норм мінеральних добрив за законом мінімуму Лібіха з урахуванням культури та стану ґрунту; формування рекомендацій щодо чергування культур у сівозміні; збереження обраної для поля культури.

В.3.2 Вимоги до інтерфейсу користувача

Інтерфейс повинен бути веборієнтованим, інтуїтивно зрозумілим та адаптивним до пристроїв різного розміру. Взаємодія з картою має забезпечувати масштабування, переміщення та малювання контурів полів. Повідомлення про помилки мають бути зрозумілими користувачеві.

В.3.3 Вимоги до надійності

Система повинна коректно опрацьовувати некоректні та неповні вхідні дані, не допускаючи аварійного завершення. Цілісність даних має забезпечуватися обмеженнями бази даних та транзакційністю операцій, що змінюють пов'язані записи.

В.3.4 Вимоги до програмного та апаратного забезпечення

Серверна частина потребує середовища виконання Java версії 21 та СУБД PostgreSQL. Клієнтська частина працює у сучасному вебпереглядачі з підтримкою JavaScript. Мінімальні апаратні вимоги до сервера: двоядерний процесор, 4 ГБ оперативної пам'яті, 10 ГБ дискового простору.

В.4 Вхідні та вихідні дані

Вхідними даними є облікові дані фермера, геометрія та характеристики полів, результати агрохімічних аналізів (вміст азоту, фосфору, калію, рівень pH) та обрана культура. Вихідними даними є рекомендовані дози мінеральних добрив із зазначенням лімітуючого фактора й плану внесення, а також перелік рекомендованих культур-послідовників для сівозміни.

В.5 Стадії та етапи розроблення

Розроблення виконується у такі стадії: аналіз предметної області та постановка задачі; проєктування моделей системи та бази даних; реалізація серверної та клієнтської частин; тестування; оформлення документації та підготовка до захисту.

В.6 Порядок контролю та приймання

Контроль якості здійснюється шляхом модульного та інтеграційного тестування, а також перевіркою відповідності реалізованих функцій вимогам цього технічного завдання. Приймання роботи виконується науковим керівником та атестаційною комісією за результатами захисту кваліфікаційної роботи.

ДОДАТОК Г. UML ДІАГРАМИ

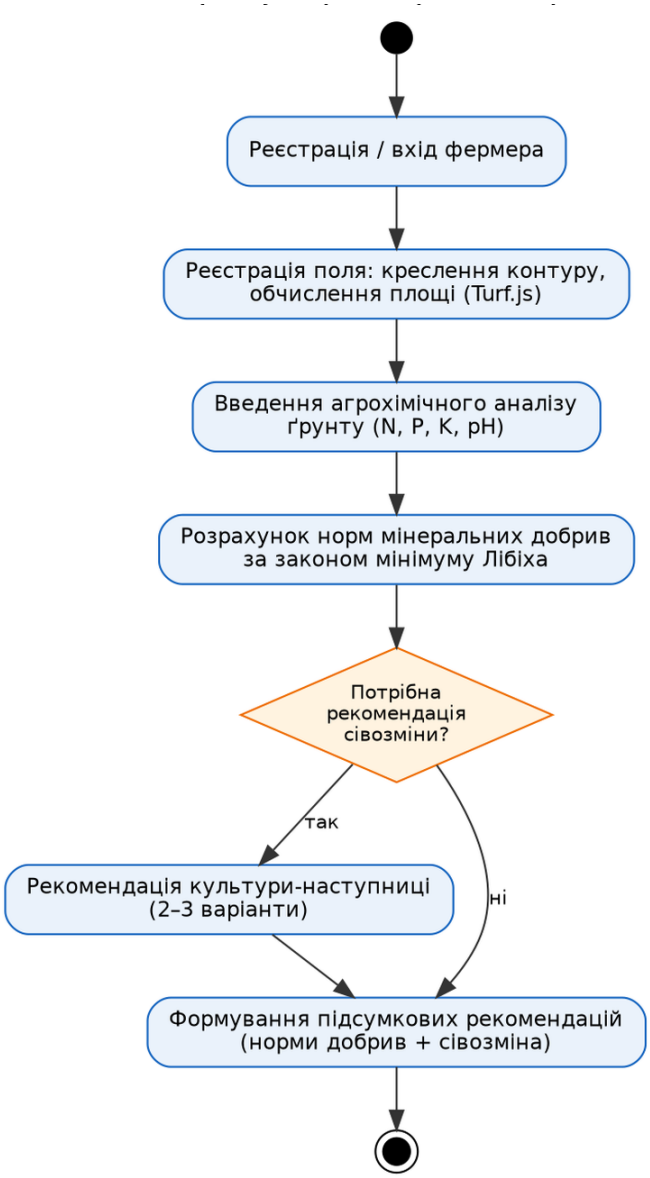

Рисунок Г.1 – Модель бізнес-процесу підтримки агротехнічних рішень

На рисунку Г.1 подано модель основного бізнес-процесу підтримки агротехнічних рішень: від реєстрації фермера та створення поля на інтерактивній карті, через уведення результатів агрохімічного аналізу ґрунту, до розрахунку норм мінеральних добрив і формування

рекомендацій із сівозміни. Діаграма відображає п’ять ключових процесів, які автоматизує система “АгроСистема”.

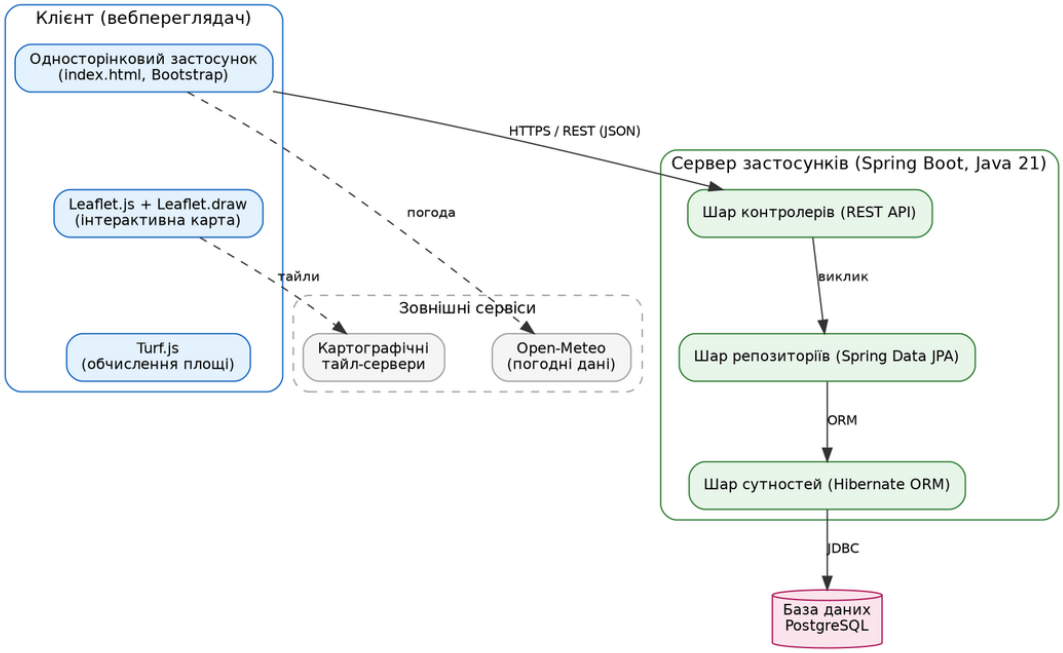

Рисунок Г.2 – Архітектура програмного комплексу «АгроСистема»

Рисунок Г.2 ілюструє трирівневу клієнт-серверну архітектуру комплексу: SPA-клієнт у браузері (Bootstrap 5, Leaflet.js, Turf.js), серверний застосунок Spring Boot із REST-API та реляційна база PostgreSQL, що взаємодіють за протоколом HTTP у форматі JSON.

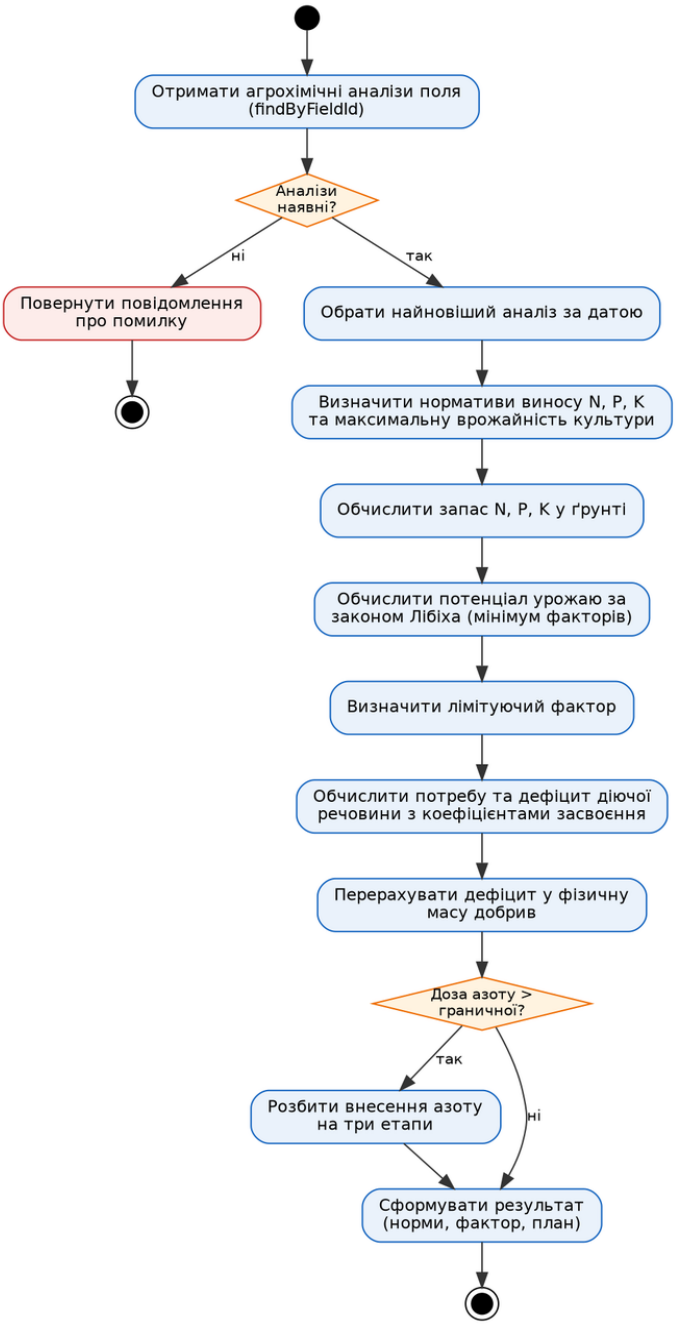

Рисунок Г.3 – Діаграма діяльності «Розрахунок норм добрив»

На рисунку Г.3 наведено діаграму діяльності процесу розрахунку норм добрив: вибір агрохімічного аналізу й культури, визначення лімітуючого елемента за законом мінімуму Лібіха, прогнозування врожаю та обчислення маси азотних, фосфорних і калійних добрив із планом внесення азоту.

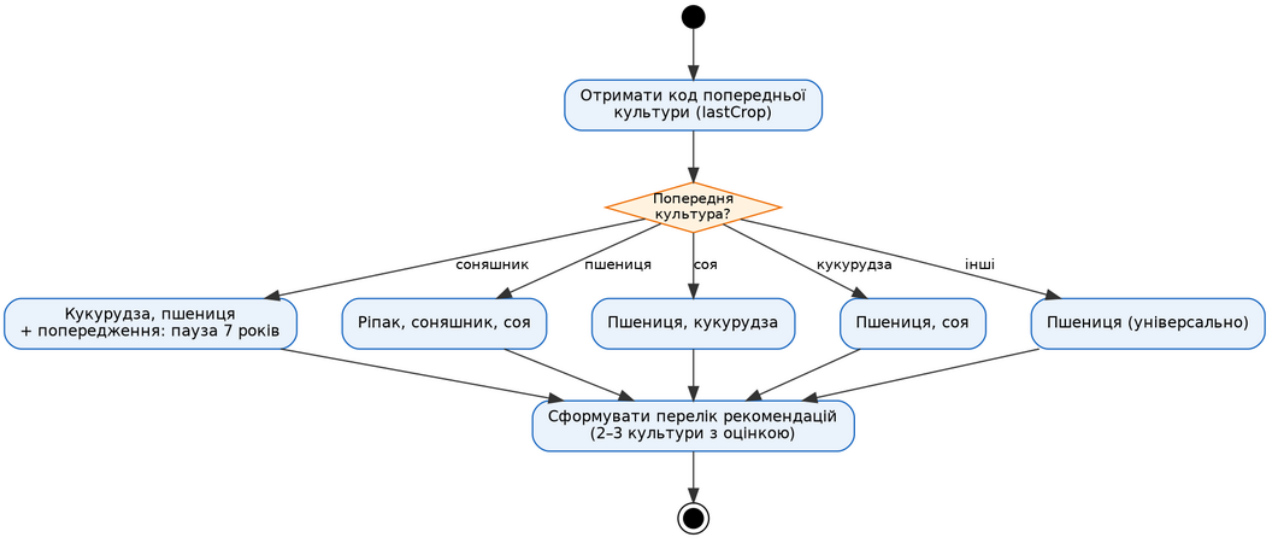

Рисунок Г.4 – Діаграма діяльності «Рекомендація сівозміни»

Рисунок Г.4 відображає діаграму діяльності процесу формування рекомендацій із сівозміни. На основі останньої вирощуваної на полі культури система генерує 2–3 рекомендації культури-наступниці з якісною оцінкою та поясненням, а також окремо видає попередження про обов’язкову паузу повернення соняшнику (7 років).