НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ** (підпис)	_____ **Белла ГОЛУБ** (підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмне забезпечення системи моніторингу стану домашніх рослин»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми к.т.н., доцент	_____ (підпис)	**Ганна Вайганг**
Керівник бакалаврської кваліфікаційної роботи	_____ (підпис)	**Георгій Бородкін**
Виконав	_____ (підпис)	**Назар Пшеничний**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

Пшеничному Назару Віталійовичу
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи
«Програмне забезпечення системи моніторингу стану домашніх рослин»

затверджена наказом від «19» грудня 2025 р. № 3204 «С»
Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): методичні рекомендації фахівців щодо утримання домашніх рослин

Перелік питань, що підлягають дослідженню:

1. Провести аналіз предметної області.
2. Дослідити стан існуючих аналогів.
3. Спроектувати інформаційне забезпечення системи.
4. Реалізувати мобільний застосунок.
5. Провести тестування системи.
6. Підготувати рекомендації щодо впровадження застосунку.

Перелік графічних документів (за необхідності).

_____

Дата видачі завдання «19» грудня 2025р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Георгій Бородкін**

Завдання взяв до виконання _____ (підпис) **Назар Пшеничний**

Зміст

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API - Application Programming Interface, програмний інтерфейс взаємодії веб-сервісів і модулів системи.

ARIMA - AutoRegressive Integrated Moving Average, модель прогнозування коротких часових рядів сенсорних вимірювань.

AS - Alert Status, показник критичності події або відхилення параметра рослини.

CPU - Central Processing Unit, центральний процесор серверного або локального пристрою.

CSV - Comma-Separated Values, формат експорту табличних даних моніторингу.

DAQ - Data Acquisition, процес отримання даних від сенсорів домашньої рослини.

DB - Database, база даних вимірювань, профілів рослин і подій системи.

DQI - Data Quality Index, індекс повноти та коректності отриманої телеметрії.

EMQX - **MQTT**-брокер, що використовується для маршрутизації IoT-повідомлень.

ESP32 - мікроконтролер сенсорного вузла для збирання показників біля вазона.

FNN - Feedforward Neural Network, нейромережева модель для базової оцінки стану рослини.

HTML - HyperText Markup Language, мова розмітки веб-панелі та звітів системи.

HTTP(S) - HyperText Transfer Protocol Secure, протокол захищеного доступу до веб-контролю.

IoT - Internet of Things, мережа підключених датчиків і виконавчих пристроїв.

JSON - JavaScript Object Notation, формат обміну даними між сенсорним вузлом і сервером.

K-means - метод групування вимірювань за подібністю умов утримання рослин.

LSTM - Long Short-Term Memory, рекурентна модель для аналізу динаміки параметрів.

Lux - одиниця вимірювання освітленості робочої зони рослини.

MAE - Mean Absolute Error, середня абсолютна похибка прогнозу вологості або температури.

MQTT - Message Queuing Telemetry Transport, легковаговий протокол передавання телеметрії.

ORM - Object-Relational Mapping, засіб взаємодії програмних об’єктів із реляційною БД.

PCA - Principal Component Analysis, метод зменшення розмірності наборів вимірювань.

PDF - Portable Document Format, формат збереження звітів користувача.

PHI - Plant Health Index, узагальнений індекс стану домашньої рослини.

pH - водневий показник кислотності ґрунтового субстрату.

RAM - Random Access Memory, оперативна пам’ять пристрою або сервера.

REST - Representational State Transfer, архітектурний стиль обміну даними веб-API.

RMSE - Root Mean Square Error, середньоквадратична похибка прогнозної моделі.

ВСТУП

Актуальність теми зумовлена поширенням домашнього вирощування декоративних і корисних рослин, а також потребою у простих цифрових засобах контролю їхнього стану. У квартирних, офісних і навчальних приміщеннях рослини часто залежать від нерегулярного поливу, нестачі світла, пересушеного повітря та несвоєчасного реагування власника. Тому інформаційна система моніторингу домашніх рослин з веб-контролем є доцільним рішенням, яке поєднує сенсорні вимірювання, автоматичну фіксацію подій, веб-панель спостереження та рекомендації щодо догляду [1].

Метою дослідження є розроблення інформаційної системи моніторингу домашніх рослин з веб-контролем, яка забезпечує збирання, зберігання, аналіз і наочне відображення параметрів мікроклімату та стану ґрунту для своєчасного догляду за рослинами.

Для досягнення поставленої мети необхідно поєднати сенсорний вузол на базі ESP32, датчики вологості ґрунту, температури, вологості повітря та освітленості, серверну частину з базою даних, веб-інтерфейс користувача і механізм сповіщень про критичні відхилення.

У межах роботи передбачено виконання таких завдань:

- проаналізувати предметну область цифрового догляду за кімнатними рослинами та типові проблеми домашнього утримання;
- дослідити існуючі рішення для розумних вазонів, сенсорного контролю та мобільного догляду за рослинами;
- сформувати логічну модель даних для збереження профілів рослин, сенсорних вимірювань, подій і рекомендацій;
- спроєктувати архітектуру системи з веб-контролем, IoT-вузлом, серверним API та аналітичним модулем;
- реалізувати програмний прототип, перевірити роботу інтерфейсу, обміну даними, сповіщень і формування звітів.

Об’єктом дослідження є процес дистанційного моніторингу параметрів середовища домашніх рослин.

Предметом дослідження є методи, моделі та програмні засоби збору, аналізу й веб-відображення даних про стан кімнатних рослин.

У роботі використано методи системного аналізу, об’єктно-орієнтованого моделювання, проєктування баз даних, аналізу часових рядів, перевірки порогових умов і візуальної аналітики. Програмна частина орієнтована на технології Python, FastAPI або Flask, SQLite/PostgreSQL, MQTT, WebSocket, HTML/CSS/JavaScript і веб-панель, що забезпечує доступ до стану рослини з комп’ютера або мобільного браузера.

Науково-практична новизна роботи полягає у формуванні цілісної моделі веб-керованого домашнього фітомоніторингу, де сенсорні вимірювання автоматично пов’язуються з профілем конкретної рослини, допустимими межами параметрів і рекомендаціями для користувача.

Практична цінність полягає у можливості застосування системи для догляду за домашніми рослинами, міні-теплицями, навчальними колекціями та офісним озелененням. Розроблене рішення дає змогу зменшити ризик пересихання ґрунту, надмірного поливу, дефіциту освітлення та втрати даних про попередній догляд.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Інформаційна система моніторингу домашніх рослин з веб-контролем призначена для регулярного збирання, збереження, аналізу та відображення параметрів, що впливають на стан кімнатної рослини. Предметна область охоплює контроль вологості ґрунту, температури й вологості повітря, рівня освітленості, кислотності субстрату, а також фіксацію подій догляду: поливу, підживлення, пересаджування та зміни місця розташування. На рис. 1.1 наведено структурну схему системи, яка показує основні елементи збору й оброблення даних.

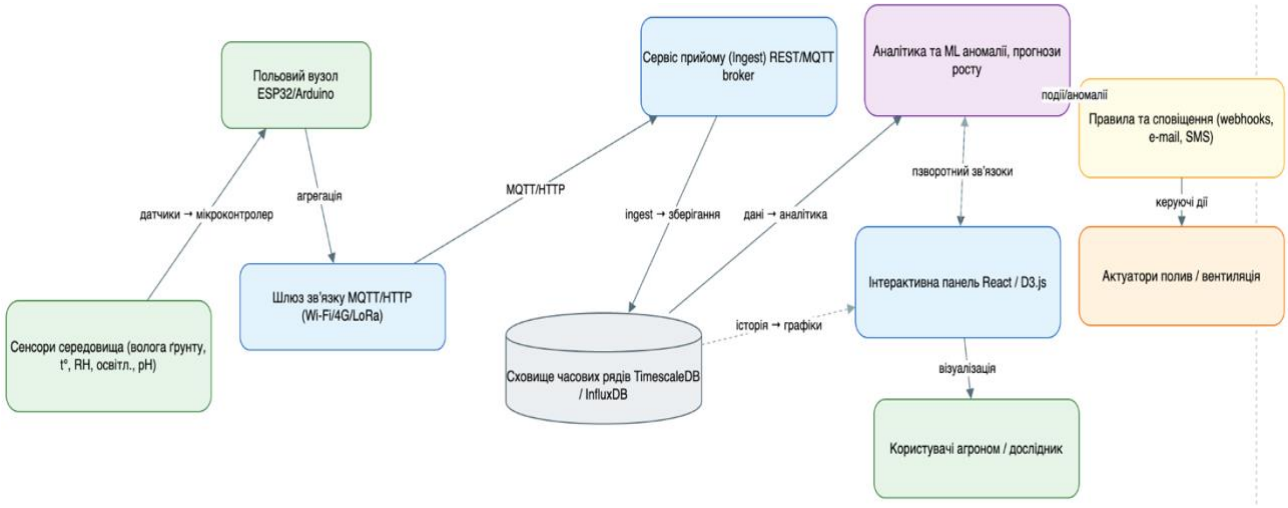

Рис. 1.1 Структура інформаційної системи моніторингу домашніх рослин з веб-контролем

Основу системи становить локальна IoT-інфраструктура, до якої входять сенсорні модулі, мікроконтролер ESP32, канал зв’язку Wi-Fi та сервер приймання даних через MQTT або HTTP API. Виміряні показники передаються до серверної частини, проходять перевірку на коректність, нормалізуються та записуються до бази даних. Такий підхід дає можливість вести історію стану

кожної рослини й аналізувати зміни не за окремим вимірюванням, а за послідовністю спостережень [3].

Після збереження дані використовуються модулем веб-контролю. Користувач бачить поточний стан рослини, графіки зміни вологості, температури та освітленості, а також повідомлення про необхідність поливу або перенесення вазона ближче до джерела світла. Додатково система може керувати виконавчими пристроями: мініпомпою поливу, фітолампою або вентилятором, якщо вони підключені до сенсорного вузла.

Основні параметри, які збирає система, наведено у табл. 1.1. Для кожного показника визначено тип сенсора, одиницю вимірювання та рекомендовану частоту опитування.

Таблиця 1.1

Основні параметри моніторингу домашніх рослин

№	Параметр	Тип сенсора	Одиниця вимірювання	Період опитування
1	Вологість ґрунту	Soil Moisture / Capacitive Sensor	%	3 хв
2	Температура повітря	DHT22 / BME280	°C	5 хв
3	Вологість повітря	DHT22 / BME280	%	5 хв
4	Освітленість	BH1750 / TSL2561	lx	5 хв
5	Кислотність субстрату	Analog pH Meter	pH	30 хв
6	Температура ґрунту	DS18B20	°C	10 хв

Предметна область системи описується циклом: вимірювання - передавання - збереження - аналіз - веб-відображення - дія користувача або автоматичного пристрою. Завдяки цьому власник рослини отримує не лише окремі числові значення, а й зрозумілий контекст: чи є умови нормальними, наскільки швидко змінюється вологість і які дії варто виконати. Подальші етапи

роботи спрямовані на формалізацію вимог, моделювання процесів і побудову програмної архітектури.

1.2 Теоретико-методологічні засади та стан наукових досліджень

Теоретичною основою розробки є багаторівнева архітектура IoT-систем, у якій фізичний рівень відповідає за вимірювання, комунікаційний рівень - за передавання даних, серверний рівень - за оброблення, а рівень веб-представлення - за взаємодію з користувачем. Для домашніх рослин такий підхід особливо важливий, оскільки навіть невеликі зміни вологості ґрунту або освітленості можуть швидко впливати на стан рослини. На рис. 1.2 наведено блок-схему архітектури збору та передавання IoT-даних.

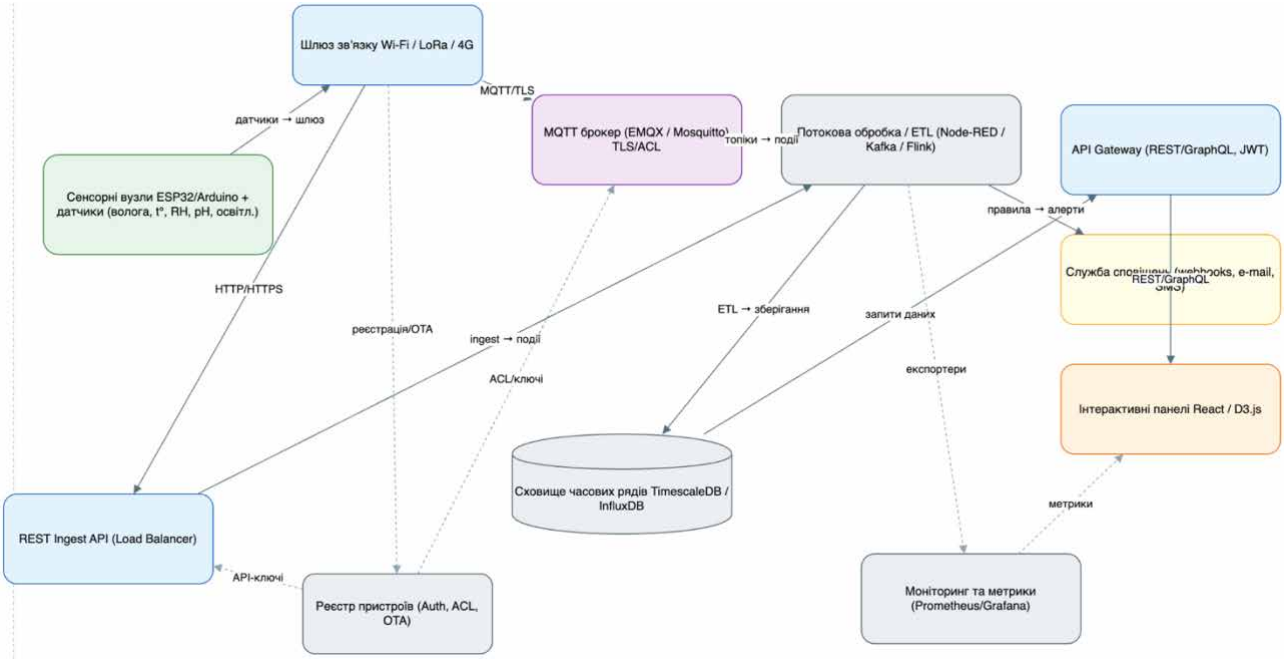

Рис. 1.2 Блок-схема архітектури збору та передавання IoT-даних

Методологія аналізу параметрів ґрунту та мікроклімату базується на роботі з часовими рядами. Короткострокове прогнозування дає змогу визначати момент можливого пересихання субстрату, а статистичні пороги дозволяють виявляти небезпечні стани без складного машинного навчання. Для розширеної аналітики

можуть застосовуватися ARIMA, LSTM або прості регресійні моделі, які порівнюють фактичні вимірювання з рекомендованими межами для конкретного виду рослини. На рис. 1.3 показано загальну логіку аналітичної обробки.

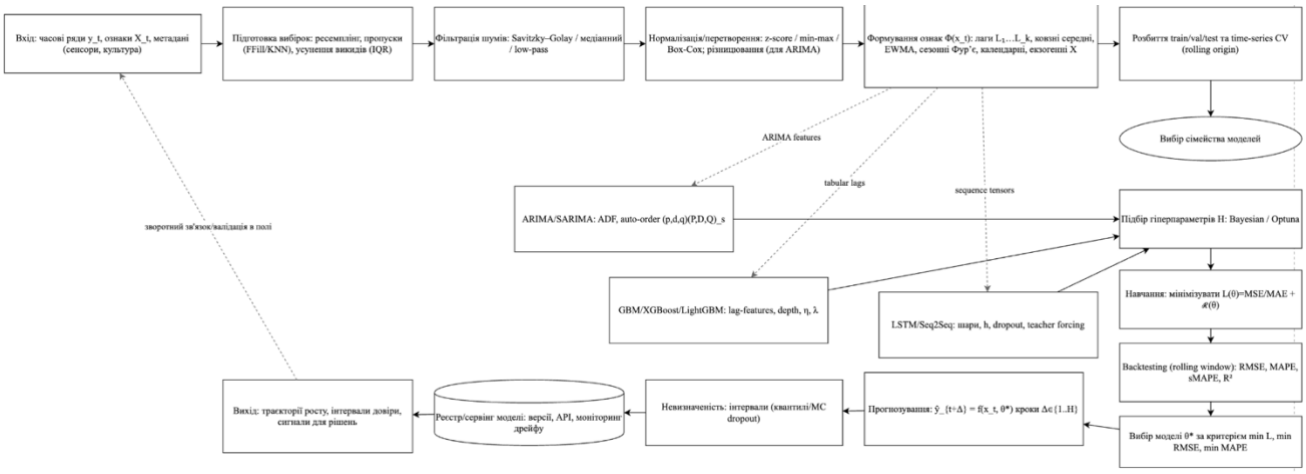

Рис. 1.3 Математична блок-схема аналітики та прогнозування стану домашньої рослини

З позиції системного аналізу домашня рослина розглядається як об’єкт спостереження, стан якого залежить від умов освітлення, вологості, температури, складу субстрату та регулярності догляду. Інформаційна система не замінює біологічне оцінювання, але забезпечує цифрову підтримку прийняття рішень: фіксує тенденції, виявляє відхилення й пропонує прості дії для стабілізації умов утримання.

Розглянуті принципи визначають основу майбутньої розробки: система має бути модульною, доступною через веб-інтерфейс, здатною працювати з кількома рослинами та підтримувати історію спостережень. У наступних підрозділах виконано аналіз готових рішень, моделювання предметної області та формування вимог до програмного забезпечення.

1.3 Аналіз існуючих рішень

Ринок цифрового догляду за рослинами включає мобільні застосунки для нагадувань, сенсорні пристрої для контролю ґрунту, розумні вазони та

універсальні платформи «розумного дому». Їхній аналіз потрібний для визначення функцій, які варто реалізувати у власній системі: веб-доступ, журнал вимірювань, сповіщення, робота з кількома рослинами та можливість підключення виконавчих пристроїв.

Одним із відомих підходів є використання мобільних застосунків для догляду за рослинами, наприклад Planta або Blossom. Такі рішення допомагають вести календар поливу, але часто залежать від ручного введення інформації й не завжди отримують реальні сенсорні дані. На рис. 1.4 наведено приклад інтерфейсу, який демонструє логіку подання параметрів і рекомендацій.

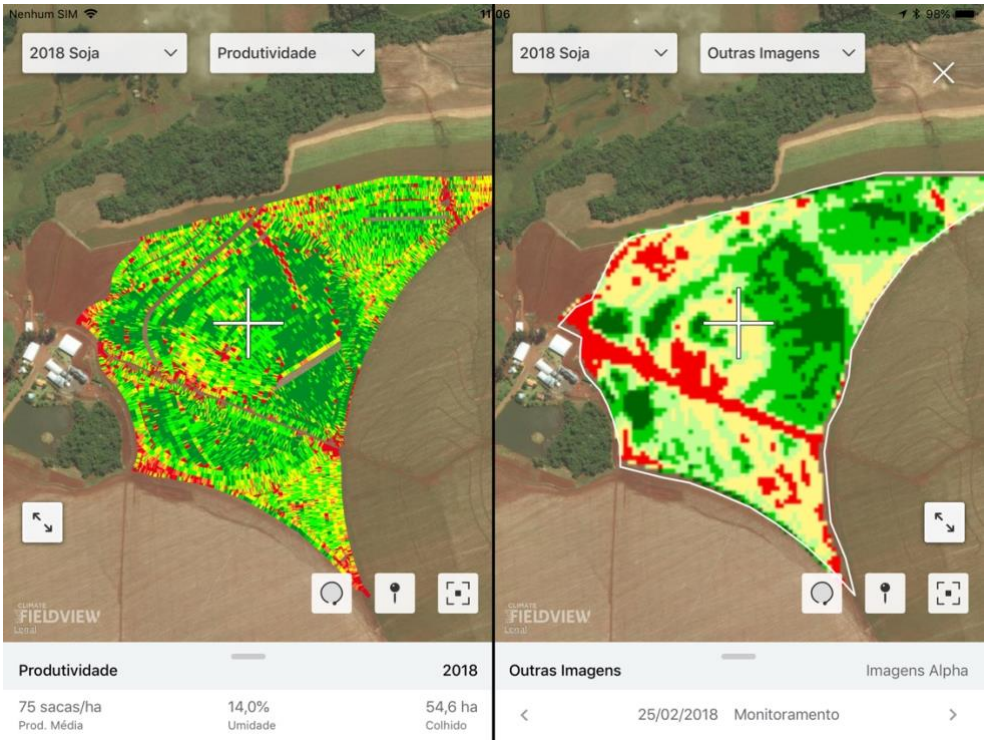

Рис. 1.4 Приклад інтерфейсу сервісу догляду за рослинами з рекомендаційними елементами

Інший клас рішень представлений розумними вазонами та датчиками ґрунту, зокрема Xiaomi Mi Flora, Parrot Flower Power або подібними BLE/IoT-сенсорами. Вони забезпечують вимірювання вологості, освітленості та температури, однак здебільшого орієнтовані на окремий мобільний застосунок, що ускладнює централізований веб-контроль декількох рослин.

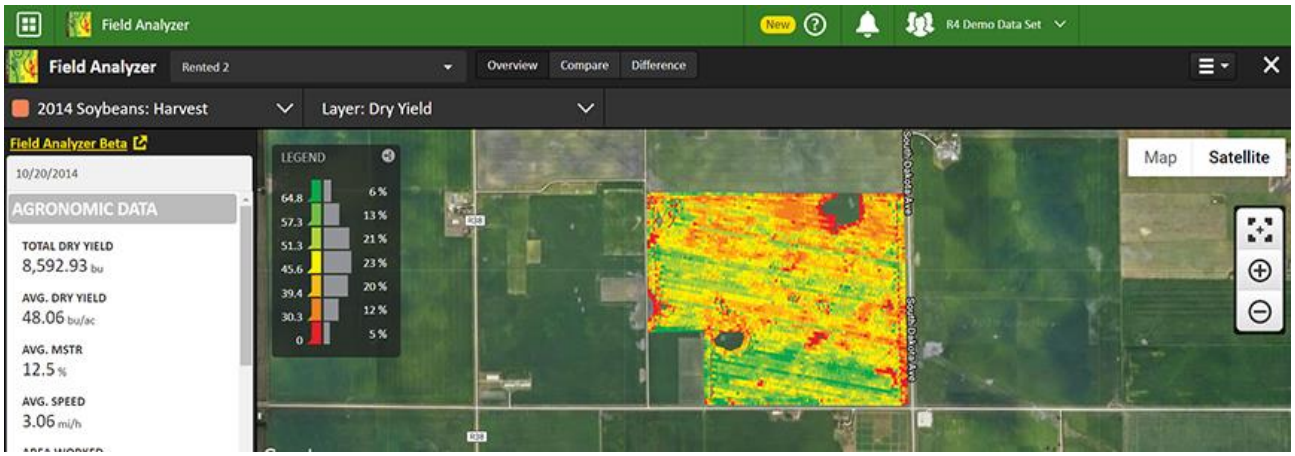

Рис. 1.5 Модуль сенсорного контролю параметрів рослини у готовому IoT-рішенні

Системи класу Home Assistant дають змогу інтегрувати датчики, реле, помпи та фітолампи в загальну інфраструктуру розумного дому. Їхня перевага полягає у гнучкій автоматизації, проте для навчального або дипломного проєкту доцільно створити спеціалізовану систему, орієнтовану саме на профілі рослин, допустимі межі показників і зрозумілу веб-панель.

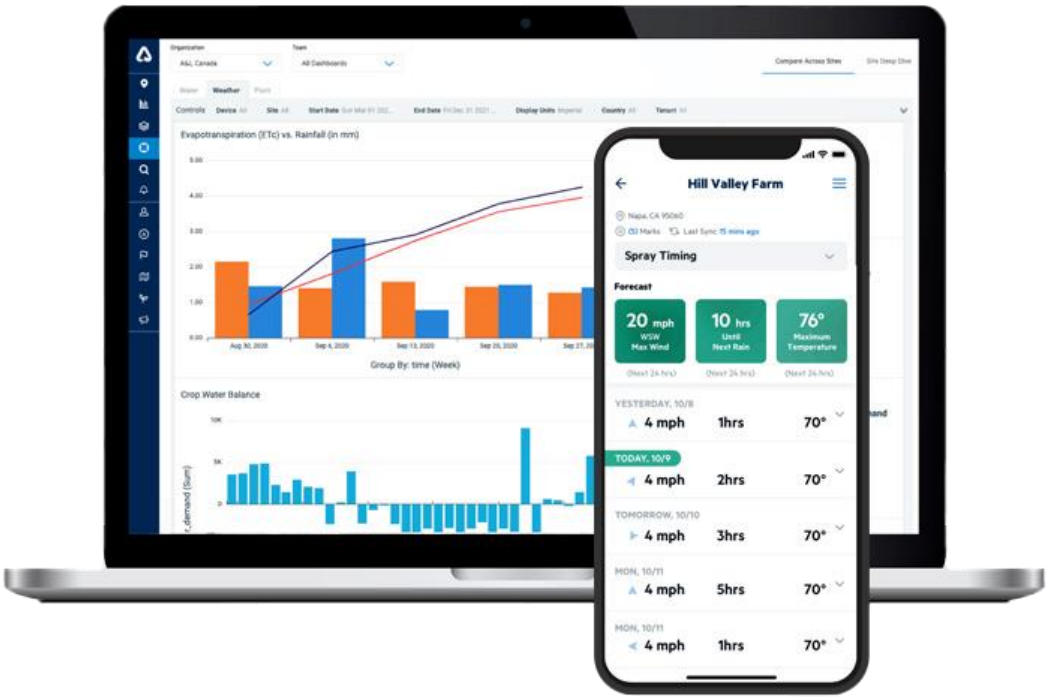

Рис. 1.6 Приклад веб-панелі з графіками параметрів мікроклімату

Також існують системи автоматизованого поливу для домашніх домашніх міні-теплиць і вертикальних ферм. Вони ефективні для групового керування

обладнанням, але часто не забезпечують детальної персоналізації під конкретний вид кімнатної рослини. Розроблюване рішення має поєднати простоту домашнього використання та достатній рівень технічного контролю.

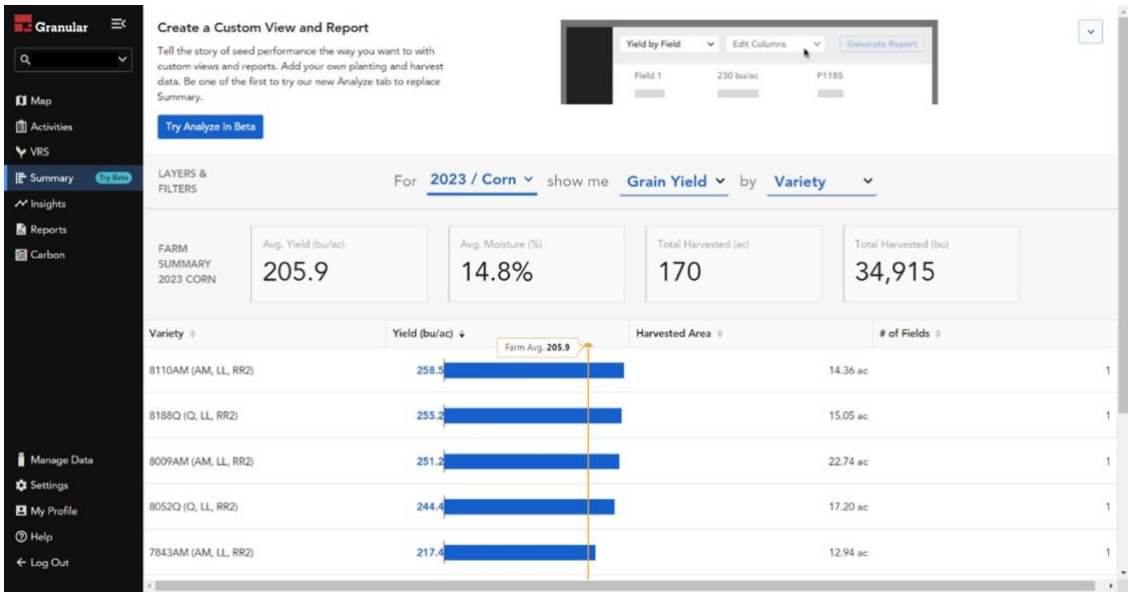

Рис. 1.7 Аналітична панель контролю рослин із відображенням показників догляду

Для порівняння наявних рішень із розроблюваною системою моніторингу домашніх рослин з веб-контролем сформовано табл. 1.2, у якій наведено джерела даних, тип аналітики, спосіб візуалізації та особливості інтеграції.

Таблиця 1.2

Порівняльна характеристика існуючих рішень і розроблюваної системи

Система	**Джерела даних**	**Аналітика / ML**	**Візуалізація**	**Інтерактивність**	**API / Обмін даними**	**Особливості**
Planta	Ручні записи, календар догляду	Рекомендації за профілем	Картки рослин, нагадування	Висока	Cloud / Mobile API	Зручний догляд без реальної телеметрії

Система	Джерела даних	Аналітика / ML	Візуалізація	Інтерактивність	API / Обмін даними	Особливості
Xiaomi Mi Flora	BLE-сенсор, ґрунт, світло	Базова оцінка параметрів	Мобільні графіки	Середня	Bluetooth / інтеграції	Працює переважно з окремим датчиком
Home Assistant	IoT-сенсори, реле, сценарії	Правила автоматизації	Веб-панелі, графіки	Висока	REST / MQTT	Гнучка платформа розумного дому
Blossom	Календар, база рослин	Персональні поради	Мобільний інтерфейс	Висока	Cloud	Орієнтація на довідник і нагадування
Розроблювана система	ESP32, датчики, SQLite/PostgreSQL	Пороги, PHI, прогноз вологості	Веб-панель, графіки, звіти	Дуже висока	MQTT + REST + WebSocket	Веб-контроль домашніх рослин у реальному часі

Проведений аналіз показує, що більшість доступних інструментів або орієнтована лише на календар догляду, або працює як окремий сенсор без розгорнутої веб-аналітики. Запропонована система відрізняється поєднанням реальних вимірювань, профілів домашніх рослин, веб-контролю, історії подій та можливості підключення автоматичного поливу чи освітлення.

1.4 Моделювання предметної області

Моделювання предметної області дає змогу формалізувати сценарії використання системи, визначити основних акторів і показати взаємодію між сенсорним вузлом, сервером, базою даних та веб-панеллю. Для цього використано UML-діаграми варіантів використання, послідовності та активності, які відображають структурну й поведінкову логіку майбутнього програмного забезпечення.

На рис. 1.8 подано діаграму варіантів використання. У ній визначено основних акторів: користувач, адміністратор системи, сенсорний вузол ESP32, веб-сервер, база даних і сервіс сповіщень.

Рис. 1.8 Діаграма варіантів використання системи моніторингу домашніх рослин з веб-контролем

Основними прецедентами є реєстрація користувача, додавання рослини, прив’язування сенсорного вузла, перегляд поточних параметрів, аналіз графіків, отримання сповіщень, налаштування порогів і формування звіту. Такий розподіл функцій дозволяє відокремити дії людини від автоматичних процесів збору й оброблення даних.

Динамічна взаємодія компонентів представлена на рис. 1.9. Типовий сценарій починається з авторизації користувача у веб-панелі, після чого сервер завантажує список рослин, останні вимірювання та статус сенсорного вузла. Далі

веб-інтерфейс надсилає запит на графіки, база даних повертає часові ряди, а модуль сповіщень перевіряє перевищення порогів вологості, температури або освітленості.

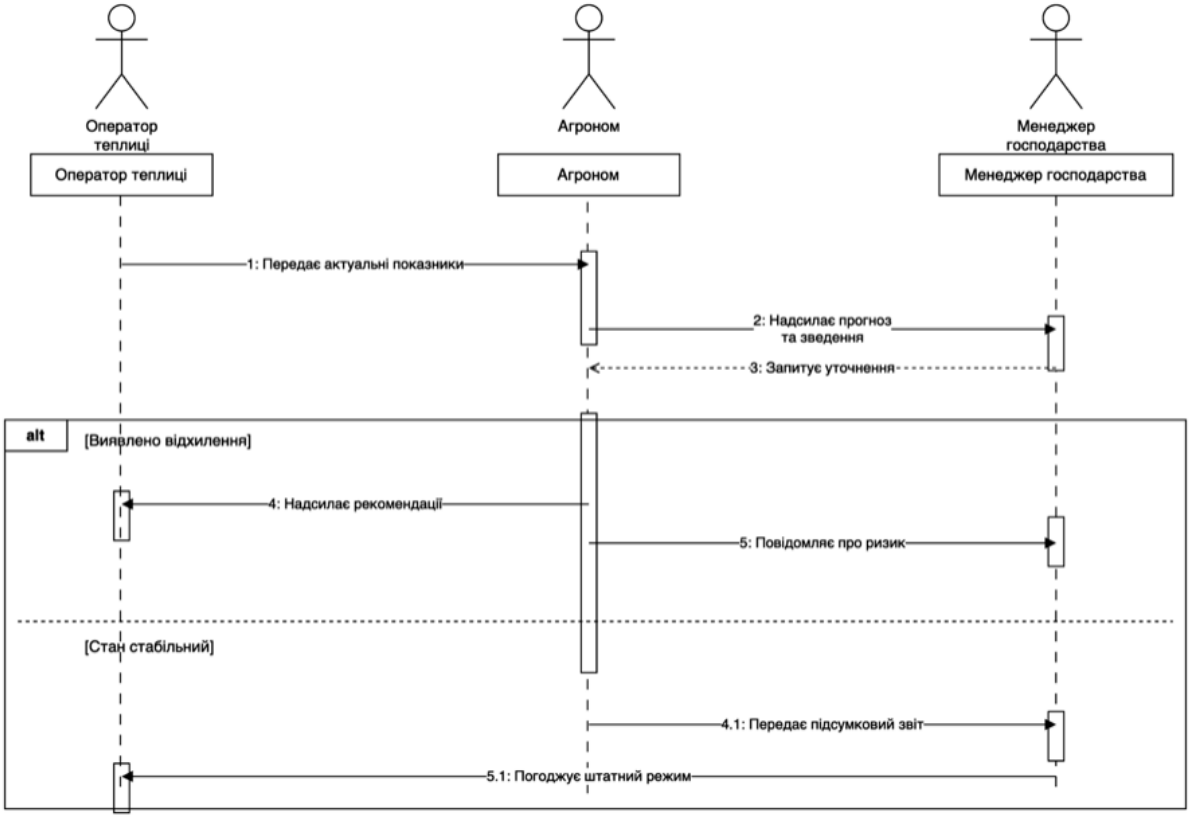

Рис. 1.9 Діаграма послідовності взаємодії користувача з веб-системою моніторингу

Процесна логіка системи відображена на рис. 1.10. Вона включає запуск сенсорного вузла, перевірку підключення, передавання вимірювань, валідацію даних, запис до бази, оновлення веб-панелі та формування сповіщення. Передбачено альтернативні гілки: робота при втраті зв’язку, повторна синхронізація та ручне внесення події догляду.

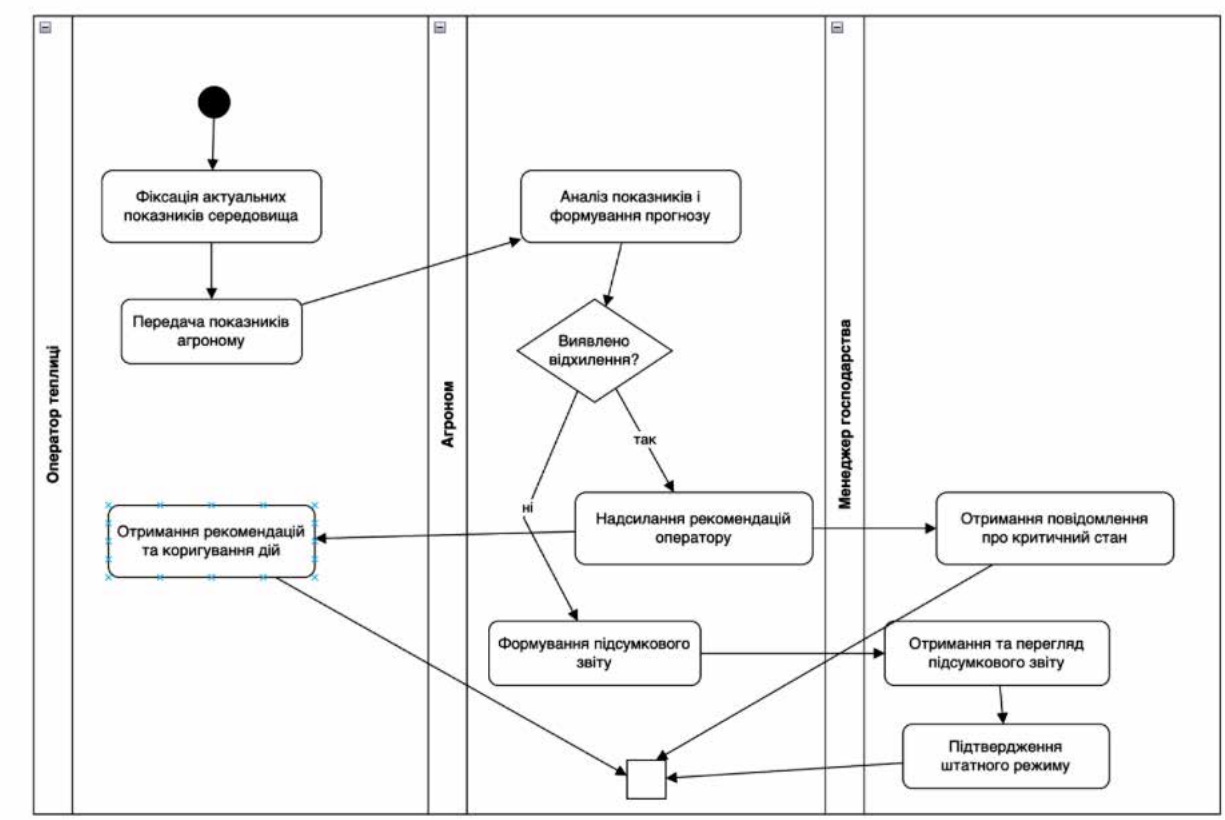

Рис. 1.10 Діаграма активності процесу функціонування системи

Завершальним етапом сценарію є фіксація дії користувача: полив, підживлення, вимкнення сповіщення або зміна порогів для конкретної рослини. Це забезпечує накопичення історії догляду й підвищує корисність подальшого аналізу.

Модель предметної області визначає узгоджену взаємодію між апаратною, серверною та інтерфейсною частинами. Розроблені діаграми створюють основу для проєктування структури даних, класів, компонентів, алгоритмів і сценаріїв тестування, розглянутих у наступних розділах.

1.5 Аналіз вимог розроблюваної системи

Аналіз вимог формує основу подальшого проєктування, оскільки визначає функціональні межі системи, якість її роботи та умови експлуатації. Для системи моніторингу домашніх рослин з веб-контролем вимоги охоплюють сенсорний

рівень, серверну обробку, веб-інтерфейс, базу даних, механізм сповіщень і можливість інтеграції з виконавчими пристроями.

За результатами аналізу предметної області систему поділено на три основні підсистеми: збирання телеметрії, веб-аналітика та керування подіями догляду. Перша відповідає за отримання вимірювань, друга - за збереження, оброблення і графічне подання, третя - за сповіщення користувача та, за потреби, запуск автоматичного поливу або освітлення.

Функціональні вимоги наведено у табл. 1.3. Вони описують конкретні дії, які система повинна виконувати для повного циклу моніторингу домашніх рослин.

Таблиця 1.3

Функціональні вимоги інформаційної системи

№	Назва вимоги	Опис функціональності	Очікуваний результат
1	Збір сенсорних даних	Отримання показників вологості, температури, освітленості та pH через MQTT/HTTP	Структурований потік вимірювань для кожної рослини
2	Валідація вимірювань	Перевірка пропусків, некоректних форматів і виходу за фізичні межі	Очищені дані для збереження та аналізу
3	Оцінювання стану рослини	Розрахунок PHI та перевірка параметрів відповідно до профілю виду	Статус «норма», «увага» або «критично»
4	Веб-візуалізація	Побудова графіків і карток рослин у веб-панелі	Зручний контроль з браузера
5	Експорт і звітність	Формування CSV, HTML або PDF-звіту	Можливість аналізу історії догляду
6	Система сповіщень	Генерація повідомлень про полив, нестачу світла або втрату зв’язку	Оперативне інформування користувача

Нефункціональні вимоги визначають продуктивність, безпеку, надійність і зручність використання системи. Їх узагальнено в табл. 1.4.

Таблиця 1.4

Нефункціональні вимоги системи

№	Категорія	Вимога	Характеристика
1	Продуктивність	Оновлення веб-панелі не довше 2 секунд	Асинхронне приймання даних і кешування останніх вимірювань
2	Безпека	Авторизація користувачів і захищені токени доступу	Обмеження доступу до профілів рослин і налаштувань
3	Масштабованість	Підключення не менше 20 сенсорних вузлів	Модульна структура API та БД
4	Надійність	Відновлення синхронізації після втрати зв’язку	Буферизація вимірювань на вузлі або повторна передача
5	Юзабіліті	Адаптивний веб-інтерфейс	Зручність перегляду з ПК і мобільного браузера
6	Сумісність	Підтримка MQTT, REST та WebSocket	Можливість інтеграції з розумним домом

Технічні вимоги встановлюють інструменти, платформи та апаратні компоненти, необхідні для реалізації веб-контрольованої системи. Їх подано в табл. 1.5.

Таблиця 1.5

Технічні вимоги програмної системи

№	Компонент	Технологія / Платформа	Призначення
1	Серверна частина	Python, FastAPI / Flask	Приймання телеметрії, API, оброблення запитів
2	Аналітичний модуль	Pandas, scikit-learn, statsmodels	Оцінка стану, прогноз вологості, кластеризація
3	Клієнтська частина	HTML, CSS, JavaScript, React / Chart.js	Веб-панель, графіки, картки рослин

4	Комунікаційний рівень	MQTT, REST, WebSocket	Передавання даних між сенсорами, сервером і браузером
5	Апаратна база	ESP32, DHT22, BH1750, capacitive soil sensor	Вимірювання параметрів домашньої рослини

Продовження таблиці 1.5

6	Розгортання	Docker Compose, Nginx, PostgreSQL	Запуск і масштабування серверної частини

Зведений аналіз вимог показує, що система повинна бути доступною з браузера, підтримувати роботу з кількома рослинами, зберігати історію вимірювань, надавати зрозумілі сповіщення та працювати стабільно навіть при нестабільному з’єднанні сенсорного вузла.

Узагальнено розроблювана система реалізує цикл «сенсор - сервер - база даних - веб-панель - дія користувача». Така модель є достатньо простою для домашнього використання, але водночас підтримує розширення за рахунок додавання нових датчиків, рослин і правил автоматизації.

1.6 Постановка завдання

Постановка завдання визначає призначення, вхідні та вихідні дані, межі функціональності й очікуваний результат розроблення. У межах теми необхідно створити інформаційну систему моніторингу домашніх рослин з веб-контролем, яка забезпечує збір показників з сенсорного вузла, їх збереження, аналітичне опрацювання, відображення у веб-інтерфейсі та інформування користувача про потребу догляду.

Під час розроблення потрібно забезпечити узгоджену роботу фізичного рівня (ESP32 і датчики), серверного рівня (API, база даних, правила перевірки), клієнтського рівня (веб-панель) і комунікаційного рівня (MQTT або HTTP). Система має бути модульною, щоб у майбутньому можна було додати керування

помпою, фітолампою, журнал підживлення або підключення до системи розумного дому [7].

Вхідні дані системи:

– показники сенсорів домашньої рослини: вологість ґрунту, температура, вологість повітря, освітленість і pH субстрату;

– профіль рослини: назва, вид, рекомендовані межі вологості, освітлення, температури та періодичність догляду;

– дані користувача і налаштування веб-доступу, включаючи пороги сповіщень і список підключених вузлів;

– історичні записи вимірювань, подій поливу, підживлення, пересаджування та змін місця розташування;

– параметри автоматизації: правила запуску помпи, фітолампи або формування критичних повідомлень.

Вихідні дані системи:

– веб-панель з поточним станом кожної домашньої рослини та статусом сенсорного вузла;

– графіки зміни вологості, температури, освітленості та інших параметрів за обраний період;

– рекомендації щодо поливу, освітлення, провітрювання або зміни умов розміщення рослини;

– автоматизовані звіти у форматах HTML, CSV або PDF для аналізу історії догляду;

– сповіщення про критичні відхилення, втрату зв’язку із сенсором або потребу ручної дії користувача.

Формально завдання можна подати як перетворення множини вхідних параметрів стану рослини

$$X = \{x1, x2, ..., xn\}$$

на множину керуючих та інформаційних результатів

$$Y = f(X, \theta),$$

де f - функція оброблення та оцінювання стану рослини, а θ - набір порогів, правил і параметрів моделі. Отже, система повинна не лише збирати дані, а й інтерпретувати їх у зрозумілі для користувача дії.

У результаті реалізації завдання буде створено програмно-апаратну систему, яка:

– об’єднує сенсорний вузол, серверну частину, базу даних і веб-інтерфейс у єдину інфраструктуру;

– дозволяє користувачу дистанційно контролювати стан домашніх рослин у режимі близькому до реального часу;

– підтримує персональні налаштування для різних видів рослин і сценаріїв догляду;

– створює основу для подальшого розвитку автоматизованого поливу, керування освітленням і інтеграції з розумним домом.

Отже, постановка завдання визначає науково-технічну логіку проєкту та забезпечує перехід від аналізу предметної області до проєктування інформаційного й програмного забезпечення системи.

1.7 Висновки до першого розділу

У першому розділі виконано аналіз предметної області, пов’язаної з цифровим контролем стану домашніх рослин. Визначено основні проблеми: нерегулярний полив, нестача світла, відсутність історії догляду, складність контролю кількох рослин і несвоєчасне реагування на зміну умов середовища.

Проаналізовано готові мобільні застосунки, сенсорні пристрої та платформи розумного дому. Встановлено, що більшість із них вирішує лише частину задачі: нагадує про полив, показує окремі сенсорні значення або потребує складного налаштування. Це підтверджує доцільність створення спеціалізованої веб-системи з орієнтацією на профілі кімнатних рослин.

У процесі моделювання сформовано концепцію системи з поділом на сенсорну, серверну, аналітичну та інтерфейсну підсистеми. За допомогою UML-діаграм описано сценарії взаємодії користувача, сенсорного вузла, сервера, бази даних і модуля сповіщень.

Сформульована постановка завдання визначила мету подальшої розробки - створити систему, яка збирає дані про стан домашніх рослин, відображає їх у веб-панелі, формує сповіщення та підтримує прийняття рішень щодо догляду. Отримані результати є основою для проєктування структури даних і програмних компонентів.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних є основою побудови сховища системи моніторингу домашніх рослин з веб-контролем. Вона забезпечує узгоджене збереження профілів рослин, сенсорних вузлів, вимірювань, подій догляду, сповіщень і користувацьких налаштувань. Модель орієнтована на повний цикл роботи: підключення рослини, отримання телеметрії, аналіз умов і подання результатів у веб-інтерфейсі.

На рис. 2.1 подано ER-діаграму логічної моделі даних. Її структура відображає рух інформації від сенсорного вузла PlantNode через датчики Sensor до таблиці Measurement, де накопичуються часові вимірювання. Далі дані використовуються для формування Alert, CareAction, Forecast і Report, що забезпечує контроль стану рослин та ведення історії догляду.

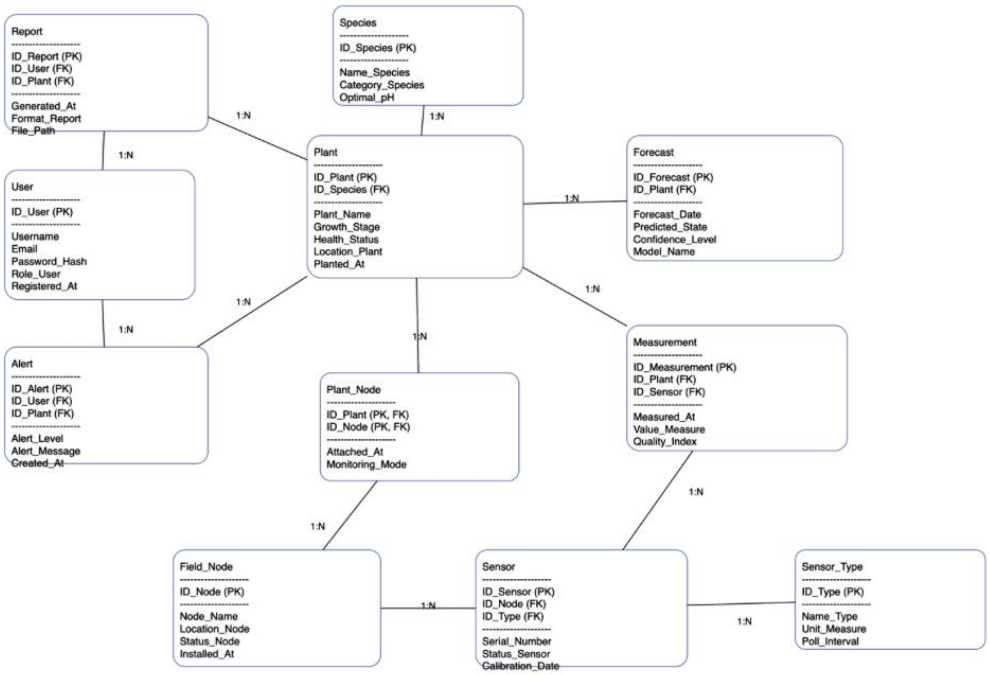

Рис. 2.1 ER-діаграма логічної моделі даних системи моніторингу домашніх рослин

Модель реалізує принцип єдиного інформаційного контуру. Сенсорний рівень відповідає за збір даних, рівень збереження - за цілісність і історичність записів, аналітичний рівень - за оцінювання стану, а користувацький рівень - за звіти, сповіщення й налаштування догляду.

Основні сутності та їхні атрибути наведено у табл. 2.1. Вони формують ядро бази даних і дозволяють нормалізувати інформацію без дублювання параметрів рослин, користувачів і сенсорів.

Таблиця 2.1

Основні сутності та атрибути логічної моделі даних

Сутність	Ключові атрибути	Тип даних	Призначення
PlantNode	id, name, location, status	int, string	Сенсорний вузол, прив’язаний до рослини або зони
Sensor	id, type, unit, freqMin	int, string	Опис фізичного датчика та одиниць вимірювання
Measurement	id, timestamp, value, sensor_id	int, datetime, float	Збереження часових сенсорних вимірювань
PlantProfile	id, plant_name, species, moisture_min, light_min	int, string, float	Профіль рослини та рекомендовані межі
Alert	id, level, message, created_at	int, string, datetime	Події та повідомлення про відхилення
User	id, username, role, email	int, string	Користувач веб-системи
CareReport	id, generated_at, file_path, user_id	int, datetime, string	Звіт про стан і догляд за рослинами

Логічна модель створює основу для фізичної реалізації бази даних, налаштування зв’язків між таблицями та інтеграції з серверною частиною. Вона забезпечує збереження вимірювань у часовій послідовності та підтримує швидке формування графіків у веб-панелі.

2.2 Діаграма класів та кооперації

Діаграма класів відображає об’єктно-орієнтовану структуру програмного забезпечення. Вона описує класи, які забезпечують реєстрацію користувача, керування рослинами, приймання телеметрії, перевірку порогів, формування сповіщень і виведення даних у веб-інтерфейс. Така модель знижує зв’язність компонентів і спрощує подальше розширення системи.

Класи згруповано за функціональними рівнями: доменний рівень PlantProfile, SensorNode і Measurement; сервісний рівень TelemetryIngestor, RuleEngine і NotificationService; рівень збереження DatabaseRepository; рівень представлення PlantWebDashboard. На рис. 2.2 наведено UML-діаграму класів із ключовими атрибутами, методами й асоціаціями.

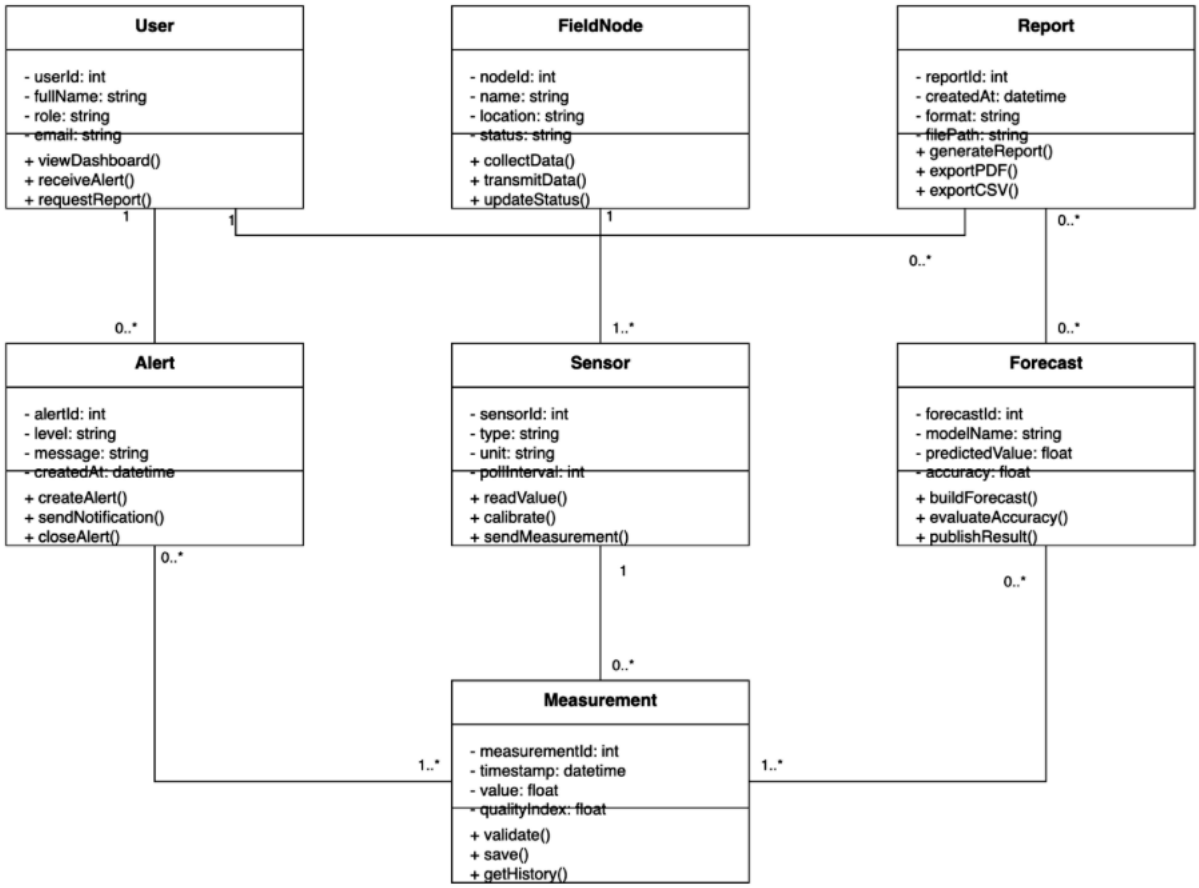

Рис. 2.2 UML-діаграма класів програмного забезпечення системи веб-контролю домашніх рослин

Зв’язки між класами показують життєвий цикл даних: SensorNode отримує вимірювання від Sensor, TelemetryIngestor перетворює повідомлення MQTT або HTTP у структурований об’єкт Measurement, DatabaseRepository записує його до БД, а RuleEngine порівнює значення з нормами профілю PlantProfile. Якщо параметр виходить за межі, NotificationService створює подію для PlantWebDashboard.

Для пояснення динаміки взаємодії компонентів розроблено три діаграми кооперацій (рис. 2.3-2.5), які демонструють типові сценарії роботи системи.

Перша кооперація на рис. 2.3 описує передавання первинних показників від датчика до сенсорного вузла, сервісу приймання і бази даних.

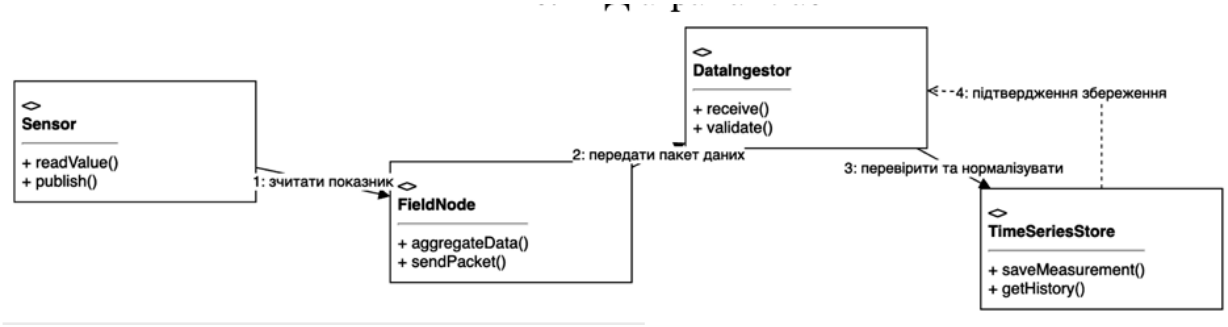

Рис. 2.3 Кооперація 1. Передавання телеметрії між Sensor, PlantNode, TelemetryIngestor та DatabaseRepository

Друга кооперація (рис. 2.4) відображає перевірку порогів: PlantWebDashboard ініціює запит поточного стану, RuleEngine аналізує останні вимірювання, а NotificationService формує повідомлення про потребу догляду.

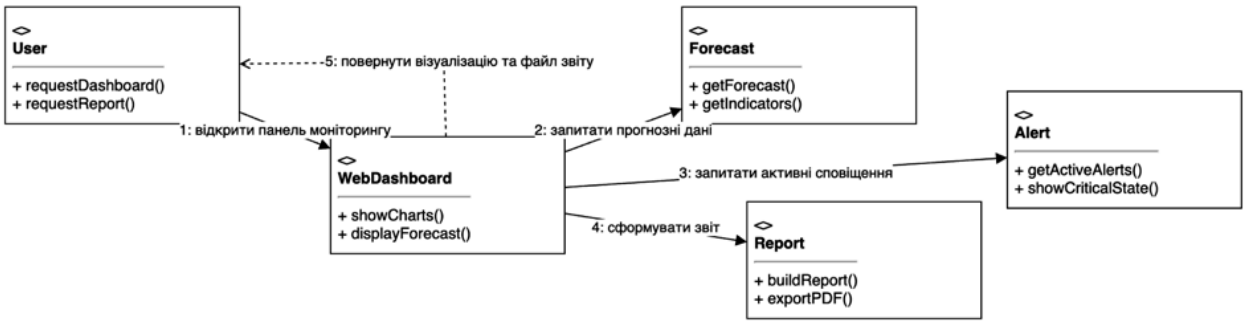

Рис. 2.4 Кооперація 2. Оцінювання стану рослини й формування сповіщення

Третя кооперація на рис. 2.5 описує перегляд історії догляду: користувач обирає рослину, веб-панель отримує дані з репозиторію, будує графік і показує рекомендації.

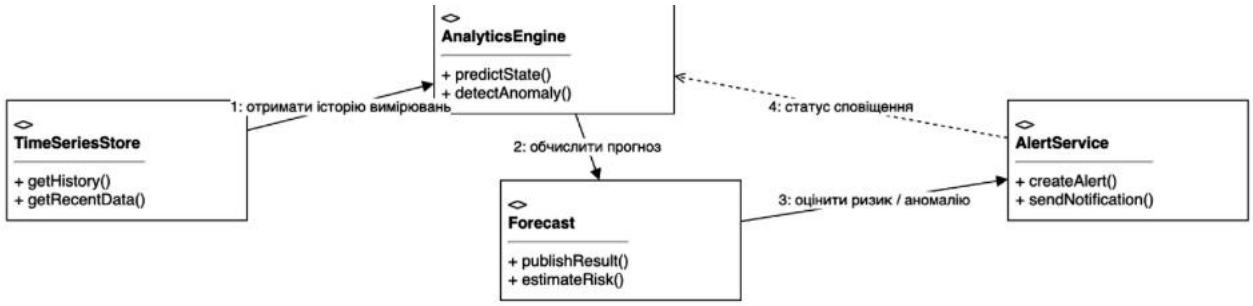

Рис. 2.5 Кооперація 3. Візуалізація історії вимірювань у веб-панелі

Об’єктна модель відповідає принципам інкапсуляції, модульності й розмежування відповідальності. Усі ключові взаємодії реалізуються через чіткі інтерфейси API, що дає змогу незалежно змінювати сенсорний вузол, серверну логіку або клієнтську частину без порушення цілісності системи.

Таблиця 2.2

Структурна узгодженість класів та їх функціональні ролі

Клас	**Рівень**	**Основна функція**	**Тип зв’язку**
PlantNode	Сенсорний	Агрегація датчиків і передавання телеметрії	Агрегація
Sensor	Сенсорний	Зчитування фізичного параметра середовища	Асоціація
TelemetryIngestor	Комунікаційний	Приймання повідомлень, парсинг, запис у БД	Асоціація
MeasurementStore	Дані	Збереження та вибірка часових рядів	Залежність
PlantAnalyticsService	Аналітика	Розрахунок PHI, прогноз, перевірка порогів	Асоціація

NotificationService	Оповіщення	Формування веб-повідомлень і подій догляду	Залежність
PlantPlantWebDashboard	Інтерфейс	Перегляд рослин, графіків, сповіщень і звітів	Асоціація

Отримана модель демонструє логіку оброблення даних домашнього фітомоніторингу та забезпечує узгоджену взаємодію сенсорної, аналітичної й інтерфейсної підсистем. Вона також підтримує додавання нових типів датчиків, профілів рослин і правил автоматизації.

2.3 Діаграма компонентів

Діаграма компонентів відображає архітектуру програмного забезпечення, у якій сенсорний вузол, брокер повідомлень, сервер API, база даних, модуль правил, система сповіщень і веб-інтерфейс поєднані в єдину структуру. Мета цієї моделі - показати розподіл відповідальності між компонентами та способи їхньої взаємодії.

На рис. 2.6 наведено UML-діаграму компонентів системи моніторингу домашніх рослин з веб-контролем. Обмін даними між рівнями здійснюється через MQTT, REST API та WebSocket, що забезпечує оновлення показників у веб-панелі без ручного перезавантаження сторінки.

Архітектура побудована за принципом розділення функцій. Сенсорний рівень формує вимірювання, комунікаційний рівень передає їх на сервер, сервісний рівень виконує валідацію та перевірку правил, база даних зберігає історію, а веб-рівень забезпечує перегляд і керування. Контур автентифікації захищає доступ до профілів рослин і налаштувань користувача.

Така структура дає змогу підтримувати стабільну роботу системи навіть при збільшенні кількості рослин. Компоненти можна оновлювати незалежно:

наприклад, змінити базу даних або додати новий сенсор без переписування веб-панелі.

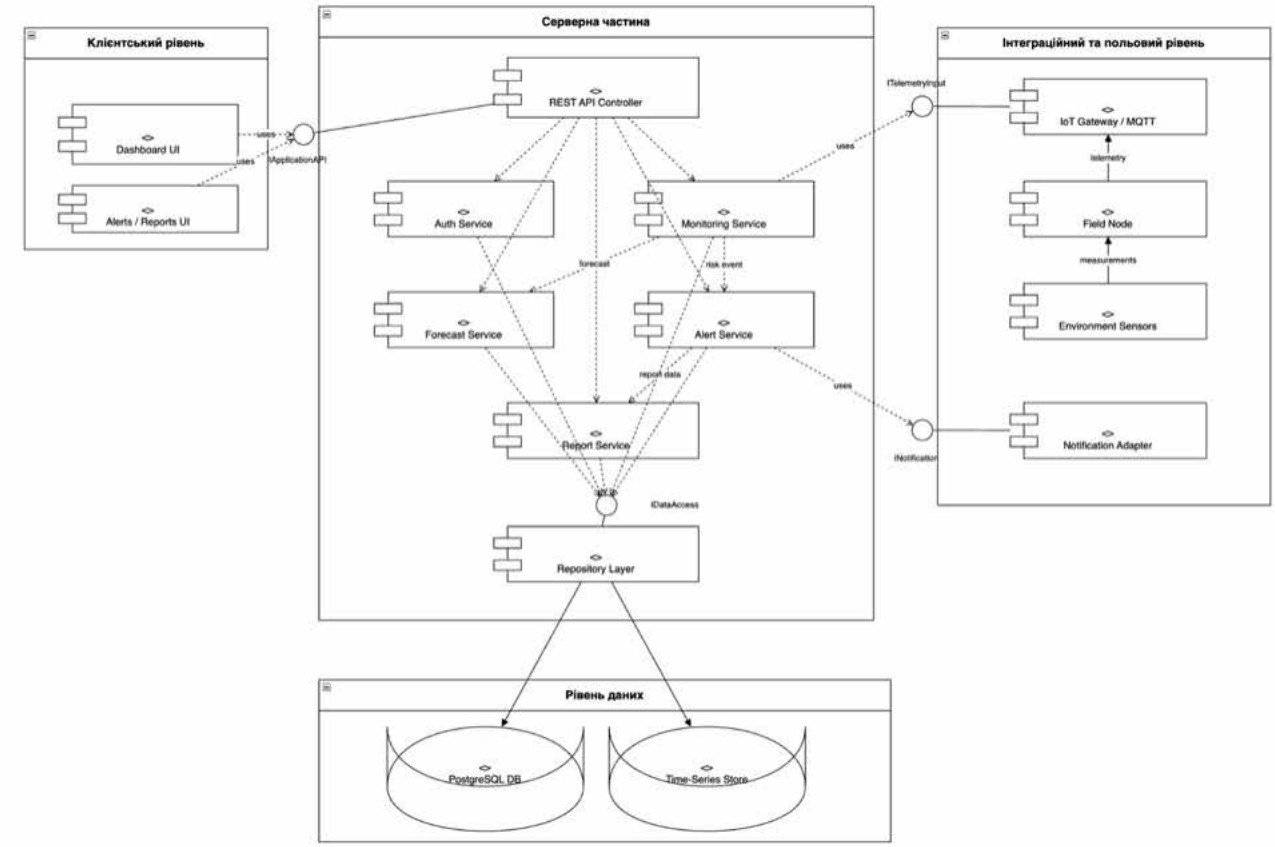

Рис. 2.6 Діаграма компонентів програмного забезпечення системи моніторингу домашніх рослин

Узагальнену характеристику компонентів подано в табл. 2.3, де визначено основні шари, взаємодії та призначення кожного рівня системи.

Таблиця 2.3

Структурно-функціональні зв’язки компонентів системи

Шар системи	Основна взаємодія	Призначення
Сенсорний	Передача показників до сервісу приймання	Отримання первинних даних від рослин
Сервісний	Валідація, нормалізація, запис у БД	Підготовка даних для веб-панелі
Аналітичний	Перевірка порогів, PHI, прогноз вологості	Формування оцінок і попереджень

Безпековий	Авторизація, токени, права доступу	Захист профілів користувачів
Користувацький	Перегляд графіків, налаштування догляду	Веб-контроль стану рослин

Результуюча архітектура забезпечує баланс між простотою домашнього використання та технічною розширюваністю. Вона підтримує локальні сенсорні вузли, серверну обробку, веб-доступ і можливість інтеграції з модулями автоматичного поливу або освітлення.

2.4 Діаграма пакетів

Діаграма пакетів системи показує логічну організацію програмного коду. Пакети розділено за призначенням: робота з пристроями, комунікація, доменна логіка, доступ до даних, безпека, веб-інтерфейс і звітність. Такий підхід спрощує супровід проєкту й дозволяє незалежно змінювати окремі модулі.

На рис. 2.7 подано UML-діаграму пакетів. Вона побудована з урахуванням принципів слабкої зв’язності та високої внутрішньої узгодженості, коли кожен пакет має власну відповідальність і взаємодіє з іншими через визначені інтерфейси.

Пакетна організація підтримує три основні напрями системної логіки:

– інформаційний потік - отримання вимірювань від сенсорного вузла, передавання на сервер і запис до БД;

– аналітичний контур - перевірка порогів, розрахунок індексу стану рослини та формування рекомендацій;

– керуючий рівень - автентифікація, веб-контроль, сповіщення та запуск дій користувача або виконавчих пристроїв.

Ключовою перевагою пакетної моделі є відокремлення доменної логіки від інтерфейсу й інфраструктурних модулів. Завдяки цьому можна змінювати спосіб

зберігання даних або дизайн веб-панелі без зміни алгоритмів оцінювання стану рослини.

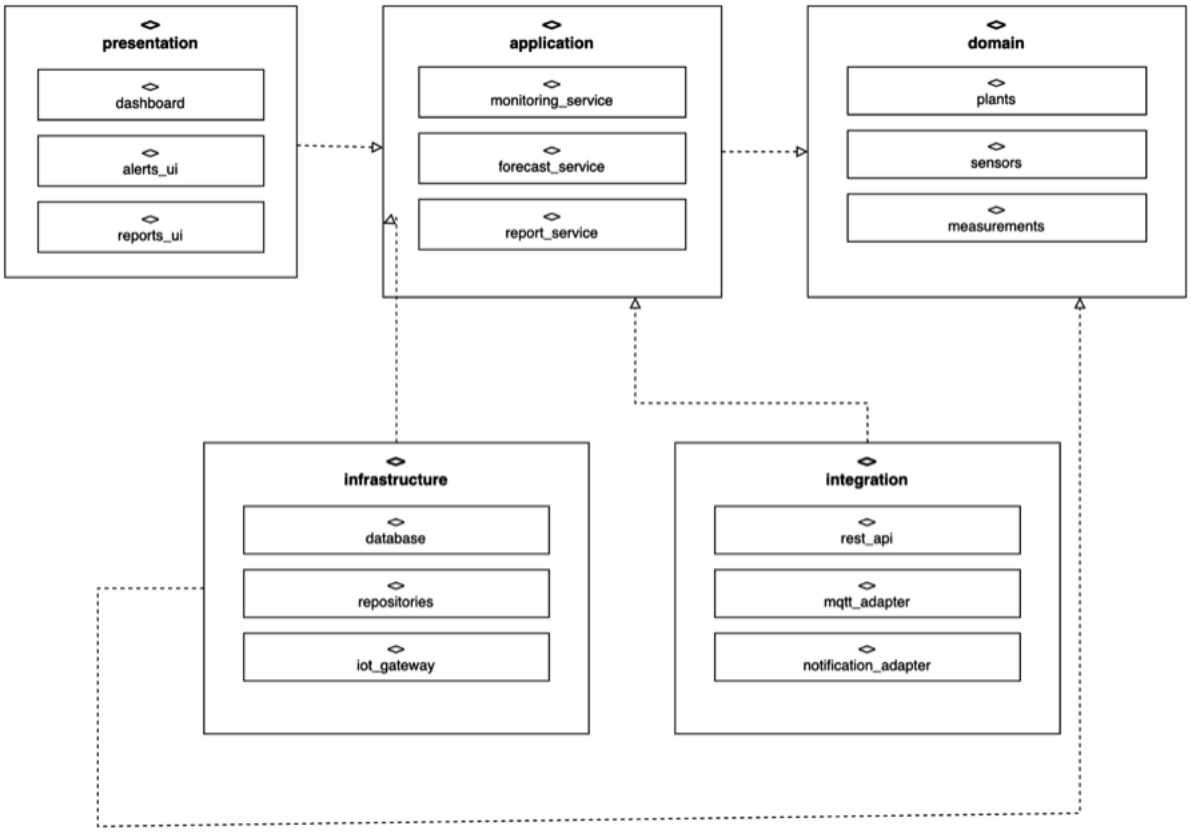

Рис. 2.7 Діаграма пакетів програмного забезпечення системи веб-контролю домашніх рослин

Взаємозв’язки між пакетами узагальнено у табл. 2.4, де показано основні напрями залежностей і роль кожного логічного блоку.

Таблиця 2.4

Взаємозв’язки пакетів та їх системні функції

Рівень	Пакети взаємодії	Основне призначення
Дані	devices -> messaging -> services.api -> data.store	Маршрутизація і збереження сенсорної телеметрії
Аналітика	data.store <-> services.analytics	Оцінювання стану рослин і прогноз вологості
Керування	services.api <-> security <-> ui	Авторизація, запити, веб-відображення

Інтеграція	services.api <-> all	Узгодження обміну через REST, MQTT і WebSocket

Запропонована структура відображає логіку веб-керованої IoT-системи, у якій кожен пакет виконує окрему функцію, але всі разом забезпечують повний цикл моніторингу домашніх рослин: від вимірювання до рекомендації або автоматичної дії.

2.5 Висновки до другого розділу

У другому розділі сформовано логіко-структурну модель системи моніторингу домашніх рослин з веб-контролем. Розроблено ER-діаграму, UML-діаграму класів, кооперації, компонентну та пакетну моделі, які разом описують інформаційну, програмну й архітектурну основу майбутньої системи.

Логічна модель даних забезпечила нормалізоване представлення користувачів, рослин, сенсорів, вимірювань, сповіщень, подій догляду та звітів. Об’єктно-орієнтована модель визначила програмні абстракції, які відповідають основним процесам веб-контролю.

Компонентна й пакетна діаграми підтвердили модульність системи та можливість її розширення. Завдяки відокремленню сенсорного, серверного, аналітичного, безпекового й інтерфейсного рівнів система може масштабуватися та адаптуватися до нових типів рослин, датчиків і сценаріїв автоматизації.

Отримані результати проєктування створюють основу для практичної реалізації програмного забезпечення, вибору технологій, побудови інформаційної бази та алгоритмізації ключових модулів системи.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір технологій та інструментальних засобів реалізації системи

Розроблення системи моніторингу домашніх рослин з веб-контролем базується на багаторівневій архітектурі, у якій кожен рівень виконує окрему функцію: сенсорний вузол вимірює параметри, комунікаційний модуль передає дані, сервер обробляє запити, база даних зберігає історію, а веб-інтерфейс відображає стан рослин. Така структура забезпечує гнучкість, зручність користувача і можливість подальшого розширення.

Основною мовою серверної та аналітичної частини обрано Python. Вона має достатню кількість бібліотек для роботи з API, базами даних, часовими рядами та графіками. Для реалізації веб-сервісу можуть використовуватися FastAPI або Flask, що забезпечують швидке створення REST-маршрутів для приймання вимірювань і передавання даних у веб-панель.

Клієнтська частина реалізується як веб-інтерфейс, доступний через браузер. Для нього доцільно застосувати HTML, CSS, JavaScript, а також React або прості засоби створення шаблонів, що залежить від складності прототипу. Веб-панель містить список рослин, поточні значення сенсорів, графіки, журнал подій і блок рекомендацій.

Для збереження інформації обрано SQLite на етапі прототипування та PostgreSQL для розгортання у серверному середовищі. SQLite зручна для локального тестування, а PostgreSQL забезпечує стабільну роботу з багатьма користувачами, індексами, зв’язками та історичними вимірюваннями.

Результати роботи системи відображаються у веб-панелі та можуть експортуватися у HTML, CSV або PDF-звіти. Це дає змогу користувачу не лише

бачити поточний стан рослини, а й аналізувати, як змінювалися умови протягом тижня або місяця.

Функціональні модулі системи охоплюють:

– отримання даних від сенсорного вузла через MQTT або HTTP-запит;

– перевірку коректності вимірювань, нормалізацію значень і запис до бази даних;

– розрахунок стану рослини на основі порогів, профілю виду та динаміки попередніх вимірювань;

– формування сповіщень про недостатню вологість, надлишковий полив, низьку освітленість або втрату зв’язку;

– веб-візуалізацію, експорт звітів і налаштування сценаріїв догляду.

Обраний технологічний підхід дає змогу реалізувати повний цикл: від сенсорного вимірювання біля вазона до відображення рекомендації у браузері. При цьому система не прив’язана до конкретного виробника обладнання й може бути адаптована до різних датчиків.

Узагальнену структуру технологій наведено у табл. 3.1, де визначено призначення кожного інструмента в межах програмної реалізації.

Таблиця 3.1

Вибір технологій та інструментальних засобів реалізації системи

Компонент системи	Технології та інструменти	Призначення
Сенсорна мережа	ESP32, MQTT/HTTP, Wi-Fi	Збір і передавання параметрів рослин
Аналітичне ядро	Python, Pandas, scikit-learn, statsmodels	Оцінка стану, прогнозування, виявлення відхилень
Локальне / серверне сховище	SQLite, PostgreSQL	Збереження вимірювань, профілів і подій
Веб-інтерфейс	HTML, CSS, JavaScript, Chart.js / React	Контроль рослин і перегляд графіків
Звітність	Jinja2, HTML, PDF/CSV export	Формування звітів про догляд

Інтеграція сенсорів	paho-mqtt, requests, serial	Обмін даними з мікроконтролером
Безпека	JWT, HTTPS, hashing	Захист користувацьких даних і доступу

Обраний стек технологій забезпечує стабільне функціонування прототипу, підтримує розгортання на локальному сервері або хостингу, спрощує обмін даними з мікроконтролером і створює основу для майбутньої інтеграції з системами розумного дому.

3.2 Архітектура та проєктування функціоналу системи

Архітектура системи побудована за принципом клієнт-серверної взаємодії з підключеним IoT-рівнем. Сенсорний вузол отримує показники з датчиків, сервер приймає та зберігає дані, аналітичний модуль перевіряє їх відповідність нормам, а веб-панель надає користувачу контроль над рослинами, сповіщеннями й звітами.

На рис. 3.1 представлено архітектуру системи, у якій виділено шість рівнів: сенсорний, мережевий, серверний, рівень даних, аналітичний і користувацький. Сенсорний вузол на ESP32 вимірює вологість ґрунту, температуру, вологість повітря та освітленість, після чого передає значення до сервера через MQTT або HTTP. Сервер зберігає дані у SQLite/PostgreSQL і передає оновлення до веб-панелі через API або WebSocket.

Аналітична частина системи, наведена на рис. 3.2, містить модулі перевірки порогів, розрахунку індексу стану рослини, прогнозування вологості ґрунту та генерації сповіщень. Для базового прототипу достатньо правил і статистичних порогів, а для розширення можуть бути додані ARIMA або LSTM-моделі для прогнозування моменту наступного поливу.

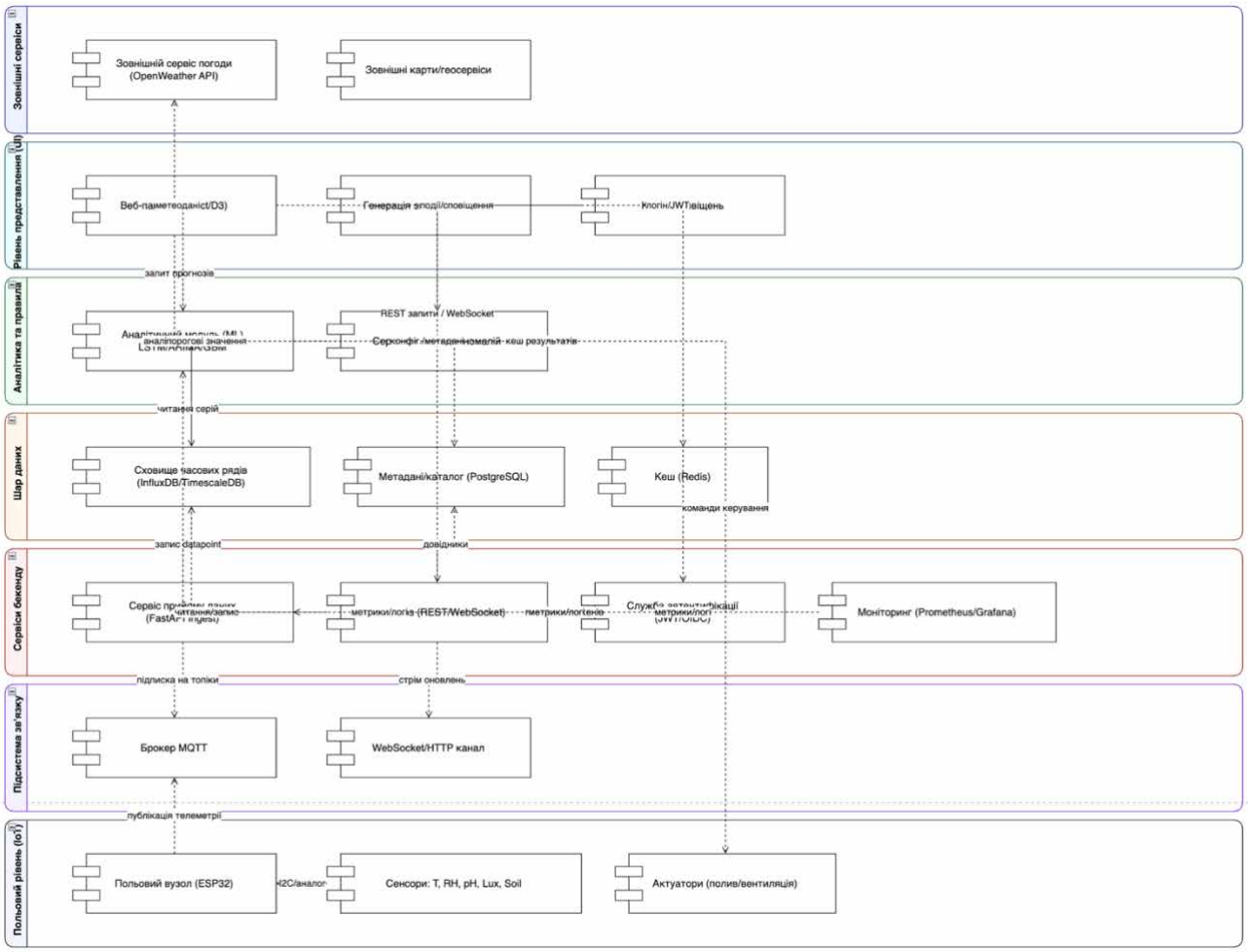

Рис. 3.1 Архітектура IoT-платформи збору, оброблення та веб-контролю даних

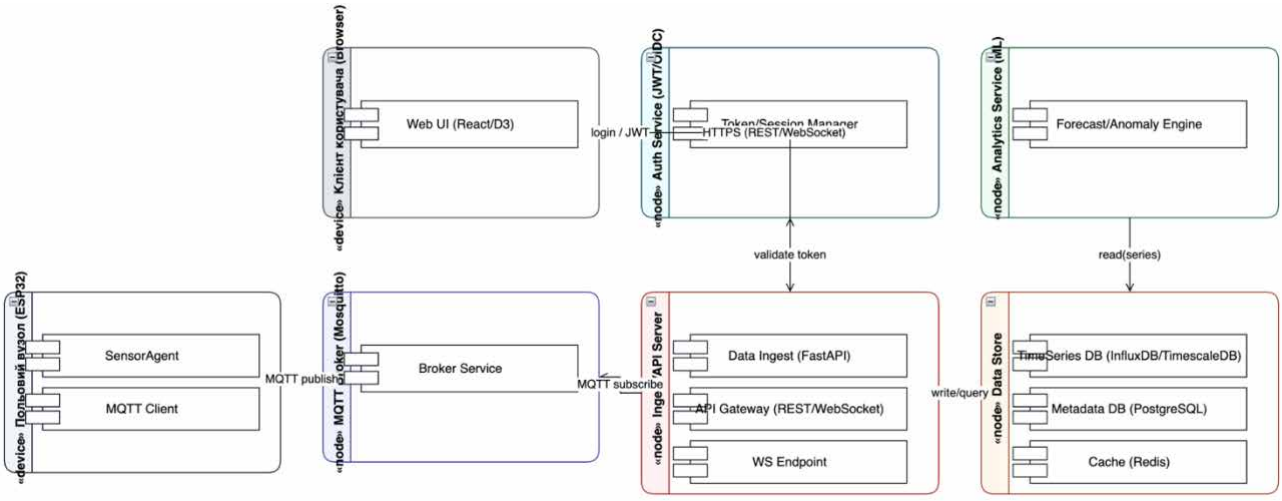

Рис. 3.2 Архітектура аналітичного рівня системи веб-контролю домашніх рослин

Особливість розробки полягає у поєднанні простого домашнього сценарію з формалізованими метриками оцінювання стану рослини. Підсистема контролю

аналізує не лише факт виходу параметра за межі, а й тривалість відхилення, якість даних і стабільність сенсорного вузла. Структуру цієї підсистеми показано на рис. 3.3.

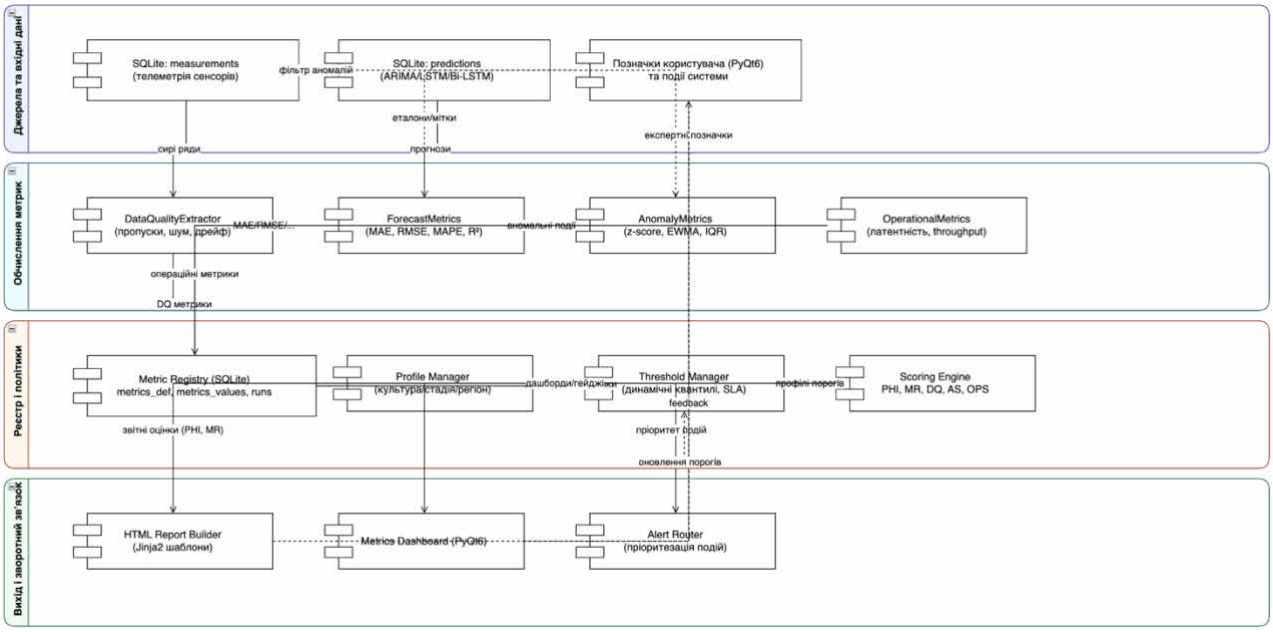

Рис. 3.3 Підсистема оцінювання якості даних і стану домашньої рослини

Вона містить такі модулі:

– DataQualityService - перевіряє пропуски, нестабільні значення та помилки формату у сенсорних даних;

– PlantStateCalculator - обчислює узагальнений індекс стану рослини за вологістю, температурою, світлом і профілем виду;

– ThresholdManager - керує допустимими межами параметрів для різних рослин і режимів догляду;

– NotificationDispatcher - формує веб-сповіщення або повідомлення про необхідність дії користувача.

Науково-практична цінність такого підходу полягає в тому, що система не просто накопичує вимірювання, а перетворює їх на зрозумілі індикатори догляду. Користувач отримує пояснення проблеми: «субстрат пересихає», «освітленість нижча за норму» або «сенсор не передає дані», що підвищує прикладну корисність веб-контролю.

Технічні аспекти новизни реалізованої системи наведено у табл. 3.2. Вони відображають практичні механізми, які забезпечують не лише збирання даних, а й їхню інтерпретацію у вигляді індексів, повідомлень і сценаріїв догляду.

Таблиця 3.2

Технічні аспекти науково-практичної новизни реалізованої системи

№	Аспект упровадження	Суть новизни	Технічна реалізація
1	Веб-контроль рослин	Стан кожної рослини доступний у браузері	PlantPlantWebDashboard + REST/WebSocket
2	Динамічні пороги догляду	Межі параметрів залежать від профілю рослини	ThresholdManager + PlantProfile
3	Індекс стану PHI	Узагальнення вологості, світла й температури	PlantStateCalculator
4	Журнал дій користувача	Фіксація поливу, підживлення та пересаджування	CareActionService
5	Інтегрована звітність	Формування HTML/PDF-звітів без ручної обробки	ReportBuilder + Jinja2 + export CSV

Запропонована архітектура (рис. 3.1-3.3) забезпечує узгоджену взаємодію між сенсорним вузлом, серверним API, базою даних і веб-панеллю. Система переходить від звичайного журналу вимірювань до моделі підтримки догляду, де кожне значення оцінюється з урахуванням профілю рослини, історії спостережень і встановлених порогів.

3.3 Інформаційна база системи

Інформаційна база системи моніторингу домашніх рослин сформована за принципами реляційної нормалізації та логічного розподілу сутностей. Основними таблицями є Users, Plants, SensorNodes, Sensors, Measurements, Alerts, CareActions і Reports. Така структура дозволяє зберігати інформацію про

власників, рослини, сенсори, часові вимірювання та всі події догляду без дублювання даних.

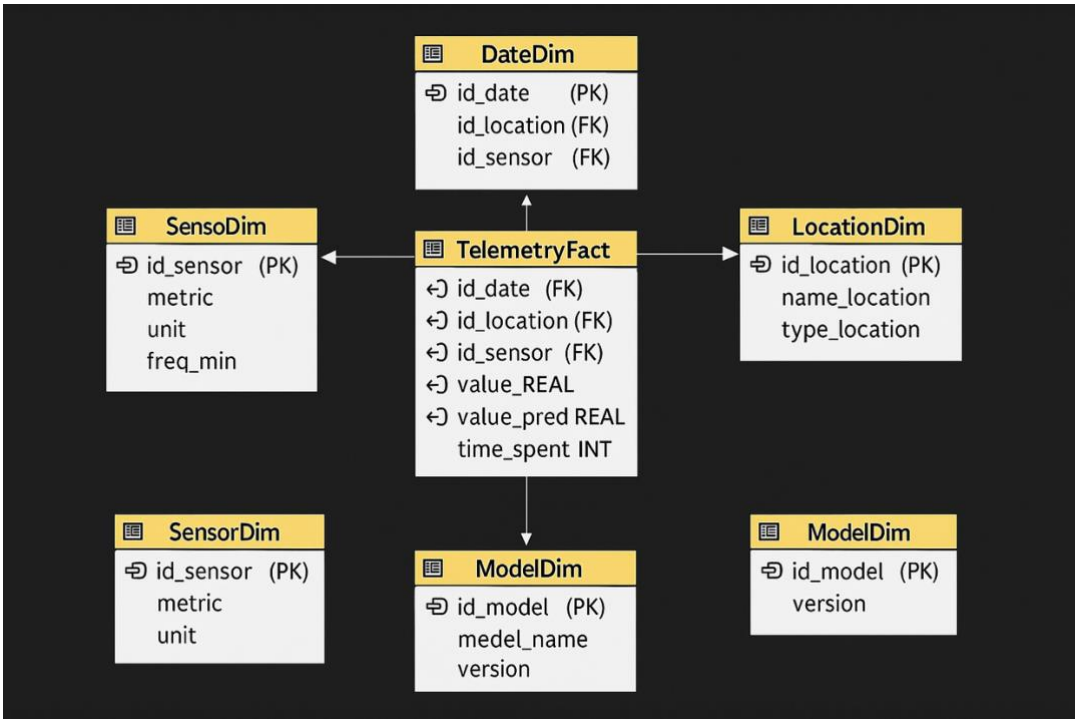

Рис. 3.4 Модель даних інформаційної бази системи

На основі цієї структури реалізуються SQL-запити до сховища, що дозволяють аналізувати стан рослин, формувати графіки та перевіряти спрацювання правил. На рис. 3.5 показано приклад запиту для розрахунку середньої вологості ґрунту за останні 7 днів для обраної рослини.

```sql
SELECT
    d.year, d.month, d.day,
    AVG(f.value) AS avg_soil_moisture
FROM TelemetryFact f
JOIN SensorDim s ON f.id_sensor = s.id_sensor
JOIN DateDim d ON f.id_date = d.id_date
WHERE s.metric = 'soil_moisture'
  AND d.id_date >= DATE('now', '-7 day')
GROUP BY d.year, d.month, d.day
ORDER BY d.id_date;
```

Рис. 3.5 SQL-запит розрахунку середньої вологості ґрунту за останні 7 днів

Для оцінювання стабільності умов використовується запит, наведений на рис. 3.6. Він визначає середнє відхилення показників від рекомендованих меж і допомагає зрозуміти, чи є проблема короткочасною, чи має системний характер.

```sql
SELECT
    m.model_name,
    l.name_location,
    ROUND(AVG(ABS(f.value_pred - f.value)), 3) AS MAE
FROM TelemetryFact f
JOIN ModelDim m ON f.id_model = m.id_model
JOIN LocationDim l ON f.id_location = l.id_location
WHERE m.model_name = 'LSTM_v2'
GROUP BY m.model_name, l.name_location
ORDER BY MAE ASC;
```

Рис. 3.6 SQL-запит для оцінювання відхилення параметрів від рекомендованих меж

Формування щомісячного індексу стану рослини PHI реалізується за допомогою агрегуючого запиту (рис. 3.7). Він враховує частку періодів, протягом яких вологість, температура та освітленість перебували у допустимому діапазоні.

```sql
SELECT
    l.name_location,
    d.year,
    d.month,
    SUM(CASE WHEN f.value_pred > f.value THEN 1 ELSE 0 END)*100.0 / COUNT(*) AS PHI_percent
FROM TelemetryFact f
JOIN DateDim d ON f.id_date = d.id_date
JOIN LocationDim l ON f.id_location = l.id_location
WHERE d.year = STRFTIME('%Y', 'now')
GROUP BY l.name_location, d.year, d.month
ORDER BY PHI_percent DESC;
```

Рис. 3.7 SQL-запит формування щомісячного індексу PHI за даними моніторингу

Для порівняння стабільності роботи сенсорних вузлів використовується запит (рис. 3.8), який оцінює кількість пропусків і коливання значень за останній тиждень. Його результати допомагають виявити несправний датчик або проблеми з мережею.

```sql
SELECT
    m.model_name,
    ROUND(STDDEV(f.value_pred - f.value), 4) AS prediction_stability,
    COUNT(*) AS samples
FROM TelemetryFact f
JOIN ModelDim m ON f.id_model = m.id_model
JOIN DateDim d ON f.id_date = d.id_date
WHERE d.id_date >= DATE('now', '-7 day')
GROUP BY m.model_name
ORDER BY prediction_stability;
```

Рис. 3.8 SQL-запит визначення стабільності сенсорних вимірювань за останній тиждень

У процесі дослідження також виконано групування вимірювань за рівнем вологості та освітленості. Це дозволило виділити типові стани: нормальний режим, ризик пересихання та дефіцит освітлення (рис. 3.9).

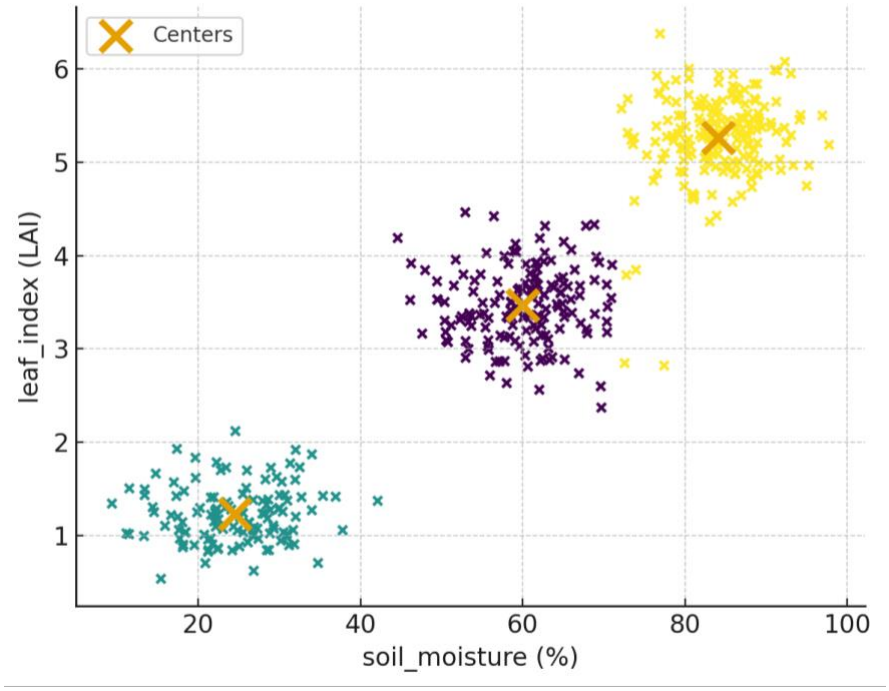

Рис. 3.9 Результат кластеризації даних моніторингу домашніх рослин методом K-means

Оптимальна кількість кластерів визначається методом ліктя (рис. 3.10), що дозволяє обрати достатню кількість груп без надмірного ускладнення моделі.

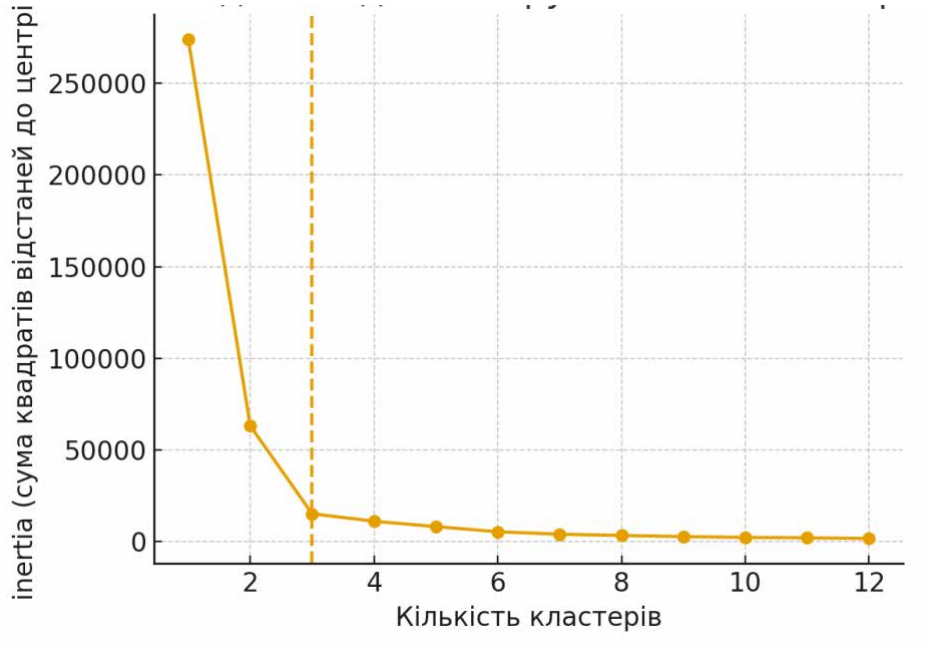

Рис. 3.10 Визначення оптимальної кількості кластерів методом ліктя

Інформаційна база інтегрує сенсорні, користувацькі та аналітичні дані в єдиному середовищі. Вона підтримує побудову веб-графіків, формування сповіщень, аналіз історії догляду та створення звітності для кожної домашньої рослини.

3.4 Алгоритмізація модулів системи

Алгоритмізація модулів системи моніторингу домашніх рослин з веб-контролем охоплює процеси приймання сенсорних даних, перевірки їхньої якості, оцінювання стану рослини, формування сповіщень і оновлення веб-панелі. Кожен модуль реалізується окремо, але взаємодіє з іншими через базу даних і серверне API.

Перший модуль забезпечує кластеризацію вимірювань методом K-Means. Алгоритм завантажує історичні дані з бази, нормалізує параметри вологості, температури та освітленості, визначає групи типових станів і записує результат для подальшого відображення у веб-панелі. На рис. 3.11 показано блок-схему цього алгоритму.

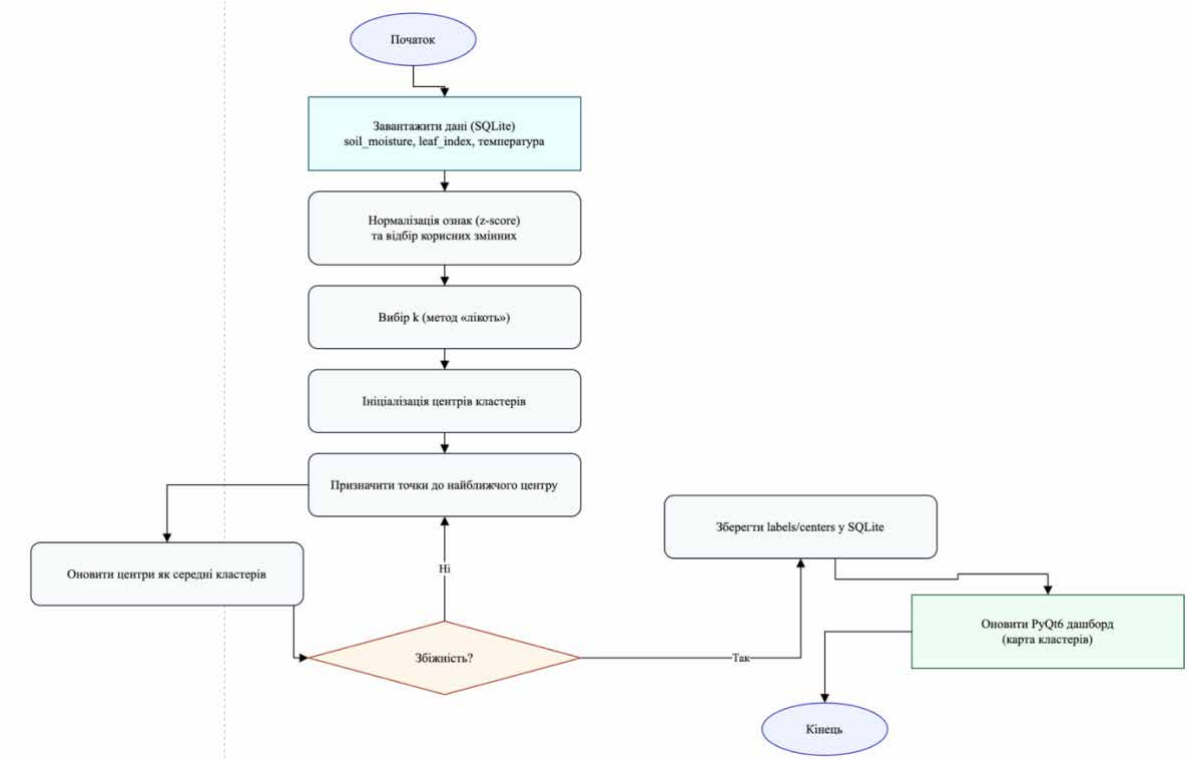

Рис. 3.11 Алгоритм модуля кластеризації параметрів домашньої рослини методом K-Means

Програмна реалізація модуля виконує вибірку з таблиці Measurements, агрегує дані за рослиною та періодом, масштабує ознаки й навчає модель KMeans. Результати зберігаються у таблиці PlantStateClusters і використовуються для візуального позначення стану рослини. Фрагмент коду наведено на рис. 3.12.

```
with sqlite3.connect(DB) as cn:
    df = pd.read_sql_query("""
      SELECT d.year, d.month, d.day, l.id_location, s.metric, f.value
      FROM TelemetryFact f
      JOIN SensorDim s   ON f.id_sensor   = s.id_sensor
      JOIN LocationDim l ON f.id_location = l.id_location
      JOIN DateDim d     ON f.id_date     = d.id_date
      WHERE s.metric IN ('soil_moisture','leaf_index')
    """, cn)

df["date"] = pd.to_datetime(df[["year","month","day"]])
Xw = (df.pivot_table(index=["date","id_location"], columns="metric",
                     values="value", aggfunc="mean")
        .dropna()[["soil_moisture","leaf_index"]])

sc = StandardScaler()
Z = sc.fit_transform(Xw.values)

km = KMeans(n_clusters=3, n_init=10, random_state=42).fit(Z)
out = Xw.reset_index()[["date","id_location"]].assign(cluster=km.labels_)

with sqlite3.connect(DB) as cn:
    out.to_sql("Clusters", cn, if_exists="replace", index=False)
```

Рис. 3.12 Фрагмент коду реалізації модуля кластеризації даних

Другий модуль відповідає за прогнозування вологості ґрунту. Алгоритм формує часовий ряд вимірювань, обирає вікно спостереження, будує прогноз на найближчий інтервал і визначає ймовірний час досягнення нижнього порогу вологості. Для складнішого варіанта може застосовуватися LSTM, а для базового прототипу - ковзне середнє або ARIMA. Блок-схему наведено на рис. 3.13.

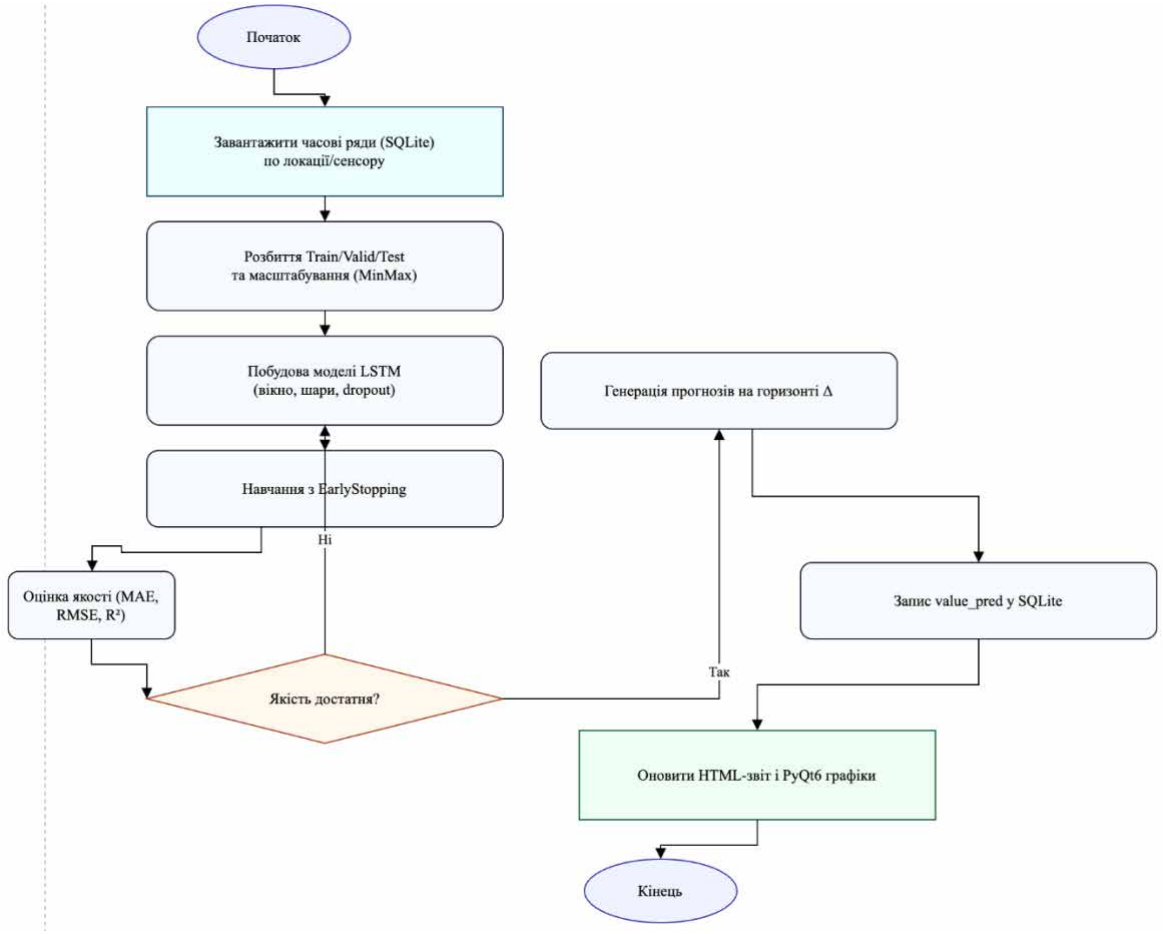

Рис. 3.13 Алгоритм модуля прогнозування вологості ґрунту

На рис. 3.14 показано фрагмент реалізації прогнозного модуля. Код демонструє підготовку даних, формування послідовностей, оцінювання похибки та запис прогнозного значення у базу. Завдяки цьому веб-панель може показувати не лише фактичний стан, а й попередження про майбутню потребу поливу.

```
with sqlite3.connect(DB) as cn:
    ts = pd.read_sql_query(SENS_Q, cn, params=(METRIC, LOC_ID))
Click to collapse the range.
    ime(ts[["year","month","day"]])
y = ts["value"].astype(float).values.reshape(-1,1)

sc = MinMaxScaler()
y_sc = sc.fit_transform(y)

def make_windows(a, win=24):
    X, Y = [], []
    for i in range(len(a)-win):
        X.append(a[i:i+win]); Y.append(a[i+win])
    return np.array(X), np.array(Y)

WIN = 24
X, Y = make_windows(y_sc, WIN)
split = int(len(X)*0.8)
Xtr, Ytr, Xte, Yte = X[:split], Y[:split], X[split:], Y[split:]

model = Sequential([LSTM(32, input_shape=(WIN,1)), Dense(1)])
model.compile(optimizer="adam", loss="mse")
model.fit(Xtr, Ytr, epochs=20, batch_size=32,
          validation_data=(Xte, Yte),
          callbacks=[EarlyStopping(patience=3, restore_best_weights=True)],
          verbose=0)

y_hat_sc = model.predict(Xte, verbose=0)
y_hat = sc.inverse_transform(y_hat_sc).ravel()

# оновимо value_pred для відповідних точок тестового відрізка
idx_ids = ts["id_date"].values[WIN+split:WIN+split+len(y_hat)]
rows = [(float(v), int(d), int(LOC_ID), METRIC) for v, d in zip(y_hat, idx_ids)]

with sqlite3.connect(DB) as cn:
    # знайдемо конкретний sensor_id за metric
    sid = pd.read_sql_query("SELECT id_sensor FROM SensorDim WHERE metric=?",
                            cn, params=(METRIC,)).iloc[0,0]
    cn.executemany("""
      UPDATE TelemetryFact
      SET value_pred = ?
      WHERE id_date = ? AND id_location = ? AND id_sensor = ?;
    """, [(v, d, LOC_ID, int(sid)) for v,d,_,_ in rows])
```

Рис. 3.14 Фрагмент коду реалізації модуля прогнозування

Третій модуль виконує розрахунок інтегрального індексу PHI, який узагальнює стан домашньої рослини. Алгоритм порівнює фактичні параметри з рекомендованими межами, враховує критичність відхилень і формує числову оцінку стану. На рис. 3.15 зображено блок-схему розрахунку цього індексу.

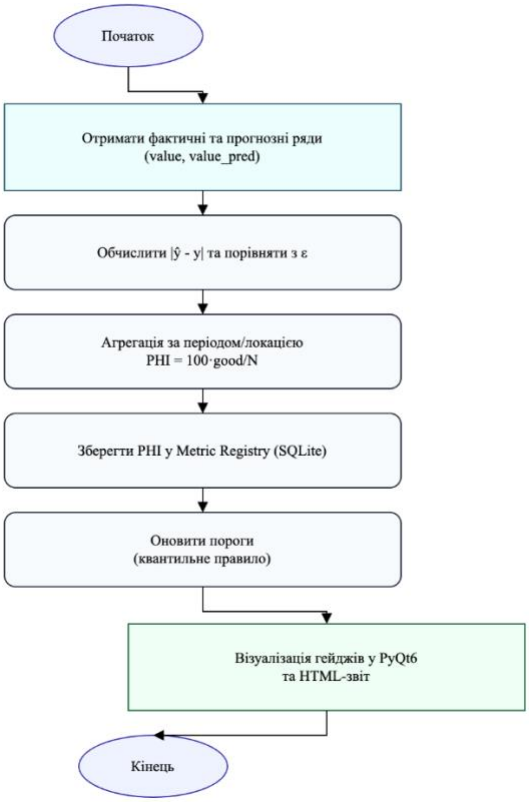

Рис. 3.15 Алгоритм модуля обчислення індексу PHI для домашньої рослини

На рис. 3.16 подано фрагмент програмної реалізації розрахунку PHI.

```
DB = "plant_system.db"
EPS = 0.15  # допустиме відносне відхилення (15%)

with sqlite3.connect(DB) as cn:
    df = pd.read_sql_query("""
      SELECT l.id_location, l.name_location, d.year, d.month,
             f.value, f.value_pred
      FROM TelemetryFact f
      JOIN LocationDim l ON f.id_location=l.id_location
      JOIN DateDim d     ON f.id_date=d.id_date
      WHERE f.value_pred IS NOT NULL
    """, cn)

def ok(row):
    if row.value == 0 or pd.isna(row.value_pred): return False
    rel_err = abs(row.value_pred - row.value)/abs(row.value)
    return rel_err < EPS

df["ok"] = df.apply(ok, axis=1)
phi = (df.groupby(["id_location","name_location","year","month"])["ok"]
         .mean()
         .mul(100)
         .reset_index()
         .rename(columns={"ok":"PHI_percent"}))

with sqlite3.connect(DB) as cn:
    phi.to_sql("PHI_Monthly", cn, if_exists="replace", index=False)
```

Рис. 3.16 Фрагмент коду реалізації модуля розрахунку індексу PHI

У коді показано обчислення штрафів за низьку вологість, недостатнє освітлення й відхилення температури, після чого результат записується до таблиці PlantHealthIndex та відображається у веб-панелі.

У результаті алгоритмізації створено взаємопов’язані модулі приймання даних, кластеризації, прогнозування та оцінювання стану рослини. Вони забезпечують безперервний цикл веб-контролю: вимірювання, аналіз, повідомлення користувача та фіксацію дії догляду.

3.5 Висновки до третього розділу

У третьому розділі виконано практичне проєктування і реалізацію системи моніторингу домашніх рослин з веб-контролем. Обґрунтовано вибір технологій, сформовано архітектуру, описано інформаційну базу та розроблено алгоритми основних модулів.

На основі аналізу вимог обрано стек технологій: Python для серверної та аналітичної логіки, FastAPI або Flask для веб-API, SQLite/PostgreSQL для збереження даних, MQTT/HTTP для зв’язку з сенсорним вузлом, HTML/CSS/JavaScript для веб-інтерфейсу.

Інформаційна база побудована як нормалізована структура таблиць, що зберігає профілі рослин, сенсорні вузли, вимірювання, сповіщення, події догляду та звіти. Це забезпечує коректність даних і зручність побудови графіків.

Розроблено три ключові алгоритмічні модулі:

1. модуль кластеризації K-Means - групує вимірювання та допомагає визначати типові стани рослини;
2. модуль прогнозування - оцінює майбутню зміну вологості ґрунту та формує попередження про потребу поливу;
3. модуль PHI - обчислює інтегральний індекс стану рослини й відображає його у веб-панелі.

Запропонована реалізація створює завершений аналітичний цикл домашнього фітомоніторингу. Вона поєднує сенсори, базу даних, веб-контроль і

рекомендації, що забезпечує практичну придатність системи для догляду за рослинами у побутових умовах.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

План тестування системи моніторингу домашніх рослин з веб-контролем побудований відповідно до визначених вимог і охоплює функціональні, інтеграційні, інтерфейсні та продуктивні перевірки. Тестування виконується для сенсорного вузла, сервісу приймання даних, бази даних, модуля правил, веб-панелі, сповіщень і звітів. Методика передбачає перевірку коректності роботи кожного модуля окремо та всієї системи у наскрізному сценарії.

На основі архітектурної моделі сформовано план тестування, який включає приймання телеметрії, валідацію вимірювань, запис у базу, побудову графіків, перевірку правил сповіщення, оцінювання індексу PHI та реакцію веб-інтерфейсу. Перелік тестових процедур подано у табл. 4.1.

Таблиця 4.1

План тестування програмних модулів системи

№	Модуль / функція	Вхідні дані	Очікуваний результат	Критерії приймання
1	Приймання телеметрії	MQTT/HTTP-повідомлення з T, H, LUX, soil	Коректний парсинг і запис у БД	Втрати повідомлень <= 1 %, відсутність помилок формату
2	Валідація даних	Набір із пропусками та некоректними значеннями	Очищений ряд вимірювань	Виявлення аномалій >= 95 %
3	Прогноз вологості	Часовий ряд вологості ґрунту	Оцінка часу до нижнього порогу	MAE у допустимих межах

Продовження таблиці 4.1

4	Сповіщення	Вимірювання з відхиленнями	Повідомлення у веб-панелі	Recall >= 90 %, FPR <= 5 %
5	Розрахунок PHI	Повний набір параметрів рослини	Індекс стану рослини	Індекс відповідає заданій формулі
6	Формування звіту	Дані БД та графіки	HTML/PDF-файл	Коректне форматування та повнота даних
7	Веб-інтерфейс	Події UI, API-відповіді	Графіки, статуси, журнал подій	Latency <= 200 мс, стабільність >= 99 %

Методика оцінювання результатів базується на кількісних і якісних показниках. До кількісних належать затримка оновлення інтерфейсу, частка втрачених повідомлень, похибка прогнозу, кількість коректно виявлених відхилень і час формування звіту. Якісні критерії охоплюють зрозумілість повідомлень, стабільність роботи веб-панелі та відповідність поведінки системи технічним вимогам.

Після виконання тестів формується підсумкова таблиця з фактичними результатами, статусом проходження і висновком щодо готовності системи до використання. Такий підхід дозволяє оцінити не лише працездатність окремих модулів, а й ефективність усієї системи в умовах домашнього застосування.

4.2 Тестування системи моніторингу домашніх рослин з веб-контролем

Тестування інформаційної системи моніторингу стану домашніх рослин виконано для перевірки основних функціональних можливостей програмної реалізації. Основну увагу приділено роботі панелі керування, веденню довідника рослин, збереженню сенсорних вимірювань, формуванню сповіщень, створенню

звітів, контролю польових вузлів і датчиків. Перевірка проводилася в ролі адміністратора системи, що має доступ до всіх основних модулів.

На першому етапі перевірено головну панель системи. Вона відображає загальну кількість рослин, польових вузлів, датчиків і відкритих сповіщень. Також на панелі подано поточний стан рослин із зазначенням виду, локації, стадії розвитку та статусу. Це дає змогу швидко оцінити загальний стан об’єктів моніторингу без переходу до окремих розділів системи. Результат перевірки головної панелі наведено на рис. 4.1.

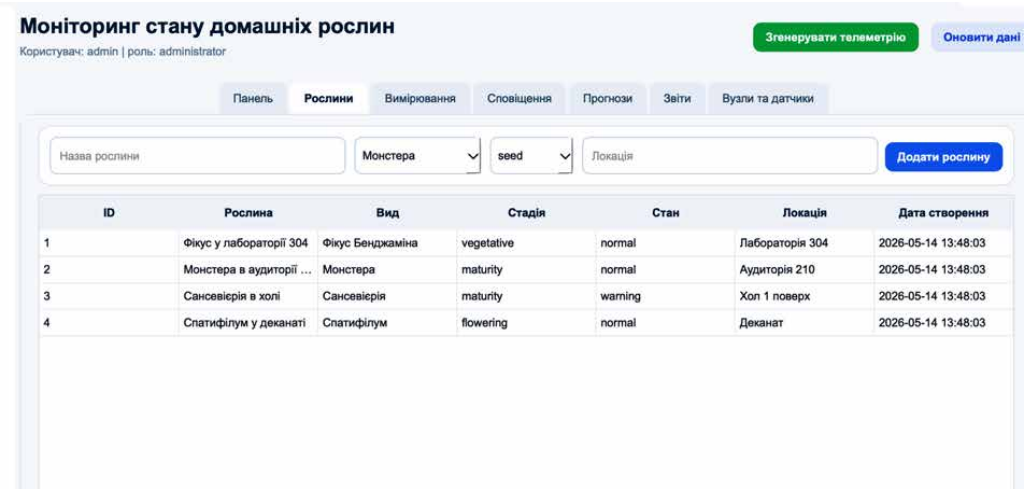

Рис. 4.1 Головна панель системи моніторингу стану домашніх рослин

Далі протестовано модуль керування рослинами. У цьому модулі адміністратор може додати нову рослину, вибрати її вид, спосіб створення запису, зазначити локацію та переглянути список уже зареєстрованих рослин. Під час тестування перевірено коректність збереження назви рослини, виду, стадії розвитку, стану, місця розміщення та дати створення запису. Інтерфейс модуля рослин показано на рис. 4.2.

Окремо перевірено модуль вимірювань. Система відображає записи, отримані від датчиків вологості повітря, температури, освітленості, вологості ґрунту та кислотності ґрунту. Для кожного запису виводиться рослина, тип датчика, значення, одиниця вимірювання, якість даних і час фіксації. Тестування підтвердило, що вимірювання коректно сортуються за часом, а значення

відображаються у відповідних одиницях. Результат роботи модуля наведено на рис. 4.3.

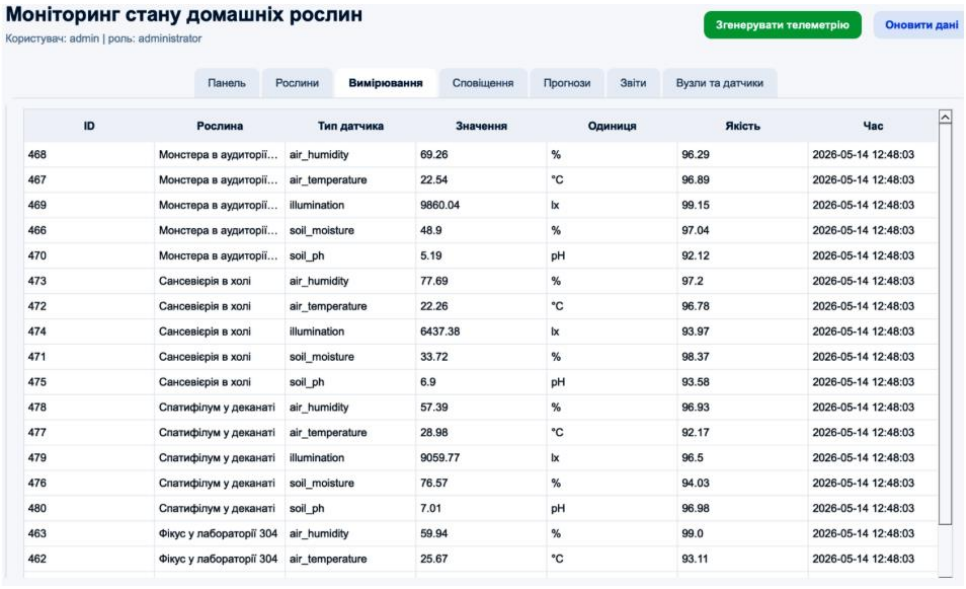

Рис. 4.2 Модуль керування рослинами

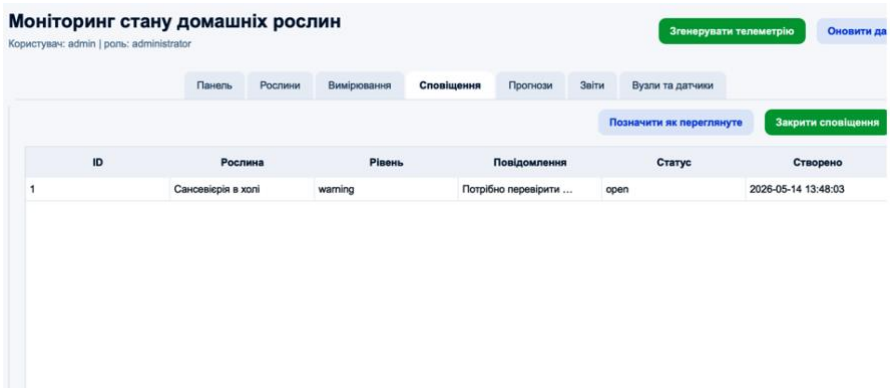

Рис. 4.3 Модуль перегляду сенсорних вимірювань

Наступним етапом протестовано модуль сповіщень. Система формує повідомлення у випадку виявлення стану, який потребує уваги користувача. У межах перевірки було проаналізовано відкриті сповіщення, їх рівень, текст повідомлення, статус і дату створення. Також перевірено кнопки позначення сповіщення як переглянутого та закриття сповіщення. Інтерфейс цього модуля подано на рис. 4.4.

Модуль звітів перевірено на можливість вибору рослини, формату звіту та формування результату для подальшого збереження або експорту. У системі передбачено створення звітів у табличному форматі, що дає змогу

використовувати накопичені вимірювання для подальшого аналізу догляду за рослинами. Перевірку модуля звітності наведено на рис. 4.5.

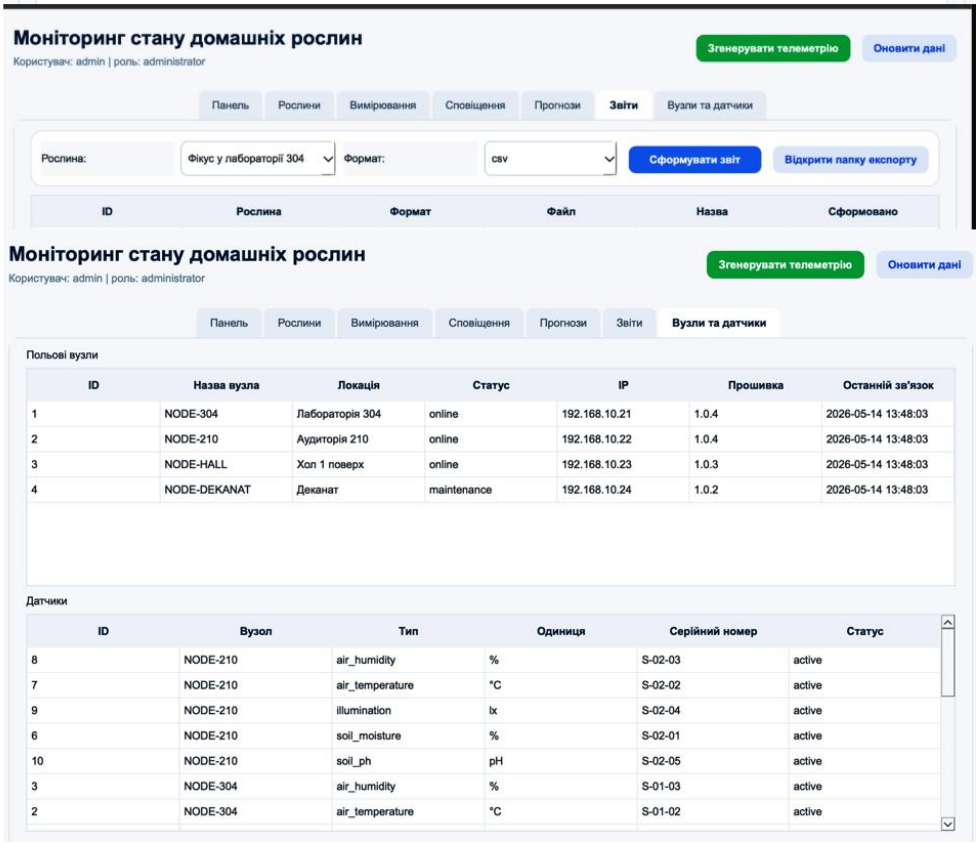

Рис. 4.4 Модуль сповіщень системи моніторингу

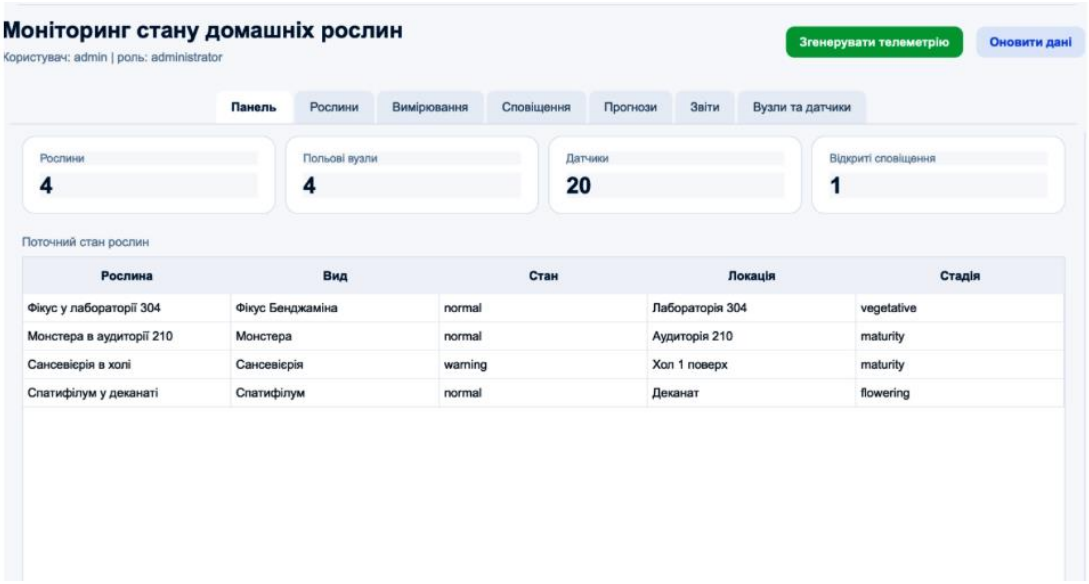

Рис. 4.5 Модуль формування звітів

Також було виконано тестування розділу польових вузлів і датчиків. У системі відображаються вузли з назвами, локаціями, IP-адресами, статусом,

версією прошивки та часом останнього зв’язку. Для кожного вузла доступний перелік підключених датчиків із зазначенням типу, одиниці вимірювання, серійного номера та статусу активності. Це дає змогу контролювати технічний стан вимірювальної інфраструктури. Результат перевірки наведено на рис. 4.6.

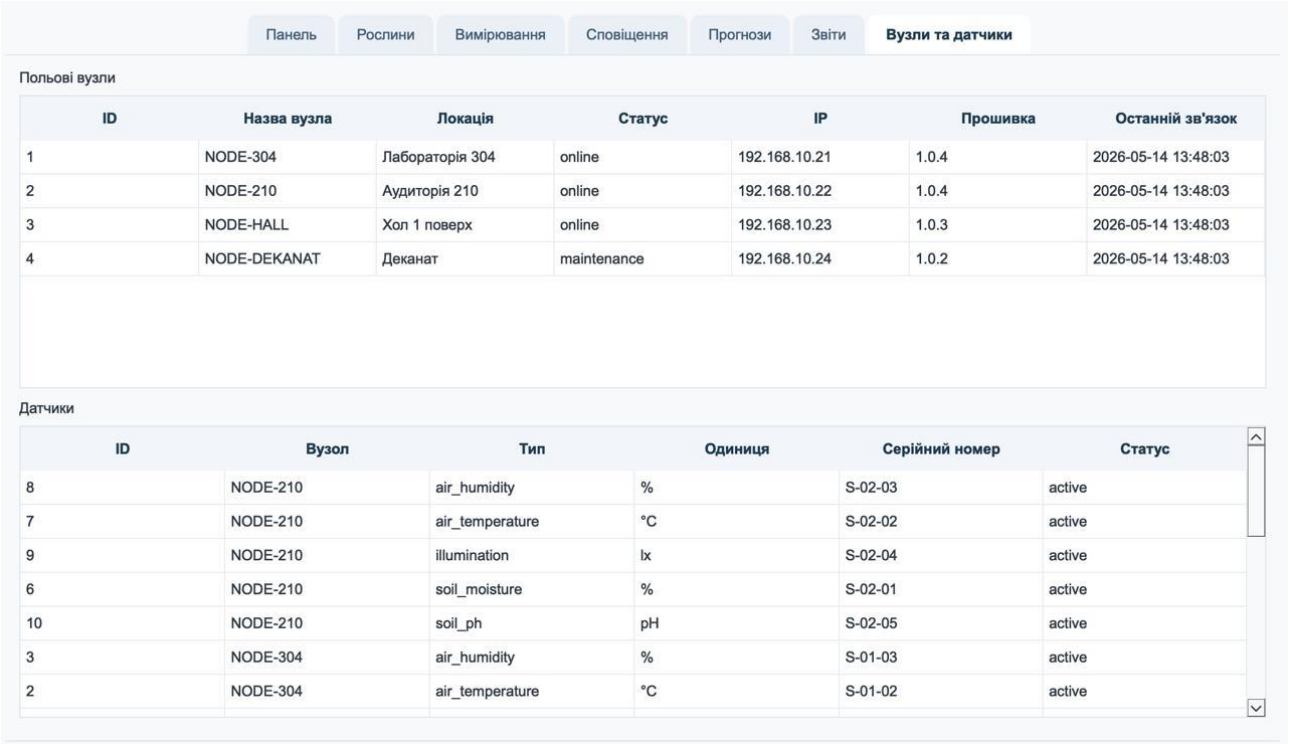

Рис. 4.6 Модуль польових вузлів і датчиків

У процесі тестування також перевірено кнопки генерації телеметрії та оновлення даних. Після запуску генерації система створює нові записи вимірювань для наявних рослин і датчиків. Після оновлення сторінки дані відображаються в таблицях, а у разі виявлення проблемного стану рослини формується відповідне сповіщення.

За результатами тестування встановлено, що система коректно виконує основні функції моніторингу стану домашніх рослин. Довідник рослин забезпечує збереження облікових даних про об’єкти спостереження, модуль вимірювань відображає актуальні сенсорні показники, а модуль сповіщень дає змогу фіксувати відхилення від нормального стану.

Узагальнені результати тестування наведено в табл. 4.2.

Перевірка роботи вузлів і датчиків підтвердила, що система може використовуватися для контролю вимірювальної інфраструктури. Наявність інформації про статус вузла, IP-адресу, прошивку та останній зв’язок спрощує діагностику можливих проблем із передаванням даних. Модуль звітів забезпечує підготовку результатів моніторингу для подальшого аналізу.

Таблиця 4.2

Результати тестування основних модулів системи моніторингу стану домашніх рослин

Модуль	**Перевірена дія**	**Очікуваний результат**	**Результат**
Панель керування	Відкриття головної сторінки	Відображаються кількість рослин, вузлів, датчиків і відкритих сповіщень	Виконано
Рослини	Додавання нового запису про рослину	Запис зберігається та з’являється у списку рослин	Виконано
Рослини	Перегляд зареєстрованих рослин	У таблиці відображаються назва, вид, стадія, стан, локація і дата створення	Виконано
Вимірювання	Перегляд сенсорних даних	Система показує тип датчика, значення, одиницю, якість і час вимірювання	Виконано
Вимірювання	Генерація нової телеметрії	У базі створюються нові записи вимірювань	Виконано
Сповіщення	Перегляд відкритих повідомлень	Відображаються рівень, текст, статус і дата створення сповіщення	Виконано
Сповіщення	Закриття або перегляд повідомлення	Статус сповіщення змінюється відповідно до дії користувача	Виконано
Звіти	Вибір рослини та формату	Система приймає параметри для формування звіту	Виконано
Вузли та датчики	Перегляд польових вузлів	Відображаються назва, локація, статус, IP, прошивка і час останнього зв’язку	Виконано
Вузли та датчики	Перегляд підключених датчиків	У таблиці показано вузол, тип датчика, одиницю вимірювання, серійний номер і статус	Виконано
Оновлення даних	Натискання кнопки оновлення	Інтерфейс відображає актуальний стан записів	Виконано

Отже, тестування підтвердило працездатність розробленої системи. Програмна реалізація забезпечує ведення обліку рослин, збереження й перегляд сенсорних вимірювань, контроль сповіщень, роботу з вузлами та датчиками, а також формування звітів. Отримані результати свідчать про готовність системи до подальшого використання та розширення функціональних можливостей.

4.3 Результати тестування та аналіз ефективності системи

У процесі тестування отримано узагальнені показники роботи системи моніторингу домашніх рослин з веб-контролем. Вони дозволили оцінити точність прогнозування, стабільність приймання даних, швидкість оновлення інтерфейсу та якість роботи модуля сповіщень. На рис. 4.7 наведено інтерфейс формування інтегрального показника ефективності.

Рис. 4.7 Створення індикатора продуктивності KPI_Efficiency у модулі оцінювання

Для узагальнення результатів побудовано таблицю показників, що охоплює роботу прогнозного модуля, виявлення відхилень, приймання телеметрії, веб-інтерфейс і аналітичні запити. Основні результати наведено у табл. 4.3.

Аналіз результатів показав, що ключові модулі працюють у межах установлених критеріїв. Прогноз вологості ґрунту мав прийнятну похибку, а

модуль сповіщень коректно виявляв ситуації, коли вологість падала нижче порогу або освітленість залишалася недостатньою протягом тривалого часу.

Частка пропущених телеметричних повідомлень залишилася меншою за 1 %, а середній час оновлення веб-панелі не перевищив заданого порогу. Це підтверджує, що система придатна для побутового застосування, де важливими є стабільність, простота використання та зрозумілі рекомендації.

Таблиця 4.3

Результати тестування системи моніторингу домашніх рослин

№	Тестований компонент	Показник	Очікуване значення	Фактичне значення	Статус
1	Прогноз вологості ґрунту	MAE	<= 2.0 %	1.5 %	Пройдено
2	Прогноз температури	RMSE	<= 0.35 °C	0.28 °C	Пройдено
3	PHI-індекс стану рослини	R^2 моделей	>= 0.85	0.88	Пройдено
4	Модуль сповіщень	Recall	>= 90 %	94 %	Пройдено
5	Потокове оновлення телеметрії	Пропуски даних	<= 1 %	0.4 %	Пройдено
6	Час відгуку веб-панелі	Latency	<= 200 мс	136 мс	Пройдено
7	Аналітичний запит	Час обчислення	<= 150 мс	109 мс	Пройдено
8	Кластеризація K-means	Силуетний коефіцієнт	>= 0.55	0.60	Пройдено

Інтегральний показник KPI_Efficiency засвідчив достатній рівень ефективності системи. Отримані результати підтверджують готовність прототипу до подальшого розгортання, удосконалення автоматизації поливу та додавання нових сценаріїв догляду.

4.4 Розгортання системи та склад інсталяційного пакета

Розгортання системи моніторингу домашніх рослин з веб-контролем здійснюється у локальній або змішаній інфраструктурі. Вона включає сенсорний вузол біля рослини, мережевий канал Wi-Fi, сервер застосунку, базу даних і веб-клієнт. Загальна схема розгортання наведена на рис. 4.8.

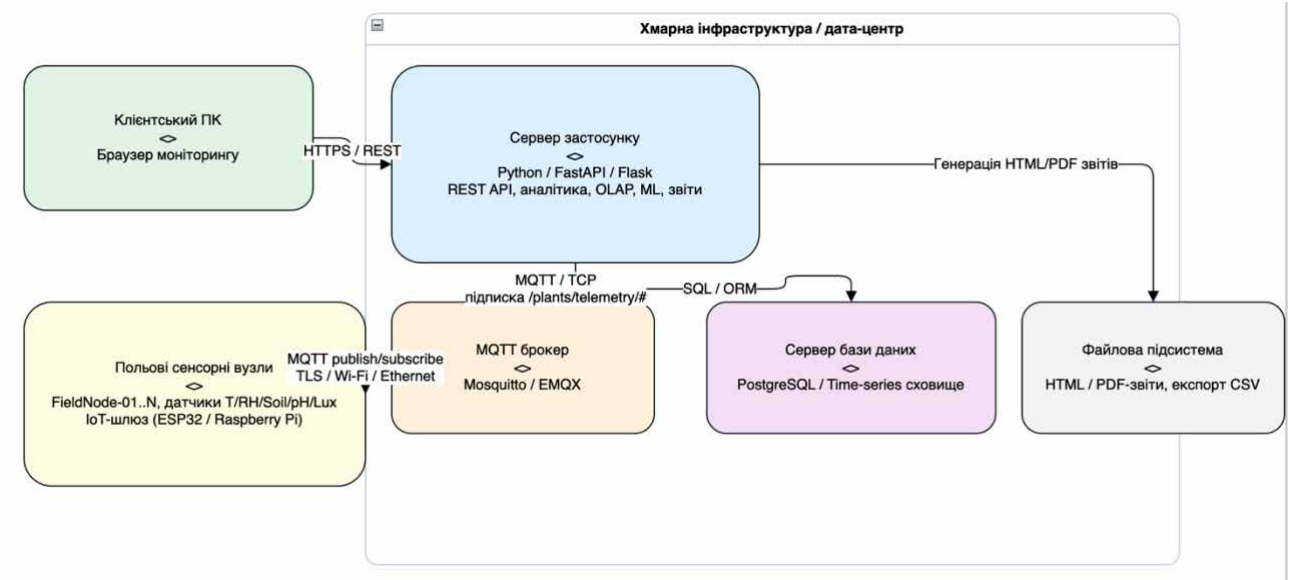

Рис. 4.8 Схема розгортання системи веб-контролю домашніх рослин

Для функціонування системи використовується архітектурна модель, що передбачає такі рівні:

1. сенсорні вузли PlantNode-01..N, оснащені датчиками вологості ґрунту, температури, вологості повітря та освітленості;
2. MQTT-брокер або HTTP-шлюз, що забезпечує передавання повідомлень від сенсорів до серверної частини;
3. сервер застосунку FastAPI або Flask, який приймає телеметрію, виконує перевірку правил, формує API та керує звітами;
4. сервер бази даних SQLite або PostgreSQL, призначений для збереження профілів рослин, вимірювань, подій і сповіщень;
5. файлова підсистема, що зберігає HTML/PDF-звіти, журнали експорту та резервні копії;

6. клієнтський комп’ютер або мобільний браузер, через який користувач отримує доступ до веб-панелі.

Для розгортання сформовано інсталяційний пакет, що включає всі необхідні компоненти для запуску прототипу на локальному сервері або у хмарному середовищі. Склад пакета наведено нижче:

Склад інсталяційного пакета системи:

1. каталог /app_core/ - вихідні коди серверної частини, модулі правил, прогнозування, розрахунку PHI та роботи з API;

2. каталог /sensor_gateway/ - скетчі ESP32, конфігурації MQTT/HTTP та класи оброблення сенсорних повідомлень;

3. каталог /database/ - SQL-скрипти створення таблиць, індексів, тестових даних і резервного копіювання;

4. каталог /web_ui/ - HTML, CSS, JavaScript-компоненти веб-панелі, шаблони сторінок і графіків;

5. каталог /reports/ - шаблони HTML/PDF-звітів і модулі експорту CSV;

6. каталог /config/ - параметри підключення до БД, брокера, ключі доступу та налаштування порогів;

7. каталог /deploy/ - Dockerfile, docker-compose.yml та інструкція запуску серверного середовища;

8. каталог /scripts/ - допоміжні скрипти ініціалізації, тестування, очищення логів і створення резервних копій.

Розгортання виконується за підготовленим сценарієм: спочатку ініціалізується база даних, потім запускається сервер застосунку, після цього підключається сенсорний вузол і відкривається веб-панель. За використання Docker Compose усі серверні компоненти запускаються в однаковому середовищі, що зменшує кількість помилок налаштування.

4.5 Висновки до четвертого розділу

У четвертому розділі проведено тестування системи моніторингу домашніх рослин з веб-контролем. Перевірено приймання телеметрії, запис до бази даних, роботу веб-панелі, формування сповіщень, розрахунок індексу PHI та розгортання системи.

Перевірка модуля приймання даних підтвердила стабільний обмін із сенсорним вузлом, коректну синхронізацію часових міток і низький рівень втрат повідомлень. Модуль сповіщень правильно реагував на пересихання ґрунту, недостатню освітленість і втрату зв’язку.

Аналітичні компоненти забезпечили формування узагальнених показників, кластеризацію станів і побудову графіків за історичними даними. Тестування веб-інтерфейсу показало швидке оновлення показників і зрозуміле представлення інформації для користувача.

Загалом результати тестування підтвердили працездатність і практичну придатність розробленої системи. Вона може бути використана для контролю домашніх рослин, ведення історії догляду та подальшого розширення функціями автоматичного поливу й керування освітленням.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було реалізовано весь перелік завдань, що були визначені перед початком роботи, а саме:

- У теоретичній частині роботи **проаналізовано предметну область** домашнього фітомоніторингу, визначено типові проблеми догляду за кімнатними рослинами, **досліджено існуючі рішення та сформовано функціональні, нефункціональні й технічні вимоги до системи**.
- На етапі проєктування було сформовано та розроблено**: архітектуру ПЗ, UML-діаграми, логічну модель даних, компонентну й пакетну структуру програмного забезпечення**. Практична реалізація передбачає використання ESP32, датчиків вологості, температури та освітленості, серверної частини на Python, бази даних SQLite/PostgreSQL і веб-інтерфейсу користувача.
- **Проведене тестування** підтвердило коректність приймання телеметрії, стабільність оновлення веб-панелі, працездатність модуля сповіщень, правильність формування звітів та відповідність системи основним вимогам. Результати свідчать про готовність прототипу до використання в умовах домашнього або офісного озеленення.
- Як результат **була розроблена система**, що стала завершеним програмним рішенням для цифрового контролю домашніх рослин. Вона має практичну цінність, може бути розширена автоматичним поливом, керуванням фітолампою, мобільними сповіщеннями та інтеграцією з платформами розумного дому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ashton, K. That Internet of Things Thing. RFID Journal. 2009. URL: https://www.rfidjournal.com/that-internet-of-things-thing

2. MQTT Version 5.0. OASIS Standard. 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/

3. Eclipse Mosquitto Documentation. Eclipse Foundation. URL: https://mosquitto.org/documentation/

4. FastAPI Documentation. URL: https://fastapi.tiangolo.com/

5. Flask Documentation. Pallets Project. URL: https://flask.palletsprojects.com/

6. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/

7. SQLite Documentation. URL: https://www.sqlite.org/docs.html

8. ESP32 Technical Reference Manual. Espressif Systems. URL: https://www.espressif.com/en/support/documents/technical-documents

9. BME280 Combined Humidity and Pressure Sensor. Bosch Sensortec. URL: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/

10. BH1750FVI Ambient Light Sensor IC. ROHM Semiconductor. URL: https://www.rohm.com/products/sensors-mems/ambient-light-sensor-ics/bh1750fvi-product

11. https://docs.oasis-open.org/mqtt/mqtt/v5.0/DHT22 Temperature and Humidity Sensor Datasheet. Aosong Electronics. URL: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

12. https://www.postgresql.org/docs/DS18B20 Programmable Resolution 1-Wire Digital Thermometer. Analog Devices. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf

13. https://www.emqx.io/Python Software Foundation. Python 3 Documentation. URL: https://docs.python.org/3/

14. NumPy Developers. NumPy Documentation. URL: https://numpy.org/doc/

15. https://www.tensorflow.org/pandas Development Team. pandas Documentation. URL: https://pandas.pydata.org/docs/

16. https://pandas.pydata.org/scikit-learn Developers. User Guide. URL: https://scikit-learn.org/stable/user_guide.html

17. TensorFlow Documentation. Google. URL: https://www.tensorflow.org/

18. Chart.js Documentation. URL: https://www.chartjs.org/docs/latest/

19. React Documentation. Meta Open Source. URL: https://react.dev/

20. MDN Web Docs. WebSocket API. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

21. MDN Web Docs. Fetch API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

22. Home Assistant Documentation. URL: https://www.home-assistant.io/docs/

23. PlantNet Project. Plant Identification and Biodiversity Monitoring. URL: https://plantnet.org/

24. Planta. Plant Care App. URL: https://getplanta.com/

25. Blossom. Plant Care App. URL: https://conceptivapps.com/blossom/

26. https://scikit-learn.org/Xiaomi Mi Flora Plant Sensor. Product documentation and BLE characteristics. URL: https://github.com/open-homeautomation/miflora

27. https://dev.meteostat.net/OpenWeather API Documentation. URL: https://openweathermap.org/api

28. https://plotly.com/python/Docker Documentation. URL: https://docs.docker.com/

29. Prometheus Documentation. URL: https://prometheus.io/docs/introduction/overview/

30. https://flask.palletsprojects.com/Grafana Documentation. URL: https://grafana.com/docs/

31. Web-технології та Web-дизайн: Застосування мови HTML для створення електронних ресурсів // Навчальний посібник. (видання друге) / – Київ: Видавництво Ліра-К, 2020. - 212 с.

32. Бородкіна І.Л., Бородкін Г.О. Теорія алгоритмів: Навчальний посібник. – Київ : Центр учбової літератури, 2018. – 184 с.

33. Бородкіна І.Л., Бородкін Г.О. Інженерія програмного забезпечення: Навчальний посібник. – Київ : Центр учбової літератури, 2018. - 204 с.

ДОДАТКИ

ДОДАТОК А

ФРАГМЕНТ ПРОГРАМНОГО КОДУ СЕРВЕРНОГО API СИСТЕМИ МОНІТОРИНГУ ДОМАШНІХ РОСЛИН

```python
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel, Field
from datetime import datetime
import sqlite3
from typing import Optional, List

DB_PATH = "plant_monitoring.db"

app = FastAPI(
    title="Plant Monitoring API",
    description="API для моніторингу стану домашніх рослин",
    version="1.0.0"
)


class TelemetryRequest(BaseModel):
    node_id: int = Field(..., description="Ідентифікатор сенсорного вузла")
    sensor_type: str = Field(..., description="Тип сенсора")
    value: float = Field(..., description="Значення вимірювання")
    unit: str = Field(..., description="Одиниця вимірювання")
    timestamp: Optional[str] = Field(None, description="Час вимірюван-ня")


class PlantResponse(BaseModel):
    id: int
    name: str
    species: str
    location: str
    status: str


def get_connection():
```

```
    connection = sqlite3.connect(DB_PATH)
    connection.row_factory = sqlite3.Row
    return connection


def init_database():
    connection = get_connection()
    cursor = connection.cursor()

    cursor.execute("""
        CREATE TABLE IF NOT EXISTS plants (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            species TEXT NOT NULL,
            location TEXT NOT NULL,
            status TEXT DEFAULT 'active',
            created_at TEXT NOT NULL
        )
    """)

    cursor.execute("""
        CREATE TABLE IF NOT EXISTS sensor_nodes (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            plant_id INTEGER NOT NULL,
            node_name TEXT NOT NULL,
            mac_address TEXT,
            status TEXT DEFAULT 'online',
            last_seen TEXT,
            FOREIGN KEY (plant_id) REFERENCES plants(id)
        )
    """)

    cursor.execute("""
        CREATE TABLE IF NOT EXISTS sensors (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
        node_id INTEGER NOT NULL,
        sensor_type TEXT NOT NULL,
        unit TEXT NOT NULL,
        min_value REAL,
        max_value REAL,
        FOREIGN KEY (node_id) REFERENCES sensor_nodes(id)
    )
    """)

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS measurements (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        node_id INTEGER NOT NULL,
        sensor_type TEXT NOT NULL,
        value REAL NOT NULL,
        unit TEXT NOT NULL,
        timestamp TEXT NOT NULL,
        quality_status TEXT DEFAULT 'valid',
        FOREIGN KEY (node_id) REFERENCES sensor_nodes(id)
    )
    """)

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS thresholds (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        plant_id INTEGER NOT NULL,
        sensor_type TEXT NOT NULL,
        min_allowed REAL,
        max_allowed REAL,
        warning_message TEXT,
        FOREIGN KEY (plant_id) REFERENCES plants(id)
    )
    """)

    cursor.execute("""
```

```
        CREATE TABLE IF NOT EXISTS alerts (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            plant_id INTEGER NOT NULL,
            node_id INTEGER NOT NULL,
            sensor_type TEXT NOT NULL,
            level TEXT NOT NULL,
            message TEXT NOT NULL,
            created_at TEXT NOT NULL,
            acknowledged INTEGER DEFAULT 0,
            FOREIGN KEY (plant_id) REFERENCES plants(id),
            FOREIGN KEY (node_id) REFERENCES sensor_nodes(id)
        )
    """)

    connection.commit()
    connection.close()

@app.on_event("startup")
def startup_event():
    init_database()

@app.get("/plants", response_model=List[PlantResponse])
def get_plants():
    connection = get_connection()
    cursor = connection.cursor()

    rows = cursor.execute("""
        SELECT id, name, species, location, status
        FROM plants
        ORDER BY id DESC
    """).fetchall()

    connection.close()

    return [
```

```
        PlantResponse(
            id=row["id"],
            name=row["name"],
            species=row["species"],
            location=row["location"],
            status=row["status"]
        )
        for row in rows
    ]


@app.post("/telemetry")
def receive_telemetry(payload: TelemetryRequest):
    connection = get_connection()
    cursor = connection.cursor()

    node = cursor.execute("""
        SELECT sn.id, sn.plant_id, p.name
        FROM sensor_nodes sn
        JOIN plants p ON p.id = sn.plant_id
        WHERE sn.id = ?
    """, (payload.node_id,)).fetchone()

    if node is None:
        connection.close()
        raise HTTPException(status_code=404, detail="Сенсорний вузол не знайдено")

    timestamp = payload.timestamp or datetime.now().isoformat(timespec="seconds")

    quality_status = "valid"
    if payload.value < -50 or payload.value > 10000:
        quality_status = "invalid"

    cursor.execute("""
```

```
        INSERT INTO measurements (
            node_id, sensor_type, value, unit, timestamp, quality_status
        )
        VALUES (?, ?, ?, ?, ?, ?)
    """, (
        payload.node_id,
        payload.sensor_type,
        payload.value,
        payload.unit,
        timestamp,
        quality_status
    ))

    cursor.execute("""
        UPDATE sensor_nodes
        SET last_seen = ?, status = 'online'
        WHERE id = ?
    """, (timestamp, payload.node_id))

    threshold = cursor.execute("""
        SELECT min_allowed, max_allowed, warning_message
        FROM thresholds
        WHERE plant_id = ? AND sensor_type = ?
    """, (node["plant_id"], payload.sensor_type)).fetchone()

    alert_created = False

    if threshold and quality_status == "valid":
        min_allowed = threshold["min_allowed"]
        max_allowed = threshold["max_allowed"]

        if payload.value < min_allowed:
            message = (
                threshold["warning_message"]
                or f"Показник {payload.sensor_type} нижчий за допустиму межу"
```

```
            )
            cursor.execute("""
                INSERT INTO alerts (
                    plant_id, node_id, sensor_type, level, message, created_at
                )
                VALUES (?, ?, ?, ?, ?, ?)
            """, (
                node["plant_id"],
                payload.node_id,
                payload.sensor_type,
                "warning",
                message,
                timestamp
            ))
            alert_created = True

        elif payload.value > max_allowed:
            message = (
                threshold["warning_message"]
                or f"Показник {payload.sensor_type} перевищує допустиму
межу"
            )
            cursor.execute("""
                INSERT INTO alerts (
                    plant_id, node_id, sensor_type, level, message, created_at
                )
                VALUES (?, ?, ?, ?, ?, ?)
            """, (
                node["plant_id"],
                payload.node_id,
                payload.sensor_type,
                "critical",
                message,
                timestamp
            ))
            alert_created = True
```

```
    connection.commit()
    connection.close()

    return {
        "status": "accepted",
        "node_id": payload.node_id,
        "sensor_type": payload.sensor_type,
        "value": payload.value,
        "quality_status": quality_status,
        "alert_created": alert_created
    }

@app.get("/plants/{plant_id}/latest")
def get_latest_measurements(plant_id: int):
    connection = get_connection()
    cursor = connection.cursor()

    rows = cursor.execute("""
        SELECT m.sensor_type, m.value, m.unit, m.timestamp
        FROM measurements m
        JOIN sensor_nodes sn ON sn.id = m.node_id
        WHERE sn.plant_id = ?
        AND m.id IN (
            SELECT MAX(id)
            FROM measurements
            GROUP BY sensor_type
        )
        ORDER BY m.sensor_type
    """, (plant_id,)).fetchall()

    connection.close()

    return {
        "plant_id": plant_id,
```

```
        "measurements": [
            {
                "sensor_type": row["sensor_type"],
                "value": row["value"],
                "unit": row["unit"],
                "timestamp": row["timestamp"]
            }
            for row in rows
        ]
    }

@app.get("/plants/{plant_id}/health-index")
def calculate_plant_health_index(plant_id: int):
    connection = get_connection()
    cursor = connection.cursor()

    rows = cursor.execute("""
        SELECT m.sensor_type, m.value, t.min_allowed, t.max_allowed
        FROM measurements m
        JOIN sensor_nodes sn ON sn.id = m.node_id
        JOIN thresholds t
            ON t.plant_id = sn.plant_id
            AND t.sensor_type = m.sensor_type
        WHERE sn.plant_id = ?
        AND m.id IN (
            SELECT MAX(m2.id)
            FROM measurements m2
            JOIN sensor_nodes sn2 ON sn2.id = m2.node_id
            WHERE sn2.plant_id = ?
            GROUP BY m2.sensor_type
        )
    """, (plant_id, plant_id)).fetchall()

    connection.close()
```

```
if not rows:
    return {
        "plant_id": plant_id,
        "phi": 0,
        "status": "no_data"
    }

scores = []

for row in rows:
    value = row["value"]
    min_allowed = row["min_allowed"]
    max_allowed = row["max_allowed"]

    if min_allowed <= value <= max_allowed:
        scores.append(100)
    else:
        interval = max_allowed - min_allowed
        if interval <= 0:
            scores.append(0)
        elif value < min_allowed:
            deviation = min_allowed - value
            scores.append(max(0, 100 - deviation / interval * 100))
        else:
            deviation = value - max_allowed
            scores.append(max(0, 100 - deviation / interval * 100))

phi = round(sum(scores) / len(scores), 2)

if phi >= 80:
    status = "normal"
elif phi >= 50:
    status = "attention"
else:
    status = "critical"
```

```
    return {
        "plant_id": plant_id,
        "phi": phi,
        "status": status
    }


@app.get("/alerts/active")
def get_active_alerts():
    connection = get_connection()
    cursor = connection.cursor()

    rows = cursor.execute("""
        SELECT a.id, p.name AS plant_name, a.sensor_type, a.level,
               a.message, a.created_at
        FROM alerts a
        JOIN plants p ON p.id = a.plant_id
        WHERE a.acknowledged = 0
        ORDER BY a.created_at DESC
    """).fetchall()

    connection.close()

    return {
        "alerts": [
            {
                "id": row["id"],
                "plant_name": row["plant_name"],
                "sensor_type": row["sensor_type"],
                "level": row["level"],
                "message": row["message"],
                "created_at": row["created_at"]
            }
            for row in rows
```

```
        ]
    }

@app.patch("/alerts/{alert_id}/ack")
def acknowledge_alert(alert_id: int):
    connection = get_connection()
    cursor = connection.cursor()

    cursor.execute("""
        UPDATE alerts
        SET acknowledged = 1
        WHERE id = ?
    """, (alert_id,))

    connection.commit()
    connection.close()

    return {
        "status": "updated",
        "alert_id": alert_id
    }
```

ДОДАТОК Б

ФРАГМЕНТ СТРУКТУРИ БАЗИ ДАНИХ СИСТЕМИ МОНІТОРИНГУ ДОМАШНІХ РОСЛИН

Сторінок 9

```sql
PRAGMA foreign_keys = ON;

CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL UNIQUE,
    email TEXT NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    role TEXT NOT NULL DEFAULT 'user',
    is_active INTEGER NOT NULL DEFAULT 1,
    created_at TEXT NOT NULL
);

CREATE TABLE plants (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    name TEXT NOT NULL,
    species TEXT NOT NULL,
    location TEXT NOT NULL,
    description TEXT,
    status TEXT NOT NULL DEFAULT 'active',
    created_at TEXT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CAS-CADE
);

CREATE TABLE plant_profiles (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    plant_id INTEGER NOT NULL,
    recommended_moisture_min REAL NOT NULL,
    recommended_moisture_max REAL NOT NULL,
    recommended_temperature_min REAL NOT NULL,
    recommended_temperature_max REAL NOT NULL,
```

```
    recommended_air_humidity_min REAL NOT NULL,
    recommended_air_humidity_max REAL NOT NULL,
    recommended_light_min REAL NOT NULL,
    recommended_light_max REAL NOT NULL,
    recommended_ph_min REAL,
    recommended_ph_max REAL,
    watering_interval_days INTEGER DEFAULT 3,
    FOREIGN KEY (plant_id) REFERENCES plants(id) ON DELETE CASCADE
);

CREATE TABLE sensor_nodes (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    plant_id INTEGER NOT NULL,
    node_name TEXT NOT NULL,
    mac_address TEXT,
    firmware_version TEXT,
    connection_type TEXT NOT NULL DEFAULT 'Wi-Fi',
    status TEXT NOT NULL DEFAULT 'offline',
    last_seen TEXT,
    created_at TEXT NOT NULL,
    FOREIGN KEY (plant_id) REFERENCES plants(id) ON DELETE CASCADE
);

CREATE TABLE sensor_types (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    code TEXT NOT NULL UNIQUE,
    name TEXT NOT NULL,
    unit TEXT NOT NULL,
    description TEXT
);

CREATE TABLE sensors (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    node_id INTEGER NOT NULL,
```

```
    sensor_type_id INTEGER NOT NULL,
    sensor_name TEXT NOT NULL,
    model TEXT,
    polling_interval_sec INTEGER NOT NULL DEFAULT 300,
    is_active INTEGER NOT NULL DEFAULT 1,
    created_at TEXT NOT NULL,
    FOREIGN KEY (node_id) REFERENCES sensor_nodes(id) ON DE-LETE
CASCADE,
    FOREIGN KEY (sensor_type_id) REFERENCES sensor_types(id)
);

CREATE TABLE measurements (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    sensor_id INTEGER NOT NULL,
    value REAL NOT NULL,
    unit TEXT NOT NULL,
    measured_at TEXT NOT NULL,
    quality_status TEXT NOT NULL DEFAULT 'valid',
    source TEXT NOT NULL DEFAULT 'sensor',
    FOREIGN KEY (sensor_id) REFERENCES sensors(id) ON DELETE CASCADE
);

CREATE TABLE thresholds (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    plant_id INTEGER NOT NULL,
    sensor_type_id INTEGER NOT NULL,
    min_allowed REAL,
    max_allowed REAL,
    warning_message TEXT,
    critical_message TEXT,
    is_active INTEGER NOT NULL DEFAULT 1,
    FOREIGN KEY (plant_id) REFERENCES plants(id) ON DELETE CASCADE,
    FOREIGN KEY (sensor_type_id) REFERENCES sensor_types(id)
);
```

```
CREATE TABLE alerts (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    plant_id INTEGER NOT NULL,
    sensor_id INTEGER,
    level TEXT NOT NULL,
    message TEXT NOT NULL,
    created_at TEXT NOT NULL,
    acknowledged INTEGER NOT NULL DEFAULT 0,
    acknowledged_at TEXT,
    FOREIGN KEY (plant_id) REFERENCES plants(id) ON DELETE CASCADE,
    FOREIGN KEY (sensor_id) REFERENCES sensors(id) ON DELETE SET NULL
);

CREATE TABLE care_actions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    plant_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    action_type TEXT NOT NULL,
    comment TEXT,
    performed_at TEXT NOT NULL,
    FOREIGN KEY (plant_id) REFERENCES plants(id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CAS-CADE
);

CREATE TABLE plant_health_index (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    plant_id INTEGER NOT NULL,
    phi_value REAL NOT NULL,
    status TEXT NOT NULL,
    calculated_at TEXT NOT NULL,
    FOREIGN KEY (plant_id) REFERENCES plants(id) ON DELETE CASCADE
);
```

```
CREATE TABLE reports (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   user_id INTEGER NOT NULL,
   plant_id INTEGER,
   report_type TEXT NOT NULL,
   period_from TEXT NOT NULL,
   period_to TEXT NOT NULL,
   file_path TEXT,
   created_at TEXT NOT NULL,
   FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CAS-CADE,
   FOREIGN KEY (plant_id) REFERENCES plants(id) ON DELETE SET NULL
);

CREATE INDEX idx_plants_user_id
ON plants(user_id);

CREATE INDEX idx_sensor_nodes_plant_id
ON sensor_nodes(plant_id);

CREATE INDEX idx_sensors_node_id
ON sensors(node_id);

CREATE INDEX idx_measurements_sensor_time
ON measurements(sensor_id, measured_at);

CREATE INDEX idx_alerts_plant_ack
ON alerts(plant_id, acknowledged);

CREATE INDEX idx_care_actions_plant_time
ON care_actions(plant_id, performed_at);

INSERT INTO sensor_types (code, name, unit, description)
VALUES
```

```
    ('soil_moisture', 'Вологість ґрунту', '%', 'Показник вологості субстрату'),
    ('air_temperature', 'Температура повітря', '°C', 'Температура повітря біля рослини'),
    ('air_humidity', 'Вологість повітря', '%', 'Відносна вологість повітря'),
    ('light', 'Освітленість', 'lx', 'Рівень освітленості зони розміщення рослини'),
    ('soil_temperature', 'Температура ґрунту', '°C', 'Температура ґрунтового субстрату'),
    ('ph', 'Кислотність ґрунту', 'pH', 'Показник кислотності субстрату');

INSERT INTO users (
    username,
    email,
    password_hash,
    role,
    is_active,
    created_at
)
VALUES (
    'admin',
    'admin@example.com',
    'pbkdf2_sha256$example_hash',
    'admin',
    1,
    datetime('now')
);

INSERT INTO plants (
    user_id,
    name,
    species,
    location,
```

```
    description,
    status,
    created_at
)
VALUES (
    1,
    'Монстера',
    'Monstera deliciosa',
    'Кімната біля вікна',
    'Домашня декоративна рослина, що потребує помірного поливу та
розсіяного світла',
    'active',
    datetime('now')
);
INSERT INTO plant_profiles (
    plant_id,
    recommended_moisture_min,
    recommended_moisture_max,
    recommended_temperature_min,
    recommended_temperature_max,
    recommended_air_humidity_min,
    recommended_air_humidity_max,
    recommended_light_min,
    recommended_light_max,
    recommended_ph_min,
    recommended_ph_max,
    watering_interval_days
)
VALUES (
    1,
    35,
    70,
    18,
    27,
```

```
    45,
    75,
    1500,
    8000,
    5.5,
    7.0,
    4
);

INSERT INTO sensor_nodes (
    plant_id,
    node_name,
    mac_address,
    firmware_version,
    connection_type,
    status,
    last_seen,
    created_at
)
VALUES (
    1,
    'PlantNode-01',
    'A0:B1:C2:D3:E4:F5',
    '1.0.0',
    'Wi-Fi',
    'online',
    datetime('now'),
    datetime('now')
);

INSERT INTO thresholds (
    plant_id,
    sensor_type_id,
    min_allowed,
```

```
    max_allowed,
    warning_message,
    critical_message,
    is_active
)
VALUES
    (1, 1, 35, 70, 'Потрібно перевірити вологість ґрунту', 'Критично низька або надмірна вологість ґрунту', 1),
    (1, 2, 18, 27, 'Температура повітря вийшла за рекомендовані межі', 'Критичне відхилення температури повітря', 1),
    (1, 3, 45, 75, 'Вологість повітря потребує уваги', 'Критичне відхилення вологості повітря', 1),
    (1, 4, 1500, 8000, 'Недостатній або надмірний рівень освітлення', 'Критичне відхилення освітленості', 1);
```

ДОДАТОК В

ФРАГМЕНТ ПРОГРАМНОГО КОДУ СЕНСОРНОГО ВУЗЛА ESP32

Сторінок 7

```cpp
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include "DHT.h"

#define WIFI_SSID "PlantWiFi"
#define WIFI_PASSWORD "password123"

#define API_URL "http://192.168.1.10:8000/telemetry"

#define NODE_ID 1

#define DHT_PIN 4
#define DHT_TYPE DHT22

#define SOIL_MOISTURE_PIN 34
#define LIGHT_SENSOR_PIN 35

DHT dht(DHT_PIN, DHT_TYPE);

unsigned long previousMillis = 0;
const unsigned long pollingInterval = 300000;

int soilDryValue = 3000;
int soilWetValue = 1200;

void connectToWiFi() {
  Serial.print("Підключення до Wi-Fi");

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

```
  int attempt = 0;

  while (WiFi.status() != WL_CONNECTED && attempt < 30) {
    delay(500);
    Serial.print(".");
    attempt++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println();
    Serial.println("Wi-Fi підключено");
    Serial.print("IP-адреса: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println();
    Serial.println("Не вдалося підключитися до Wi-Fi");
  }
}

float readSoilMoisturePercent() {
  int rawValue = analogRead(SOIL_MOISTURE_PIN);

  int moisturePercent = map(
    rawValue,
    soilDryValue,
    soilWetValue,
    0,
    100
  );

  if (moisturePercent < 0) {
    moisturePercent = 0;
  }
```

```
  if (moisturePercent > 100) {
    moisturePercent = 100;
  }

  return (float)moisturePercent;
}

float readLightLevel() {
  int rawValue = analogRead(LIGHT_SENSOR_PIN);

  float voltage = rawValue * (3.3 / 4095.0);
  float lightPercent = (voltage / 3.3) * 100.0;

  return lightPercent;
}

bool sendTelemetry(String sensorType, float value, String unit) {
  if (WiFi.status() != WL_CONNECTED) {
    connectToWiFi();
  }

  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Передавання неможливе: Wi-Fi недоступний");
    return false;
  }

  HTTPClient http;

  http.begin(API_URL);
  http.addHeader("Content-Type", "application/json");

  StaticJsonDocument<256> payload;

  payload["node_id"] = NODE_ID;
```

```
  payload["sensor_type"] = sensorType;
  payload["value"] = value;
  payload["unit"] = unit;

  String jsonBody;
  serializeJson(payload, jsonBody);

  int responseCode = http.POST(jsonBody);

  if (responseCode > 0) {
    Serial.print("HTTP код відповіді: ");
    Serial.println(responseCode);
    String response = http.getString();
    Serial.println(response);
  } else {
    Serial.print("Помилка передавання: ");
    Serial.println(responseCode);
  }

  http.end();

  return responseCode >= 200 && responseCode < 300;
}

void readAndSendMeasurements() {
  float airTemperature = dht.readTemperature();
  float airHumidity = dht.readHumidity();
  float soilMoisture = readSoilMoisturePercent();
  float lightLevel = readLightLevel();

  if (!isnan(airTemperature)) {
    sendTelemetry("air_temperature", airTemperature, "°C");
  } else {
    Serial.println("Помилка зчитування температури повітря");
```

```
    }

    delay(1000);

    if (!isnan(airHumidity)) {
      sendTelemetry("air_humidity", airHumidity, "%");
    } else {
      Serial.println("Помилка зчитування вологості повітря");
    }

    delay(1000);

    sendTelemetry("soil_moisture", soilMoisture, "%");

    delay(1000);

    sendTelemetry("light", lightLevel, "%");

    Serial.println("Цикл вимірювання завершено");
}

void setup() {
    Serial.begin(115200);

    pinMode(SOIL_MOISTURE_PIN, INPUT);
    pinMode(LIGHT_SENSOR_PIN, INPUT);

    dht.begin();

    connectToWiFi();

    readAndSendMeasurements();
}
```

```
void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= pollingInterval) {
    previousMillis = currentMillis;
    readAndSendMeasurements();
  }
}
```

Фрагмент файлу requirements.txt для запуску серверної частини:

```txt
fastapi==0.115.0
uvicorn==0.30.6
pydantic==2.8.2
python-multipart==0.0.9
jinja2==3.1.4
pandas==2.2.2
numpy==2.0.1
scikit-learn==1.5.1
matplotlib==3.9.2
```

Фрагмент команди запуску серверної частини:

```bash
uvicorn main:app --host 0.0.0.0 --port 8000 --reload
```

Фрагмент структури інсталяційного пакета:

```txt
plant_monitoring_system/
```

```
│
├── app/
│   ├── main.py
│   ├── database.py
│   ├── models.py
│   ├── services.py
│   └── reports.py
│
├── database/
│   ├── schema.sql
│   └── plant_monitoring.db
│
├── sensor_node/
│   └── esp32_plant_node.ino
│
├── web_ui/
│   ├── index.html
│   ├── dashboard.html
│   ├── style.css
│   └── app.js
│
├── reports/
│   └── generated/
│
├── requirements.txt
└── README.md
```

ДОДАТОК Г

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Пшеничний Назар Віталійович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Програмне забезпечення моніторингу стану домашніх рослин» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [ ] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - [ ] перекладу іншомовних джерел;
        - [ ] стилістичного редагування та перевірки граматики;
        - [ ] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Назар ПШЕНИЧНИЙ**
(підпис)