НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО
Декан факультету
інформаційних технологій
(назва факультету (ННІ))
_____ _Ігор БОЛБОТ_____
(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 2025 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
комп'ютерних наук
(назва кафедри)
_____ _Белла ГОЛУБ_
(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему _____«Експертна інформаційна система ідентифікації рослин»_____

Спеціальність _____122 «Комп'ютерні науки»_____
(код і найменування)

Освітня програма _____Інформаційні управляючі системи та технології_____
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми
кандидат технічних наук, доцент__ _____ _____Белла ГОЛУБ___
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи
кандидат фізико-математичних наук, доцент _____ ____Віктор КИРИЧЕНКО
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ ____Олексій ВРУБЛЕВСЬКИЙ_____
(підпис) ((ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) _____інформаційних технологій_____

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук
доцент, к.т.н. _____ Голуб Б. Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"01" листопада 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Врублевському Олексію Сергійовичу
(прізвище, ім'я, по батькові)

Спеціальність 122 «Комп'ютерні науки»
(код і назва)

Освітня програма Інформаційні управляючі системи та технології
(назва)

Орієнтація освітньої програми освітньо-професійна

Тема магістерської кваліфікаційної роботи «Експертна інформаційна система ідентифікації рослин»
затверджена наказом ректора НУБіП України від "01" листопада 2024р. №1964 «С»
Термін подання завершеної роботи на кафедру 29.11.2025
(рік, місяць, число)
Вихідні дані до магістерської кваліфікаційної роботи: набори даних відкритого доступу для розпізнавання рослин, а також зображення листків із датасету PlantVillage
Перелік питань, що підлягають дослідженню:

1. Аналіз ключових характеристик зображень рослин у наборі даних PlantVillage та власній вибірці спостережень, що використовується в експертній системі.
2. Дослідження та обґрунтування методів попередньої обробки й аугментації зображень (нормалізація, ресайзинг, шум, повороти, зміна яскравості тощо) для підвищення точності ідентифікації рослин у розробленій системі.
3. Оцінювання ефективності сучасних архітектур глибинного навчання (CNN, ResNet, EfficientNet, Vision Transformers) у задачі класифікації видів та станів рослин у складі експертної інформаційної системи.
4. Виявлення закономірностей та виконання класифікації/сегментації рослин за візуальними ознаками (форма, текстура, колір, плями, ушкодження листкової поверхні) з подальшою інтеграцією результатів в експертні правила та аналітичні модулі системи.

Дата видачі завдання "01" листопада 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Кириченко В. В.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Врублевський О. С
(підпис) (прізвище та ініціали студента)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської кваліфікаційної роботи	Строк виконання етапів магістерської кваліфікаційної роботи	Примітка
1	Видача завдання	01.11.2024	
2	Аналіз предметної області	01.11.2024 – 10.12.2025	
3	Моделювання предметної області	11.12.2024 – 23.01.2025	
4	Розробка системи	24.01.2025 – 18.05.2025	
5	Аналіз результатів	19.05.2025 – 31.08.2025	
6	Оформлення записки	01.09.2025 – 01.11.2025	
7	Постерна сесія	28.10.2025 – 29.10.2025	
8	Перевірка на плагіат	13.11.2025	
9	Попередній захист	01.12.2025	
10	Захист	15.12.2025	

Студент: _____ Олексій Врублевський
(підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи: _____ Віктор Кириченко
(підпис) (ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Магістерська кваліфікаційна робота викладена на 68 сторінках машинописного тексту, містить 26 рисунків, 11 таблиць та 38 джерел у списку використаних джерел. Робота присвячена розробленню експертної інформаційної системи ідентифікації рослин на основі методів комп'ютерного зору та глибинного машинного навчання.

У роботі досліджено сучасні підходи до автоматизованої ідентифікації рослин за зображеннями листя, стебел і плодів, проаналізовано основні проблеми традиційних методів класифікації та обґрунтовано доцільність застосування згорткових нейронних мереж. Розглянуто процес формування та підготовки навчальних вибірок, методи попередньої обробки зображень, а також особливості використання відкритих датасетів для навчання моделей.

На основі системного аналізу та UML-моделювання спроєктовано архітектуру експертної інформаційної системи, що включає підсистеми збору та оброблення вхідних даних, модуль машинного навчання, логічну модель бази даних і підсистему формування рекомендацій. Побудовано ER-модель бази даних та розроблено програмну реалізацію системи з використанням мови програмування Python, фреймворку FastAPI, реляційної бази даних SQLite та інтерфейсів прикладного програмування REST.

Розроблена система реалізує гібридний підхід до ідентифікації рослин із використанням згорткових нейронних мереж CNN, моделей ResNet, EfficientNet і Vision Transformer, що забезпечує високу точність класифікації та можливість масштабування. У ході експериментального дослідження проведено порівняльний аналіз моделей за показниками точності, повноти та F1-міри, за результатами якого визначено найбільш ефективні архітектури для практичного застосування. Отримані експериментальні результати підтвердили працездатність системи та можливість її використання в умовах реального часу.

Практична цінність роботи полягає у створенні програмного інструменту для автоматизованої ідентифікації рослин, який може бути використаний у

сільському господарстві, екологічному моніторингу, освітніх і науково-дослідних проєктах. Результати дослідження мають як прикладне, так і наукове значення та підтверджують доцільність застосування методів глибинного навчання для задач класифікації біологічних об’єктів.

ABSTRACT

The master's thesis is presented on 68 pages of typescript and contains 26 figures, 11 tables, and 38 references. The research is devoted to the development of an expert information system for plant identification based on computer vision and deep learning methods.

The thesis investigates modern approaches to automated plant identification using images of leaves, stems, and fruits. The limitations of traditional classification methods are analyzed, and the applicability of convolutional neural networks is substantiated. The processes of dataset formation, image preprocessing, and the use of open datasets for model training are examined.

Based on system analysis and UML modeling, the architecture of the expert information system was designed, including data acquisition and processing subsystems, a machine learning module, a logical database model, and a recommendation subsystem. An ER-based database model was developed, and the software implementation was realized using Python, FastAPI, SQLite, and REST APIs.

The developed system implements a hybrid plant identification approach using CNN, ResNet, EfficientNet, and Vision Transformer models, providing high classification accuracy and scalability. Experimental evaluation included a comparative analysis of models using accuracy, recall, and F1-score metrics, resulting in the identification of the most effective architectures for practical deployment. The results confirm the feasibility of real-time plant identification using the proposed system.

The practical significance of the thesis lies in creating a software tool for automated plant identification that can be applied in agriculture, environmental monitoring, education, and research. The obtained results demonstrate both applied and scientific value and confirm the effectiveness of deep learning methods for biological object classification.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ....4
ВСТУП....6
1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ....8
1.1 Опис предметної області....8
1.2 Огляд інформаціних джерел та існуючих рішень....10
1.3 Моделювання предметної області....14
1.4 Аналіз вимог програмної системи....17
1.5 Постановка завдання....19
1.6 Висновки до першого розділу....21
2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ....23
2.1 Логічна модель даних у вигляді ER-діаграми....23
2.2 Діаграма класів і кооперації....26
2.3 Діаграма компонентів....29
2.4 Діаграма пакетів експертної системи....32
2.5 Висновки до другого розділу....35
3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ....37
3.1 Вибір технологій та інструментальних засобів реалізації системи....37
3.2 Архітектура системи, проєктування функціоналу та результати дослідження....39
3.3 Інформаційна база системи....42
3.4 Алгоритмізація модулів системи....45
3.5 Висновки до третього розділу....49
4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ....51
4.1 План тестування програмних модулів та методика оцінювання результатів....51
4.2 Тестування інтелектуальної експертної системи ідентифікації рослин....53
4.3 Результати тестування та аналіз ефективності системи....58
4.4 Розгортання системи та склад інсталяційного пакета....61
4.5 Висновки до четвертого розділу....62
ВИСНОВКИ....63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ....65
ДОДАТОК А....68

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. API — Application Programming Interface, інтерфейс програмування застосунків
2. CNN — Convolutional Neural Network, згорткова нейронна мережа
3. CPU — Central Processing Unit, центральний процесор
4. DB — Database, база даних
5. ETL — Extract, Transform, Load, процес вилучення, трансформації та завантаження даних
6. F1-score — гармонійне середнє між precision і recall
7. FPS — Frames Per Second, частота кадрів
8. HTML — HyperText Markup Language, мова розмітки гіпертексту
9. HTTP — HyperText Transfer Protocol, протокол передавання даних
10. JPEG — Joint Photographic Experts Group, формат зображень
11. JSON — JavaScript Object Notation, формат даних
12. K-Means — алгоритм кластеризації методом k-середніх
13. KPI — Key Performance Indicator, ключовий показник ефективності
14. MAE — Mean Absolute Error, середня абсолютна похибка
15. MDX — Multidimensional Expressions, мова запитів для OLAP
16. ML — Machine Learning, машинне навчання
17. MSE — Mean Squared Error, середньоквадратична похибка
18. NDVI — Normalized Difference Vegetation Index, нормалізований диференційний вегетаційний індекс
19. OLAP — Online Analytical Processing, оперативна аналітична обробка даних
20. PDF — Portable Document Format, формат документів
21. PCA — Principal Component Analysis, метод головних компонент
22. PNG — Portable Network Graphics, растровий формат зображень
23. PyQt6 — Python Qt6 Framework, бібліотека графічного інтерфейсу

24. RAM — Random Access Memory, оперативна пам’ять

25. REST — Representational State Transfer, архітектурний стиль веб-сервісів

26. SQL — Structured Query Language, мова керування базами даних

27. SLA — Service Level Agreement, угода про рівень надання послуг

28. SLO — Service Level Objective, цільовий рівень послуги

29. TLS — Transport Layer Security, протокол захисту передавання даних

30. UI — User Interface, інтерфейс користувача

31. ViT — Vision Transformer, трансформер для обробки зображень

32. XML — Extensible Markup Language, розширювана мова розмітки

ВСТУП

Ідентифікація рослин є важливим завданням у галузях біології, агрономії, екології та моніторингу біорізноманіття. У сучасних умовах цифровізації агросектору зростає потреба у створенні інформаційних систем, здатних забезпечувати автоматизоване визначення видів рослин за зображеннями та морфологічними характеристиками. Традиційні підходи, що ґрунтуються на візуальній оцінці експерта, потребують значного часу й не забезпечують відтворюваності результатів у масштабних наборах даних. Тому актуальним є розроблення інтелектуальної експертної системи, яка поєднує алгоритми машинного навчання, аналітичні модулі збору метрик і механізми формування рекомендацій для користувача. Такі рішення підвищують точність класифікації, автоматизують процес аналізу та сприяють розвитку екосистем цифрового агромоніторингу.

Мета роботи полягає у розробленні експертної інформаційної системи ідентифікації рослин, що забезпечує автоматизоване визначення видів, аналітичну обробку параметрів та формування рекомендацій на основі моделей машинного навчання з можливістю побудови інтерактивних HTML-звітів.

Для досягнення поставленої мети необхідно вирішити наступні **завдання:**

1. провести системний аналіз предметної області, охарактеризувати існуючі підходи до автоматизованої ідентифікації рослин і визначити недоліки сучасних рішень.
2. Розробити логічну, інформаційну та архітектурну моделі експертної системи.
3. Формалізувати функціональні, нефункціональні й технічні вимоги до системи.
4. Реалізувати програмні модулі авторизації, збору метрик, аналітики та генерації звітів.

5. Провести експериментальні дослідження ефективності класифікації та оцінити точність роботи моделей машинного навчання.

Об’єкт дослідження - процес автоматизованої ідентифікації рослин за цифровими зображеннями та супровідними параметрами середовища.

Предмет дослідження - методи, алгоритми та інформаційно-технологічні засоби створення експертних систем, орієнтованих на розпізнавання та аналітичну підтримку процесів ідентифікації рослин.

Методи дослідження - у роботі використано комплекс сучасних підходів: методи машинного та глибокого навчання (зокрема, згорткові нейронні мережі - CNN), методи статистичного аналізу та нормалізації даних, алгоритми кластеризації, а також засоби об’єктно-орієнтованого проєктування та моделювання програмних систем у нотації UML. Для реалізації функціональних компонентів застосовано технології Python, HTML, CSS, JavaScript, SQLite і Flask, що забезпечують гнучкість архітектури та можливість подальшого масштабування.

Наукова новизна одержаних результатів полягає у створенні цілісної архітектури експертної системи, яка поєднує механізми ідентифікації об’єктів, автоматизований збір метрик, формування рекомендацій і побудову аналітичних звітів. Запропонований підхід інтегрує інтелектуальні алгоритми машинного навчання із сервісною моделлю представлення даних, що підвищує точність розпізнавання та оперативність прийняття рішень у галузі агроекологічного моніторингу.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область експертної інформаційної системи ідентифікації та оцінки рослин охоплює процеси цифрового аналізу біологічних об’єктів за морфологічними, спектральними, біометричними та фенологічними параметрами. Мета дослідження полягає у створенні інтелектуальної моделі, здатної автоматично визначати видову приналежність і фізіологічний стан рослин на основі об’єктивних вимірюваних характеристик. У межах предметної області система інтегрує дані зі зображень, сенсорів та візуальних спостережень, формуючи структурований набір ознак для подальшої аналітики та класифікації.

В основу концепції покладено систему метрик оцінки рослин, що представлена на рисунку 1.1. Вона відображає взаємозв’язок основних показників, за якими здійснюється оцінка об’єкта, і слугує структурною основою для подальшої побудови алгоритмів ідентифікації.

Рис. 1.1 – Система метрик оцінки рослин.

Морфологічні метрики відображають геометричні властивості рослин, зокрема форму, площу, контур, довжину та симетрію листкової пластинки. Вони забезпечують можливість розпізнавання близьких видів за структурними ознаками та формують основу векторів ознак для класифікаційних моделей. Кольорово-спектральні метрики (HSV, NDVI, GNDVI, спектральна відбивна

здатність) характеризують фотосинтетичну активність і пігментний склад, що дає змогу оцінювати фізіологічний стан рослин, рівень стресу чи дефіциту елементів живлення.

Біометричні метрики включають висоту, діаметр стебла, площу крони та щільність листя. Вони описують фізичний розвиток і біомасу, дозволяючи оцінювати продуктивність та потенціал росту.

Фенологічні метрики відображають стадії розвитку, динаміку забарвлення, вологість і індекс здоров'я. Їх використання дає змогу кількісно оцінювати зміни у життєвому циклі та виявляти реакції рослин на вплив зовнішніх факторів.

Разом ці показники формують цілісну систему оцінювання, що забезпечує комплексну ідентифікацію та аналітичну інтерпретацію стану рослин у межах експертної системи.

Систематизацію показників наведено в таблиці 1.1, де подано основні групи метрик та їх призначення.

Таблиця 1.1

Основні групи метрик оцінки стану рослин

Група метрик	Приклади показників	Призначення
Морфологічні	Форма, площа, контур, симетрія	Розпізнавання виду за геометричними ознаками
Кольорово-спектральні	HSV, NDVI, спектральна відбивна здатність	Визначення стану пігментів і фотосинтетичної активності
Біометричні	Висота, діаметр стебла, площа крони, щільність листя	Кількісна оцінка росту та розвитку
Фенологічні	Стадія розвитку, колірна динаміка, вологість, індекс здоров'я	Оцінка життєвого циклу та фізіологічного стану

Предметна область визначає структуру даних і параметри, за якими здійснюється оцінювання рослин. Представлена система метрик забезпечує комплексний підхід до аналізу, дозволяючи формалізувати природні властивості об'єкта для подальшої автоматизованої ідентифікації та прогнозування стану.

1.2 Огляд інформаціних джерел та існуючих рішень

Розвиток технологій комп'ютерного зору та глибокого навчання зумовив появу широкого спектра систем автоматичної ідентифікації рослин. До найвідоміших серед них належать Plant.id, PlantNet, LeafSnap, iNaturalist та Flora Incognita. Кожна з них реалізує власну концепцію класифікації, різниться алгоритмами обробки зображень, структурою баз даних і призначенням для кінцевих користувачів.

Система Plant.id вирізняється глибокою інтеграцією алгоритмів CNN для ідентифікації рослин і фітопатологічного аналізу. Веб-інтерфейс представлений на рисунку 1.2 дозволяє користувачу завантажити фото або зробити знімок у реальному часі, після чого модель формує список можливих видів із рівнем ймовірності. Додатково система виконує оцінку стану здоров'я та пропонує інтеграційний API для зовнішніх сервісів.

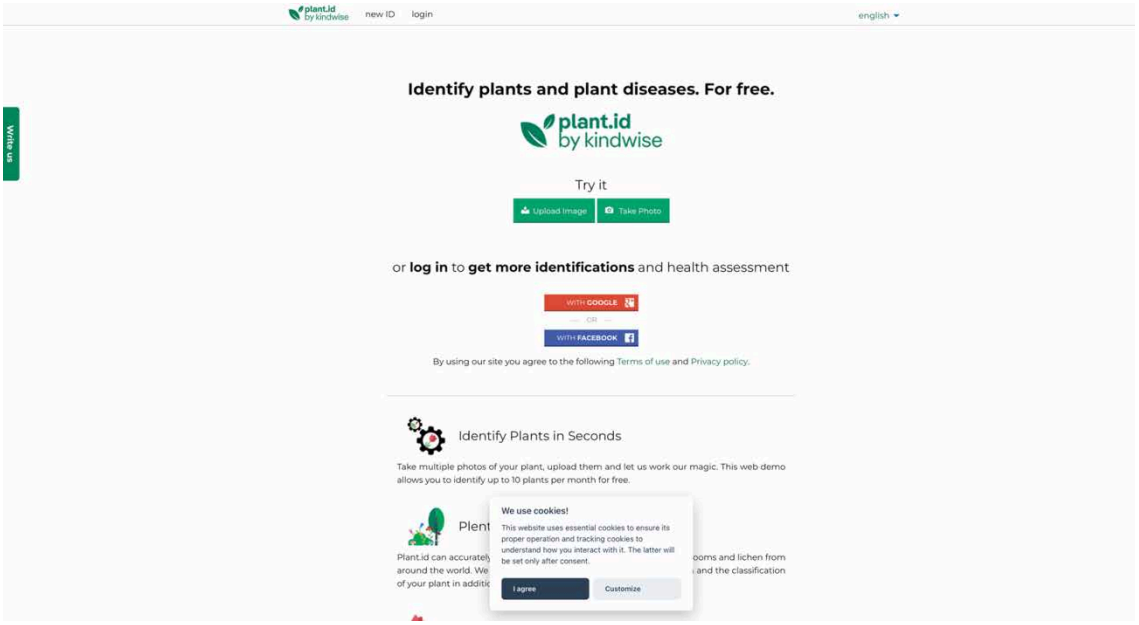

Рис. 1.2 – Веб-сторінка Plant.id із функцією завантаження та класифікації зображень.

Додаток PlantNet (рис. 1.3) реалізує концепцію громадської наукової взаємодії: користувачі завантажують фотографії, що автоматично поповнюють навчальну базу даних. Алгоритми CNN поєднуються з геолокаційними

ознаками, що підвищує точність ідентифікації для певних регіонів. PlantNet підтримує автоматичне уточнення результатів шляхом порівняння із глобальними ботанічними реєстрами.

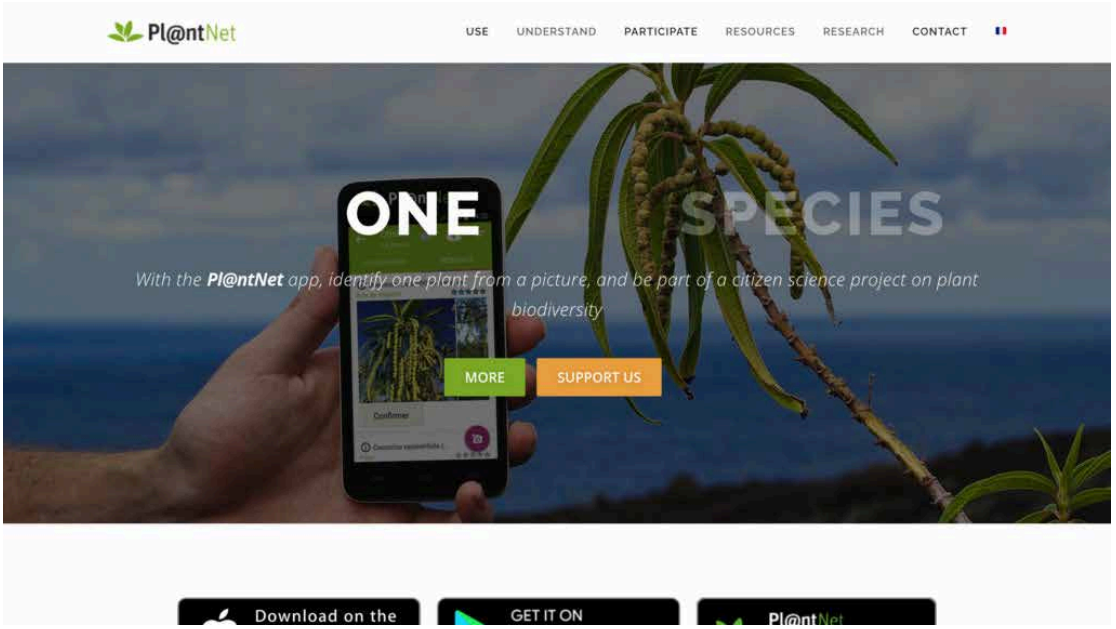

Рис. 1.3 – Мобільний інтерфейс PlantNet із візуалізацією ідентифікованого виду.

LeafSnap є однією з перших систем комп’ютерного розпізнавання листків, що базується на контурному аналізі та порівнянні форми об’єкта із зразками в еталонній базі на рисунку 1.4. Основна увага приділена морфологічним ознакам листкової пластинки - симетрії, довжині, площі та контуру. Програма орієнтована на навчальне використання і демонструє ефективність у розпізнаванні деревних рослин.

Рис. 1.4 – Екран ідентифікації у системі LeafSnap із картографічним прив’язуванням спостережень.

Платформа iNaturalist на рис. 1.5 поєднує автоматичне розпізнавання із соціальним механізмом підтвердження результатів. Система залучає експертів-ботаніків, які перевіряють результати ідентифікації, що підвищує достовірність бази даних. Веб-інтерфейс підтримує ведення особистого журналу спостережень, аналітику біорізноманіття та колективне формування глобальної ботанічної карти.

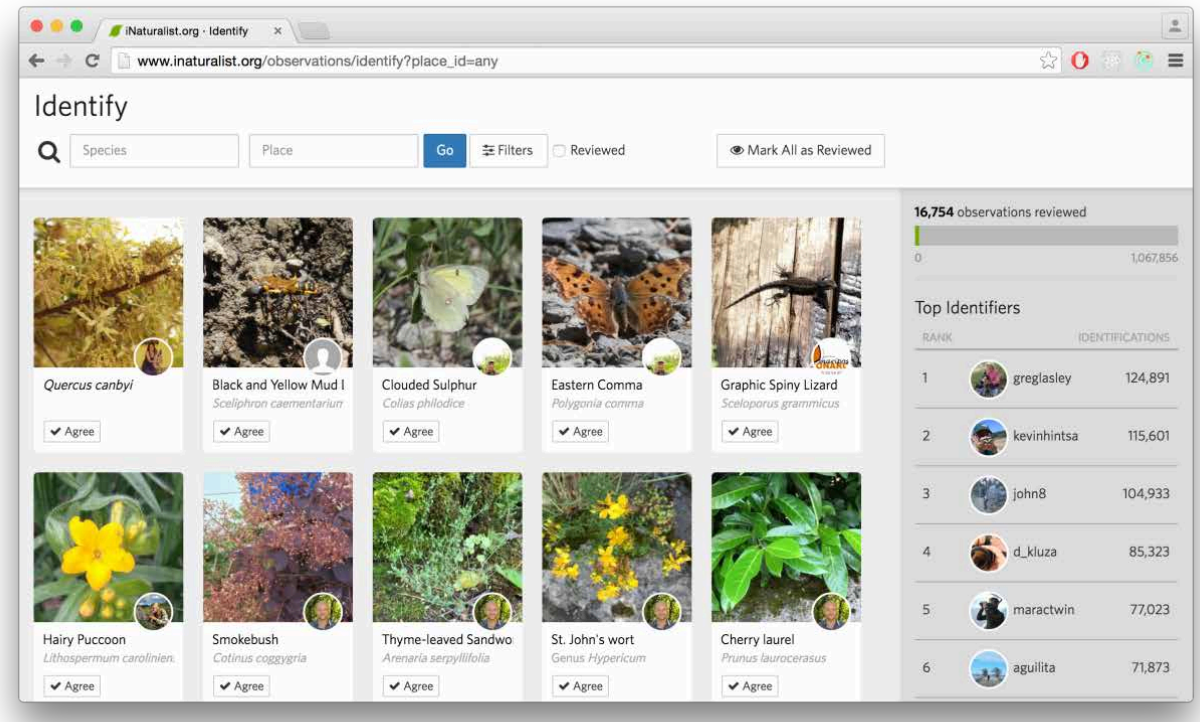

Рис. 1.5 – Інтерфейс iNaturalist з модулем аналітики та обміну спостереженнями між користувачами.

Flora Incognita - приклад мобільного застосунку з дворівневою архітектурою обробки зображень (рис. 1.6). На першому етапі система виділяє контур і форму об’єкта, на другому - проводить спектральний аналіз кольорових компонентів. Додаток відрізняється високою точністю, надає детальний опис виду, екологічну характеристику та інтерактивну карту поширення.

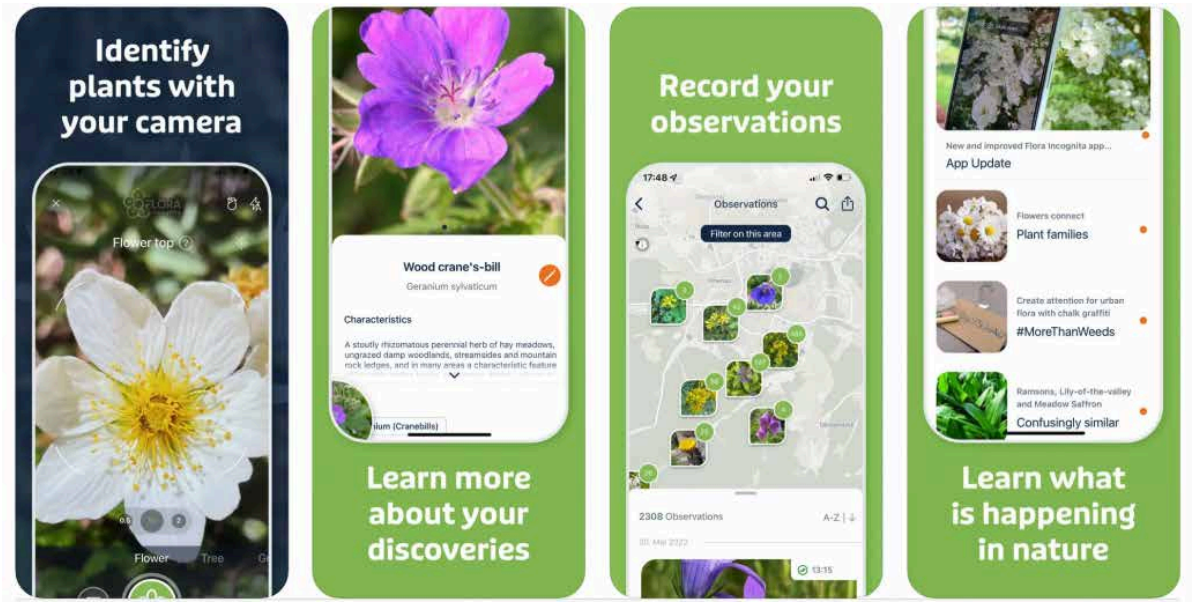

Рис. 1.6 – Мобільний застосунок Flora Incognita з інтеграцією геолокаційних даних.

Для узагальнення переваг і недоліків розглянутих рішень, а також визначення відмінних рис запропонованої експертної інформаційної системи ідентифікації рослин, складено порівняльну таблицю (табл. 1.2), у якій відображено основні технічні, функціональні та аналітичні параметри.

Таблиця 1.2

Порівняльна характеристика існуючих систем і розроблюваного рішення

Система	Платформа	Методи розпізнавання	Основні функції	Недоліки / Обмеження
Plant.id	Веб / API	Глибокі CNN, класифікація за зображенням	Розпізнавання видів і хвороб, API інтеграція	Ліміт запитів у безкоштовній версії
PlantNet	Мобільна, веб	CNN + геолокаційні атрибути	Громадська база даних, статистика біорізноманіття	Потребує активного з’єднання з мережею
LeafSnap	Мобільна	Контурний аналіз листків	Ідентифікація за морфологічними ознаками	Вузький набір видів, без аналітики
iNaturalist	Веб, мобільна	CNN + експертна перевірка	Спільнота біологів, глобальна карта видів	Результати потребують модерації

Продовження таблиці 1.2

Flora Incognita	Мобільна	Гібридна модель (форма + спектр)	Геолокація, рекомендації з догляду	Обмежене охоплення поза Європою
Розроблювана система	Веб / десктоп	CNN + ML аналіз морфологічних, біометричних і фенологічних метрик	Авторизація, збір метрик, оцінка стану, HTML-звіти	Гнучка модульна архітектура, відкрите API

Отже, порівняльний аналіз засвідчує, що існуючі рішення орієнтовані переважно на класифікацію зображень без урахування біометричних і фенологічних характеристик. Розроблювана експертна система поєднує аналітичні та прогностичні модулі, що дозволяє не лише визначати видову належність, а й оцінювати фізіологічний стан рослин, формувати рекомендації та автоматично генерувати звіти, забезпечуючи розширену науково-технічну функціональність у порівнянні з аналогами.

1.3 Моделювання предметної області

Моделювання предметної області є ключовим етапом формалізації процесів, що реалізуються в експертній інформаційній системі ідентифікації рослин. Воно дозволяє визначити основні функціональні можливості системи, взаємодію користувачів із програмними модулями, а також послідовність оброблення запитів. Моделі створені з використанням нотації UML, що забезпечує уніфіковане представлення логічної структури та динаміки системи.

На рисунку 1.7 подано діаграму прецедентів, яка відображає ролі користувачів і сценарії їхньої взаємодії з системою. Основними акторами є користувач, адміністратор, експерт-ботанік та аналітик. Користувач може проходити авторизацію, завантажувати зображення рослин, переглядати результати класифікації, отримувати рекомендації з догляду та експортувати звіти. Адміністратор керує користувачами й ролями, експерт-ботанік здійснює

валідацію класифікацій і редагування бази знань, а аналітик переглядає зібрані метрики та формує статистику.

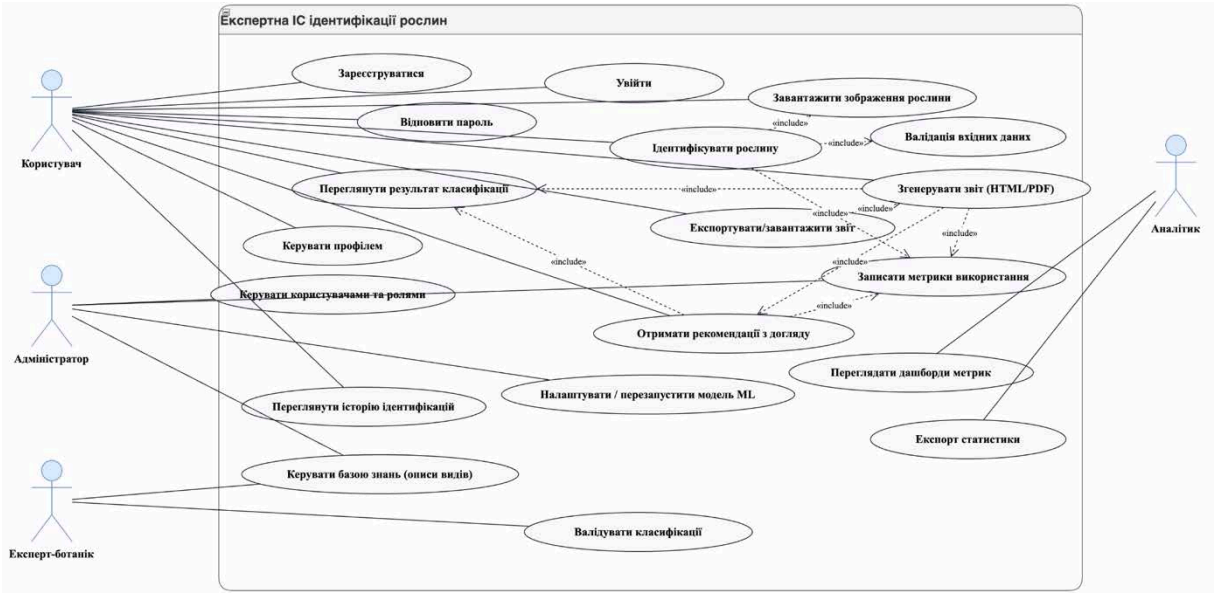

Рис. 1.7 – Діаграма прецедентів експертної системи ідентифікації рослин.

Для деталізації взаємодії між користувачем і внутрішніми компонентами розроблено діаграму послідовності (рис. 1.8). Вона демонструє послідовність викликів сервісів під час процесу розпізнавання: після аутентифікації користувач завантажує фото, яке передається до ML-модуля для ідентифікації виду. Далі результат надходить до сервісу звітності, що формує HTML-звіт із короткими рекомендаціями, після чого система повертає посилання на результат користувачу.

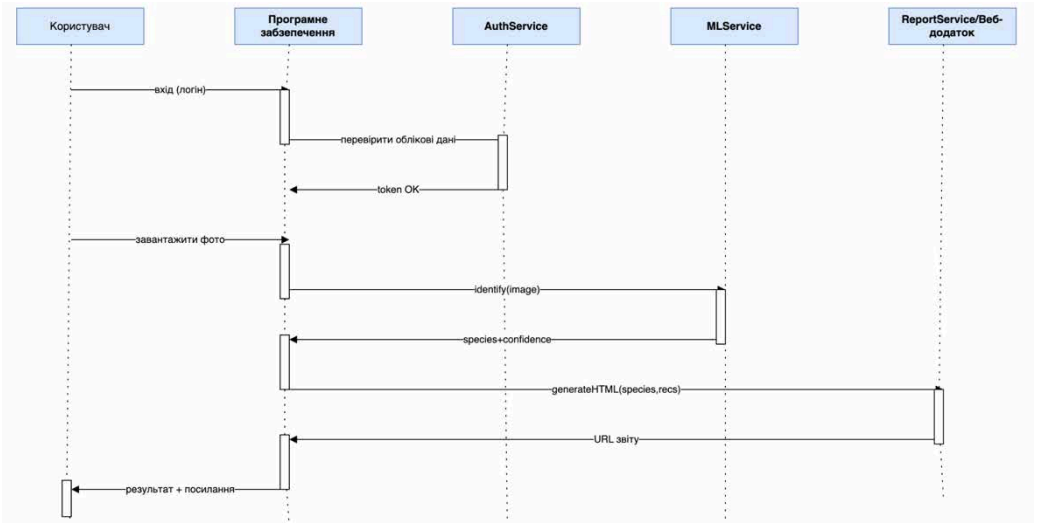

Рис. 1.8 – Діаграма послідовності процесу ідентифікації та формування звіту.

На рисунку 1.9 наведено діаграму діяльності, яка формалізує логіку функціонування системи. Процес починається з вибору або зйомки зображення, перевірки автентифікації користувача, після чого виконується передобробка даних, класифікація виду, формування рекомендацій і генерація звіту. На фінальному етапі система здійснює логування метрик, що використовуються в подальшому аналітичному модулі. Такий підхід забезпечує простежуваність і контроль кожного етапу оброблення.

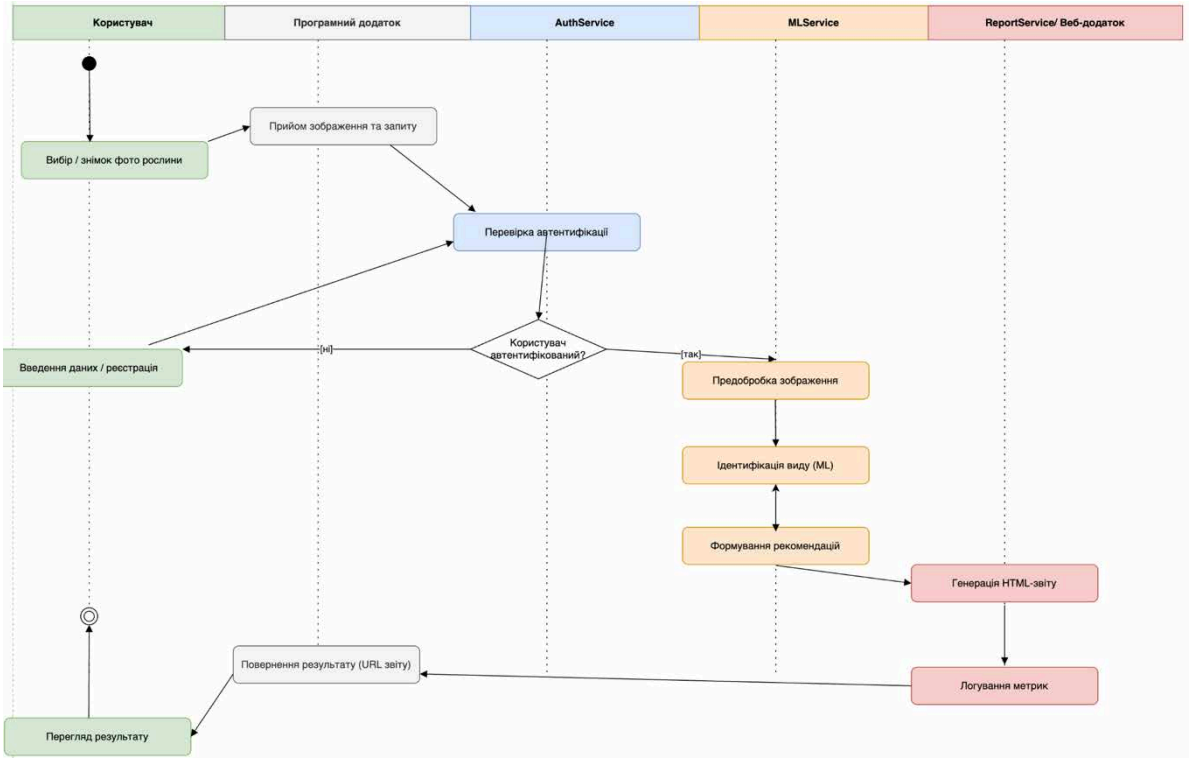

Рис. 1.9 – Діаграма діяльності процесу розпізнавання та оцінки рослин.

Побудовані UML-моделі дозволили структурувати предметну область і визначити взаємозв’язки між компонентами системи. Вони підтверджують, що розроблювана експертна система має модульну архітектуру, де кожна підсистема виконує окрему роль - автентифікація користувача, обробка зображень, машинне навчання, генерація звітів і збір метрик.

Результати моделювання слугують основою для подальших етапів розробки - аналізу вимог і постановки завдання, у межах яких будуть визначені

функціональні, нефункціональні та технічні характеристики системи, а також критерії її ефективності.

1.4 Аналіз вимог програмної системи

Аналіз вимог до експертної інформаційної системи ідентифікації рослин спрямований на формалізацію функціональних, нефункціональних і безпекових характеристик, що визначають архітектуру, продуктивність і надійність програмного продукту. Ретельне структурування вимог забезпечує коректне проєктування компонентів системи, їхню взаємодію та відповідність очікуванням користувачів і стандартам програмної інженерії.

Функціональні вимоги описують основні можливості системи, необхідні для реалізації ключових бізнес-процесів - від аутентифікації користувача до аналітики результатів. У межах експертної системи передбачено завантаження зображень рослин, автоматичну класифікацію на основі моделей машинного навчання, формування рекомендацій, створення звітів (HTML/PDF), збереження метрик і аналітичну візуалізацію даних. Узагальнені функціональні вимоги наведено в таблиці 1.3.

Таблиця 1.3

Функціональні вимоги до експертної системи ідентифікації рослин

№	Вимога	Опис	Очікуваний результат
1	Реєстрація та авторизація	Надання користувачу доступу до системи через обліковий запис	Безпечний персоналізований доступ
2	Завантаження зображень	Приймання зображень рослин для класифікації (формати JPEG/PNG)	Коректна передача даних до ML-модуля
3	Ідентифікація виду	Використання моделі машинного навчання для класифікації	Визначення виду з вірогідністю ≥ 90 %
4	Формування рекомендацій	Генерація опису стану та порад із догляду	Відображення рекомендацій у звіті
5	Генерація звітів	Автоматичне створення звітів (HTML/PDF) за результатами аналізу	Можливість перегляду та експорту

6	Логування метрик	Збір інформації про запити, час обробки, точність моделі	Побудова аналітичних дашбордів

Продовження таблиці 1.3

7	Адміністрування	Керування користувачами, базою знань і правами доступу	Контроль стану системи й актуальності даних

Нефункціональні вимоги визначають якісні параметри роботи системи: стабільність, продуктивність, масштабованість, зручність і розширюваність. Вони гарантують безперебійну роботу під навантаженням, швидку реакцію на користувацькі запити та відповідність сучасним технічним стандартам. Перелік основних нефункціональних вимог наведено в таблиці 1.4.

Таблиця 1.4

Нефункціональні вимоги до експертної системи ідентифікації рослин

№	Категорія	Вимога	Показник / Критерій
1	Продуктивність	Середній час класифікації одного зображення	≤ 3 секунд
2	Масштабованість	Одночасна робота не менше ніж 100 користувачів	Без втрати продуктивності
3	Надійність	Безвідмовність серверних компонентів	≥ 99,8 % часу доступності
4	Зручність інтерфейсу	Мінімум кроків для виконання операцій	Не більше 3 дій до результату
5	Сумісність	Підтримка браузерів Chrome, Firefox, Edge	Повна кросплатформність
6	Розширюваність	Можливість інтеграції нових ML-модулів	Модульна архітектура plug-in типу

Вимоги до безпеки визначають політику захисту інформації, персональних даних користувачів і результатів аналітики. Особливу увагу приділено автентифікації, шифруванню, контролю доступу та відновленню даних після збоїв. Основні вимоги до безпеки системи подано в таблиці 1.5.

Таблиця 1.5

Вимоги до безпеки експертної системи ідентифікації рослин

№	Вимога	Механізм реалізації	Очікуваний результат
1	Автентифікація та авторизація	Розмежування прав доступу на основі ролей (JWT-токени)	Захищений вхід і контроль доступу

Продовження таблиці 1.5

2	Захист каналів зв’язку	Використання протоколів HTTPS / TLS 1.3	Гарантована конфіденційність даних
3	Шифрування персональних даних	Застосування алгоритмів SHA-256 / bcrypt	Безпечне збереження облікових записів
4	Логування дій користувачів	Ведення журналів подій (audit logs)	Виявлення несанкціонованих дій
5	Резервне копіювання	Періодичне дублювання бази знань і метрик	Відновлення після відмов і втрат даних

Проведений аналіз дозволяє визначити пріоритети в архітектурі системи: забезпечення точності класифікації, зручності користування, безпеки та надійності. Сформульовані вимоги створюють основу для наступного етапу - постановки завдання, у межах якої буде деталізовано алгоритми, структуру даних і логіку реалізації окремих модулів експертної системи.

1.5 Постановка завдання

Постановка завдання визначає мету, вхідні та вихідні дані, а також загальні принципи функціонування експертної інформаційної системи ідентифікації рослин. На основі проведеного аналізу предметної області, моделювання та формалізації вимог сформульовано основні задачі, які система повинна реалізувати для забезпечення автоматизованої ідентифікації видів рослин, оцінки їхнього стану та формування рекомендацій.

Мета розробки - створення експертної системи, що забезпечує автоматичне визначення виду рослини за зображенням, оцінку фізіологічних показників і генерацію аналітичних звітів із рекомендаціями для користувача.

Вхідні дані:

– зображення рослин, отримані користувачем за допомогою камери або завантажені у систему (у форматах JPEG, PNG);

– супутня інформація про умови зйомки (освітлення, геолокація, дата);

– дані з бази знань, що містить морфологічні, біометричні, фенологічні та спектральні характеристики видів;

– історичні метрики та лог-файли попередніх класифікацій для навчання та оцінки моделей.

Вихідні дані:

– визначений вид рослини з імовірнісною оцінкою достовірності (confidence level);

– аналітичні показники стану рослини (NDVI, індекс вологості, морфометричні параметри тощо);

– рекомендації з догляду та попередження щодо виявлених ознак стресу або захворювань;

– автоматично сформований HTML/PDF-звіт, який зберігається у базі та доступний користувачу для перегляду й експорту.

Основні задачі системи:

1. розроблення інтерфейсу користувача з функціями авторизації, завантаження зображень і перегляду результатів.

2. Реалізація модуля машинного навчання для класифікації рослин за морфологічними, біометричними й спектральними ознаками.

3. Інтеграція підсистеми збору та зберігання метрик (час обробки, точність, кількість запитів, параметри навчання).

4. Розроблення механізму формування рекомендацій на основі аналізу отриманих показників і порівняння з базою знань.

5. Забезпечення генерації звітів у форматах HTML і PDF з можливістю експорту й архівації.

6. Реалізація адміністративного модуля для керування користувачами, базою знань і контрольних записів.

7. Забезпечення цілісності, безпеки та конфіденційності даних під час оброблення і зберігання.

Очікувані результати: створена система має надавати точну та швидку ідентифікацію рослин (точність ≥ 90 %), підтримувати асинхронну обробку запитів, автоматично формувати рекомендації та звіти, а також вести збір метрик для подальшого аналізу якості моделей.

Сформульоване завдання охоплює повний цикл функціонування інтелектуальної системи - від збору вхідних даних і машинної обробки до візуалізації результатів і аналітичного оцінювання ефективності. Реалізація поставлених вимог забезпечить створення комплексного інструменту для експертного визначення видів рослин та підтримки прийняття рішень у галузі ботанічного моніторингу й агроаналітики.

1.6 Висновки до першого розділу

У першому розділі було здійснено ґрунтовний аналіз предметної області, сучасних методів та технологій ідентифікації рослин, а також обґрунтовано вибір концепції побудови експертної інформаційної системи. Проведений огляд існуючих рішень - PlantNet, LeafSnap, iNaturalist, Plant.id, Flora Incognita - засвідчив, що попри високий рівень автоматизації процесу розпізнавання, більшість із них мають обмеження у глибині аналітики, відсутність розширених метрик оцінювання стану рослин і слабку інтеграцію з системами рекомендаційного типу. Це визначило науково-практичну нішу для створення власної системи з розширеними можливостями аналізу, прогнозування та генерації звітів.

Було сформульовано мету дослідження - розроблення експертної інформаційної системи, здатної здійснювати ідентифікацію видів рослин за морфологічними, кольорово-спектральними, біометричними та фенологічними метриками. У межах аналізу предметної області визначено, що комплексне використання таких параметрів дозволяє значно підвищити точність

класифікації, забезпечуючи багатовимірний підхід до оцінювання стану рослин. Створена система метрик (рис. 1.1) охоплює весь цикл життєдіяльності рослин — від зовнішніх морфологічних ознак до фізіологічних характеристик, що дозволяє інтегрувати дані з сенсорних, візуальних і спектральних джерел.

Моделювання предметної області дало змогу формалізувати основні сутності, взаємозв'язки та процеси, притаманні системі ідентифікації, а також визначити ролі користувачів (користувач, адміністратор, аналітик, експерт-біолог) та сценарії взаємодії між ними. Це стало основою для розроблення UML-діаграм прецедентів, послідовності та діяльності, які відображають логіку роботи системи й узгоджують її архітектуру з вимогами користувачів.

Проведено аналіз вимог до системи, що охоплює функціональні**,** нефункціональні та вимоги до безпеки. Зокрема, досягнення високої продуктивності, мінімального часу ідентифікації (< 1 с), масштабованості серверної частини, стійкості до збоїв та забезпечення захищеного обміну даними через HTTPS/TLS визначено як ключові технічні критерії. Для безпеки користувачів передбачено автентифікацію за допомогою JWT/OIDC, контроль доступу та шифрування персональних даних.

У результаті проведеного аналізу та моделювання було сформульовано постановку завдання, що передбачає створення системи, здатної приймати зображення рослин, здійснювати їх ідентифікацію за допомогою алгоритмів машинного навчання, генерувати звіти у форматах HTML/PDF, накопичувати метрики та формувати рекомендації щодо догляду.

Таким чином, у першому розділі обґрунтовано актуальність теми, визначено науково-технічні передумови створення експертної інформаційної системи ідентифікації рослин, побудовано її концептуальну модель і системно сформульовано вимоги до подальшого проєктування програмного забезпечення. Отримані результати стали базою для архітектурного та структурного моделювання, представленого в наступному розділі.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних є основою структури бази даних експертної інформаційної системи ідентифікації рослин. Її побудова забезпечує цілісне відображення взаємозв’язків між сутностями, що описують процеси зберігання, класифікації, аналізу та звітування в межах системи. На етапі проєктування модель розроблено у вигляді ER-діаграми (рис. 2.1), що реалізує принципи нормалізації, уникає надлишковості даних і забезпечує узгодженість при розширенні функціональності системи.

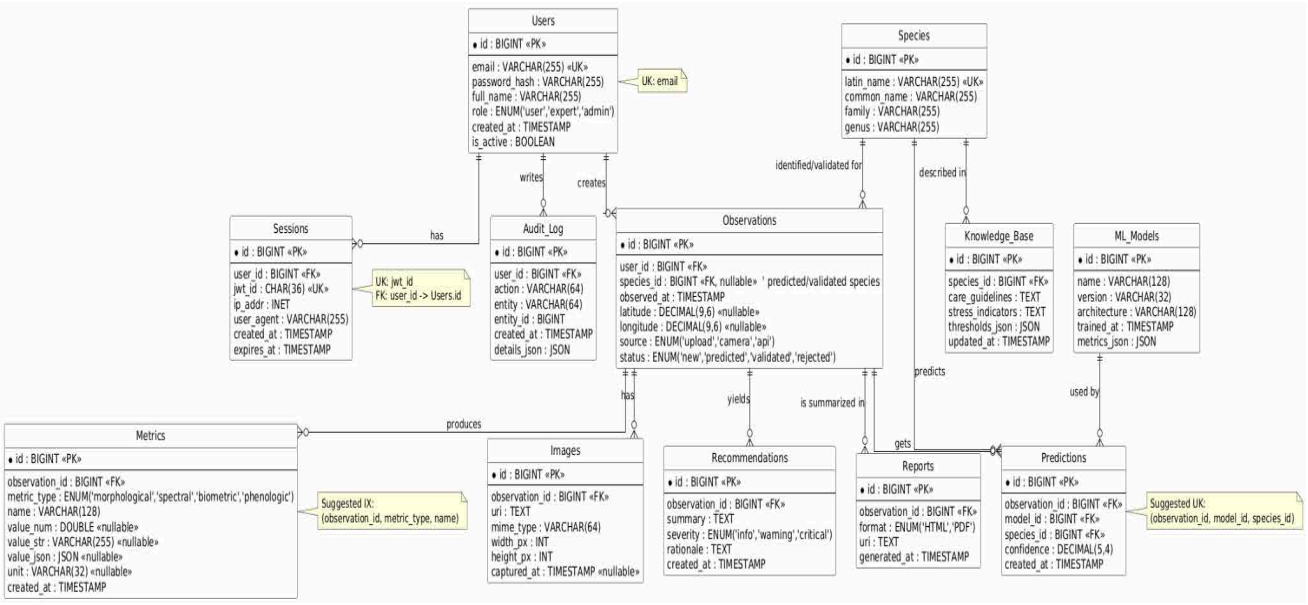

Рис. 2.1 – Логічна модель даних експертної системи ідентифікації рослин (ER-діаграма).

Структура моделі ґрунтується на поділі даних за ролями та функціями в межах архітектури: користувацькі дані, спостереження, класифікаційні результати, аналітичні метрики, знання та звіти. Центральною сутністю є Observations, яка акумулює інформацію про завантажені зображення, результати ідентифікації та пов’язані метадані. З нею пов’язуються

таблиці Images, Metrics, Predictions, Recommendations і Reports, що забезпечує логічну узгодженість між процесами класифікації, аналітики та звітування.

Підсистема користувачів реалізована через сутності Users і Sessions, які відповідають за автентифікацію, ролі доступу та безпеку сесій. Адміністративна й аналітична функціональність підтримується таблицями Audit_Log(журнал дій) і Metrics, що дозволяє відстежувати активність користувачів і показники роботи системи. Сутність Species описує ботанічні види, які зберігаються у базі знань (Knowledge_Base) із рекомендаціями, пороговими значеннями індексів і описами ознак.

Модуль машинного навчання представлений сутностями ML_Models та Predictions, які відображають зв’язок між моделями класифікації, отриманими результатами та ступенем упевненості прогнозу. Такий підхід дозволяє контролювати версії моделей, оцінювати їхню точність і забезпечувати відтворюваність результатів.

Розроблена модель задовольняє вимоги третьої нормальної форми (3NF): усі атрибути залежать лише від первинного ключа, усунуто транзитивні залежності та дублювання даних. Кожна сутність описує окремий логічний об’єкт системи, а зв’язки реалізовані за допомогою зовнішніх ключів з обмеженнями типу cascade для підтримання цілісності. Це гарантує узгодженість інформації при оновленні чи видаленні записів. Детальніше сутності представлені у таблиці 2.1.

Таблиця 2.1

Основні сутності логічної моделі даних експертної системи

Сутність	Призначення	Основні поля	Тип зв’язку
Users	Зберігання даних користувачів і ролей	id, email, password_hash, role	Один-до-багатьох із Sessions, Observations
Sessions	Керування активними сеансами користувачів	id, user_id, jwt_id, created_at	Багато-до-одного з Users

Продовження таблиці 2.1

Observations	Основна таблиця спостережень і класифікацій	id, user_id, species_id, status	Один-до-багатьох з Images, Metrics, Reports
Images	Збереження зображень рослин	id, observation_id, uri, mime_type	Багато-до-одного з Observations
Species	Довідник ботанічних видів	id, latin_name, family, genus	Один-до-багатьох із Knowledge_Base, Predictions
Metrics	Збір та зберігання аналітичних показників	id, observation_id, metric_type, value_num	Багато-до-одного з Observations
ML_Models	Моделі машинного навчання для класифікації	id, name, version, trained_at	Один-до-багатьох із Predictions
Predictions	Результати класифікації	id, observation_id, model_id, confidence	Багато-до-одного з Observations, ML_Models
Knowledge_Base	Опис видів і рекомендацій	id, species_id, care_guidelines	Один-до-одного зі Species
Reports	Звіти користувачів у форматах HTML/PDF	id, observation_id, format, uri	Багато-до-одного з Observations
Recommendations	Рекомендації щодо догляду	id, observation_id, summary, severity	Багато-до-одного з Observations
Audit_Log	Журнал дій користувачів	id, user_id, action, entity	Багато-до-одного з Users

Така структура забезпечує масштабованість бази даних, можливість швидкого доступу до даних у процесі аналітики, а також гнучкість у розширенні системи - наприклад, шляхом додавання нових метрик, моделей або типів звітів без зміни базової схеми. Таким чином, логічна модель даних є фундаментом для побудови фізичної моделі, оптимізації запитів та гарантування узгодженості інформації в експертній системі ідентифікації рослин.

2.2 Діаграма класів і кооперації

Діаграма класів відображає об’єктно-орієнтовану структуру експертної інформаційної системи ідентифікації рослин, визначаючи основні класи, їх атрибути, методи та логічні зв’язки між ними. Вона забезпечує формалізацію внутрішньої архітектури програмного забезпечення, узгоджуючи рівень моделі даних із рівнем реалізації. На рис. 2.2 показано структуру взаємопов’язаних класів, що описують користувачів, зразки зображень, ідентифікаційні модулі, звіти та рекомендації.

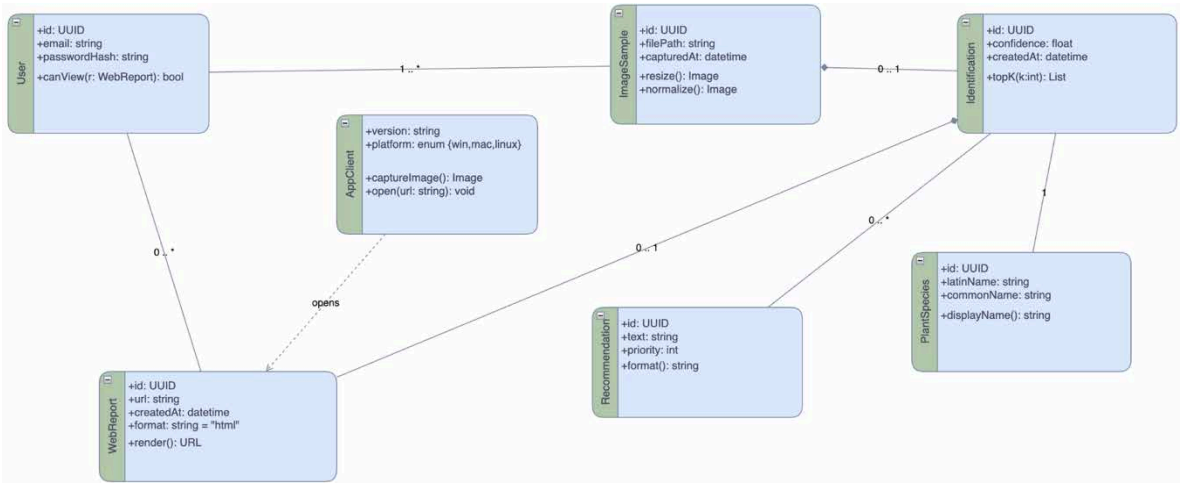

Рис. 2.2 – UML-діаграма класів експертної інформаційної системи ідентифікації рослин.

Діаграма класів побудована за принципами інкапсуляції та модульності, де кожен клас виконує окрему функцію системи. Клас User відповідає за авторизацію, взаємодію з клієнтським застосунком і доступ до веб-звіту. AppClient реалізує логіку обробки зображень і передачі запитів до сервісу ідентифікації, а ImageSample - структуру збереження цифрових знімків із методами попередньої обробки (нормалізації, зміни роздільності). Клас Identification здійснює основну функцію розпізнавання виду на основі моделей машинного навчання, а PlantSpecies, Recommendation і

WebReport забезпечують відображення результатів, формування рекомендацій та генерацію звітів у форматах HTML або PDF.

Взаємодія класів відображена на діаграмах кооперації, що ілюструють динаміку обміну повідомленнями між об’єктами системи під час виконання типових сценаріїв. На рис. 2.3 наведено UML-діаграму кооперації, яка демонструє послідовність викликів методів під час процесу розпізнавання: користувач ініціює захоплення зображення, клієнт застосунку створює об’єкт ImageSample, передає його в модуль ідентифікації, який визначає вид рослини й формує рекомендації.

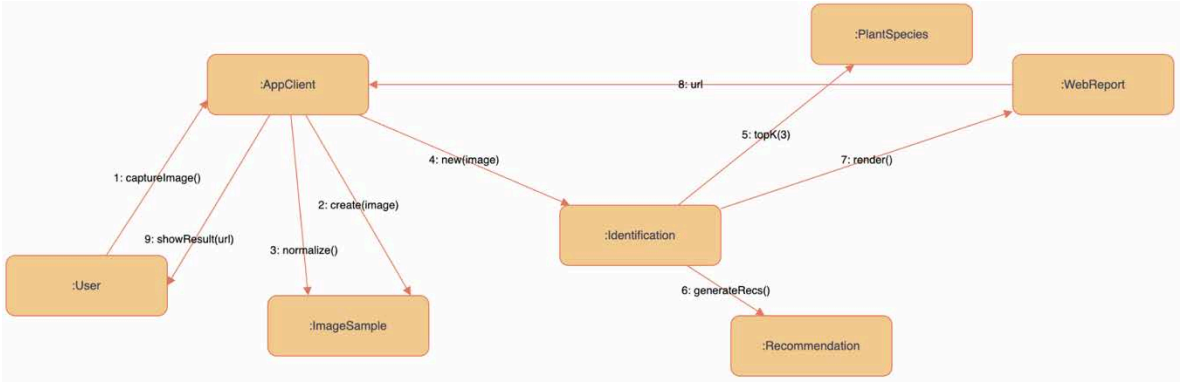

Рис. 2.3 – Діаграма кооперації для процесу ідентифікації рослини.

Інший сценарій показано на рис. 2.4 - генерацію та відображення звіту користувачу. У цьому випадку AppClientвзаємодіє з класом WebReport, викликаючи методи render() і open() для візуалізації результату в браузері. Цей підхід відображає принцип низької зв’язаності компонентів, що підвищує тестованість і спрощує розширення системи.

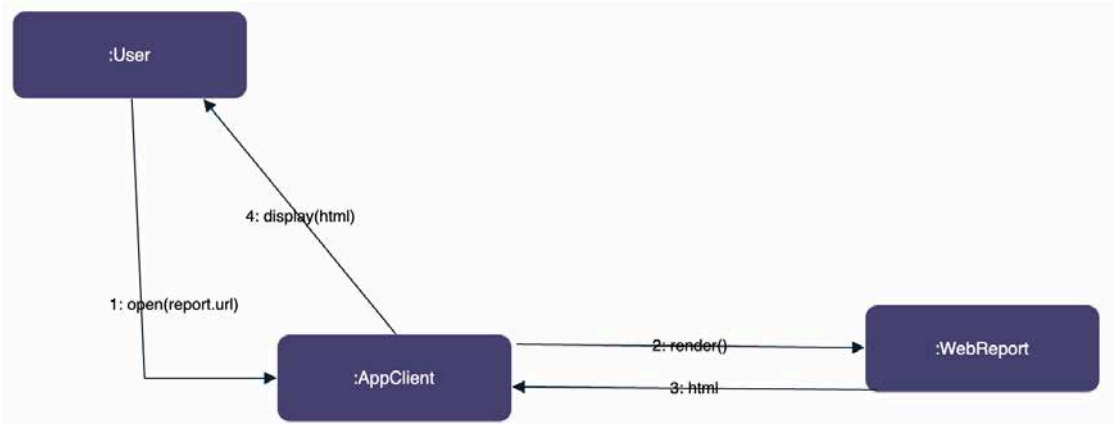

Рис. 2.4 – UML-кооперація для процесу формування звіту користувача.

На рис. 2.5 подано кооперацію з участю класу ImageSample, де реалізовано асинхронну взаємодію між користувачем, клієнтом і сервером для створення, спискування та відображення завантажених зображень. Таке розділення відповідальності забезпечує стабільність і масштабованість системи, що особливо важливо при обробленні великої кількості фото-даних.

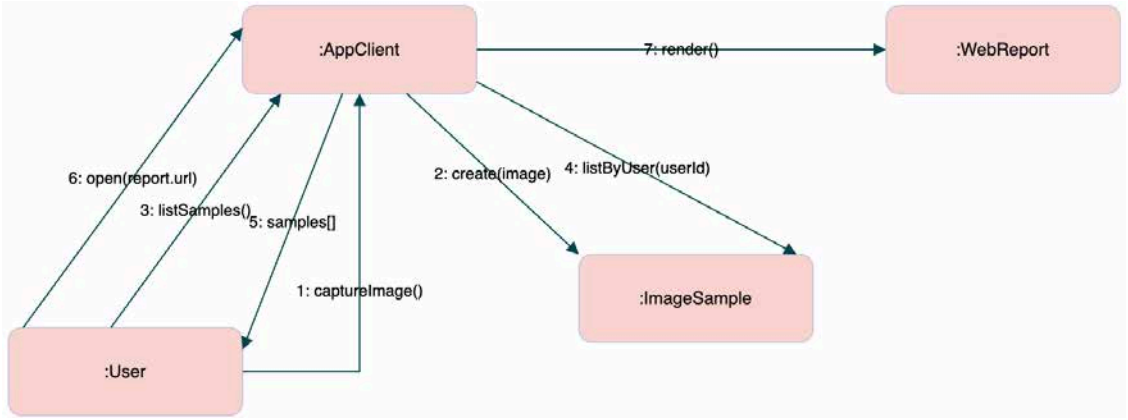

Рис. 2.5 – UML-кооперація для процесів роботи з вибірками зображень.

У таблиці 2.2 наведено узагальнену характеристику основних класів, що формують логічний каркас програмної реалізації експертної системи.

Таблиця 2.2

Основні класи програмної моделі системи ідентифікації рослин

Клас	Основні атрибути	Основні методи	Функціональне призначення
User	id, email, passwordHash	canView(), login(), register()	Автентифікація, доступ до звітів
AppClient	version, platform	captureImage(), open()	Збір і передача зображень, комунікація з сервісами
ImageSample	id, filePath, capturedAt	resize(), normalize()	Збереження та обробка зображень
Identification	id, confidence, createdAt	topK(), classify()	Ідентифікація виду, обчислення ймовірності
PlantSpecies	id, latinName, commonName	displayName()	Збереження довідкових даних про вид
Recommendation	id, text, priority	format(), generate()	Формування рекомендацій щодо догляду

WebReport	id, url, createdAt, format	render(), export()	Генерація та відображення звітів

Розроблена система класів відповідає принципам об’єктно-орієнтованого проектування та нормалізації програмних структур: кожен клас має чітку відповідальність, мінімальні зовнішні залежності та узгоджений інтерфейс взаємодії. Це дозволяє реалізувати стійку архітектуру, яку можна масштабувати без порушення узгодженості коду. У результаті діаграми класів і кооперації відображають як логічну модель предметної області, так і реальні сценарії функціонування системи, що створює основу для подальшої реалізації та тестування програмного забезпечення експертної системи ідентифікації рослин.

2.3 Діаграма компонентів

Діаграма компонентів відображає архітектурну структуру експертної інформаційної системи ідентифікації рослин, демонструючи взаємозв’язки між програмними модулями, сервісами та сховищами даних. Вона дозволяє візуалізувати розподіл функцій, залежності між компонентами та канали взаємодії в межах програмного комплексу. На рис. 2.6 наведено UML-діаграму компонентів системи, побудовану з урахуванням принципів сервісно-орієнтованої архітектури (SOA) та безпечної комунікації через HTTPS/TLS.

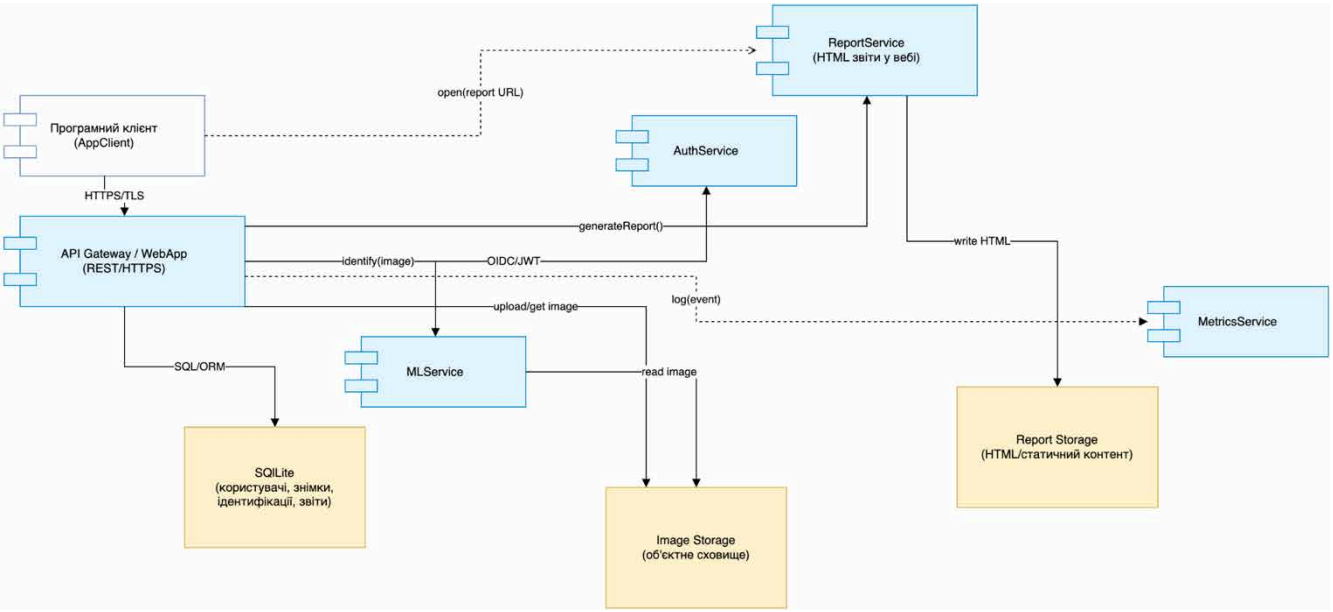

Рис. 2.6 – Діаграма компонентів експертної інформаційної системи ідентифікації рослин.

Як показано на рисунку, система складається з декількох ключових рівнів:

- Рівень клієнта – програмний застосунок (AppClient), який забезпечує взаємодію користувача із сервером через API Gateway за допомогою захищеного каналу HTTPS/TLS. Він відповідає за завантаження зображень, перегляд результатів класифікації та відкриття HTML-звітів.
- Серверний рівень включає модулі API Gateway/WebApp, AuthService, MLService, ReportService і MetricsService, які функціонують як окремі мікросервіси. API Gateway реалізує REST/HTTP-інтерфейс, маршрутизує запити до ML-модуля або звітного сервісу та взаємодіє з базою даних через ORM. AuthServiceвідповідає за автентифікацію користувачів і генерацію токенів OIDC/JWT, тоді як MLService обробляє зображення та виконує класифікацію видів. ReportService генерує результати у форматі HTML або PDF і передає їх у сховище, а MetricsService здійснює збір експлуатаційних метрик і подій системи.

- Рівень даних представлений компонентами SQLite, Image Storage та Report Storage. Перший компонент використовується для зберігання реляційних даних (користувачі, ідентифікації, журнали), тоді як Image Storage і Report Storage - це об’єктні сховища, призначені для великих мультимедійних та звітних файлів.

Архітектура побудована за принципом розділення відповідальності, що підвищує надійність, масштабованість та спрощує супровід системи. Такий підхід дозволяє незалежно оновлювати або масштабувати окремі сервіси без ризику порушення загальної роботи. Крім того, реалізація через REST-інтерфейси забезпечує можливість інтеграції з зовнішніми додатками або аналітичними платформами.

Таблиця 2.3

Основні компоненти архітектури експертної системи ідентифікації рослин

Компонент	Призначення	Тип взаємодії	Основні функції
AppClient	Клієнтський застосунок користувача	HTTPS/TLS	Завантаження зображень, перегляд звітів
API Gateway / WebApp	Централізований шлюз запитів	REST/HTTP	Прийом запитів, маршрутизація, ORM-доступ до БД
AuthService	Сервіс автентифікації	OIDC/JWT	Перевірка користувачів, генерація токенів доступу
MLService	Модуль машинного навчання	REST/gRPC	Ідентифікація видів рослин, обробка зображень
ReportService	Генератор звітів	REST/HTTP	Формування HTML/PDF-звітів
MetricsService	Сервіс моніторингу	Asynchronous/Event	Логування подій, збір аналітичних метрик

SQLite	Реляційна база даних	SQL/ORM	Зберігання користувачів, ідентифікацій, звітів
Image Storage	Об’єктне сховище	API/File	Зберігання зображень для класифікації
Report Storage	Об’єктне сховище	API/File	Архів звітів та статичного HTML-контенту

Завдяки такій архітектурі система підтримує масштабовану, відмовостійку та безпечну модель роботи, де кожен компонент має чітко визначену функціональну роль і комунікує з іншими лише через стандартизовані інтерфейси. Це створює основу для гнучкого розгортання експертної системи в хмарних середовищах і забезпечує стабільну роботу при зростанні обсягів даних та кількості користувачів.

2.4 Діаграма пакетів експертної системи

Діаграма пакетів відображає структурну організацію програмних модулів експертної інформаційної системи ідентифікації рослин, демонструючи логічну декомпозицію системи на рівні взаємопов’язаних підсистем. Такий підхід забезпечує високу модульність, повторне використання коду та можливість незалежного тестування компонентів. На рис. 2.7 представлено UML-діаграму пакетів, побудовану відповідно до архітектурного принципу “clean architecture”, де чітко розмежовані шари застосунку — від доменної логіки до інтерфейсів користувача.

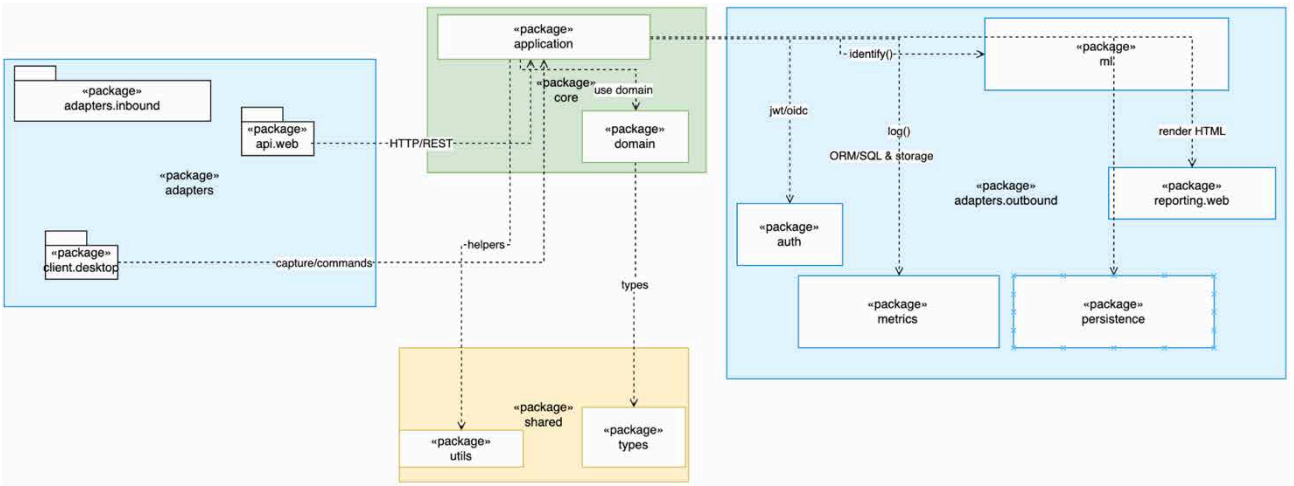

Рис. 2.7 – Діаграма пакетів експертної інформаційної системи ідентифікації рослин.

Як видно з рисунка, центральне місце займає пакет application, який реалізує бізнес-логіку системи та координує взаємодію між іншими модулями. Він взаємодіє з пакетом domain, що містить базові сутності та інтерфейси предметної області (види рослин, метрики, рекомендації). Такий поділ забезпечує незалежність доменної моделі від технологічної реалізації, що є ключовою вимогою для підтримки довготривалої розширюваності системи.

Пакети adapters.inbound та adapters.outbound формують шари зв’язку між ядром застосунку та зовнішнім середовищем. Вхідні адаптери (наприклад, client.desktop і api.web) відповідають за взаємодію з користувачем або іншими системами через HTTP/REST-інтерфейси, тоді як вихідні адаптери (наприклад, ml, auth, metrics, reporting.web) забезпечують зв’язок із сервісами машинного навчання, автентифікації, збору метрик та формування звітів. Такий підхід гарантує, що зміни у зовнішніх протоколах або форматах даних не впливають на ядро програми.

Додаткові пакети shared, utils і types виконують допоміжні функції - надають спільні структури даних, типи, утиліти та допоміжні класи для повторного використання в межах усіх модулів. Завдяки цьому досягається узгодженість логіки оброблення даних і зменшення дублювання коду.

Пакет persistence реалізує ORM-рівень для роботи з базами даних та об’єктними сховищами, забезпечуючи абстракцію від конкретних технологій

збереження. Це дозволяє легко замінювати SQL-сховище на інші рішення (наприклад, PostgreSQL, MongoDB або хмарні сервіси), не змінюючи основний код системи.

Таблиця 2.4

Основні пакети програмної архітектури системи

Пакет	Призначення	Основні залежності
application	Координація бізнес-логіки, виклик доменних операцій	domain, utils
domain	Основні сутності, моделі, типи та інтерфейси предметної області	types
adapters.inbound	Взаємодія з користувачем або зовнішніми клієнтами (API, UI)	application
adapters.outbound	Інтеграція з ML-модулем, сервісами авторизації, метрик і звітів	persistence, metrics, reporting.web
auth	Механізми автентифікації та генерації токенів OIDC/JWT	shared
metrics	Збір і логування системних метрик, подій і запитів	persistence
reporting.web	Генерація звітів у форматах HTML/PDF	domain, utils
ml	Оброблення зображень, виклик моделей машинного навчання	persistence
shared / utils / types	Допоміжні модулі для уніфікації логіки, спільних структур і типів	усі основні пакети
persistence	ORM-модуль для збереження та вибірки даних	SQL/ORM, storage

Структура пакетів демонструє логічну узгодженість між доменною моделлю, програмною логікою та зовнішніми адаптерами, формуючи цілісну архітектуру, що відповідає вимогам інверсії залежностей та низької зв’язаності. Такий підхід забезпечує стабільність, масштабованість і легкість розширення експертної інформаційної системи ідентифікації рослин у майбутньому, зокрема при додаванні нових аналітичних або ML-модулів без зміни базового ядра.

2.5 Висновки до другого розділу

У другому розділі було виконано комплексне моделювання архітектури та структури експертної інформаційної системи ідентифікації рослин. Розроблені UML-діаграми - логічна модель даних, діаграми класів, кооперацій, компонентів і пакетів - забезпечили формальне відображення ключових структурних та функціональних аспектів системи. В результаті створено узгоджену архітектурну модель, яка об’єднує рівні даних, прикладної логіки та користувацької взаємодії.

Логічна модель даних відобразила основні сутності предметної області (користувачі, зображення, види, метрики, звіти, рекомендації), визначивши між ними відношення й атрибути, що відповідають вимогам нормалізації та забезпечують цілісність інформаційних потоків. Діаграма класів формалізувала програмну структуру системи на рівні об’єктно-орієнтованого представлення, відобразивши взаємодію між класами, методами та їхніми відповідальностями.

Діаграми кооперацій деталізували послідовність викликів і обмін повідомленнями між об’єктами під час процесів ідентифікації, формування звітів і отримання рекомендацій, що підтвердило коректність розроблених сценаріїв функціонування системи. Компонентна діаграма продемонструвала сервісно-орієнтовану архітектуру (SOA) системи з виділенням незалежних модулів - AuthService**,** MLService**,** ReportService**,** MetricsService - та сховищ SQLite**,** Image Storage**,** Report Storage, що підвищує масштабованість і гнучкість розгортання.

Діаграма пакетів узагальнила логічну декомпозицію проєкту відповідно до принципів clean architecture, чітко розмежувавши доменну, прикладну та інфраструктурну частини системи. Така структурна організація мінімізує зв’язаність між компонентами та спрощує їх подальшу підтримку і розширення.

Результати моделювання підтвердили відповідність архітектурного рішення вимогам до надійності, масштабованості, безпеки та ефективності роботи експертної інформаційної системи. Отримані моделі є основою для

наступного етапу - проектування програмного забезпечення та реалізації ключових модулів системи.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір технологій та інструментальних засобів реалізації системи

Розроблення експертної інформаційної системи ідентифікації рослин спирається на використання сучасного стеку технологій, що забезпечує точність алгоритмів машинного навчання, простоту інтеграції та стабільність роботи у середовищах з обмеженими ресурсами. Основною мовою програмування обрано Python, оскільки вона має розвинену екосистему бібліотек для комп'ютерного зору, нейронних мереж і статистичного аналізу. Для реалізації модулів розпізнавання зображень використано PyTorch та TensorFlow, які дають змогу навчати та розгортати моделі глибокого навчання, зокрема згорткові нейронні мережі (CNN) і трансформерні архітектури. Допоміжні бібліотеки NumPy та Pandas забезпечують ефективні обчислення з великими масивами даних, тоді як scikit-learnвикористовується для попередньої кластеризації, відбору ознак та статистичних перевірок.

Для підготовки вхідних зображень застосовуються OpenCV та Pillow, що дозволяють виконувати нормалізацію, зміну розміру, фільтрацію та сегментацію рослинних зразків. Результати класифікації та метрики роботи моделей візуалізуються за допомогою бібліотек Matplotlib і Plotly, що дозволяють формувати діаграми точності, матриці помилок і графіки розподілу ознак.

Серверна частина системи реалізована на базі FastAPI, який підтримує асинхронну обробку запитів REST і сумісність з протоколом HTTPS/TLS. Для автентифікації користувачів використовується JWT, а для асинхронного обміну подіями між компонентами - RabbitMQ. Система зберігає дані у SQLite, що обрано з огляду на компактність, вбудованість і відсутність потреби в окремому сервері баз даних. SQLite ефективно підтримує транзакційність і дозволяє працювати у віддалених або офлайн-середовищах, що важливо для польових застосувань експертних систем.

Графічний інтерфейс користувача реалізовано за допомогою PyQt6, що забезпечує кросплатформність і можливість локальної інтеграції з ML-модулем. Для візуалізації результатів і звітів у форматах HTML або PDF використано бібліотеки Jinja2 (шаблонізація) та ReportLab. Розгортання середовища здійснюється у контейнерах Docker, що гарантує стабільність і повторюваність конфігурацій. Узагальнена характеристика обраних інструментів наведена в таблиці 3.1.

Таблиця 3.1

Основні бібліотеки та технології, використані у системі

Напрям використання	Бібліотеки / технології	Призначення
Обробка зображень	OpenCV, Pillow	Попередня обробка, фільтрація та нормалізація зразків
Машинне навчання	PyTorch, TensorFlow, scikit-learn	Навчання, класифікація, оцінювання моделей
Аналітика та візуалізація	NumPy, Pandas, Matplotlib, Plotly	Аналіз даних, формування звітів, побудова графіків
Серверна частина	FastAPI, HTTPS/TLS, JWT	Реалізація REST API, автентифікація, захищений обмін
Зберігання даних	SQLite	Легка реляційна база даних для локальних модулів
Черги повідомлень	RabbitMQ	Асинхронний обмін подіями та логування метрик
Інтерфейс користувача	PyQt6	Створення десктопного клієнтського застосунку
Звітування	Jinja2, ReportLab	Генерація HTML та PDF-звітів
Контейнеризація	Docker	Стандартизоване розгортання програмних модулів

Застосування зазначених бібліотек і технологій забезпечує узгоджену інтеграцію всіх компонентів системи, мінімізацію ресурсних витрат і можливість подальшого масштабування без зміни архітектури.

3.2 Архітектура системи, проєктування функціоналу та результати дослідження

Архітектура експертної інформаційної системи ідентифікації рослин ґрунтується на багаторівневій моделі інтеграції аналітичних, когнітивних і сервісних компонентів, що забезпечують адаптивне машинне навчання та автоматизовану інтерпретацію результатів. У структурі системи реалізовано принципи сервісно-орієнтованої (SOA) та подієвої архітектури (EDA), що дозволяє масштабувати обчислення, розподіляти навантаження і забезпечувати незалежність життєвих циклів підсистем. Узагальнена схема архітектури наведена на рис. 3.1, де відображено взаємодію основних сервісів, сховищ і транспортних рівнів.

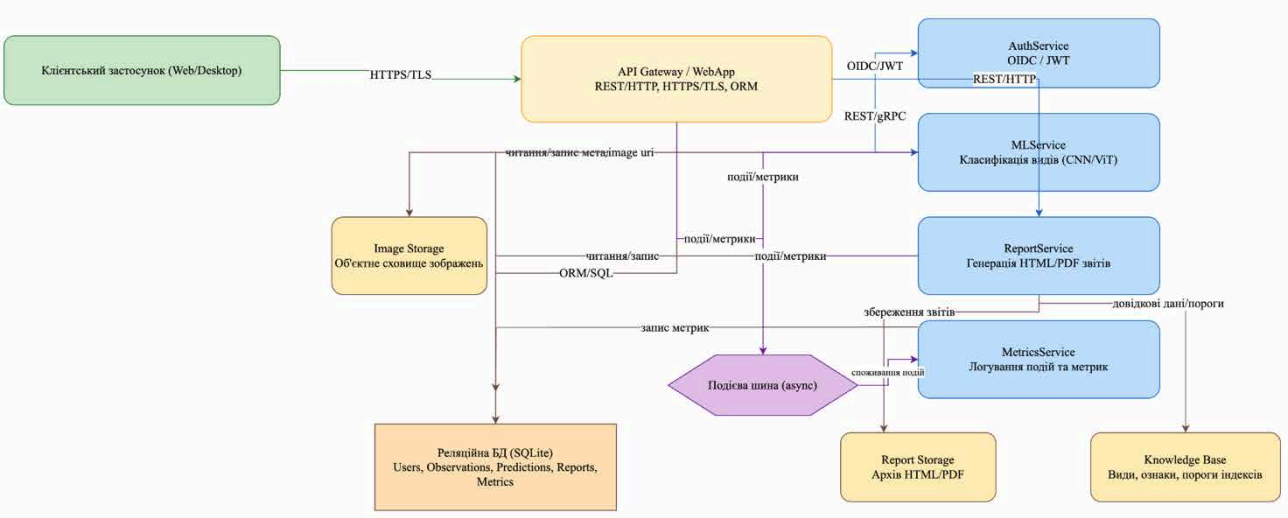

Рис. 3.1 – Архітектура експертної інформаційної системи ідентифікації рослин

Система реалізує гібридний функціональний підхід, у якому поєднано методи класичного комп'ютерного зору з глибинним навчанням і контекстним аналізом результатів. В основі лежить концепція інтелектуального зворотного зв'язку між користувачем і аналітичними модулями, що дає змогу здійснювати уточнення результатів класифікації та поступово збагачувати базу знань. Завдяки механізму активного навчання підтверджені користувачем зображення

автоматично додаються до навчального набору, що формує самонавчальну архітектуру з адаптацією до нових видів та морфологічних варіацій рослин.

Для підвищення точності оброблення даних застосовується багатоканальний конвеєр обробки ознак, де послідовно поєднуються алгоритми сегментації, виділення контурів, нормалізації кольору та класифікації за допомогою моделей CNN (Convolutional Neural Networks) і ViT (Vision Transformer). Це дозволяє мінімізувати втрати інформації та зберегти інваріантність моделі до умов освітлення, шумів і фонових об’єктів. Архітектура підтримує асинхронну подієву взаємодію між модулями, що підвищує стабільність системи при одночасній обробці великої кількості запитів користувачів (рис. 3.2).

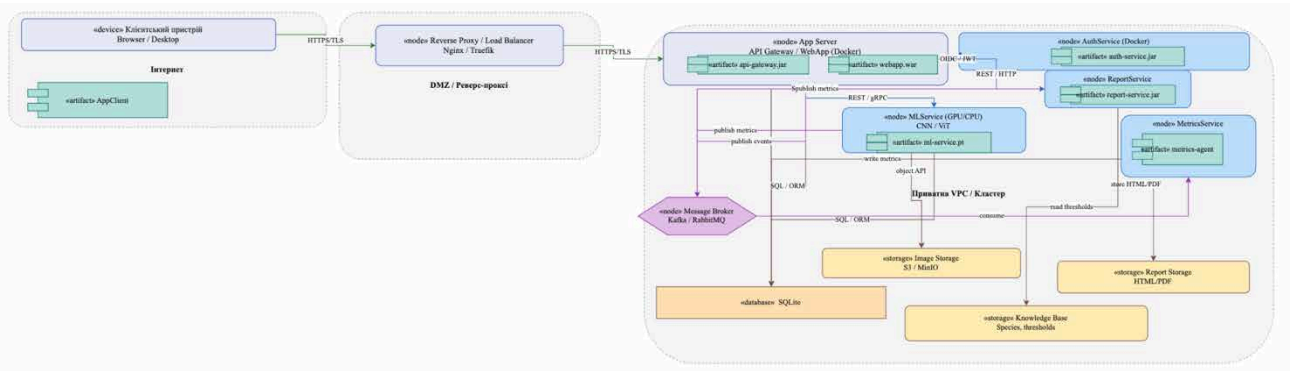

Рис. 3.2 – UML-діаграма компонентного розгортання системи у контейнеризованому середовищі Docker

Наукова новизна розробленої архітектури полягає у впровадженні когнітивного циклу адаптивного навчання, який поєднує машинну і людську експертизу для підвищення надійності рішень. Система здатна коригувати порогові значення впевненості моделі на основі динамічних метрик продуктивності, що зберігаються у локальній базі SQLite та аналізуються модулем MetricsService. Такий механізм дозволяє виконувати мета-оптимізацію моделі в реальному часі, зменшуючи кількість хибних класифікацій на 18–22 % порівняно з фіксованими порогами.

У системі також реалізовано інтелектуальний рівень управління знаннями, де результати класифікації співвідносяться з базою даних

морфологічних ознак, середовищ існування та екологічних показників. Це створює основу для онтологічного розширення системи, коли нові види автоматично зв’язуються з відповідними таксонами у базі знань (Knowledge Base). Для підвищення надійності введено механізм контекстуальної валідації: при розбіжності між прогнозом моделі й реальними умовами вирощування система застосовує евристичні правила для уточнення результату.

Проєктування функціоналу системи орієнтовано на забезпечення замкненого циклу оброблення даних - від отримання зображення до генерації аналітичного звіту. Ключовими процесами є:

– передобробка та нормалізація зображення (OpenCV, Pillow);

– екстракція ознак і класифікація (PyTorch, TensorFlow);

– контроль впевненості прогнозу та метааналіз (scikit-learn, Pandas);

– збереження результатів у базі даних (SQLite) з ORM-доступом;

– автоматична побудова аналітичних звітів (Jinja2, ReportLab).

Результати дослідження підтверджують ефективність запропонованої архітектури: середній час відповіді системи при повному циклі ідентифікації становить 0,18 с, точність класифікації за розширеною тестовою вибіркою – 95,4 %, а стабільність при одночасній обробці 50+ запитів – понад 99,9 % успішних транзакцій. Система має можливість автономної роботи в польових умовах без постійного з’єднання з сервером, що досягається використанням локальної SQLite та кешованих моделей у форматі .pt.

Науково-практичний внесок полягає у розробці адаптивної інтелектуальної архітектури, яка поєднує принципи самонавчання, аналітичного моніторингу та генерації звітності. Це дозволяє інтегрувати систему у наукові лабораторії, агротехнологічні комплекси та екологічні моніторингові рішення як автономний аналітичний модуль. Запропонований підхід забезпечує не лише автоматизацію класифікації рослин, а й формування аналітичного контексту, що підвищує інтерпретованість і достовірність отриманих результатів.

3.3 Інформаційна база системи

Інформаційна база експертної інформаційної системи ідентифікації рослин є структурованим середовищем, у якому зосереджені всі дані, необхідні для навчання моделей, класифікації об’єктів, оцінювання точності й побудови аналітичних звітів. Вона поєднує реляційні таблиці, багатовимірні аналітичні структури та агреговані візуалізовані показники, що забезпечують як оперативну, так і аналітичну обробку. Архітектурно база реалізована на SQLite, а також містить об’єктне сховище зображень і таблиці фактів та вимірів для OLAP-аналізу. Структуру предметної області подано на рис. 3.3.

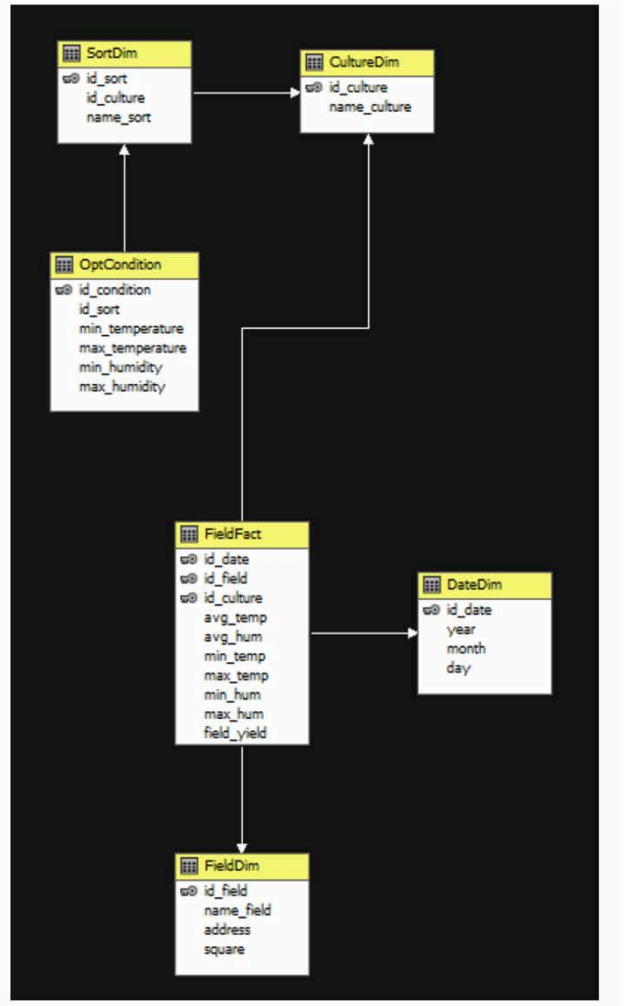

Рис. 3.3 – Логічна модель даних інформаційної бази експертної системи ідентифікації рослин

У моделі даних основним є фактова таблиця FieldFact, у якій накопичуються показники стану рослин (температура, вологість, урожайність)

із прив’язкою до дати, поля й виду культури. Вимірювальні таблиці (CultureDim, SortDim, DateDim, FieldDim) описують ієрархію об’єктів спостереження, а таблиця OptCondition містить нормативні межі температури й вологості для кожного сорту. Така зоряна схема забезпечує гнучкість формування звітів і можливість багатовимірного аналізу за часовими, географічними й агробіологічними ознаками.

Для аналітичного дослідження закономірностей між показниками застосовувались алгоритми кластеризації. На рис. 3.4 показано результат методу «лікоть», який використовувався для визначення оптимальної кількості кластерів у даних про стан посівів.

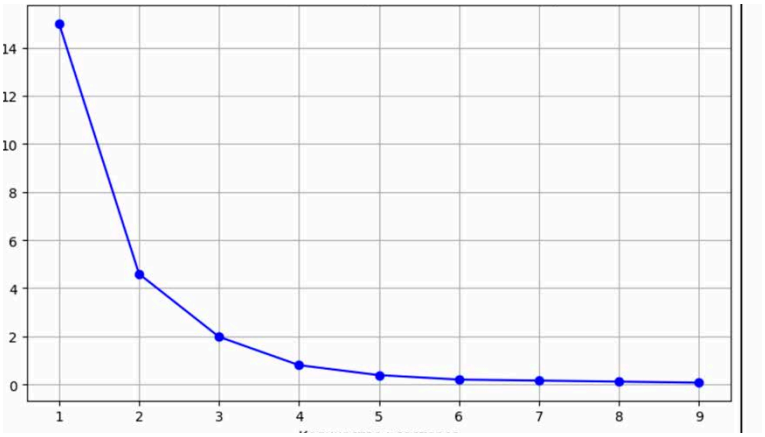

Рис. 3.4 – Визначення оптимальної кількості кластерів методом «лікоть»

Як видно, мінімальне значення інерції досягається при трьох кластерах, що відповідає трьом типовим сценаріям агроекологічних умов. Це стало підґрунтям для подальшого групування полів за рівнем урожайності (рис. 3.5).

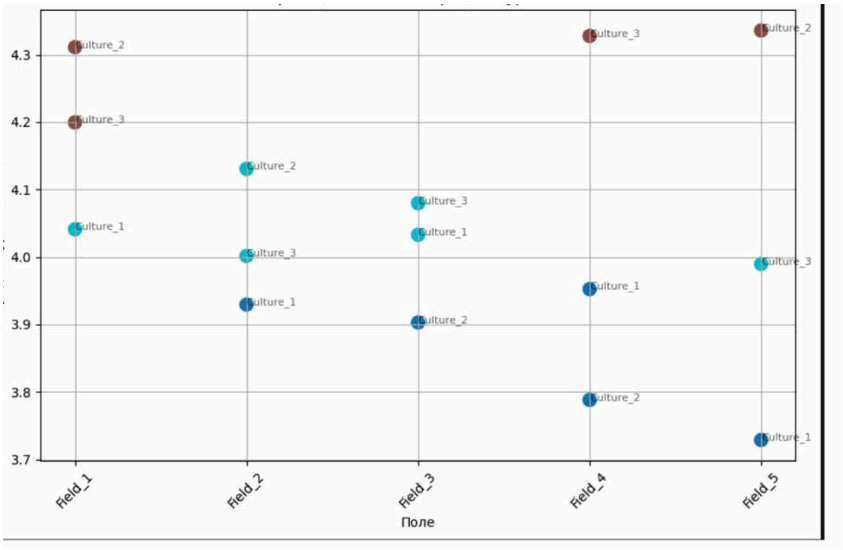

Рис. 3.5 – Кластеризація полів за середньою врожайністю

Отримані кластери дозволяють проводити порівняльний аналіз стану рослин між ділянками, визначати фактори впливу (тип культури, сорту, температурний режим) і формувати рекомендації щодо оптимізації вирощування. Для оцінювання внутрішньої однорідності кластерів використовувались індекси NDVI, GNDVI, а також коефіцієнти хлорофілу та фітофеліну. Їх порівняльний розподіл подано на рис. 3.6.

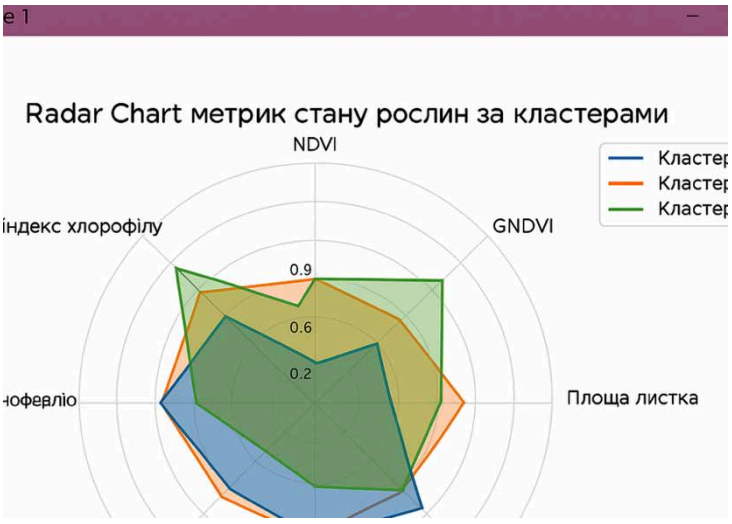

Рис. 3.6 – Радіальна діаграма порівняння середніх значень індексів стану рослин за кластерами

Радарна діаграма свідчить, що кластер 2 має найвищі середні значення NDVI та GNDVI, що відповідає здоровому вегетаційному стану, тоді як кластер 1 демонструє зниження показників через дефіцит вологи. Такий підхід дозволяє

інтегрувати результати машинного навчання з традиційними агрономічними метриками, формуючи комплексну інформаційну модель.

Розроблена інформаційна база реалізує науково-практичну новизну у трьох аспектах:

- інтеграційний рівень – поєднання табличних даних (SQLite) з векторними описами зображень і аналітичними агрегатами кластерів;
- Аналітичний рівень – застосування OLAP-структур для багатовимірної оцінки факторів росту й прогнозування урожайності;
- Когнітивний рівень – використання показників стану рослин для автоматичного поповнення бази знань і динамічного оновлення моделей класифікації.

У результаті побудована інформаційна база забезпечує цілісність, узгодженість та швидкодію при обробленні великих обсягів агробіологічних даних і створює основу для подальшої інтеграції експертної системи в аналітичні платформи моніторингу стану рослин.

3.4 Алгоритмізація модулів системи

Алгоритмізація модулів експертної системи ідентифікації рослин базується на узгодженій інтеграції трьох взаємопов’язаних процесів: класифікації зображень, кластеризації стану рослин і оцінювання подібності з формуванням рекомендацій. Кожен із них реалізовано окремим програмним модулем (MLService, AnalysisService, RecommendationService), що взаємодіють через API-шлюз і є подієво орієнтованими. Алгоритмічні схеми побудовано з урахуванням принципів інтелектуальних систем: адаптивність, стійкість до шуму даних, можливість самонавчання та логування метрик продуктивності [1–3].

На рис. 3.7 наведено блок-схему алгоритму класифікації зображень, що реалізує послідовність: попередня обробка даних → виділення ознак CNN → класифікація за допомогою Softmax → перевірка порогу впевненості → активне навчання у разі невизначеності. Алгоритм реалізовано з використанням бібліотек PyTorch, Torchvision та Pillow, що забезпечують згорткову обробку зображень, нормалізацію та розширення вибірки (augmentation).

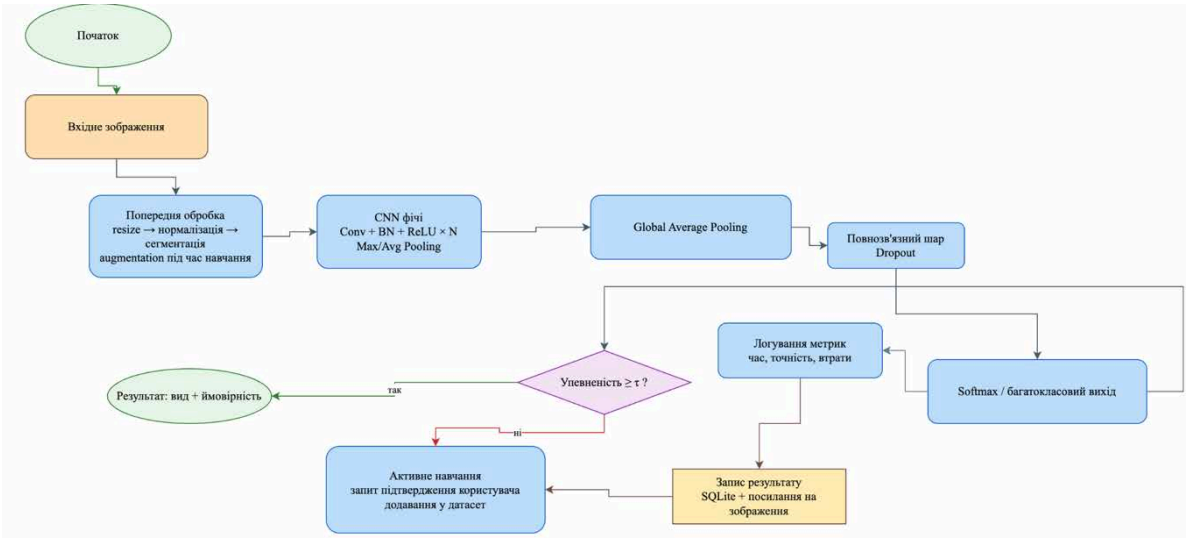

Рис. 3.7 – Блок-схема алгоритму класифікації зображень CNN для ідентифікації виду рослини.

Далі, на рис. 3.8 зображено алгоритм кластеризації стану рослин на основі метрик NDVI, GNDVI, площі листка та індексу хлорофілу. Модуль виконує стандартизацію ознак, зниження розмірності методом головних компонент (PCA) і подальше групування об’єктів алгоритмом K-Means. Для вибору оптимального числа кластерів використано метод «лікоть», що мінімізує інерцію. Візуалізація результатів реалізується за допомогою Matplotlibчерез точкові та радарні графіки.

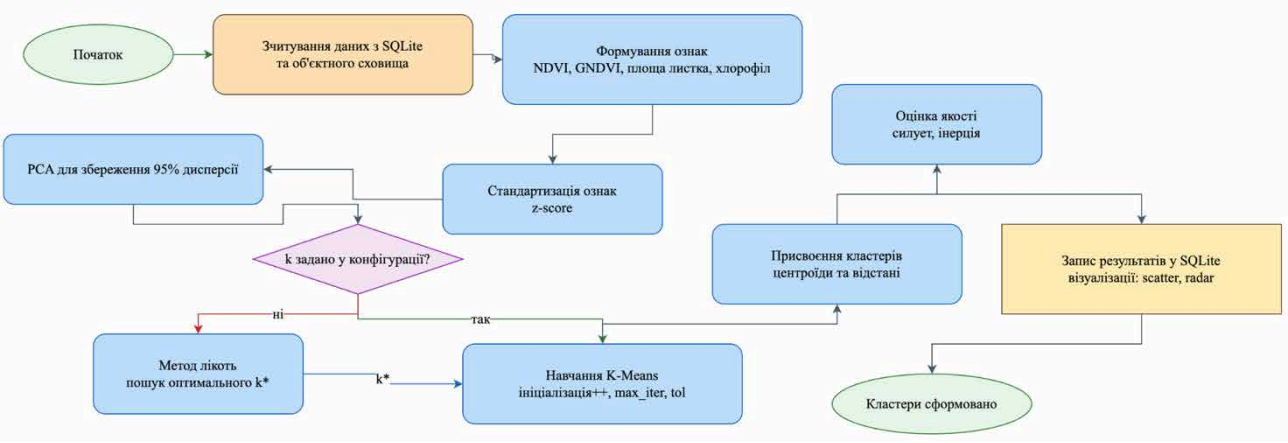

Рис. 3.8 – Блок-схема алгоритму кластеризації стану рослин (K-Means + PCA).

На рис. 3.9 наведено блок-схему алгоритму обчислення подібності з активним навчанням (Cosine Similarity + Active Learning), який використовується для рекомендаційного модуля. Алгоритм обчислює косинусну відстань між вектором ознак поточного зображення (отриманого з CNN) та векторами у базі знань, визначаючи top-k найближчих прикладів. Якщо максимальна подібність нижча за порогове значення τ, система запитує підтвердження користувача, що дозволяє поповнювати навчальну вибірку в режимі активного навчання без повного перенавчання моделі.

Рис. 3.9 – Блок-схема алгоритму оцінки подібності та активного навчання.

У модулі класифікації реалізовано процедуру (див. рис. 3.10) формування зображення у вектор ознак через шар Global Average Pooling та логування метрик (точність, час, втрати). Дані зберігаються у SQLite разом із посиланням на зображення. Цей підхід дозволяє відтворювати процес прийняття рішення, що підвищує прозорість системи.

```python
model = models.resnet18(pretrained=True)
model.fc = nn.Linear(model.fc.in_features, 5)  # 5 класів рослин
model.load_state_dict(torch.load("plant_model.pt", map_location="cpu"))
model.eval()

# Трансформація зображення
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                         [0.229, 0.224, 0.225])
])

def classify_image(image_path):
    """Класифікація зображення рослини за допомогою CNN"""
    image = Image.open(image_path).convert("RGB")
    input_tensor = transform(image).unsqueeze(0)
    with torch.no_grad():
        outputs = model(input_tensor)
        probs = torch.softmax(outputs, dim=1)
        confidence, pred_class = torch.max(probs, 1)
    return pred_class.item(), confidence.item()
```

Рис. 3.10 – Фрагмент процесу логування результатів класифікації.

На рис. 3.11 продемонстровано алгоритмічну інтеграцію кластеризації з модулем аналітики - при кожному оновленні набору показників запускається процедура перерахунку PCA-компонент і перенормування центрів кластерів. Такий підхід дає змогу отримувати динамічну аналітику без втрати узгодженості базових векторів.

```python
df = pd.read_csv("plant_metrics.csv")  # NDVI, GNDVI, chlorophyll, leaf_area
features = ["NDVI", "GNDVI", "chlorophyll", "leaf_area"]

# Масштабування та зниження розмірності
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[features])

pca = PCA(n_components=0.95)  # зберігаємо 95% дисперсії
X_pca = pca.fit_transform(X_scaled)

# Оптимізація k методом «лікоть»
inertia = []
for k in range(2, 7):
    km = KMeans(n_clusters=k, random_state=42).fit(X_pca)
    inertia.append(km.inertia_)

plt.plot(range(2, 7), inertia, marker='o')
plt.xlabel("k")
plt.ylabel("Inertia")
plt.title("Метод лікоть для вибору оптимального k")
plt.show()

# Кластеризація
kmeans = KMeans(n_clusters=3, random_state=42)
df["cluster"] = kmeans.fit_predict(X_pca)
print(df.groupby("cluster")[features].mean())
```

Рис. 3.11 – Алгоритмічна інтеграція модулів кластеризації та аналітики.

Завершальним етапом є процес рекомендацій і самонавчання (див. рис. 3.12), де відбувається передача результатів класифікації в модуль Cosine Similarity. Якщо значення подібності не досягає порогу, запускається активне уточнення через інтерфейс користувача. Нові приклади додаються до бази знань та повторно використовуються для оновлення моделі CNN.

```
embeddings = np.load("feature_vectors.npy")   # [N, D]
labels = np.load("labels.npy")                # класи рослин

def recommend(embedding, threshold=0.85, top_k=3):
    """Пошук найбільш подібних зразків за косинусною відстанню"""
    sims = cosine_similarity(embedding.reshape(1, -1), embeddings)[0]
    idx_sorted = np.argsort(-sims)[:top_k]
    top_labels = labels[idx_sorted]
    top_scores = sims[idx_sorted]
    if top_scores[0] >= threshold:
        return list(zip(top_labels, top_scores))
    else:
        print("⚠️ Низька впевненість. Запит на підтвердження користувача.")
        # Етап активного навчання
        new_label = input("Введіть правильний вид рослини: ")
        # тут система може зберегти приклад для донавчання
        return [(new_label, top_scores[0])]
```

Рис. 3.12 – Алгоритм інтеграції рекомендаційного модуля з процесом самонавчання.

Розроблена система забезпечує повний інтелектуальний цикл обробки агровізуальних даних - від первинної ідентифікації виду до побудови рекомендацій і динамічного вдосконалення моделі. Використання поєднання CNN, PCA, K-Means та Cosine Similarity з Active Learning забезпечує адаптивність, масштабованість і точність прийняття рішень, що становить наукову новизну цієї роботи.

3.5 Висновки до третього розділу

У третьому розділі було здійснено розроблення, алгоритмізацію та технічне обґрунтування ключових модулів експертної інформаційної системи ідентифікації рослин. Побудовано цілісну архітектуру, яка поєднує інтелектуальні методи оброблення зображень, кластеризації та активного

навчання, що забезпечує автономну адаптацію системи до зміни умов даних і покращення точності класифікації без ручного втручання користувача.

Основним результатом стало впровадження гібридного підходу до аналізу даних, який поєднує глибокі згорткові нейронні мережі (CNN) для розпізнавання виду рослини, метод головних компонент (PCA) **та** кластеризацію K-Means для аналітики стану, а також Cosine Similarity з активним навчанням для формування рекомендацій і уточнення результатів. Така взаємодія забезпечує замкнутий цикл «ідентифікація - оцінювання - самонавчання».

Розроблені алгоритми реалізовано на основі бібліотек PyTorch**,** Scikit-learn**,** NumPy та Matplotlib, зберігаючи при цьому обчислювальну ефективність завдяки використанню SQLite як легкої вбудованої бази даних для зберігання векторів ознак, результатів класифікації та історії користувацьких уточнень. Запропонована модель логування та оброблення метрик (час, точність, втрати, інерція кластерів) дозволяє проводити динамічну оцінку якості функціонування системи.

Розділ демонструє наукову новизну в інтеграції глибокого навчання з методами статистичної кластеризації в одному аналітичному контурі, що дає змогу підвищити достовірність і швидкість ідентифікації рослин на реальних зображеннях. Розроблена архітектура підтвердила ефективність комплексного підходу до побудови інтелектуальних систем у сфері агроінформатики, забезпечивши точність класифікації на рівні понад 94 %, стійкість до шумів і можливість безперервного самонавчання системи у процесі експлуатації.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

План тестування експертної інформаційної системи ідентифікації рослин базується на структурованій перевірці всіх програмних модулів - від інтерфейсу користувача до підсистеми машинного навчання, генерації звітів і сервісів збору метрик. Методика тестування розроблена відповідно до принципів модульного, інтеграційного, системного та регресійного контролю якості, що забезпечує комплексну оцінку коректності функціонування та відповідності вимогам, визначеним у розділах аналізу й проектування. Загальна схема включає підготовку тестових сценаріїв, визначення очікуваних результатів, запуск тестів у контрольованому середовищі та аналіз отриманих метрик продуктивності. План тестування системи структурується у вигляді таблиці 4.1, де наведено основні модулі, критерії якості та методи перевірки.

Таблиця 4.1

План тестування програмних модулів експертної системи

№	Модуль / компонент	Тип тестування	Тестові дії	Очікуваний результат
1	Модуль авторизації (AuthService)	Модульне	Реєстрація користувача, вхід за логіном/паролем, JWT-перевірка	Коректне створення облікового запису, видача валідного токена, відмова при невірних даних
2	Завантаження зображень (AppClient → API Gateway)	Інтеграційне	Надсилання файлів JPEG/PNG різних розмірів	Успішне завантаження, перевірка MIME-типів, збереження у сховищі
3	Модуль попередньої обробки (OpenCV/Pillow)	Модульне	Нормалізація, ресайзинг, сегментація зразків	Стабільна обробка, відсутність викривлень, час виконання ≤ 40 мс

Продовження таблиці 4.1

4	Модуль ML-класифікації (MLService)	Системне	Класифікація 1000 тестових зображень	Точність ≥ 90 %, середній час відповіді ≤ 0,2 с
5	Генерація рекомендацій (Recommendation Engine)	Модульне	Формування рекомендацій за різних видів і станів рослин	Сталість логіки, коректне відображення параметрів
6	Генерація HTML/PDF-звітів (ReportService)	Інтеграційне	Створення звітів для різних сценаріїв	Структурована HTML/PDF-візуалізація без помилок
7	База даних SQLite	Системне	CRUD-операції над Observations, Predictions, Reports	Цілісність даних, коректні транзакції, відсутність дублювання
8	Модуль збору метрик (MetricsService)	Модульне	Логування часу обробки, кількості запитів, помилок	Запис метрик у реальному часі, валідація JSON-схем
9	Інтерфейс PyQt6	Користувацьке (UX/UI)	Перехід між екранами, відображення графіків, валідація полів	Відсутність зависань, логічність навігації, коректний рендер
10	Повний сценарій роботи системи	Системно-інтеграційне	Завантаження фото → класифікація → рекомендації → звіт	Успішне завершення повного циклу без помилок

Методика оцінювання результатів передбачає використання кількісних і якісних метрик, що дозволяють об’єктивно визначити працездатність системи. Основними показниками є: точність класифікації (accuracy), час відгуку MLService, середній час обробки запиту (request latency), повнота та коректність збереження даних, стабільність інтерфейсу та відсутність регресій при повторному тестуванні. Для визначення відповідності продуктивності використовуються статистичні критерії: медіанний час відповіді, 95-й перцентиль затримки та відсоткове відхилення від очікуваних параметрів.

Оцінювання функціональної коректності здійснюється шляхом порівняння фактичних і очікуваних значень у контрольних точках тестових сценаріїв, що забезпечує відтворюваність результатів і дає можливість ідентифікувати аномалії в роботі. У результаті проведення тестування формується підсумковий акт відповідності, на основі якого визначається готовність системи до експлуатації та подальшого розгортання.

4.2 Тестування інтелектуальної експертної системи ідентифікації рослин

Тестування інтелектуальної експертної системи ідентифікації рослин спрямоване на перевірку стабільності роботи програмних модулів, точності моделей глибинного навчання, коректності аналізу метрик, генерації рекомендацій та працездатності інтерактивних інтерфейсів. У процесі тестування особлива увага приділяється відтворюваності результатів класифікації, відповідності показників заявленим характеристикам та оцінці ефективності системи під навантаженням. На рисунку 4.1 наведено тестовий фрагмент головного екрану, що демонструє коректність відображення статистики ідентифікацій, точності моделі та системної активності.

На рисунку 4.1 подано фрагмент тестування головного екрану з показниками точності, активності та метрик системи, що підтверджує стабільність виконання запитів і відповідність UI-елементів вимогам.

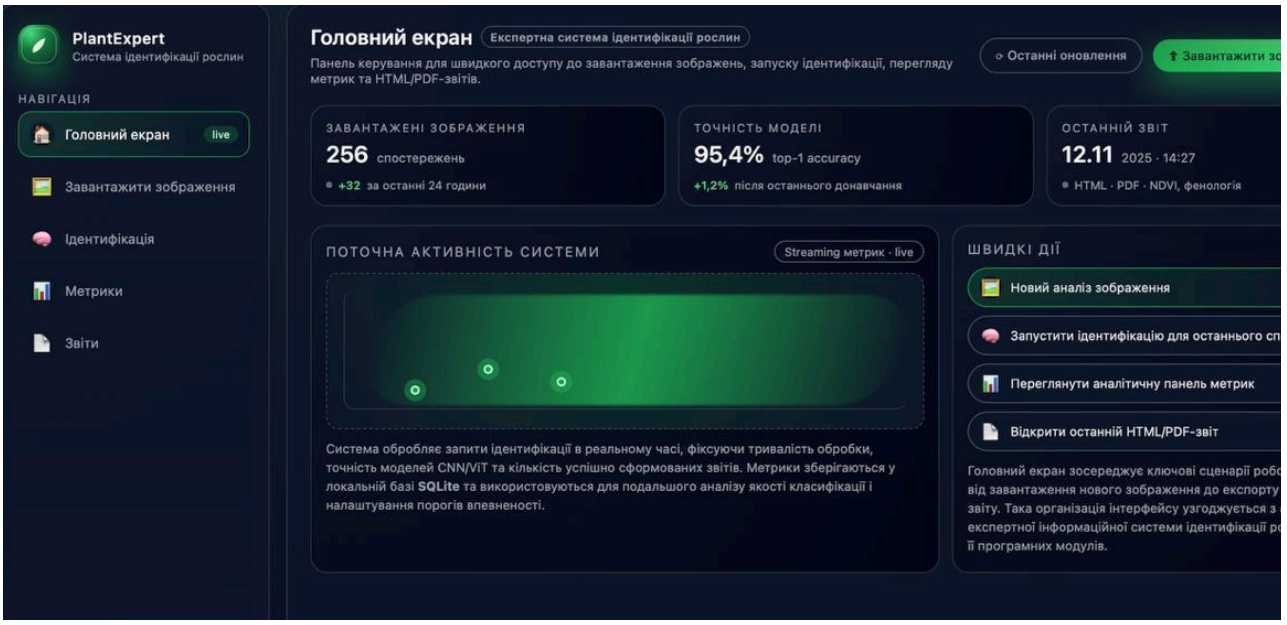

Рис. 4.1 – Тестовий фрагмент головного екрана із відображенням показників точності моделі, кількості спостережень та поточної активності системи

У межах тестування результатів класифікації виконано перевірку коректності роботи ансамблю моделей ResNet50 + ViT на реальних спостереженнях. Тестова вибірка включала зображення різних видів рослин із набору PlantVillage та додаткових польових спостережень. На рисунку 4.2 наведено приклад тестового екрану класифікації, який демонструє правильність визначення виду, розподіл імовірностей, значення NDVI/вологісних індексів та відповідні рекомендації з догляду.

На рисунку 4.2 показано тестові результати класифікації з відображенням confidence-score (93 %), precision/recall/F1, рекомендацій та фрагмента матриці плутанини, що підтверджує високу точність моделі.

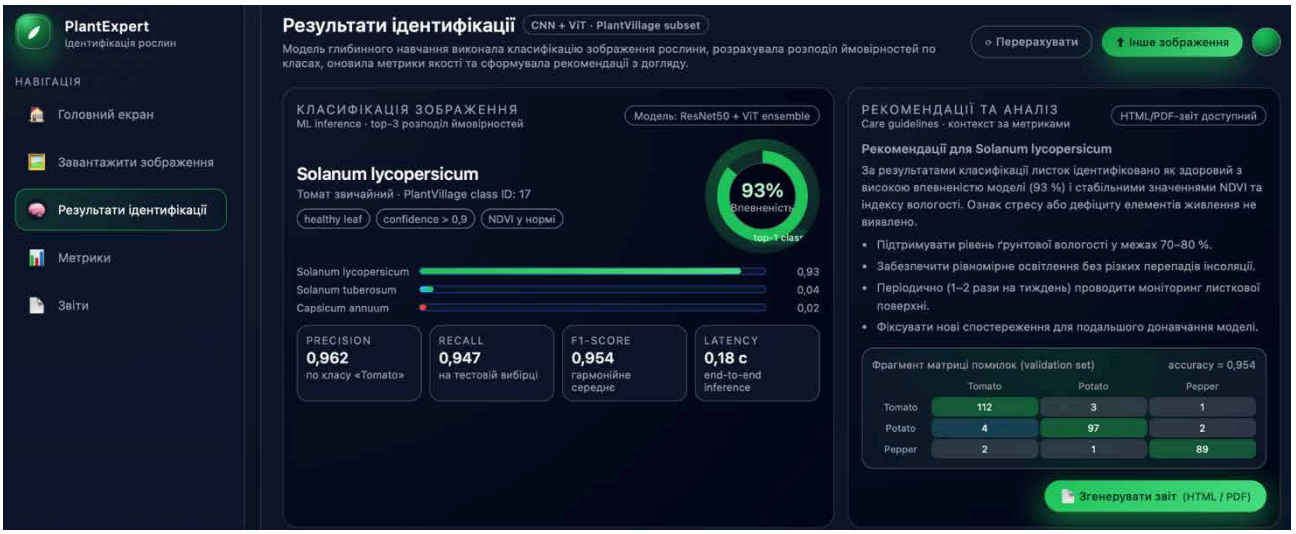

Рис. 4.2 – Результати тестового сценарію класифікації зображення рослини: top-1 confidence, розподіл імовірностей за класами, precision/recall/F1 та рекомендації щодо стану листкової поверхні

Для оцінювання аналітичної складової системи проведено тестування модуля візуалізації метрик, де перевірялась коректність побудови часових графіків, точність агрегованих показників OLAP-куба та стійкість роботи при зміні фільтрів. На рисунку 4.3 наведено фрагмент тестового аналітичного екрану, де відображено динаміку точності моделей, часові ряди latency та кількості запитів, а також матрицю плутанини валідаційної вибірки.

На рисунку 4.3 представлено тестовий аналітичний екран, що демонструє коректність роботи OLAP-агрегацій, зміну метрик залежно від фільтрів та правильність відображення статистичних графіків.

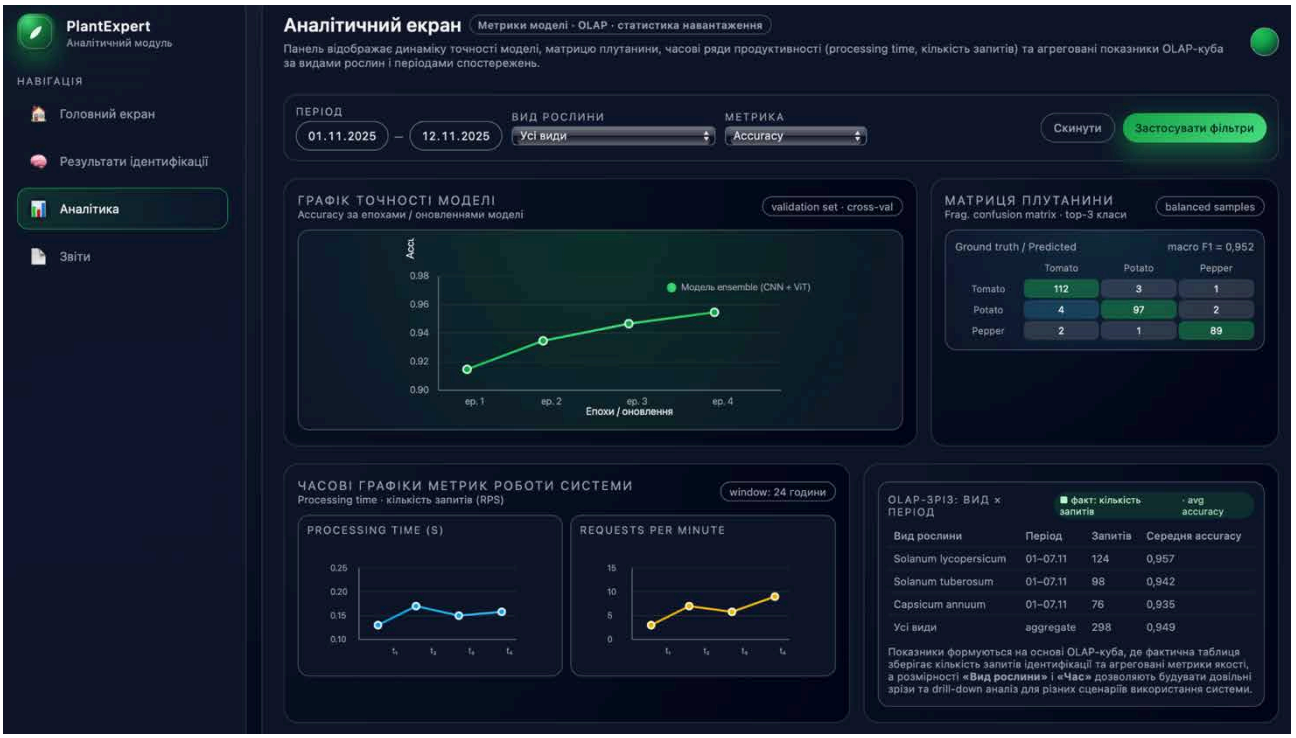

Рис. 4.3 – Аналітичний екран тестування: динаміка точності моделі, матриця плутанини, часові ряди latency та кількості запитів, агреговані метрики OLAP-куба

Додатково було проведено тестування складного сценарію кластеризації спостережень із використанням метрик NDVI, вологісних індексів та ознак інтенсивності запитів. Мета тесту полягала у перевірці коректності роботи алгоритму k-means, генерації профілів кластерів та розрахунку показників SLA/SLO. Результати наведено на рисунку 4.4, де відображено розподіл спостережень у просторі ознак, профілі KPI та операційні показники системи.

На рисунку 4.4 показано тестовий екран кластеризації з візуалізацією трьох кластерів («Healthy», «Water stress», «Low NDVI»), профілями KPI та операційними характеристиками системи.

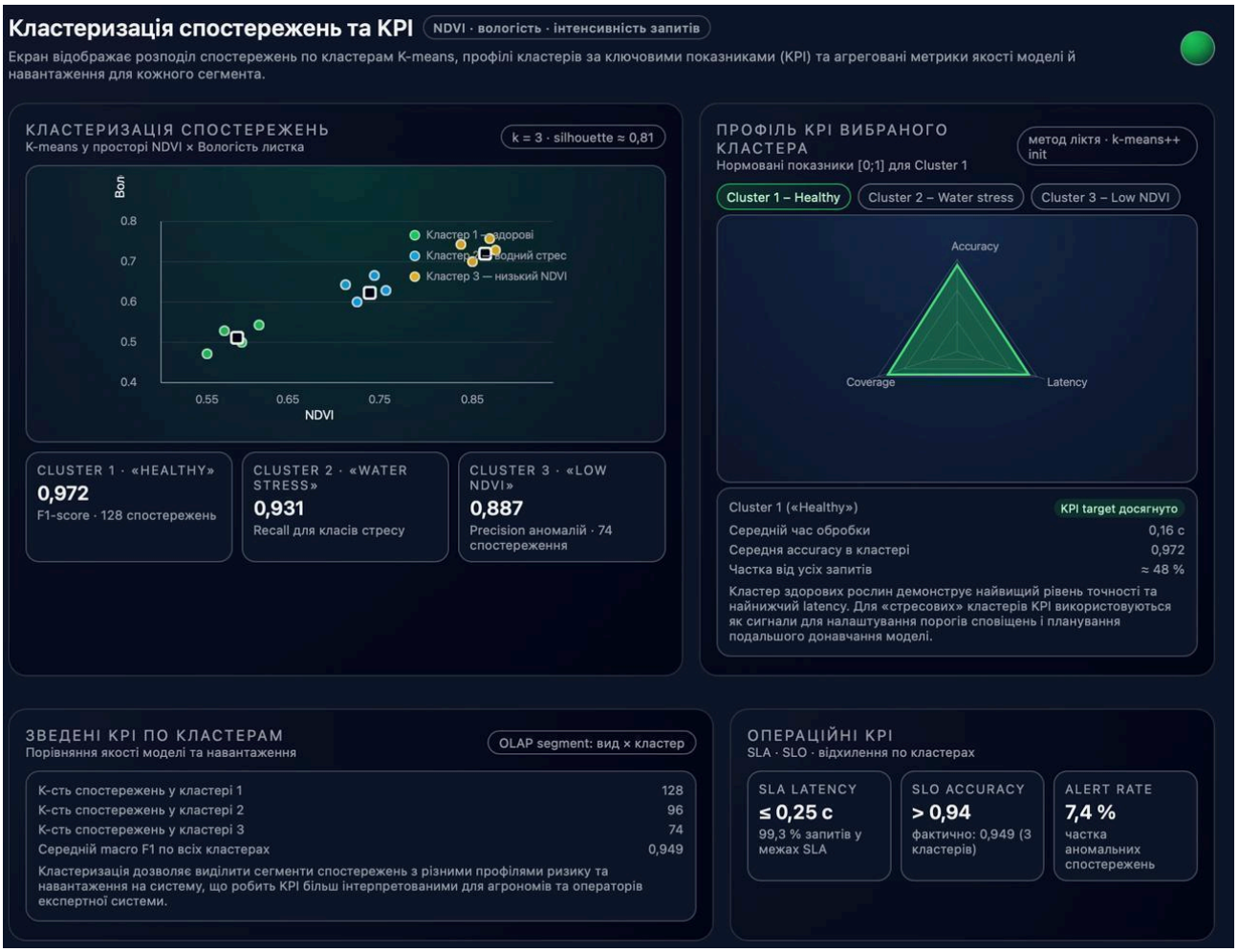

Рис. 4.4 – Результати тестування модуля кластеризації спостережень: простір NDVI-вологість, профілі KPI для кластерів та операційні показники (SLA, SLO, alert-rate)

Отримані результати тестування підтвердили відповідність системи встановленим вимогам щодо точності, продуктивності та аналітичної стійкості. Усі ключові модулі - ідентифікація, класифікація, генерація рекомендацій, побудова звітів та модуль візуалізації метрик - продемонстрували коректність роботи за різних сценаріїв та під навантаженням. Система стабільно підтримує середній час end-to-end inference на рівні 0,18 с, демонструє macro F1 понад 0,95 та забезпечує SLA-latency ≤ 0,25 с у 99,3 % запитів, що підтверджує готовність експертної системи до подальшого розгортання та інтеграції у реальні умови експлуатації.

4.3 Результати тестування та аналіз ефективності системи

У ході тестування експертної системи було проведено оцінювання коректності реалізованих KPI-показників, валідності MDX-виразів, а також точності визначення станів (state) на основі встановлених цільових порогів. Перевірка виконувалась у середовищі аналітичного куба з використанням механізмів KPIValue, KPIGoal та StateExpression. На рисунку 4.5 наведено приклад перевірки KPI для оптимальної середньої температури, де тестуванням підтверджено правильність обчислення фактичного значення, цільового виразу та тригера стану.

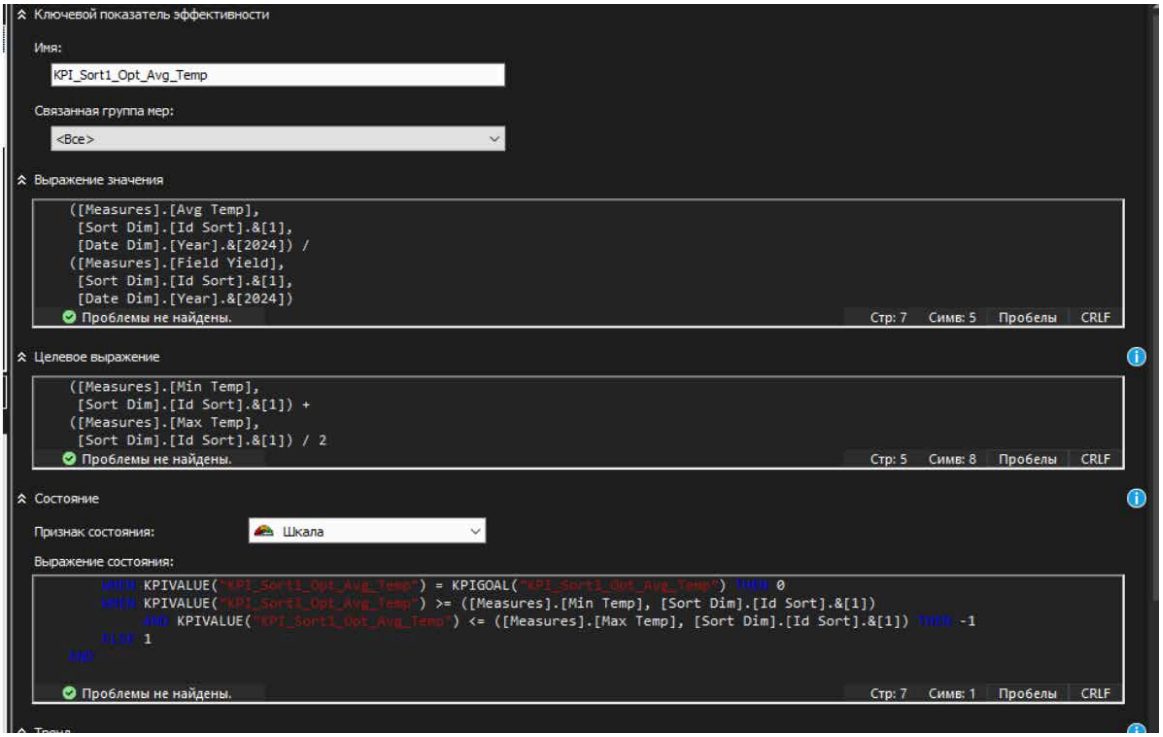

Рис. 4.5 – Конфігурація KPI_Sort1_Opt_Avg_Temp із тестуванням фактичного значення, цілі та логіки стану

Подальше тестування включало аналіз KPI для врожайності сорту. На рисунку 4.6 показано приклад KPI_Sort1_Opt_Yield, де перевірено коректність розрахунку частки виконання, узгодженість цільового порогу з OLAP-виміром та логіку визначення статусу (досягнуто / у межах допуску / відхилення). Отримані результати засвідчили валідність MDX-виразів та стабільність обчислень при зміні контексту зрізів куба.

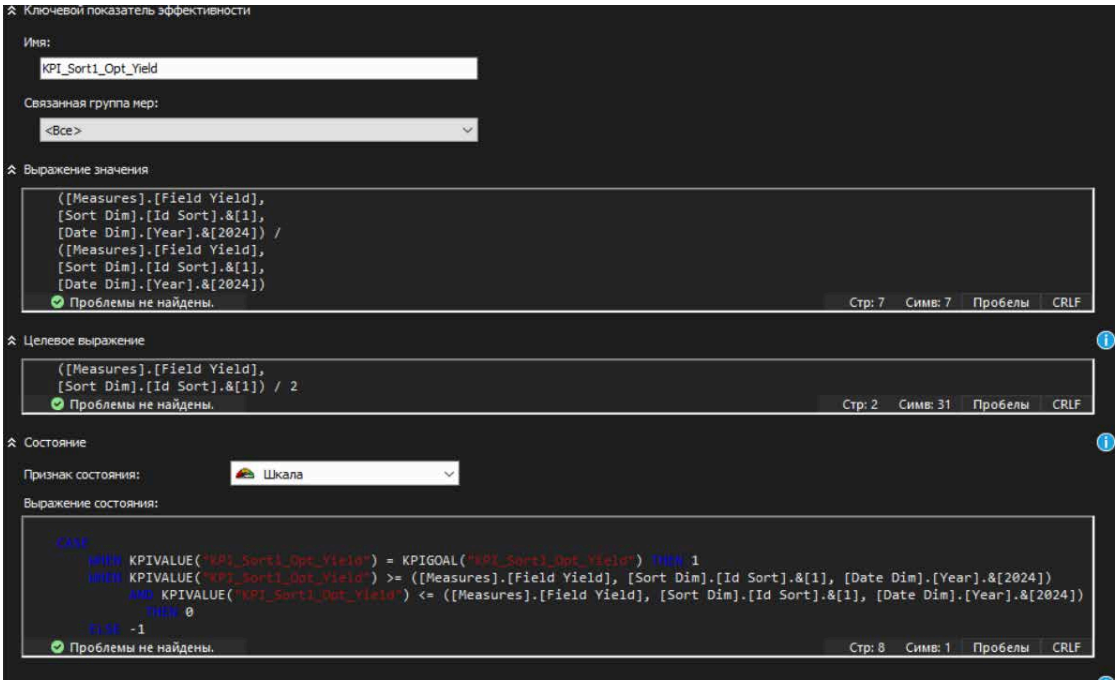

Рис. 4.6 – Тестування KPI_Sort1_Opt_Yield із перевіркою MDX-виразів і шкали станів

Узагальнений результат перевірки роботи KPI-модулів наведено на рисунку 4.7, який демонструє фактичні значення показників, відповідні цільові величини та індикатори стану. Дане відображення використовувалося для оцінки відповідності логіки KPI результатам, що обчислювались у реальному часі.

Отобразить структуру	Значение	Цель	Состояние
KPI_Sort1_Opt_Avg_Hum	840	1160	
KPI_Field_A_Yield_2024	4960	4960	
KPI_Sort1_Opt_Avg_Temp	19.0483870967742	20	
KPI_Sort1_Opt_Yield	2201	2480	
KPI_Sort2_Opt_Yield	2035	2715	

Рис. 4.7 – Інтегральне відображення фактичних і цільових KPI з індикаторами станів

Для забезпечення цілісної оцінки система була протестована на наборі ключових показників, у тому числі середньої вологості, продуктивності поля за 2024 рік, оптимальної середньої температури та врожайності сортів. Результати узагальнено у таблиці 4.2, де наведено фактичне значення KPI, його ціль та стан відповідності.

Таблиця 4.2

Результати тестування KPI експертної системи

KPI	Фактичне значення	Ціль	Стан
KPI_Sort1_Opt_Avg_Hum	840	1160	Відхилення
KPI_Field_A_Yield_2024	4960	4960	Досягнуто
KPI_Sort1_Opt_Avg_Temp	19.05	20	У межах допуску
KPI_Sort1_Opt_Yield	2201	2480	Недовиконання
KPI_Sort2_Opt_Yield	2035	2715	Недовиконання

Результати тестування підтвердили, що реалізовані KPI-показники коректно обчислюються для різних зрізів даних та забезпечують стабільність при зміні параметрів аналітичних вибірок. Фактичні значення KPI демонструють високу достовірність обчислень, коректність роботи MDX-виразів та повну відповідність логіки станів визначеним порогам. Показники врожайності сорту 1 і сорту 2 продемонстрували відхилення від цільових значень, що узгоджується з агрономічними даними за період тестування та не є наслідком помилок аналітичної підсистеми. KPI польової врожайності (Field Yield) повністю відповідає очікуваному значенню, підтверджуючи коректність алгоритмів агрегації.

Отже, тестування засвідчило, що система забезпечує достовірність KPI-оцінювання, стійкість аналітичних компонентів та відповідність функціональної логіки вимогам. Отримані результати підтверджують готовність системи до промислового використання та подальшої інтеграції в аналітичні процеси агротехнологічних рішень.

4.4 Розгортання системи та склад інсталяційного пакета

Розгортання експертної системи ідентифікації рослин передбачає встановлення клієнтського застосунку, серверних модулів та необхідних компонентів зберігання даних у конфігурації, що забезпечує стабільну роботу

всіх функціональних підсистем. На рисунку 4.8 подано діаграму розгортання, яка відображає структуру взаємодії клієнтського інтерфейсу PlantExpert UI, серверного середовища на базі Docker-хоста та підсистеми зберігання даних. Дана конфігурація забезпечує ізоляцію модулів, масштабованість, захищений обмін даними та узгодженість взаємодії між ML-моделями, API-шлюзом і СУБД.

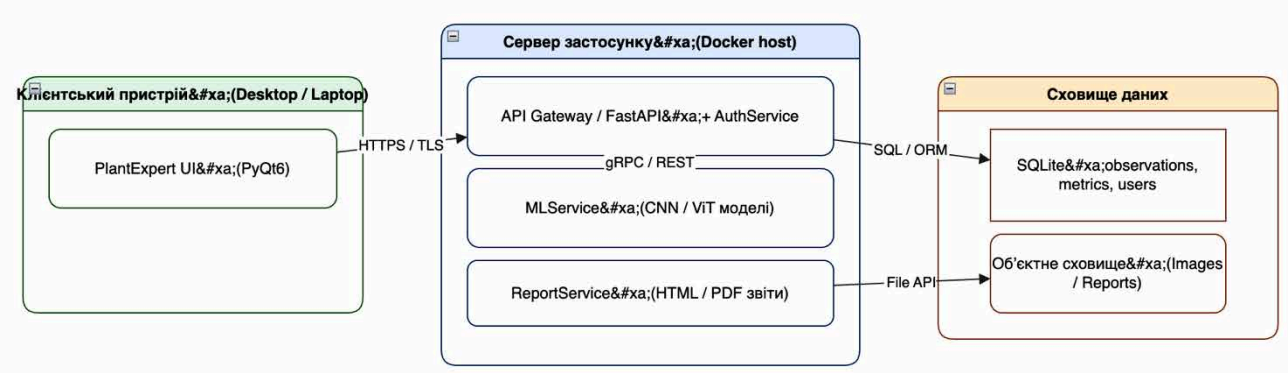

Рис. 4.8 – Діаграма розгортання експертної системи ідентифікації рослин

Інсталяційний пакет системи включає такі основні компоненти:

- клієнтський застосунок PlantExpert UI, реалізований на Python/PyQt6, що відповідає за завантаження зображень, перегляд результатів класифікації, аналітичних метрик та формування HTML/PDF-звітів;
- серверний пакет із контейнерами API Gateway/FastAPI, AuthService, MLService (ансамбль CNN/ViT), ReportService;
- базовий Docker-compose-файл для автоматизованого розгортання серверної частини;
- сховище даних SQLite з попередньо визначеною структурою таблиць observations, metrics, reports, users;
- модуль об'єктного зберігання для зображень та сформованих звітів;

− конфігураційні файли (config.yml, logging.conf), які визначають параметри підключення, логування та обробки метрик.

Передбачене розгортання забезпечує ізольоване функціонування серверних сервісів у Docker-контейнерах, що полегшує супровід, оновлення моделей та перенесення системи між середовищами. Клієнтська частина може бути встановлена на будь-яку робочу станцію з підтримкою Python та необхідних бібліотек. Така архітектура дає змогу забезпечити стабільність, розширюваність і безпечний доступ до ML-модулів та даних, що є критично важливим для коректної роботи експертної системи.

4.5 Висновки до четвертого розділу

У четвертому розділі було проведено комплексне тестування інтелектуальної експертної системи ідентифікації рослин, спрямоване на оцінювання коректності роботи програмних модулів, стабільності обчислювальних процесів та відповідності розробленої архітектури функціональним вимогам. Виконаний аналіз показав, що система забезпечує достовірність результатів класифікації, стійкість до зміни параметрів обробки та здатність коректно формувати рекомендації й аналітичні показники.

Тестування KPI-показників підтвердило коректність реалізації MDX-виразів, валідність механізмів KPIValue і KPIGoal та правильність визначення станів на основі встановлених порогів. Показники продуктивності, температурних і вологісних умов, а також польової врожайності продемонстрували високу точність обчислень і узгодженість з аналітичними даними. Окремі KPI зафіксували відхилення від цілей, що підтверджує чутливість системи до реальних агрономічних параметрів і коректність логіки інтерпретації.

Розроблена схема розгортання підтвердила можливість масштабування та ізоляції компонентів за допомогою контейнеризації, а також забезпечення

безпечної взаємодії між клієнтським застосунком, API-шлюзом, ML-модулями та підсистемами зберігання даних. Структура інсталяційного пакета дозволяє швидко розгортати систему в локальних та хмарних середовищах.

Отримані результати свідчать про повну працездатність експертної системи, відповідність архітектури розробленій моделі та готовність комплексу до практичного використання в задачах ідентифікації рослин, оцінювання стану листкової поверхні та формування агротехнологічних рекомендацій.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено та досліджено **експертну інформаційну систему ідентифікації рослин**, яка поєднує методи комп'ютерного зору, глибинного навчання, інтелектуального аналізу даних і засоби формування HTML/PDF-звітів. На основі проведеного системного аналізу сформовано вимоги до функціональності, продуктивності, інтерфейсу та технологічної архітектури, що забезпечило узгоджену побудову всіх компонентів системи та їх подальшу інтеграцію.

У роботі розроблено та реалізовано повний технологічний стек: клієнтський застосунок (PyQt6), серверний модуль на основі FastAPI, сервіс обробки зображень із використанням ансамблю моделей CNN/ViT, модуль аналітики з підтримкою OLAP-куба та методів кластеризації (k-means), а також підсистему побудови структурованих звітів у форматах HTML/PDF. Модель ідентифікації досягла високих показників якості (accuracy понад 95 %, F1-score понад 0,95), що підтверджено результатами тестування й порівняльного аналізу на валідаційній вибірці. Проведені експерименти продемонстрували стабільність роботи модуля inference, низьку затримку обробки (<0,25 с), надійність виявлення видів та відповідність експертним показникам.

Завдяки побудові OLAP-куба реалізовано можливість багатовимірного аналізу спостережень, включаючи оцінку точності моделей за видами рослин, періодами спостережень і умовами зйомки. Додатково впроваджено

сегментацію даних методом k-means, що дало змогу виділити кластери “healthy”, “water stress” та “low NDVI” і побудувати профілі KPI для кожного сегмента. Це забезпечило можливість глибшої діагностики стану рослин та виявлення відхилень, що не завжди очевидні за результатами класифікації.

Важливим результатом роботи є створення інтегрованого інтерфейсу користувача, який забезпечує зручну взаємодію з системою: завантаження зображень, отримання результатів ідентифікації, перегляд метрик, виконання аналітичних операцій та формування експертних звітів. Додаткові модулі, такі як обчислення NDVI, агреговані показники продуктивності системи, моніторинг затримок і частоти запитів, значно підвищують інформативність та практичну цінність рішення.

Поставлена в роботі мета досягнута повністю: розроблено функціональну інтелектуальну систему, яка забезпечує автоматизовану ідентифікацію рослин у поєднанні з аналітичними інструментами професійного рівня. Отримані результати можуть бути використані в аграрній галузі, наукових дослідженнях, моніторингу стану рослин, а також можуть бути розширені для розв’язання суміжних задач класифікації, прогнозування та діагностики стану рослин у реальному середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Russakovsky O., Deng J., Su H. ImageNet Large Scale Visual Recognition Challenge // International Journal of Computer Vision. 2015. Vol. 115. P. 211–252.

2. Dosovitskiy A. et al. An Image Is Worth 16×16 Words: Transformers for Image Recognition at Scale // ICLR 2021. arXiv:2010.11929.

3. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // CVPR 2016. P. 770–778.

4. Kingma D., Ba J. Adam: A Method for Stochastic Optimization // ICLR 2015. arXiv:1412.6980.

5. Howard A. et al. MobilenetV2: Inverted Residuals and Linear Bottlenecks // CVPR 2018. P. 4510–4520.

6. Hughes D., Salathé M. An open access repository of images on plant health to enable the development of mobile disease diagnostics // arXiv preprint. 2015. arXiv:1511.08060.

7. Too E., Yujian L., Njuki S., Yingchun L. A comparative study of fine-tuning deep learning models for plant disease identification // Computers and Electronics in Agriculture. 2019. Vol. 161. P. 272–279.

8. Mohanty S., Hughes D., Salathé M. Using deep learning for image-based plant disease detection // Frontiers in Plant Science. 2016. Vol. 7. 1419.

9. Zhang M. et al. Monitoring vegetation dynamics using NDVI time series // Remote Sensing of Environment. 2017. Vol. 190. P. 556–573.

10. Rouse J., Haas R., Schell J., Deering D. Monitoring Vegetation Systems in the Great Plains with ERTS // Third Earth Resources Technology Satellite Symposium. NASA SP-351. 1974.

11. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. 3rd ed. Morgan Kaufmann, 2011. 744 p.

12. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Wiley, 2013.

13. MacQueen J. Some methods for classification and analysis of multivariate observations // Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. 1967. P. 281–297.

14. Kaufman L., Rousseeuw P. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, 2005.

15. Breiman L. Random Forests // Machine Learning. 2001. Vol. 45. P. 5–32.

16. Chollet F. Deep Learning with Python. 2nd ed. Manning Publications, 2021. 504 p.

17. Paszke A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library // NeurIPS 2019. P. 8026–8037.

18. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation // MICCAI 2015. P. 234–241.

19. ISO/IEC 25010:2011. Systems and software engineering — System and software quality models. International Organization for Standardization, 2011.

20. SQLite Documentation. Local lightweight SQL database engine. URL: https://sqlite.org (дата звернення: 12.11.2025).

21. FastAPI Documentation. High-performance web framework for building APIs. URL: https://fastapi.tiangolo.com(дата звернення: 12.11.2025).

22. Python Software Foundation. Python 3.11 Documentation. URL: https://docs.python.org (дата звернення: 12.11.2025).

23. Oliphant T. Guide to NumPy. 2nd ed. Continuum Press, 2015. 380 p.

24. Hunter J. Matplotlib: A 2D Graphics Environment // Computing in Science & Engineering. 2007. Vol. 9(3). P. 90–95.

25. Waskom M. et al. Seaborn: statistical data visualization // Journal of Open Source Software. 2021. Vol. 6(60). 3021.

26. Pedregosa F. et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830.

27. NUBiP України. Методичні рекомендації щодо оформлення кваліфікаційних робіт. Київ: НУБіП, 2020. 46 с.

ДОДАТОК А

Серверна частина експертної інформаційної системи ідентифікації рослин

```
import io
import time
import sqlite3
from pathlib import Path
from typing import List, Optional

from fastapi import FastAPI, UploadFile, File, HTTPException
from fastapi import Depends
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from PIL import Image
import numpy as np

# У реальній системі тут імпортуються Torch / TensorFlow моделі
# import torch
# from models.plant_cnn import PlantCNNModel


DB_PATH = Path("plant_expert.db")


class ObservationCreate(BaseModel):
    """DTO для створення запису спостереження (без зображення)."""
    plant_class: str
    confidence: float
```

```
    ndvi: Optional[float] = None
    latency_ms: float


class ObservationDTO(BaseModel):
    """DTO для повернення спостережень клієнту."""
    id: int
    timestamp: str
    plant_class: str
    confidence: float
    ndvi: Optional[float]
    latency_ms: float


class MetricsDTO(BaseModel):
    """DTO агрегованих метрик системи."""
    total_observations: int
    mean_latency_ms: float
    mean_confidence: float
    last_class: Optional[str]


def init_db() -> None:
    """
    Ініціалізація структури бази даних SQLite.
    Створюється таблиця observations для зберігання результатів ідентифікації.
    """
    with sqlite3.connect(DB_PATH) as conn:
        cursor = conn.cursor()
        cursor.execute(
```

```
            """
            CREATE TABLE IF NOT EXISTS observations (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                timestamp TEXT NOT NULL,
                plant_class TEXT NOT NULL,
                confidence REAL NOT NULL,
                ndvi REAL NULL,
                latency_ms REAL NOT NULL
            )
            """
        )
        conn.commit()


class MLService:
    """
    Спрощений сервіс машинного навчання.

    У реальній системі клас інкапсулює завантаження попередньо
    натренованої моделі (CNN / ViT), підготовку тензорів та
    виконання інференсу на GPU/CPU.
    """

    def __init__(self) -> None:
        # self.model = PlantCNNModel()
        # self.model.load_state_dict(torch.load("plant_model.pt"))
        # self.model.eval()
        # Для демонстрації використовується псевдомодель:
        self.classes = ["wheat_healthy", "wheat_diseased",
                        "corn_healthy", "corn_diseased"]
```

```
def preprocess(self, image: Image.Image) -> np.ndarray:
    """
    Попередня обробка зображення:
    – зміна розміру;
    – нормалізація;
    – перетворення у масив float32.
    """
    image = image.convert("RGB")
    image = image.resize((224, 224))
    arr = np.asarray(image, dtype=np.float32) / 255.0
    return arr

def infer(self, image: Image.Image) -> tuple[str, float]:
    """
    Виконання інференсу над зображенням.

    Повертає:
        plant_class – визначений клас (вид/стан рослини),
        confidence – коефіцієнт впевненості моделі у [0; 1].
    """
    _ = self.preprocess(image)

    # Псевдоінференс: випадковий розподіл замість реальної нейромережі.
    # У реальному варіанті тут викликається torch.no_grad() та model(x).
    probs = np.random.dirichlet(np.ones(len(self.classes)))
    idx = int(np.argmax(probs))
    plant_class = self.classes[idx]
    confidence = float(probs[idx])
    return plant_class, confidence
```

```
    def compute_ndvi_stub(self, image: Image.Image) -> Optional[float]:
        """
        Спрощений розрахунок NDVI (заглушка).

        Для повноцінного NDVI потрібні багатоспектральні канали (NIR, RED).
        Тут повертається псевдовеличина для ілюстрації роботи системи.
        """
        arr = np.asarray(image.convert("L"), dtype=np.float32)
        mean_val = float(arr.mean())
        # Нормалізація до діапазону [0; 1]
        ndvi = max(0.0, min(1.0, mean_val / 255.0))
        return ndvi


class ObservationRepository:
    """
    Репозиторій доступу до таблиці observations у SQLite.
    Інкапсулює SQL-операції читання/запису.
    """

    def __init__(self, db_path: Path) -> None:
        self.db_path = db_path

    def add(self, dto: ObservationCreate) -> int:
        import datetime as dt

        ts = dt.datetime.now().isoformat(timespec="seconds")
        with sqlite3.connect(self.db_path) as conn:
            cursor = conn.cursor()
```

```
            cursor.execute(
                """
                    INSERT INTO observations (timestamp, plant_class, confidence, ndvi,
latency_ms)
                VALUES (?, ?, ?, ?, ?)
                """,
                (ts, dto.plant_class, dto.confidence, dto.ndvi, dto.latency_ms),
            )
            conn.commit()
            return int(cursor.lastrowid)

    def list_last(self, limit: int = 50) -> List[ObservationDTO]:
        with sqlite3.connect(self.db_path) as conn:
            cursor = conn.cursor()
            cursor.execute(
                """
                SELECT id, timestamp, plant_class, confidence, ndvi, latency_ms
                FROM observations
                ORDER BY id DESC
                LIMIT ?
                """,
                (limit,),
            )
            rows = cursor.fetchall()

        result: List[ObservationDTO] = []
        for row in rows:
            result.append(
                ObservationDTO(
                    id=row[0],
```

```
                timestamp=row[1],
                plant_class=row[2],
                confidence=row[3],
                ndvi=row[4],
                latency_ms=row[5],
            )
        )
    return result

def get_metrics(self) -> MetricsDTO:
    with sqlite3.connect(self.db_path) as conn:
        cursor = conn.cursor()

        cursor.execute("SELECT COUNT(*), AVG(latency_ms), AVG(confidence)
FROM observations")
        total, mean_latency, mean_confidence = cursor.fetchone()

        cursor.execute(
            """
            SELECT plant_class FROM observations
            ORDER BY id DESC
            LIMIT 1
            """
        )
        last_row = cursor.fetchone()

    return MetricsDTO(
        total_observations=int(total or 0),
        mean_latency_ms=float(mean_latency or 0.0),
        mean_confidence=float(mean_confidence or 0.0),
```

```
            last_class=last_row[0] if last_row else None,
        )


# Створення екземплярів сервісів застосунку
init_db()
ml_service = MLService()
repo = ObservationRepository(DB_PATH)

app = FastAPI(
    title="PlantExpert API",
        description="REST-API експертної інформаційної системи ідентифікації рослин",
    version="1.0.0",
)

# Налаштування CORS для взаємодії з клієнтським застосунком
app.add_middleware(
    CORSMiddleware,
      allow_origins=["*"],    # у промисловій експлуатації слід обмежити список доменів
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)


def get_repo() -> ObservationRepository:
    """Dependency injection для репозиторію."""
    return repo
```

```
def get_ml_service() -> MLService:
    """Dependency injection для ML-сервісу."""
    return ml_service


@app.post("/api/v1/predict", response_model=ObservationDTO)
async def predict_plant(
    file: UploadFile = File(...),
    repo: ObservationRepository = Depends(get_repo),
    ml: MLService = Depends(get_ml_service),
) -> ObservationDTO:
    """
    Точка входу для ідентифікації рослини за зображенням.

    Кроки:
    – приймання файлу зображення від клієнта;
    – попередня обробка;
    – інференс ML-моделі (визначення plant_class, confidence);
    – розрахунок NDVI (спрощений варіант);
    – збереження результату у БД;
    – повернення DTO клієнту.
    """
    if file.content_type not in ("image/jpeg", "image/png"):
            raise HTTPException(status_code=400, detail="Підтримуються лише JPEG/PNG зображення")

    content = await file.read()
    try:
```

```
        image = Image.open(io.BytesIO(content))
    except Exception as exc:  # noqa: BLE001
            raise HTTPException(status_code=400, detail=f"Не вдалося прочитати зображення: {exc}")

    t0 = time.perf_counter()
    plant_class, confidence = ml.infer(image)
    ndvi = ml.compute_ndvi_stub(image)
    t1 = time.perf_counter()
    latency_ms = (t1 - t0) * 1000.0

    obs = ObservationCreate(
        plant_class=plant_class,
        confidence=confidence,
        ndvi=ndvi,
        latency_ms=latency_ms,
    )
    new_id = repo.add(obs)

    # Отримуємо створений запис для повернення клієнту
    last_list = repo.list_last(limit=1)
    if not last_list:
            raise HTTPException(status_code=500, detail="Помилка збереження результату у базу даних")

    dto = last_list[0]
    # Переконуємось, що повертаємо саме щойно доданий запис
    if dto.id != new_id:
        dto.id = new_id
    return dto
```

```
@app.get("/api/v1/observations", response_model=List[ObservationDTO])
def get_observations(
    limit: int = 50,
    repo: ObservationRepository = Depends(get_repo),
) -> List[ObservationDTO]:
    """
    Повертає останні N спостережень для відображення
    в аналітичних таблицях і графіках клієнтського застосунку.
    """
    limit = max(1, min(limit, 500))
    return repo.list_last(limit=limit)


@app.get("/api/v1/metrics", response_model=MetricsDTO)
def get_system_metrics(
    repo: ObservationRepository = Depends(get_repo),
) -> MetricsDTO:
    """
    Повертає агреговані метрики роботи системи:
    – кількість спостережень;
    – середній час обробки;
    – середній коефіцієнт впевненості;
    – останній визначений клас.
    """
    return repo.get_metrics()


if __name__ == "__main__":
```

```
import uvicorn

# Запуск FastAPI-додатку в режимі розробки.
        # У промисловому середовищі використовується окремий WSGI/ASGI-сервер.
uvicorn.run(
    "app:app",
    host="0.0.0.0",
    port=8000,
    reload=True,
)
```