**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

15.03 — КМР. 2001– “С” 2023.11. 01 ПЗ

ГРИГУРКА ДАНИЛА ОЛЕГОВИЧА

2024 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

**Факультет інформаційних технологій**

**УДК**

| **«ПОГОДЖЕНО»** | **«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»** |
|---|---|
| декан факультету | Завідувач кафедри комп'ютерних наук |
| інформаційних технологій | |
| Глазунова О.Г., д.п.н., професор | Голуб Б.Л., к.т.н., доцент |
| _____ | _____ |
| ____ _____ **2024 р.** | ____ _____ **2024 р.** |

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему** __Експертна інформаційно-управляюча система тепличного виробництва_

Спеціальність ___122 - Комп'ютерні науки_____
(код і назва)

Освітня програма ____Інформаційно управляючі системи та технології_
(назва)

Орієнтація освітньої програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

**Гарант освітньої програми**

_____ _____ ___Волошин С.М.__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

**Керівник магістерської кваліфікаційної роботи**

_____к.т.н., доцент_____ _____ _____Дудник А.О._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

**Виконав** _____ _____Григурко Д.О._____
(підпис) (ПІБ студента)

**КИЇВ - 2024**

**НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет (ННІ)** _____Інформаційних технологій_____

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри**___комп'ютерних наук__

_____

____к.т.н., доцент_____ _____ _Голуб Б.Л._

(науковий ступінь, вчене звання) (підпис) (ПІБ)

"____" _____ 2024 року

**З А В Д А Н Н Я**

**ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ**

_____Григурко Данило Олегович_____

(прізвище, ім'я, по батькові)

Спеціальність __122 - Комп'ютерні науки _____

(код і назва)

Освітня програма ____Інформаційно управляючі системи та технології_____

(назва)

Орієнтація освітньої програми_____освітньо-професійна_____

(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи__Експертна інформаційно-управляюча система тепличного виробництва_____

затверджена наказом ректора НУБіП України від "____"_____20___р. №_____

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи __створення системи, яка на основі аналізу даних зможе виявляти відхилення і надавати рекомендації для кращого вирощування овочей

Перелік питань, що підлягають дослідженню:

1. _Системний аналіз предметної області_____
2. _Моделювання системи_____
3. _Розробка системи_____
4. _Результати дослідження_____

Перелік графічного матеріалу (за потреби) _____

_____

**Дата видачі завдання** "_____"_____20___ р.

**Керівник магістерської кваліфікаційної роботи** _____ _____Дудник А.О.___

( підпис ) (прізвище та ініціали)

**Завдання прийняв до виконання** _____ _____Григурко Д.О._____

(підпис ) (прізвище та ініціали студента)

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання магістерської кваліфікаційної роботи | Строк виконання етапів магістерської кваліфікаційної роботи | Примітка |
|---|---|---|---|
| 1. | Отримання теми завдання | 01.11.2023 | |
| 2. | Формування технічного завдання | 02.11.2023 – 20.11.2023 | |
| 3. | Розробка структури інформаційної системи | 21.11.2023 – 10.12.2023 | |
| 4. | Розробка інтерфейсу | 11.12.2023 – 31.01.2024 | |
| 5. | Реалізація базового функціоналу | 01.02.2024 – 31.03.2024 | |
| 6. | Розробка моделі прийняття рішень та системи рекомендацій | 01.04.2024 – 30.04.2024 | |
| 7. | Тестування, усунення помилок та покращення системи | 01.05.2024 – 31.07.2024 | |
| 8. | Впровадження алгоритмів та налаштування | 01.08.2024 – 31.08.2024 | |
| 9. | Оформлення пояснювальної записки | 01.09.2024 – 15.10.2024 | |
| 10. | Дослідницька практика за темою магістерської кваліфікаційної роботи | 19.08.2024 – 01.11.2024 | |
| 11. | Перевірка роботи на академічний плагіат | 01.10.2024 – 15.10.2024 | |
| 12. | Проходження нормконтролю та отримання зовнішньої рецензії | 16.10.2024 – 31.10.2024 | |
| 13. | Передзахист та постерна сесія | 11.11.2024 – 28.11.2024 | |
| 14. | Захист магістерської кваліфікаційної роботи | 02.12.2024 – 14.12.2024 | |

**Студент** _____ Григурко Д.О.

(підпис) (прізвище та ініціали)

**Керівник бакалаврської кваліфікаційної роботи** _____ Дудник А.О

(підпис) (прізвище та ініціали)

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

**SQL** - Мова структурованих запитів

**БД** – База даних

**СД** – Сховище даних

**ІАД** - інтелектуальний аналіз даних

**СУБД** – система управління базами даних

# АНОТАЦІЯ

Експертна інформаційно-управляюча система тепличного виробництва – це сучасне інтегроване рішення, що забезпечує ефективний контроль за процесами вирощування овочів у тепличних умовах. Система збирає дані з численних сенсорів, що фіксують ключові параметри середовища, такі як температура, вологість, освітлення та вміст CO2, що дає змогу агроному оперативно відслідковувати умови та вчасно вносити зміни для оптимізації вирощування.

Система складається з декількох модулів, кожен із яких відповідає за конкретний аспект її роботи: модуль збору даних, модуль аналізу, модуль формування рекомендацій та звітів. Усі ці модулі тісно взаємодіють між собою для забезпечення безперервного циклу моніторингу та управління. Ключовою перевагою цієї системи є її здатність не лише фіксувати показники, але й аналізувати їх у режимі реального часу та автоматично формувати рекомендації для агронома. Це дає можливість зменшити ризики втрати врожаю та запобігти можливим проблемам, пов'язаним із змінами кліматичних умов у теплиці.

Окрім оперативних рекомендацій, система дозволяє агроному переглядати детальні звіти про стан середовища за останні 24 години та за триваліші періоди. Це сприяє виявленню довгострокових тенденцій, що допомагає у плануванні роботи теплиці, а також в удосконаленні стратегій вирощування.

Завдяки історичним даним та звітам аналітики можуть проводити детальний аналіз впливу різних факторів на врожайність і давати рекомендації щодо оптимізації процесів у майбутньому. Це дозволяє компаніям отримувати більшу стабільність у виробничих процесах і забезпечувати високу якість продукції.

# ABSTRACT

The expert information and management system of greenhouse production is a modern integrated solution that provides effective control over the processes of growing vegetables in greenhouse conditions. The system collects data from numerous sensors that capture key environmental parameters such as temperature, humidity, lighting and CO2 content, which allows the agronomist to quickly monitor conditions and make changes in time to optimize cultivation.

The system consists of several modules, each of which is responsible for a specific aspect of its work: a data collection module, an analysis module, a module for generating recommendations and reports. All these modules work closely together to ensure a continuous cycle of monitoring and control. The key advantage of this system is its ability not only to record indicators, but also to analyze them in real time and automatically generate recommendations for the agronomist. This makes it possible to reduce the risks of crop loss and prevent possible problems associated with changes in climatic conditions in the greenhouse.

In addition to operational recommendations, the system allows the agronomist to view detailed reports on the state of the environment for the last 24 hours and for longer periods. This helps to identify long-term trends, which helps in planning the operation of the greenhouse, as well as in improving cultivation strategies.

Thanks to historical data and reports, analysts can conduct a detailed analysis of the impact of various factors on yields and make recommendations for optimizing processes in the future. This allows companies to obtain greater stability in production processes and ensure high quality products.

With access to historical data and reports, analysts can conduct thorough assessments of how various factors impact yields and provide optimization recommendations for future processes. This approach offers companies greater stability in production processes and ensures the high quality of their products.

# ВСТУП

Експертна інформаційно-управляюча система для тепличного виробництва є програмним комплексом, що забезпечує моніторинг та аналіз параметрів мікроклімату в теплицях. Система інтегрує дані від різних сенсорів для кожного виду та сорту рослин, що дозволяє оптимізувати умови вирощування і підвищувати врожайність. Застосовуючи сучасні методи аналізу даних та алгоритми машинного навчання, система здатна здійснювати прогнозування, виявлення аномалій та виявлення асоціативних правил, що допомагають аграріям приймати обґрунтовані рішення для покращення процесів вирощування.

Зростаючий попит на якісні сільськогосподарські продукти та необхідність оптимізації тепличного виробництва стимулюють впровадження сучасних технологій для автоматизованого контролю та аналізу мікроклімату. Традиційні методи управління в теплицях часто не враховують складну взаємодію параметрів навколишнього середовища, що може призвести до субоптимальних умов вирощування. Інформаційно-управляюча система дозволяє покращити якість врожаю, мінімізувати витрати на ресурси та забезпечити стабільний моніторинг і оперативне реагування на зміну умов. Це особливо важливо в умовах глобальних кліматичних змін, де стале виробництво є ключовим для підтримання продовольчої безпеки.

**Об’єкт дослідження** – процеси функціонування теплиць, їх автоматизації та впровадження сучасних інформаційних технологій для моніторингу й управління параметрами середовища. **Предмет** – інтегровані методи збору, зберігання, аналізу та візуалізації даних у тепличному господарстві, а також алгоритми, що використовуються для класифікації, прогнозування й оптимізації управлінських рішень.

**Метою** дослідження є розробка експертної інформаційно-управляючої системи для автоматизації тепличного виробництва, яка базується на сучасних підходах до обробки даних та машинного навчання. Система покликана підвищити ефективність вирощування сільськогосподарських культур шляхом впровадження інструментів для аналізу великих даних, оптимізації умов середовища та покращення процесу прийняття рішень.

Для досягнення мети дослідження будуть вирішені такі **завдання**:

1. Аналіз літератури та досвіду. Провести огляд існуючих рішень у сфері автоматизації тепличного виробництва, виявити їх недоліки та потенційні напрямки вдосконалення.

2. Розробка структури бази даних. Створити нормалізовану модель бази даних, яка забезпечує зберігання даних про культури, умови середовища, дані сенсорів та результати аналізу.

3. Розробка алгоритмів обробки даних. Реалізувати алгоритми класифікації, прогнозування та асоціативного аналізу для ідентифікації аномалій і закономірностей у тепличних даних.

4. Інтеграція інтерфейсів. Створити програмний інтерфейс для взаємодії користувачів із системою, що включає візуалізацію даних, моніторинг і контроль.

5. Тестування та верифікація. Перевірити систему на реальних даних, провести її оптимізацію та оцінити ефективність запропонованих рішень у порівнянні з традиційними підходами.

У цьому дослідженні використовуватиметься комбінація **методів дослідження**:

1. Математичне моделювання. Використання для побудови прогнозів щодо змін параметрів середовища.

2. Машинне навчання. Реалізація алгоритмів класифікації (1-Rule, Naive Bayes), кластеризації (k-means) та асоціативного аналізу (алгоритм Apriori).

3. Бази даних. Застосування реляційних баз даних для організації зберігання великих обсягів структурованих даних.

4. Програмні засоби. Розробка з використанням Python, SQLAlchemy, PostgreSQL та бібліотек для обробки даних.

5. Методи візуалізації. Побудова графіків, діаграм і візуалізація зв’язків між даними для полегшення інтерпретації результатів аналізу.

**Наукова новизна** роботи полягає в комплексному підході до автоматизації тепличного виробництва:

1. Вперше запропоновано інтеграцію алгоритмів класифікації, кластеризації й асоціативного аналізу в єдину експертну систему для моніторингу та управління параметрами середовища.

2. Розроблено оригінальну структуру бази даних, яка підтримує збереження та аналіз великих масивів даних сенсорів.

3. Розширено підходи до виявлення аномалій у тепличних даних на основі аналізу гіперкуба даних, що включає показники температури, вологості, освітленості та рівня $CO_2$.

**Апробація результатів дослідження:** результати дослідження були представлені на конференціях та опубліковані у відповідних наукових виданнях:

- тези, представлені на щорічній конференції з питань великих даних у сфері охорони здоров’я, 2024 р.

**Структура даної магістерської роботи** включає чотири основні розділи:

1. Системний аналіз предметної області – у цьому розділі буде розглянуто існуючі рішення для моніторингу фітнесу, визначено ключові технології та висвітлено поточні обмеження в галузі;

2. Моделювання системи – у цьому розділі розглядатимуться концептуальний дизайн і моделювання системи моніторингу та аналізу придатності;

3. Розробка системи – у цьому розділі буде описано процес розробки запропонованої системи, включаючи ключові деталі впровадження;

4. Результати дослідження – у заключному розділі будуть представлені результати впровадження та тестування системи, оцінка її продуктивності та надання рекомендацій щодо подальшого вдосконалення.

Магістерська робота містить 73 сторінки, в тому числі 30 рисунків, 1 таблицю, 2 додатки. Вона посилається на 30 джерел, включаючи наукові статті, книги та електронні ресурси.

# 1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Опис предметної області

Тепличне виробництво є спеціалізованим сектором сільського господарства, основною метою якого є вирощування сільськогосподарських культур у контрольованих умовах. Цей підхід дозволяє забезпечити стабільний урожай незалежно від сезонних чи кліматичних змін. Головною перевагою теплиць є можливість керування параметрами середовища, такими як температура, вологість, освітлення та концентрація вуглекислого газу ($CO_2$), що дозволяє створювати оптимальні умови для різних видів культур.

На сучасному етапі розвитку технологій тепличне виробництво активно інтегрує автоматизацію, сенсорні мережі та інформаційні системи для моніторингу й управління. Використання таких систем дозволяє мінімізувати людський фактор, підвищити точність контролю параметрів та оптимізувати використання ресурсів (води, добрив, енергії), ключових параметрів тепличного середовища

Основними параметрами, що впливають на успішність вирощування культур, є:

Температура повітря та ґрунту є критично важливими для фотосинтезу, росту та розвитку рослин. Занадто високі або низькі температури можуть уповільнювати біологічні процеси або викликати стрес у рослин. Оптимальний діапазон: для більшості овочевих культур це +18...+25°C вдень і +12...+18°C вночі.

Параметр, що впливає на транспірацію рослин та доступність води. Висока вологість може викликати захворювання, а низька – призводить до зневоднення. Оптимальна вологість: 60–80%.

Освітленість має визначальне значення для фотосинтезу. У теплицях часто використовуються додаткові джерела світла, щоб компенсувати дефіцит

сонячного світла в зимовий період. Мінімальна освітленість: 5000–10000 люкс залежно від культури.

Вуглекислий газ є основним елементом, що використовується рослинами для синтезу органічних речовин у процесі фотосинтезу. Підвищення концентрації $CO_2$ сприяє інтенсивнішому росту рослин. Оптимальний рівень: 800–1200 ppm.

Тепличне виробництво активно використовує інноваційні підходи, зокрема сенсорні технології, що дозволяють здійснювати постійний моніторинг стану середовища. Інформація зі сенсорів обробляється в реальному часі, що дає змогу швидко реагувати на зміни.

Системи автоматичного керування, регулюють роботу обігрівачів, зволожувачів, систем вентиляції та освітлення залежно від даних сенсорів. Це дозволяє підтримувати оптимальні умови з мінімальним втручанням людини.

Системи аналізу даних дозволяють виявляти закономірності, прогнозувати зміни параметрів середовища, оцінювати ефективність застосованих заходів та приймати оптимальні управлінські рішення.

Ефективна інформаційно-управляюча система для тепличного виробництва повинна забезпечувати збір даних у реальному часі, зберігання великих обсягів даних. Тобто, система має інтегрувати дані з численних сенсорів, що фіксують температуру, вологість, освітленість та рівень $CO_2$. Для аналізу та прогнозування необхідно зберігати історичні дані за тривалий період.[1]

Система повинна включати алгоритми класифікації, прогнозування та виявлення аномалій. Це дозволяє оптимізувати параметри середовища залежно від типу культури.

Для зручності користувачів необхідно забезпечити доступ до візуалізації даних і керування системою через простий інтерфейс.

Основними проблемами тепличного виробництва залишаються:

- висока вартість впровадження автоматизованих систем;

- необхідність навчання персоналу для роботи з такими системами;
- технічні обмеження в малих тепличних господарствах.

Перспективи розвитку пов’язані з інтеграцією технологій штучного інтелекту, розширенням аналітичних можливостей, зниженням вартості сенсорних мереж та впровадженням «розумних» теплиць, що використовують автономне управління.

## 1.2 Огляд існуючих рішень

Ринок рішень для автоматизації тепличного виробництва пропонує широкий спектр програмних і апаратних засобів, які інтегрують сенсорні технології, машинне навчання та алгоритми аналізу даних. Одним із найбільш популярних і успішних прикладів є система "Priva Connext", яка вважається провідним інструментом для управління теплицями. Priva Connext вирізняється високим рівнем автоматизації, адаптивними алгоритмами управління параметрами середовища, інтеграцією з аналітичними платформами та зручним інтерфейсом для користувачів. З Priva Connext у вас є одна система для управління всіма вашими системами та установками, які мають відношення до вирощування; будь то світло, клімат-контроль, управління водними ресурсами або управління своїми енергетичними ресурсами ефективно. [2] Зв'язування всіх ваших систем і елементів управління неймовірно легко, надійно і стабільно.

Priva Connext повністю самоуправляє і працює 24/7. Завдяки зручним операційним програмам ви відразу зрозумієте всі поточні зростаючі процеси. Програми пропонують вам всі можливості для оптимізації вашого врожаю і клімату. Priva Connext Operator - це веб-додаток для планшета або смартфона. Це дає вам уявлення про ваші процеси і дозволяє робити найважливіші зміни у ваших системах завжди і скрізь. Приклад інтерфейсу представлений на рисунку 1.

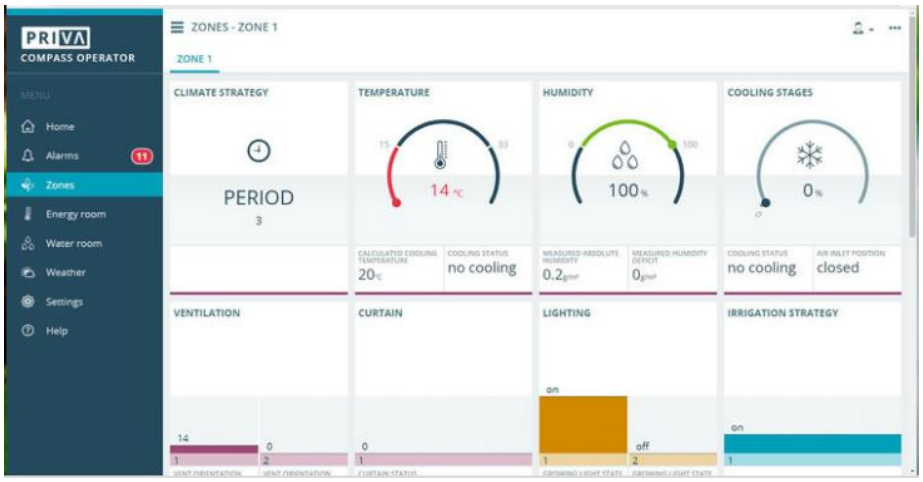


Рис.1 Інтерфейс додатку «Priva Context»

Priva Connext – це програмно-апаратний комплекс, призначений для управління мікрокліматом у теплицях, системами поливу, вентиляції та освітлення. Система використовує дані зі встановлених сенсорів і прогнозує зміни в параметрах середовища для забезпечення оптимальних умов для рослин.

Priva Connext складається з декількох основних компонентів. Система використовує сучасні сенсори для збору даних про температуру, вологість, освітлення, рівень $CO_2$, склад ґрунту та інші параметри. Контролери виконують команди системи, регулюючи роботу обігрівачів, вентиляції, зволожувачів та систем освітлення. Централізоване програмне забезпечення відповідає за аналіз даних і прийняття рішень. Воно має інтуїтивно зрозумілий графічний інтерфейс, де користувач може налаштувати параметри середовища, створювати графіки та переглядати звіти. Для великих тепличних комплексів система пропонує хмарні рішення, які забезпечують зберігання великих обсягів даних та доступ до них з будь-якої точки світу. Система використовує аналітичні алгоритми для прогнозування змін кліматичних умов, виявлення аномалій у даних сенсорів та оптимізації параметрів середовища.

Priva Connext дозволяє налаштовувати температурні режими, рівень вологості, концентрацію $CO_2$ та інші параметри залежно від типу культури, що вирощується. Автоматизація процесу знижує людський фактор і підвищує

точність підтримки умов. Система керує поливом на основі даних сенсорів вологості ґрунту, що дозволяє уникнути надмірного або недостатнього зволоження. Система аналізує споживання енергії в реальному часі та пропонує рішення для його оптимізації, наприклад, шляхом підвищення ефективності роботи систем опалення або вентиляції. Інтерфейс дозволяє створювати графіки змін параметрів середовища, генерувати звіти за певний період і виявляти потенційні проблеми в роботі системи. [3] Користувачі можуть отримувати доступ до даних системи через мобільні пристрої, що значно підвищує зручність управління. Приклад систему представлений на рисунку 2.

Рис.2 Графік контролю температури

Ця експертна система отримала світову популярність завдяки автоматизації система значно зменшує час, необхідний для управління теплицями, та забезпечує стабільність параметрів середовища. Модульна архітектура дозволяє адаптувати систему до потреб будь-якого господарства, незалежно від його розміру. Система оптимізує споживання енергії та води, що дозволяє знизити операційні витрати на 15–20%. Priva Connext легко інтегрується з іншими

рішеннями, такими як метеостанції або ERP-системи. Priva Connext успішно використовується у великих тепличних господарствах Європи, Північної Америки та Азії. Наприклад, у Нідерландах система впроваджена у кількох агропромислових комплексах, де вона забезпечила підвищення врожайності на 25% завдяки точному контролю параметрів середовища. Скорочення споживання води на 30% через впровадження автоматизованого поливу.Зниження витрат на електроенергію на 15% завдяки оптимізації режимів роботи систем освітлення та опалення.

Попри численні переваги, система має деякі обмеження:

- висока вартість впровадження;
- Priva Connext є досить дорогою системою, що може бути проблемою для невеликих господарств.
- необхідність навчання персоналу;
- робота з системою вимагає попереднього навчання користувачів, особливо для роботи з аналітичними модулями;
- залежність від стабільного інтернет-з'єднання;
- для використання хмарної платформи необхідне надійне підключення до інтернету.

## 1.3 Постановка завдання

Головним завданням є розробка експертної інформаційно-управляючої системи для тепличного виробництва, яка забезпечить автоматизований контроль, аналіз і оптимізацію умов вирощування рослин з метою підвищення врожайності та зниження витрат на ресурси.

Мета проекту створити інноваційну систему управління тепличним виробництвом, яка поєднає автоматизацію операцій, аналіз даних та інтеграцію сучасних технологій для забезпечення сталого розвитку агропромислового комплексу.

Основною умовою розробки є забезпечення точного збору, аналізу та інтерпретації даних про стан тепличного середовища, а також адаптація системи до різних типів культур і масштабів господарств. Розроблена система повинна забезпечити автоматизацію тепличних процесів із можливістю адаптації до різних видів культур. Очікується зниження витрат на енергію, воду та добрива завдяки автоматизації та точному контролю параметрів. Завдяки стабільним і оптимальним умовам вирощування рослин. Система повинна мати інтуїтивно зрозумілий інтерфейс і гнучкі налаштування.

# 2 МОДЕЛЮВАННЯ СИСТЕМИ

## 2.1 Загальні відомості

Функціональний підхід до моделювання є одним із найдавніших і найпоширеніших підходів у проєктуванні інформаційних систем. Він базується на аналізі функцій, які система повинна виконувати, і розробці моделі, що відображає ці функції, а також потоки даних між ними.

Основна ідея функціонального підходу зосереджується на тому, що система робить, а не на тому, як вона це робить. Основною метою є визначення набору функцій, необхідних для досягнення цілей системи, і способу їхньої взаємодії. У цьому підході система розглядається як сукупність взаємопов’язаних функцій, кожна з яких приймає на вхід певні дані, обробляє їх і видає результат. Усі функції пов’язані між собою потоками даних, що створює цілісну модель роботи системи.

Контекстна діаграма показує систему як єдине ціле, її взаємодію з зовнішніми середовищами та основні потоки даних між ними. Цей елемент дозволяє визначити межі системи та окреслити основні входи і виходи. Основна функція системи ділиться на підфункції, які, у свою чергу, можуть бути розбиті на ще дрібніші частини. Це дозволяє побудувати ієрархічну структуру функцій і забезпечити краще розуміння завдань кожного компонента системи.

Процеси позначають окремі функції чи операції. Дані - матеріали, які обробляються системою. Джерела даних це зовнішні системи або користувачі, які взаємодіють із системою. Словник даних це набір описів усіх елементів даних, які використовуються в системі, що дозволяє уніфікувати термінологію і спростити проєктування.

Етапами побудови функціональної моделі є : аналіз завдань, визначення функцій, декомпозиція функцій.

Визначення основних завдань, які система повинна вирішувати. Наприклад, у системі тепличного виробництва завданням може бути моніторинг кліматичних умов, контроль поливу та прогнозування врожайності. Функціональний підхід забезпечує легке сприйняття системи завдяки її поділу на конкретні функції. Це особливо корисно на етапі аналізу вимог і початкового проєктування. Акцент на функціях системи дозволяє розробникам краще зрозуміти цілі системи та орієнтуватися на їх досягнення. Кожна функція системи може бути змінена чи вдосконалена незалежно, що сприяє гнучкості й адаптивності системи. [4] У контексті розробки експертної інформаційно-управляючої системи для тепличного виробництва функціональний підхід дозволяє розділити систему на такі функції:

- система отримує дані про температуру, вологість, освітлення та рівень co2 через сенсори;
- обробляє ці дані для виявлення відхилень від норми прогнозування врожайності;
- використовує історичні дані та алгоритми аналізу;
- оцінює ймовірність отримання оптимального врожаю за поточних умов;
- регулює обігрів, зрошення й вентиляцію залежно від результатів моніторингу;
- звіти та графіки для оцінки ефективності впроваджених заходів.

Функціональний підхід дозволяє створити просту у використанні модель, що ідеально підходить для розробки систем автоматизації, які відповідають специфічним потребам тепличного виробництва.

Python — це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис і динамічна

типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ. Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для настроюваних програм. У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Програмне забезпечення на Пайтоні оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

- у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;
- гнучкість та масштабованість Пайтона дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;
- розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;
- Пайтон – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;
- у Python — колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність. Ця мова програмування хоч і універсальна, проте повільніша за

інші. Велика кількість ресурсів, необхідних для роботи та «прив’язаність» до системних бібліотек.

Python підтримує кілька парадигм програмування: об’єктно-орієнтоване, процедурне та функціональне. Інтерпретована природа дозволяє виконувати код без компіляції, що прискорює розробку.Динамічна типізація полегшує роботу з типами даних і спрощує написання програм [5].

Широке застосування: Python використовується в галузях аналізу даних, штучного інтелекту, веб-розробки, автоматизації, фінансів. Python дозволяє аналізувати великі обсяги даних, зібраних із сенсорів у теплицях, створювати прогнози врожайності та розробляти алгоритми автоматизації.

UML (Unified Modelling Language або UML) — це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення. UML не є методом розробки, іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає вам наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками. Розробкою UML керує Object Management Group (OMG). Ця мова є загальноприйнятим стандартом графічного опису програмного забезпечення.

UML розроблено для розробки структури зорієнтованого на об’єкти програмного забезпечення, ця мова має дуже обмежену користь для програмування на основі інших парадигм. Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення. Елементи UML використовуються для побудови діаграм, які відповідають певній частині системи або точці зору на систему. Як видно, UML дозволяє візуалізувати структуру вашої програми. Крім того, UML не обмежується лише цим. За допомогою UML можна описувати різні процеси

всередині компанії або описувати бізнес-процес, в рамках якого працює функція.

Сховище даних – це особлива форма організації бази даних, що призначена для зберігання в погодженому вигляді агрегованої інформації, що отримується на основі баз даних різних систем та зовнішніх джерел. Сховища повинні надавати можливість параметризації даних за різними ознаками, наприклад банківські операції під час їх аналізу необхідно групувати за часом їх виконання, за клієнтами, за їх обсягами у вартісному виразі, за контрагентами, видами валют та іншими ознаками. Дані мають бути подані у сховищі таким чином, щоб надавати можливість їх багатовимірного аналізу.

Найбільш вдалою формою подання даних, що надасть можливість багатовимірної їх параметризації і подання даних у вигляді багатовимірної моделі. В основу OLAP-систем покладено поняття гіперкуба, тобто багатовимірного куба, у комірках якого зберігаються необхідні для аналізу дані.

Нині існує три варіанти побудови систем на основі сховищ даних: MOLAP, ROLAP і HOLAP.

У MOLAP – системі гіперкуб реалізується як спеціальна модель нереляційної структури, яка швидше забезпечує доступ до даних, ніж реляційні моделі, але вимагає додаткових витрат пам'яті.

У KOLAP – системах гіперкуб це лише користувацький інтерфейс, який моделюється на традиційній реляційній базі даних. Дані в сховищі подаються у вигляді моделі, що дістала назву "зірка". У цих системах зберігаються агреговані дані. Такий підхід дозволяє зберігати великі обсяги даних, але вони не досить ефективні при виконанні аналітичних операцій.

HOLAP – системи – це комбінований варіант зберігання даних, який використовує обидва типи СУБД. У багатовимірній СУБД зберігаються агрегати даних, а дані, які мають невеликий обсяг, зберігаються в реляційній СУБД.

Сховище даних характеризуються предметною орієнтацією, інтегрованістю, підтримкою хронології, незмінністю та мінімальною надлишковістю. Дані в сховищі даних організовані відповідно до основних напрямів діяльності організації. У цьому полягає відмінність сховищ даних від оперативної БД, в якій дані подаються відповідно до процесів (відвантаження товару, виписування рахунків і т.п.). Предметна організація даних не лише спрощує аналіз, а й значно прискорює проведення аналітичних розрахунків. Тобто сховища орієнтовані на бізнес-поняття, а не на бізнес – процеси.[6]

Первинні дані оперативних баз даних перевіряються, певним чином добираються, зводяться до одного вигляду, необхідною мірою агрегуються (тобто обраховуються сумарні показники) і завантажуються у сховище даних. Такі інтегровані дані набагато простіше аналізувати.

Дані, які вибираються з оперативних баз даних, накопичуються в сховищі даних у вигляді "історичних пластів", кожен із яких характеризує певний період часу. Це дозволяє проводити аналіз зміни показників у часі.

Дані сховища даних, що характеризують кожен "історичний пласт", ні в якому разі не підлягають змінам. Це теж є суттєвою відмінністю даних, що зберігаються у сховища даних, від оперативних даних. Оперативні дані можуть дуже часто змінюватися, з даними сховища можливі лише операції їх первинного завантаження, пошуку та їх читання.

Незважаючи на те, що інформація до сховищ даних завантажується з БД, це не призводить до надлишковості даних. Зведення до мінімуму надлишковості даних забезпечується тим, що перш ніж завантажувати дані до сховищ, їх фільтрують і певним чином очищають від таких даних, які не потрібні і не можуть бути використані в системах.

В основу інтелектуального аналізу даних покладена концепція шаблонів, що відбивають фрагменти багатоаспектних взаємин у даних. Ці шаблони являють собою закономірності, властиві підвибіркам даних, які можуть бути

компактно виражені у зрозумілій людині формі. Пошук шаблонів проводиться методами, не обмеженими апріорними припущеннями про структуру вибірки, та видами розподілів значень аналізованих показників. Важливе положення ІАД – нетривіальність розшукуваних шаблонів. Це означає, що знайдені шаблони повинні відбивати неочевидні, несподівані регулярності в даних, що становлять так звані приховані знання.

До методів і алгоритмів ІАД належать такі: штучні нейроні мережі, дерева рішень, символьні правила, методи найближчого сусіда і к-найближчого сусіда, метод опорних векторів, байєсові мережі, лінійна регресія, кореляційно-регресійний аналіз; ієрархічні методи кластерного аналізу, неієрархічні методи кластерного аналізу, зокрема і алгоритми к-середніх і к-медіани; методи пошуку асоціативних правил, зокрема алгоритм Apriori; метод обмеженого перебору, еволюційне програмування і генетичні алгоритми, різноманітні методи візуалізації даних і безліч інших методів. Варто зазначити, що більшість методів ІАД була розроблена у межах теорії штучного інтелекту. Єдиної думки щодо того, які задачі необхідно зараховувати до ІАД, немає. Більшість авторитетних джерел перераховує такі: класифікація, кластеризація, прогнозування, асоціація, візуалізація, аналіз виявлення відхилень, оцінювання, аналіз зв'язків, підведення підсумків.

Класифікація це найпростіша і найпоширеніша задача ІАД. В результаті розв’язання задачі класифікації виявляються ознаки, які характеризують групи об'єктів досліджуваного набору даних – класи; за цими ознаками новий об'єкт можна зарахувати до того чи іншого класу. Для розв’язання задачі класифікації можуть використовуватися методи: найближчого сусіда; к-ближнього сусіда; байєсових мереж; індукції дерев рішень; нейронних мереж [7].

Кластеризація є логічним продовженням ідеї класифікації. Це є складніша задача. Особливість кластеризації полягає у тому, що класи об'єктів спочатку не визначені. Результатом кластеризації є розбиття об'єктів на групи. Прикладом

методу задачі кластеризації є особливий вид нейроних мереж, що самоорганізовуються без вчителя.

Асоціація. У процесі розв’язання задачі пошуку асоціативних правил відшукуються закономірності міжзв'язаними подіями в наборі даних. Відмінність асоціації від двох попередніх задач ІАД: пошук закономірностей здійснюється не на основі властивостей об'єкта, що аналізується, а між кількома подіями, які відбуваються одночасно. Найвідоміший алгоритм розв’язку задачі пошуку асоціативних правил – алгоритм Apriori.

## 2.2 Об’єктне та функціональне моделювання

**2.2.1 Діаграма прецедентів.** Діаграма прецедентів є одним із ключових інструментів у методології UML (Unified Modeling Language) і використовується для моделювання функціональних вимог до системи. Вона забезпечує зручний спосіб опису сценаріїв використання системи, показуючи взаємодію між користувачами (акторами) і самою системою, що сприяє кращому розумінню очікуваної поведінки системи та її функціональних можливостей.

Діаграма прецедентів відображає взаємодію між системою та її зовнішнім середовищем, зосереджуючись на ролях користувачів і функціональності, яку вони очікують. Її основною метою є ідентифікація всіх можливих сценаріїв використання системи, а також їх візуалізація в наочній формі.

На діаграмі прецедентів кожна дія позначається еліпсом, що символізує функцію або процес у системі. Актори, які взаємодіють із цими прецедентами, зображуються у вигляді фігурок людини, а зв'язки між акторами та прецедентами позначаються лініями [8].

Основні елементи діаграми прецедентів

Діаграма прецедентів складається з таких ключових компонентів:

- Актори (Actors): Користувачі або зовнішні системи, які взаємодіють із системою. Вони можуть бути як людськими (наприклад, агроном), так і технічними (наприклад, сенсорна система);
- Прецеденти (Use Cases): Сценарії використання системи, які представляють певні дії або процеси, що виконуються в системі. Наприклад, "Збір даних із сенсорів" або "Генерація звіту";
- Система (System): Вона зазвичай відображається у вигляді прямокутника, що охоплює всі прецеденти, показуючи межі системи;
- Зв'язки (Associations): Лінії, що з'єднують акторів із прецедентами. Вони вказують на те, які актори беруть участь у виконанні певного сценарію;
- Включення (Include): Відношення між прецедентами, яке показує, що один прецедент є обов’язковою частиною іншого. Наприклад, "Авторизація користувача" може бути частиною "Перегляду даних сенсорів";
- Розширення (Extend): Відношення між прецедентами, що показує, що додатковий прецедент виконується лише за певних умов.

Діаграми прецедентів дозволяють зосередитися на функціональності системи, яку очікують користувачі, і забезпечити її чітке розуміння як замовниками, так і розробниками. Вони можуть бути деталізованими або абстрактними залежно від потреб проєкту.

Спроєктована діаграма прецедентів представлена на рис.3.

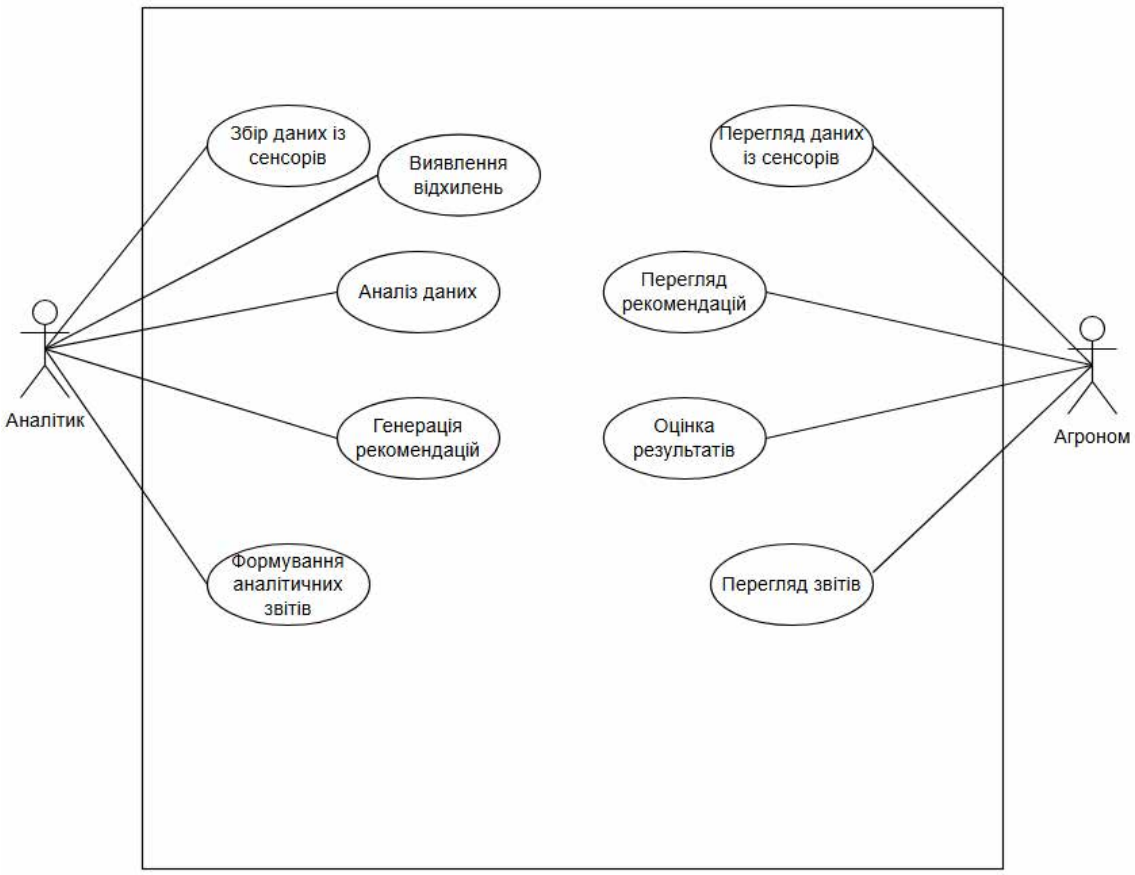


Рис. 3 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- Агроном;
- Аналітик.

Актор «Аналітик» включає такі прецеденти:

- Збір даних із сенсорів;
- Виявлення відхилень;
- Аналіз даних;
- Генерація рекомендацій;
- Формування аналітичних звітів.

Актор «Агроном» включає такі прецеденти:

- Перегляд даних із сенсорів;

- Перегляд рекомендацій;
- Оцінка результатів;
- Перегляд звітів.

**2.2.2 Діаграма послідовності.** Діаграма послідовності є одним із найважливіших інструментів для моделювання взаємодії між елементами системи у часовому контексті. Вона дозволяє детально відобразити, як відбувається обмін повідомленнями між об'єктами, які дії виконуються в системі та в якій послідовності. Це робить діаграми послідовності незамінним засобом для проєктування та аналізу складних систем, особливо тих, що мають багато взаємодій і процесів.

Діаграма послідовності використовується для відображення динамічного аспекту роботи системи, показуючи, як об'єкти взаємодіють між собою для виконання певного сценарію. Вона дозволяє простежити логіку виконання дій у конкретному сценарії, виділивши часову складову. Це особливо важливо для систем із послідовними або асинхронними операціями, де порядок виконання дій є критичним [9].

Ключовою особливістю діаграми послідовності є її здатність описати, хто саме ініціює ті чи інші дії, як відбувається обмін інформацією між компонентами системи та як формується результат виконання операцій. Діаграма послідовності придставлена на рисунку 4.

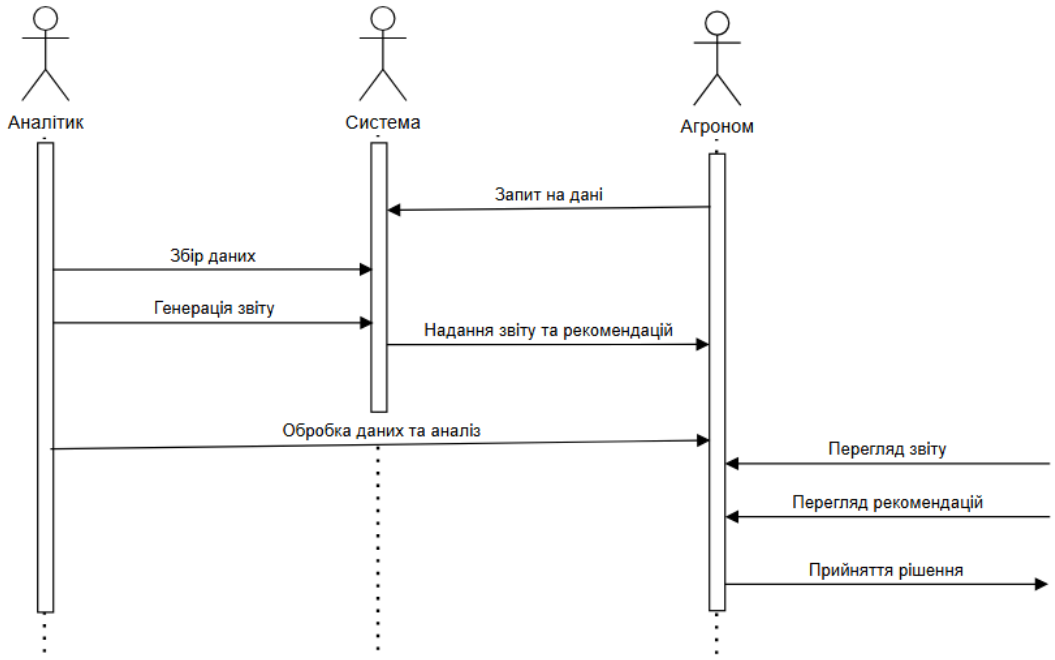


Рис. 4 Діаграма послідовності

Діаграма послідовності включає кілька основних елементів, які дозволяють відобразити логіку взаємодії об'єктів у системі:

- Актори (Actors): Елементи, що представляють користувачів або зовнішні системи, які взаємодіють із системою. Вони зазвичай позначаються фігурками людини
- Об'єкти (Objects): Елементи системи, які виконують певні дії або зберігають дані. На діаграмі вони позначаються прямокутниками;
- Життєві лінії (Lifelines): Вертикальні лінії, що починаються від акторів або об'єктів. Вони відображають час існування об'єкта або актора у межах сценарію;
- Повідомлення (Messages): Лінії зі стрілками, що показують обмін інформацією між актором і об'єктом або між об'єктами. Вони можуть бути синхронними, асинхронними або відкладеними;
- Активність (Activation): Прямокутники, розташовані на життєвій лінії, які позначають період, коли об'єкт виконує певну дію;

- Повернення (Return): Повідомлення, що позначають результат виконання дії.

Особливості моделювання діаграми послідовності

Однією з найважливіших характеристик діаграми послідовності є її здатність візуалізувати, як і коли об'єкти взаємодіють у межах сценарію. На відміну від статичних діаграм, таких як діаграми класів, діаграма послідовності акцентує увагу на динаміці процесів.

Часова шкала діаграми представлена вертикально: зверху вниз відображається послідовність дій. Це дозволяє легко простежити, які об'єкти і в якому порядку виконують операції.

Діаграми послідовності широко застосовуються для різних цілей у проєктуванні систем:

- Моделювання сценаріїв використання. Вони дозволяють детально описати, як система реагує на запити користувачів або інших систем;
- Відображення логіки роботи операцій. Діаграми послідовності показують, які методи викликаються у процесі виконання сценарію, як об'єкти обмінюються даними та як формується результат;
- Синхронізація процесів. Вони допомагають візуалізувати синхронні та асинхронні взаємодії, а також визначити можливі конфлікти;
- Аналіз взаємозалежностей. Завдяки діаграмі послідовності можна зрозуміти, як різні компоненти системи залежать один від одного;

У сучасних системах, що працюють у багатозадачних середовищах, важливо враховувати паралельність і асинхронність операцій. Діаграма послідовності дозволяє моделювати ці аспекти завдяки використанню спеціальних типів повідомлень. Синхронні повідомлення позначаються стрілкою з закритим наконечником, тоді як асинхронні — стрілкою з відкритим наконечником. Це дозволяє чітко розрізняти дії, що виконуються одразу, і ті, які можуть бути виконані пізніше [10].

**2.2.3 Діаграма діяльності**. Діаграма діяльності є одним із найважливіших інструментів для моделювання процесів у складних інформаційних системах. Її основне призначення — наочне представлення послідовності дій і логіки процесу, що дозволяє краще зрозуміти його структуру, взаємозв'язки між етапами, а також можливі варіанти розвитку сценаріїв. У сучасних системах, зокрема у проєктах із великою кількістю користувачів або паралельних обчислень, діаграми діяльності допомагають ефективно аналізувати і проєктувати різноманітні аспекти системи.

Діаграми діяльності є аналогом блок-схем, але з більшими можливостями для опису складних процесів. Вони відображають послідовність виконання дій у вигляді орієнтованого графу, де вершини представляють собою дії або операції, а ребра визначають переходи між ними. Це дозволяє візуалізувати логіку процесу в умовах, коли сценарії можуть мати альтернативні або паралельні гілки виконання. Діаграма діяльності придставлена на рисунку 5.

Діаграма діяльності включає кілька ключових елементів, що дозволяють детально описати логіку роботи процесу:

- Дія (Activity): Окрема операція або завдання, що виконується в рамках процесу. Це базовий елемент діаграми, який позначається прямокутником із закругленими кутами;
- Потік управління (Control Flow): Лінії, що з'єднують дії, показуючи послідовність виконання;
- Розгалуження (Decision): Позначається ромбом і використовується для створення умовних переходів між діями. Залежно від умови процес іде по одному з декількох напрямків;
- Злиття (Merge): Об'єднує декілька можливих шляхів у єдиний потік;
- Паралельність (Fork and Join): Позначається вертикальними лініями і використовується для поділу процесу на паралельні потоки або їх об'єднання;

- Об'єктні потоки (Object Flow): Вказують на передачу даних між діями
- Початкова і кінцева точки: Початок процесу позначається заповненим кругом, а завершення — колом із вкладеним заповненим кругом.

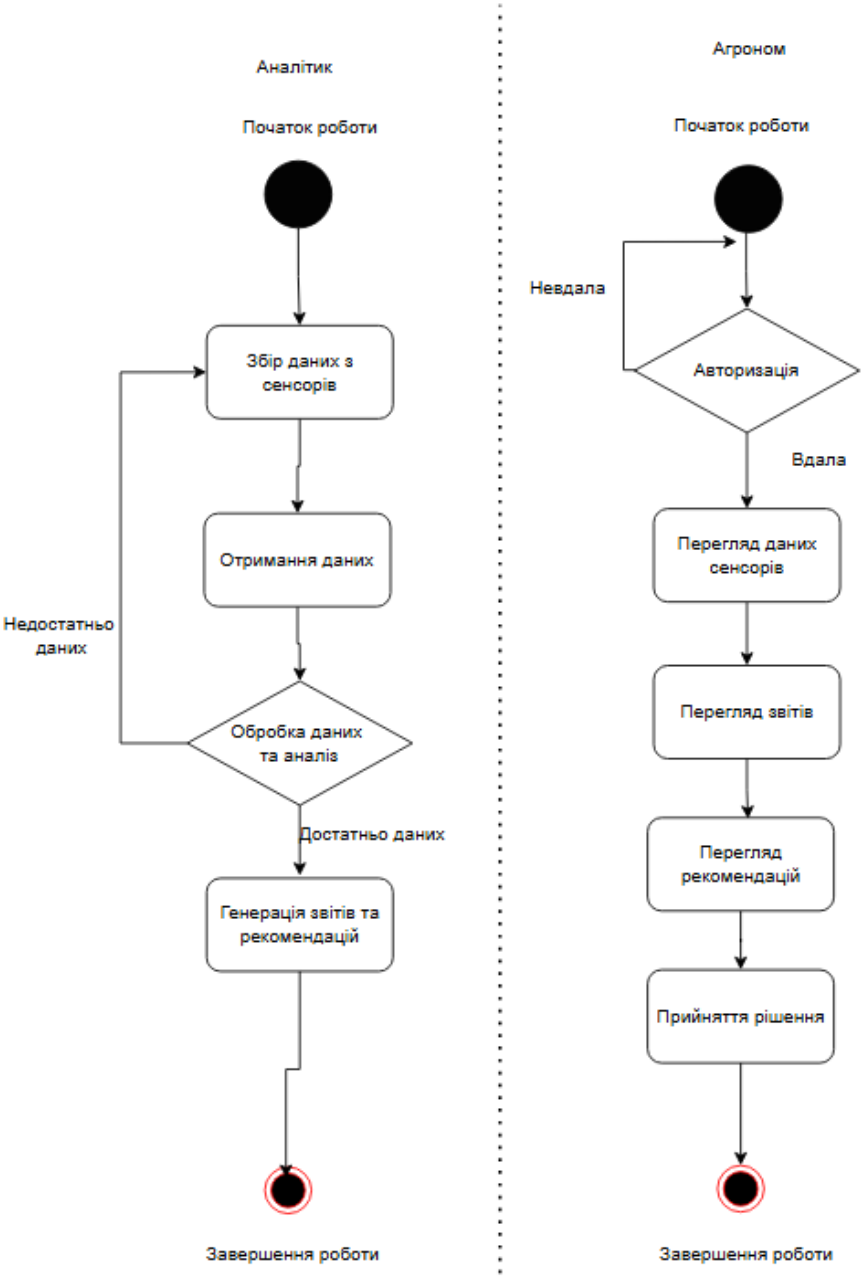


Рис. 5 Діаграма діяльності

Діаграми діяльності знаходять застосування у різних сферах розробки інформаційних систем. Зокрема, вони використовуються для:

Аналізу сценаріїв застосування. Діаграми дозволяють детально відобразити процес виконання сценаріїв, виділивши кожен окремий етап і визначивши, які дії мають виконуватися послідовно, а які — паралельно.

Вивчення взаємодії потоків робіт. У системах, що включають різноманітні сценарії, діаграми діяльності допомагають відстежити, як різні потоки взаємодіють між собою, і виявити можливі конфлікти або перехресні залежності.

Проєктування систем із багатопроцесорними обчисленнями. В умовах, коли виконання операцій може бути розподілене між кількома процесорами, діаграми діяльності дозволяють спланувати розподіл завдань і забезпечити їхню синхронізацію [11].

Однією з важливих особливостей діаграми діяльності є можливість моделювання паралельних процесів. Це особливо актуально для складних систем, які виконують кілька операцій одночасно. Паралельність забезпечується використанням елементів Fork і Join. Fork дозволяє поділити потік виконання на кілька незалежних, тоді як Join синхронізує їх і об'єднує назад в єдиний потік.

Паралельні процеси є невід'ємною частиною сучасних інформаційних систем, зокрема тих, які працюють із великими обсягами даних або використовують асинхронні алгоритми. Діаграма діяльності дозволяє визначити, як ці процеси мають взаємодіяти, і забезпечити коректність виконання.

Іншою важливою функцією діаграми діяльності є моделювання альтернативних шляхів виконання процесу. Це досягається за допомогою елементів Decision і Merge. Decision використовується для створення умовних переходів, які визначають, яким шляхом піде процес у залежності від певної умови. Merge дозволяє об'єднати ці шляхи назад після виконання.

Завдяки цьому діаграми діяльності стають потужним інструментом для опису сценаріїв, де рішення приймаються на основі даних, отриманих у ході виконання процесу.

Діаграми діяльності також широко використовуються для опису алгоритмів виконання операцій у рамках класів об'єктів. Це дозволяє детально візуалізувати, як саме реалізуються методи класів і які дії виконуються у процесі. Кожен стан на діаграмі відповідає виконанню окремої елементарної операції, а перехід до наступного стану відбувається лише після завершення поточної операції.

У рамках розробки експертної інформаційної системи для тепличного виробництва діаграми діяльності можуть бути використані для моделювання таких процесів, як збір даних із сенсорів, аналіз цих даних і генерація рекомендацій. Наприклад:

Збір даних із сенсорів розпочинається з ініціалізації запитів до сенсорних пристроїв. Потоки розподіляються між різними типами сенсорів, а після збору даних об'єднуються для аналізу. Після обробки даних алгоритмами кластеризації або виявлення аномалій діаграма відображає, як обирається подальший шлях: формування рекомендацій або попередження про критичні умови. Паралельні процеси можуть включати в себе одночасний збір даних та їхній аналіз.

Діаграми діяльності забезпечують низку переваг, зокрема:

- чітке уявлення про логіку виконання процесів;
- можливість відобразити складні сценарії з альтернативами та паралелями;
- використання зрозумілих графічних елементів, що полегшує сприйняття інформації;
- широкі можливості для аналізу і проєктування систем.

# 3 РОЗРОБКА СИСТЕМИ

ERwin Data Modeler (Enterprise Architecture and Modeling Tool) — це провідне програмне забезпечення для моделювання даних, яке допомагає фахівцям створювати та оптимізувати концептуальні, логічні й фізичні моделі баз даних. Воно широко використовується у сфері розробки баз даних, інтеграції даних, управління метаданими та інших аспектів управління даними. Інструмент надає інтуїтивний інтерфейс і потужний функціонал, що дозволяє як початківцям, так і досвідченим фахівцям ефективно працювати з різними типами моделей.

Основні можливості ERwin Data Modeler

ERwin Data Modeler призначений для створення комплексних моделей даних, що відображають структуру, взаємозв'язки та логіку роботи системи. Програмне забезпечення підтримує весь життєвий цикл роботи з даними, починаючи з аналізу вимог і закінчуючи впровадженням у виробниче середовище [12].

Основні компоненти ERwin діаграм включають:

- Сутності (Entities) - вони представляють реальні або концептуальні об'єкти в системі. Сутності зображаються у вигляді прямокутників з назвою сутності всередині;
- Атрибути (Attributes) - вони представляють властивості або характеристики сутностей. Атрибути пов'язані зі сутністю та відображаються як овали всередині прямокутника сутності;
- Зв'язки (Relationships) - вони відображають зв'язки між сутностями. Зв'язки можуть мати тип один-до-одного, один-до-багатьох або багато-до-багатьох. Зв'язки показуються лініями, що з'єднують сутності, і мають назву та тип;

- Ключі (Keys) - вони використовуються для ідентифікації унікальних записів у таблицях бази даних. Ключі можуть бути первинними ключами (Primary Keys) або зовнішніми ключами (Foreign Keys). Вони позначаються спеціальними символами на діаграмі.

Однією з ключових переваг ERwin є підтримка всього життєвого циклу роботи з даними. Програмне забезпечення дозволяє почати з моделювання на концептуальному рівні, поступово деталізуючи структуру бази даних. На етапі фізичного моделювання користувач може автоматично генерувати SQL-скрипти для створення бази даних у різних системах управління, таких як PostgreSQL, MySQL, Oracle чи Microsoft SQL Server. Також є можливість імпортувати існуючі бази даних для їх зворотного інжинірингу, створюючи візуальні моделі на основі вже наявної структури.

ERwin Data Modeler також ефективно вирішує завдання управління метаданими. Він дозволяє зберігати й систематизувати інформацію про дані, що полегшує інтеграцію між різними системами та забезпечує збереження цілісності даних у корпоративному середовищі. Інструмент широко використовується для документування баз даних, що спрощує їх супровід і полегшує розуміння складної архітектури даних для нових членів команди.

Ще однією важливою перевагою ERwin є його здатність до візуалізації. Завдяки можливостям графічного представлення моделей користувачі можуть легко аналізувати структуру бази даних, виявляти можливі недоліки, такі як дублювання даних або слабкі місця у взаємозв’язках, і оптимізувати проект ще до його впровадження.

Серед областей, де ERwin знаходить застосування, можна виділити управління корпоративними даними, розробку програмного забезпечення, бізнес-аналітику та освітню сферу. У корпоративному середовищі ERwin допомагає стандартизувати підходи до управління даними, інтегрувати інформацію з різних джерел та створювати сховища даних. У сфері розробки ПЗ

інструмент використовується для моделювання баз даних, їх документування та автоматизації ключових процесів. Бізнес-аналітики застосовують ERwin для створення моделей, які допомагають аналізувати великі обсяги даних, а також формувати основи для аналітичних платформ.

Водночас ERwin має певні обмеження. Наприклад, вартість програмного забезпечення може бути бар’єром для невеликих компаній або окремих користувачів. Також інструмент вимагає сучасного комп’ютерного обладнання для ефективної роботи. Незважаючи на ці недоліки, його функціонал і зручність значно переважають недоліки, роблячи ERwin одним із найпотужніших рішень для проектування баз даних. Приклад ER діаграми придставлений на рисунку 6.

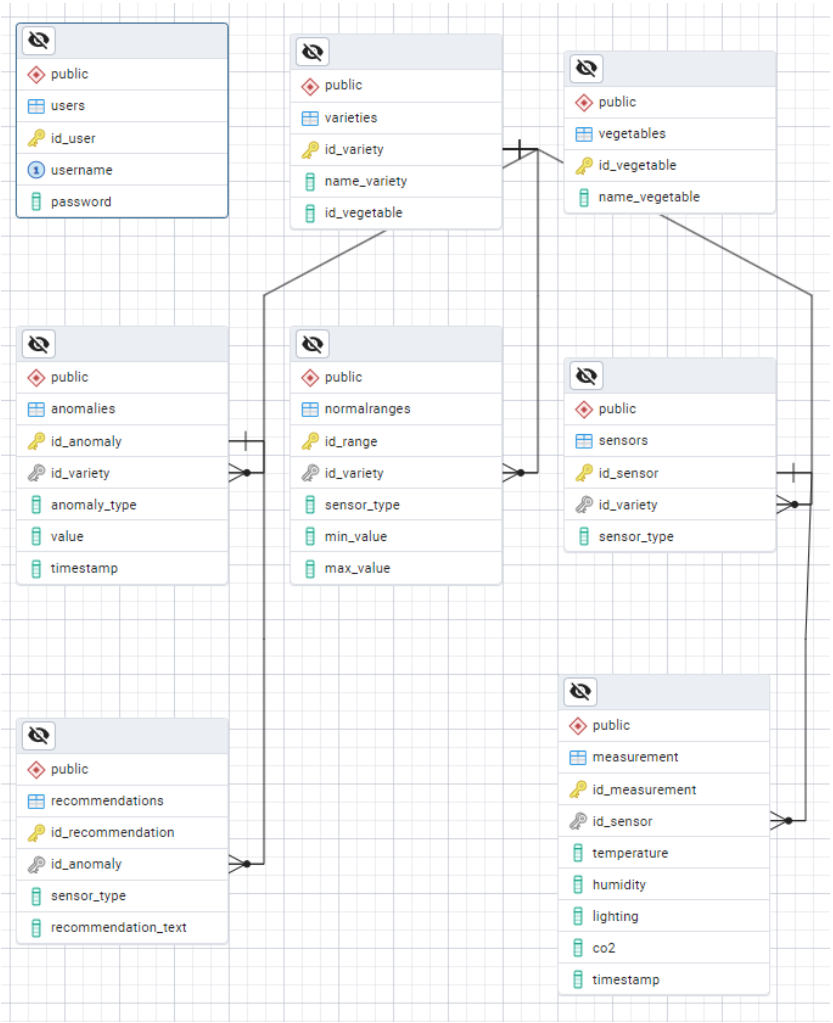


Рис.6 ER діаграма

Сутність Users (Користувачі) служить для авторизації агрономів, що користуватимуться експертною системою. Вона містить атрибути:

- id_user (PRIMARY KEY);
- username;
- password.

Сутність Recommendations служить для генерації рекомендації по вирощуванню овочів. Вона містить атрибути:

- id_recommendation (PRIMARY KEY);
- id_anomaly (FOREIGN KEY);
- sensor_type;
- recommendation_text.

Сутність Sensors містить в собі інформацію про сенсори. Вона містить атрибути:

- id_sensor (PRIMARY KEY);
- id_variety(FOREIGN KEY);
- sensor_type.

Сутність Vegetables містить в собі інформацію про овочі, що вирощуються в теплиці. Вона містить атрибути:

- id_vegetable (PRIMARY KEY);
- name_vegetable.

Сутність Varieties містить в собі дані про сорти овочей, що вирощуються в теплиці. Вона містить атрибути:

- id_variety (PRIMARY KEY);

- id_vegetable (FOREIGN KEY);
- name_variety.

Сутність Measurement зберігає в собі дані вимірювать. Вона містить атрибути:

- id_measurement (PRIMARY KEY);
- id_sensor (FOREIGN KEY);
- temperature;
- humidity;
- lighting;
- co2;
- timestamp.

Сутність Anomaly служить для фіксації відхилень і зберігання усіх зафіксованих вимірювань. Вона містить атрибути:

- id_anomaly(PRIMARY KEY);
- id_variety(FOREIGN KEY);
- anomaly_type;
- value;
- timestamp.

Сутність NormalRanges зберігає в собі рекомендовані значення показників для різних рослин і сортів. Вона містить атрибути:

- id_range(PRIMARY KEY);
- id_variety(FOREIGN KEY);
- sensor_type;
- min_value;
- max_value.

## 3.1 Вибір системи управління базою даних та її реалізація

Система Системи управління базами даних (СУБД) є ключовою складовою сучасного інформаційного середовища. Вони забезпечують ефективне створення, організацію, зберігання, доступ, обробку та керування даними. СУБД – це програмне забезпечення, яке взаємодіє з базами даних, дозволяючи користувачам додавати, змінювати, видаляти та аналізувати дані в організованій формі.СУБД відіграють важливу роль у забезпеченні зручності та надійності доступу до великих обсягів даних. Вони використовуються в різноманітних галузях, від банківської сфери до медицини, від логістики до веб-розробки. Основна мета СУБД – забезпечити користувачам простий і зрозумілий інтерфейс для роботи з даними, приховуючи складні аспекти їх зберігання та обробки [13].

Однією з ключових характеристик СУБД є їх здатність забезпечувати цілісність даних. Вони стежать за тим, щоб дані були збережені коректно і не порушували встановлених правил. Наприклад, СУБД не дозволить зберегти запис про замовлення без даних про клієнта. Важливою є також підтримка транзакцій – функції, яка забезпечує виконання послідовності дій, що повинні або повністю завершитися успішно, або повністю скасуватися у разі помилки.

СУБД поділяються на різні типи залежно від їхньої структури даних і способу роботи. Реляційні СУБД, такі як PostgreSQL, працюють з таблицями, що складаються зі стовпців і рядків. Інші типи включають документно-орієнтовані системи, графові бази даних і системи, побудовані на ключах і значеннях. Кожен тип має свої переваги й обмеження, і вибір залежить від конкретних вимог проєкту.

PostgreSQL є однією з найпотужніших і найпоширеніших реляційних СУБД. Вона виділяється своєю відкритою ліцензією, яка дозволяє будь-кому використовувати, змінювати та поширювати її без обмежень. PostgreSQL була

створена в 1986 році під назвою POSTGRES як дослідницький проєкт Каліфорнійського університету в Берклі. З часом система еволюціонувала в потужну платформу для роботи з даними, яка отримала широке визнання в різних галузях.

Основною особливістю PostgreSQL є її відповідність стандарту ACID (атомарність, узгодженість, ізольованість і довговічність), що гарантує надійність і безпечність транзакцій. Це забезпечує коректність операцій навіть у разі технічних збоїв або одночасної роботи багатьох користувачів. PostgreSQL також пропонує підтримку розширених типів даних, таких як JSON, XML, геопросторові дані, масиви й користувацькі типи, що робить її універсальним інструментом для роботи з будь-якими обсягами та видами інформації.

Однією з найважливіших переваг PostgreSQL є її модульність і розширюваність. Вона дозволяє створювати власні функції, оператори, типи даних і навіть спеціалізовані індекси, які відповідають унікальним вимогам проєкту. Це забезпечує високу гнучкість і адаптивність системи. PostgreSQL також підтримує складні запити й аналітику, дозволяючи виконувати обчислення безпосередньо в базі даних, що знижує навантаження на зовнішнє програмне забезпечення [14].

Ще однією перевагою PostgreSQL є її масштабованість і продуктивність. Вона здатна ефективно обробляти як невеликі, так і великі бази даних, завдяки оптимізованим алгоритмам зберігання й пошуку даних. Розподілена архітектура PostgreSQL дозволяє використовувати її в середовищах із великим навантаженням, забезпечуючи високу доступність і відмовостійкість.

PostgreSQL підтримує механізми реплікації й шардінгу, які дозволяють розподіляти дані між різними серверами для підвищення продуктивності й забезпечення відмовостійкості. Це особливо важливо для критичних застосунків, таких як банківські системи, платформи електронної комерції та

інші сервіси, де затримки або втрата даних є неприйнятними. Приклад заповненої таблиці представлений на рисунку 7.

| vegetable_name character varying (50) | variety_name character varying (50) | date_measurement timestamp without time zone | temperature double precision | humidity double precision | lighting double precision | co2 double precision |
|---|---|---|---|---|---|---|
| Cucumber | Acord F1 | 2023-04-01 00:00:00 | 16.772507371869963 | 89.58896566020486 | 50.95357767506752 | 828.7882554721737 |
| Cucumber | Acord F1 | 2023-04-01 01:00:00 | 15.355213998442176 | 86.98923714889972 | 60.650832226819205 | 992.1040435344448 |
| Cucumber | Acord F1 | 2023-04-01 02:00:00 | 18.572936767983705 | 89.58613315597262 | 29.085884952643625 | 891.1923945911238 |
| Cucumber | Acord F1 | 2023-04-01 03:00:00 | 17.00553489315641 | 89.115124659024 | 52.990632015764575 | 956.6902256730508 |
| Cucumber | Acord F1 | 2023-04-01 04:00:00 | 15.043421343863342 | 75.26842821002886 | 20.934510031248777 | 905.9274222373406 |
| Cucumber | Acord F1 | 2023-04-01 05:00:00 | 16.10991527498293 | 73.70883662453012 | 26.58221425131353 | 987.2049309111901 |
| Cucumber | Acord F1 | 2023-04-01 06:00:00 | 20.763531589114002 | 82.29746226224756 | 564.1990073252259 | 939.6144761607914 |
| Cucumber | Acord F1 | 2023-04-01 07:00:00 | 23.822899605365237 | 71.21950951528949 | 469.8175655018199 | 831.2790777168456 |
| Cucumber | Acord F1 | 2023-04-01 08:00:00 | 24.079410558045282 | 76.76048431089315 | 405.0008446661127 | 850.1509827610448 |
| Cucumber | Acord F1 | 2023-04-01 09:00:00 | 27.508005779803675 | 75.169673889536 | 518.5185046710101 | 819.9999081974879 |
| Cucumber | Acord F1 | 2023-04-01 10:00:00 | 22.921705482525027 | 82.62380766723707 | 688.1823028816048 | 823.6386468408682 |
| Cucumber | Acord F1 | 2023-04-01 11:00:00 | 23.108527737680422 | 87.10418701833393 | 471.9243736678743 | 855.6568601213353 |
| Cucumber | Acord F1 | 2023-04-01 12:00:00 | 20.484289019137687 | 79.49221785955368 | 585.5615055685803 | 886.3443682866857 |
| Cucumber | Acord F1 | 2023-04-01 13:00:00 | 26.587030361325706 | 74.25783129908655 | 607.297134066804 | 884.2516852871581 |
| Cucumber | Acord F1 | 2023-04-01 14:00:00 | 24.39631669869918 | 88.84698855032819 | 509.67328522193736 | 873.697386995238 |
| Cucumber | Acord F1 | 2023-04-01 15:00:00 | 20.769994671077832 | 80.68414954879111 | 568.4876336830312 | 901.2657940054594 |
| Cucumber | Acord F1 | 2023-04-01 16:00:00 | 27.37322909665432 | 88.84744734323245 | 510.2746488541976 | 940.0147859648209 |
| Cucumber | Acord F1 | 2023-04-01 17:00:00 | 27.790768308283187 | 88.7832273240411 | 627.9834410075596 | 983.9351443174143 |
| Cucumber | Acord F1 | 2023-04-01 18:00:00 | 18.220704406936 | 78.6651129838844 | 82.21556930529411 | 938.0602251719004 |
| Cucumber | Acord F1 | 2023-04-01 19:00:00 | 16.85956748934936 | 85.8772814555366 | 4.368089514110807 | 900.4917456365597 |
| Cucumber | Acord F1 | 2023-04-01 20:00:00 | 19.78892661327393 | 77.83759506457984 | 61.9560949406723 | 898.4914031964122 |
| Cucumber | Acord F1 | 2023-04-01 21:00:00 | 14.451851997454378 | 84.12633498382459 | 34.40091864265036 | 974.4924230567707 |
| Cucumber | Acord F1 | 2023-04-01 22:00:00 | 18.22716541733387 | 84.68008683135116 | 28.68609013518387 | 848.2199684723705 |
| Cucumber | Acord F1 | 2023-04-01 23:00:00 | 17.948111390345122 | 85.12598455386318 | 70.09566943867223 | 855.3659096819767 |

Рис.7 Приклад даних, що зберігаються у сховищі даних

Що стосується безпеки, PostgreSQL надає багаторівневий підхід до захисту даних. Вона підтримує шифрування з'єднань, контроль доступу на основі ролей і детальне налаштування прав доступу до об'єктів бази даних. Це дозволяє забезпечити конфіденційність і захист інформації навіть у складних середовищах з великою кількістю користувачів.

Завдяки своїм широким можливостям і стабільності, PostgreSQL отримала визнання серед розробників і підприємств по всьому світу. Вона використовується в різних галузях, включаючи фінанси, телекомунікації, охорону здоров'я, розробку програмного забезпечення й державні організації.

PostgreSQL є основою для багатьох хмарних сервісів і платформ, таких як AWS RDS, Google Cloud SQL і Heroku.

У тепличному виробництві PostgreSQL ідеально підходить для створення експертних систем завдяки її підтримці складних аналітичних запитів, роботи з великими обсягами даних і інтеграції з інструментами машинного навчання. Вона дозволяє зберігати дані з сенсорів, вести історичний аналіз і виконувати прогнозування, що є ключовими аспектами для оптимізації виробничих процесів і підвищення врожайності.

## 3.2 Використання 1-Rule для класифікації

Алгоритм 1-Rule працює шляхом створення єдиного правила класифікації на основі одного атрибуту (ознаки) в наборі даних. Він досліджує кожен атрибут окремо, визначає найпоширеніший клас для кожного значення атрибута, а потім вибирає атрибут, який виробляє найнижчу частоту помилок як основу для правила класифікації. Процес простий:

Вибір атрибутів: Алгоритм оцінює кожен атрибут (або особливість) в наборі даних. Для кожного атрибута він розглядає всі можливі значення і визначає, наскільки добре вони передбачають цільову змінну.

Створення правила: Для кожного значення атрибута 1R призначає найбільш частий клас, який відповідає цьому значенню. Це створює просте правило, яке відображає кожне значення атрибута в клас.

Розрахунок помилки: Алгоритм обчислює частоту помилок для правила шляхом порівняння прогнозованого класу з фактичними мітками класу в наборі даних. Частота помилок - це частка випадків, коли прогноз неправильний.

Найкращий вибір правил: атрибут з найнижчою швидкістю помилок вибирається як основа для остаточного правила. Це правило потім використовується для класифікації нових екземплярів [15].

Класифікація: Коли новий екземпляр представлений, значення обраного атрибута використовується для передбачення класу відповідно до правила.

Простота алгоритму 1R полягає в його фокусі лише на одному атрибуті, що робить його легким для розуміння та інтерпретації. Однак це також означає, що він не може захоплювати складні взаємодії між кількома атрибутами. Тим не менш, він часто служить цінним інструментом для вивчення аналізу даних та розуміння важливості окремих атрибутів у наборі даних.

Резюме та агрегація: Нарешті, код агрегує результати за різноманітністю, обчислюючи загальну кількість вимірювань, кількість вимірювань нижче та вище середнього, середню температуру, загальну похибку та загальну класифікацію успіху. Результати використання алгоритму відображені на рисунках 8,9 та 10.

```
PS C:\Users\Danil\MyWinFormsProject> python 1rule.py
  variety_name  total_measurements  below_avg  above_avg  avg_temperature  total_error success
0   HabaHot F1                  24         12         12        20.391412    84.710608  Низька
1   Porteca F1                  24         12         12        20.454450    88.027030  Низька
```

Рис. 8 Правила для предиктора «Температури»

```
PS C:\Users\Danil\MyWinFormsProject> python 1rule.py
  variety_name  total_measurements  below_avg  above_avg  avg_humidity  total_error success
0   HabaHot F1                  24         11         13     74.139301   171.498607  Висока
1   Porteca F1                  24         13         11     72.295839   195.925423  Низька
```

Рис. 9 Правила для предиктора «Вологості»

```
PS C:\Users\Danil\MyWinFormsProject> python 1rule.py
  variety_name  total_measurements  below_avg  above_avg  avg_lighting  total_error success
0   HabaHot F1                  24         12         12    340.473847  6915.877348  Низька
1   Porteca F1                  24         12         12    326.741504  6675.019339  Низька
```

Рис.10 Правила для предиктора «Освітлення»

## 3.3 Використання методу Наївного Байєса

Naive Bayes - сімейство імовірнісних алгоритмів, заснованих на теоремі Баєса, що використовуються для задач класифікації. Він названий «наївним», оскільки припускає, що функції в наборі даних незалежні один від одного, що часто не є випадком у реальних даних. Незважаючи на це спрощення, було виявлено, що класифікатори Наївного Баєса надзвичайно добре виконують різноманітні складні завдання класифікації, включаючи фільтрацію спаму, аналіз настроїв та медичну діагностику.

Теорема Баєса є математичною основою наївного алгоритму Баєса. Він описує ймовірність події на основі попереднього знання умов, які можуть бути пов'язані з подією.

«Наївний» аспект алгоритму наївного Баєса відноситься до припущення, що ознаки умовно незалежні з урахуванням мітки класу [16]. Це означає, що алгоритм передбачає, що наявність або відсутність певної ознаки не впливає на наявність або відсутність будь-якої іншої ознаки. Матриця плутанити розміщена на рисунку 11.

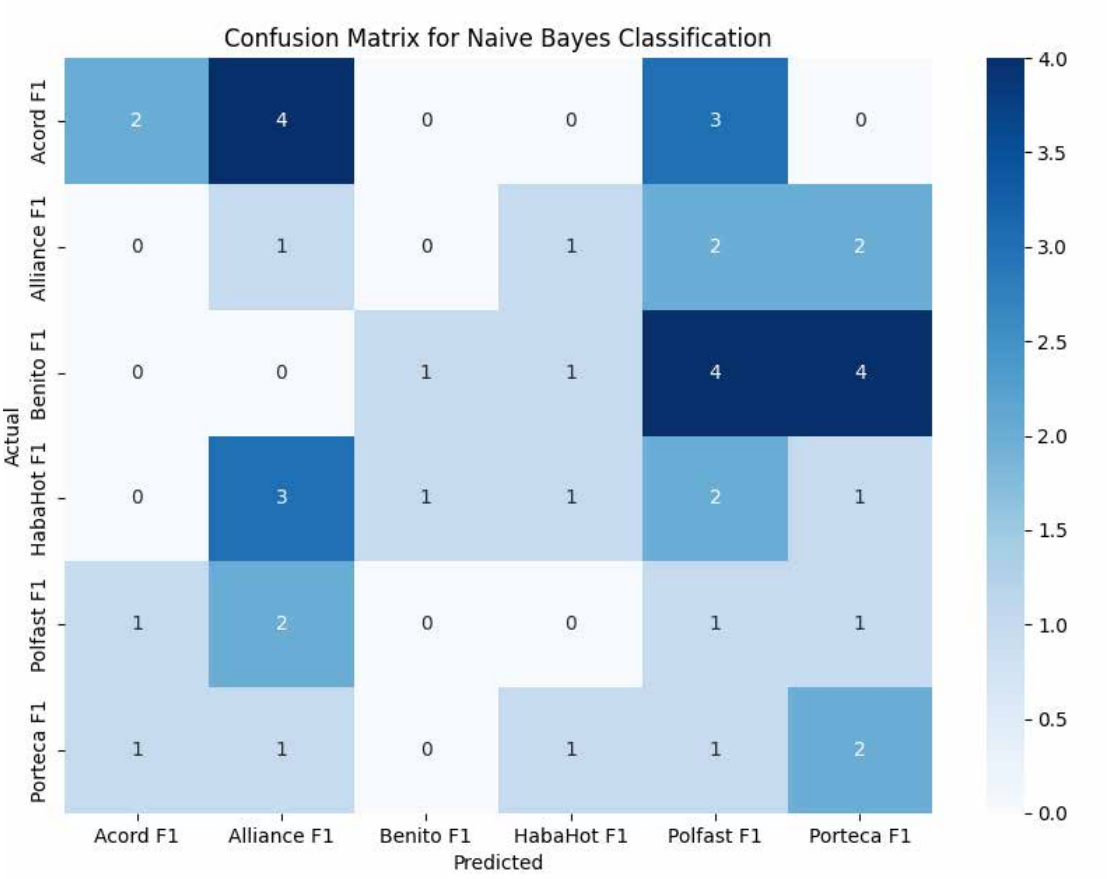


Рис. 11 Матриця плутанини

Це припущення спрощує обчислення, роблячи алгоритм високоефективним і масштабованим, навіть для великих наборів даних. Однак насправді особливості часто демонструють певний рівень кореляції, який наївний Баєс не враховує. Незважаючи на це, алгоритм все ще може добре працювати, особливо коли залежності ознак слабкі або правильна класифікація залежить в першу чергу від внеску окремих ознак.

Існує кілька варіантів наївного класифікатора Баєса, кожен з яких пристосований до різних типів даних:

- Гауссовий наївний Баєс: Припускає, що неперервні ознаки слідують гауссовому (нормальному) розподілу. Цей варіант часто використовується, коли функції є безперервними і нормально розподіленими;
- Поліноміальний наївний Байес: Зазвичай використовується для дискретних ознак, таких як кількість слів у задачах класифікації тексту;

- Бернуллі Наївний Баєс: Підходить для бінарних/булевих ознак, де кожна особливість моделюється як присутня або відсутня.

## 3.4 Пошук асоціативних правил

Пошук асоціативних правил є одним із основних напрямів інтелектуального аналізу даних (Data Mining). Цей метод дозволяє виявляти закономірності між елементами у великих масивах даних, що є особливо корисним для бізнесу, управління та наукових досліджень. Асоціативні правила допомагають формувати рекомендації, приймати рішення та розробляти стратегії на основі даних. Наприклад, у тепличному виробництві можна виявити, як певні умови (температура, вологість, освітлення) впливають на врожайність різних сортів овочів.

Основна мета пошуку асоціативних правил — ідентифікація закономірностей, які визначають спільне виникнення або співвідношення певних елементів у базі даних. При цьому використовується база транзакцій, де кожна транзакція є набором елементів. Класичним прикладом є аналіз кошика покупок у супермаркеті, коли система визначає, які товари найчастіше купують разом.

Асоціативні правила формуються у вигляді імплікацій: $A \rightarrow B$, де:

- A та B — множини елементів (або елементи), які належать до бази даних;
- A — ліва частина правила, яка називається антецедент;
- B — права частина правила, або консеквент.

Суть правила полягає у тому, що якщо в даних є елементи з множини A, то з високою ймовірністю є й елементи з множини B. Для оцінки значущості асоціативних правил використовують кілька метрик:

- Support (Підтримка): частка транзакцій, де зустрічається як A, так і B. Вона показує, наскільки поширене це правило у вибірці.
- Confidence (Достовірність): ймовірність, з якою наявність A передбачає наявність B. Це міра точності правила.
- Lift (Підсилення): співвідношення достовірності правила до очікуваної ймовірності появи B незалежно від A. Lift > 1 вказує на позитивну залежність між A і B.

Найпоширенішим алгоритмом є Apriori. Його принцип полягає у побудові частих наборів елементів, з яких потім генеруються правила. Спочатку визначаються всі часті елементи, що перевищують заданий поріг підтримки. Потім формуються більші набори елементів, які також перевіряються на підтримку. У кінці з частих наборів генеруються правила, що мають високу достовірність і підсилення. Наприклад, якщо в базі даних аналізується 1000 транзакцій, і в 200 з них зустрічається набір {температура = висока, вологість = низька}, то підтримка такого набору становить 20%. Якщо з цих 200 випадків у 180 спостерігалося збільшення врожайності, то достовірність правила {температура = висока, вологість = низька} → {врожайність = висока} становить 90%. Інший популярний алгоритм — FP-Growth, який будує дерево частих наборів і значно зменшує кількість ітерацій порівняно з Apriori. Знаходження правил відображено на рисунку 12.

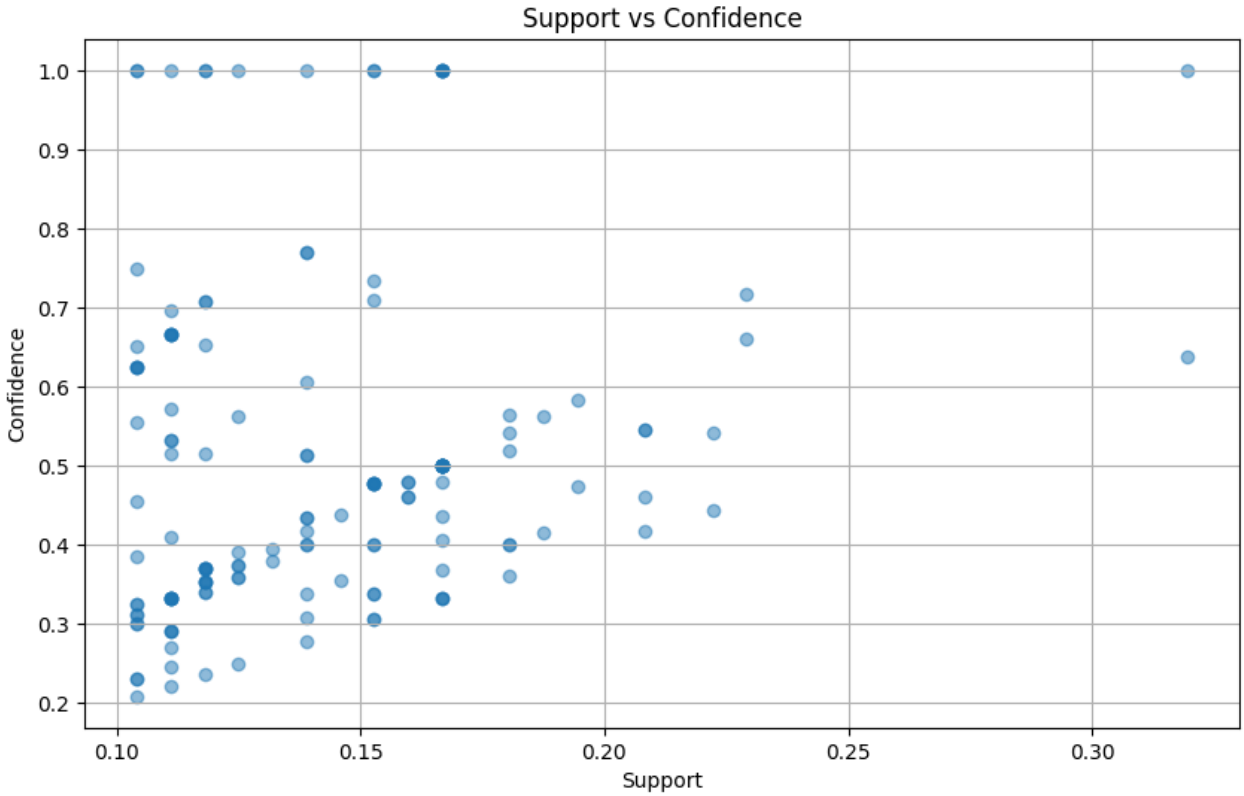


Рис. 12 Відображення знайдених правил під час обробки моделі

У тепличному виробництві асоціативні правила можуть допомогти визначити, що певна комбінація рівнів освітлення і вологості сприяє оптимальному росту конкретного сорту рослин. Це дає змогу ефективніше використовувати ресурси, знижувати витрати і підвищувати якість продукції. Попри свою корисність, пошук асоціативних правил має певні обмеження. Для великих баз даних кількість можливих комбінацій елементів експоненційно зростає, що ускладнює аналіз.  Якщо в даних є багато випадкових або нерелевантних записів, це може знижувати якість знайдених правил. Знайдені правила можуть бути складними для розуміння без глибокого знання предметної області. Правила на основі на фактичних даних відображено на рисунку 13.

| | antecedents | consequents | support | confidence | lift |
|---|---|---|---|---|---|
| 0 | (variety_name_Acord F1) | (vegetable_name_Cucumber) | 0.166667 | 1.000000 | 3.000000 |
| 1 | (vegetable_name_Cucumber) | (variety_name_Acord F1) | 0.166667 | 0.500000 | 3.000000 |
| 2 | (vegetable_name_Cucumber) | (variety_name_Alliance F1) | 0.166667 | 0.500000 | 3.000000 |
| 3 | (variety_name_Alliance F1) | (vegetable_name_Cucumber) | 0.166667 | 1.000000 | 3.000000 |
| 4 | (temperature_Medium) | (vegetable_name_Cucumber) | 0.111111 | 0.333333 | 1.000000 |
| .. | ... | ... | ... | ... | ... |
| 149 | (humidity_High, lighting_Low) | (co2_High) | 0.111111 | 0.516129 | 1.351320 |
| 150 | (co2_High, lighting_Low) | (humidity_High) | 0.111111 | 0.533333 | 1.181538 |
| 151 | (humidity_High) | (co2_High, lighting_Low) | 0.111111 | 0.246154 | 1.181538 |
| 152 | (co2_High) | (humidity_High, lighting_Low) | 0.111111 | 0.290909 | 1.351320 |
| 153 | (lighting_Low) | (humidity_High, co2_High) | 0.111111 | 0.222222 | 1.066667 |

Рис.13 Знайдені правила на основі фактичних даних

Сучасні програмні інструменти, такі як Python, R і SQL, дозволяють ефективно виконувати пошук асоціативних правил. Зокрема, бібліотеки mlxtend у Python або пакети arules в R мають вбудовані функції для реалізації алгоритмів Apriori та FP-Growth. Дані зазвичай зберігаються у вигляді транзакцій у реляційних базах або у спеціалізованих сховищах даних.

Зв'язки між наборами предметів, також відомими як «асоціації», лежать в основі видобутку правил асоціації. Ці зв'язки показують, як елементи спільно відбуваються в рамках транзакцій і дають уявлення про основні відносини всередині даних.

Часті набори предметів: основна увага приділяється виявленню наборів елементів, які часто з'являються разом. Наприклад, у системі парникового виробництва часті набори предметів можуть включати конкретні комбінації показань датчиків, які вказують на оптимальні умови вирощування для певних культур.

Ієрархії набору елементів: ітеративний підхід алгоритму Апріорі часто виявляє ієрархії всередині наборів елементів. Наприклад, частий набір предметів розміру 2 може бути частиною більшого набору частих предметів розміру 3. Ці ієрархії забезпечують більш глибоке розуміння того, як елементи пов'язані в різних контекстах.

Міцність асоціації: міцність зв'язків між наборами предметів визначається такими показниками, як підтримка, впевненість та підйом. Ці показники

допомагають визначити, які асоціації є достатньо сильними, щоб вважатися надійними для прийняття рішень.

Кінцевою метою створення правил асоціації є виявлення значущих та дієвих правил, які можуть інформувати про прийняття рішень. У нашому проекті виявлені правила можуть надати уявлення про оптимальні умови росту сільськогосподарських культур або взаємозв'язок між різними факторами навколишнього середовища .

Дієві висновки: правила, отримані з алгоритму Apriori, можуть призвести до дієвої інформації. Наприклад, якщо правило припускає, що певна комбінація температури та вологості призводить до підвищення врожайності для певної культури, менеджери теплиць можуть використовувати цю інформацію для оптимізації своїх систем клімат-контролю.

Прогностичні можливості: правила асоціації також можуть служити в якості прогнозних інструментів, що дозволяє зацікавленим сторонам передбачати результати на основі певних умов.

Порівняння моделі з ідеальною моделлю є важливим етапом у процесі розробки системи аналізу даних або побудови прогностичних моделей. Це порівняння дозволяє зрозуміти, наскільки успішно модель справляється з поставленим завданням і наскільки близько вона наближається до ідеальних умов, що задовольняють усі вимоги та критерії. Модель, яку розробляють для практичного застосування, часто може бути далека від ідеальної через різноманітні обмеження, проте порівняння допомагає не лише оцінити її ефективність, але й визначити напрямки для подальшого вдосконалення [17].

Ідеальна модель, на відміну від реальної, є теоретичним ідеалом, що втілює найкращі можливі результати. Вона відповідає всім вимогам, не має обмежень та неточностей, її рішення є завжди оптимальними. У реальності створити таку модель практично неможливо через наявність шуму в даних, обмеження в обчислювальних ресурсах, необхідність прийняття рішень в

умовах невизначеності. Втім, ідеальна модель служить орієнтиром для розробки реальних моделей, які повинні прагнути до досягнення максимальних результатів.

Порівняння побудованої моделі з ідеальною полягає в оцінці точності і ефективності розробленої системи. Порівняння може допомогти виявити, на скільки модель відповідає вимогам, наскільки вона точна в прогнозуванні або класифікації, а також чи здатна вона правильно розподіляти ресурси або приймати оптимальні рішення на основі аналізу даних. У випадку зі складними моделями, такими як штучні нейронні мережі або складні статистичні методи, порівняння з ідеальною моделлю допомагає побачити, де саме модель не досягає максимальної точності. Зв’язки між наборами елементів відображено на рисунку 14.

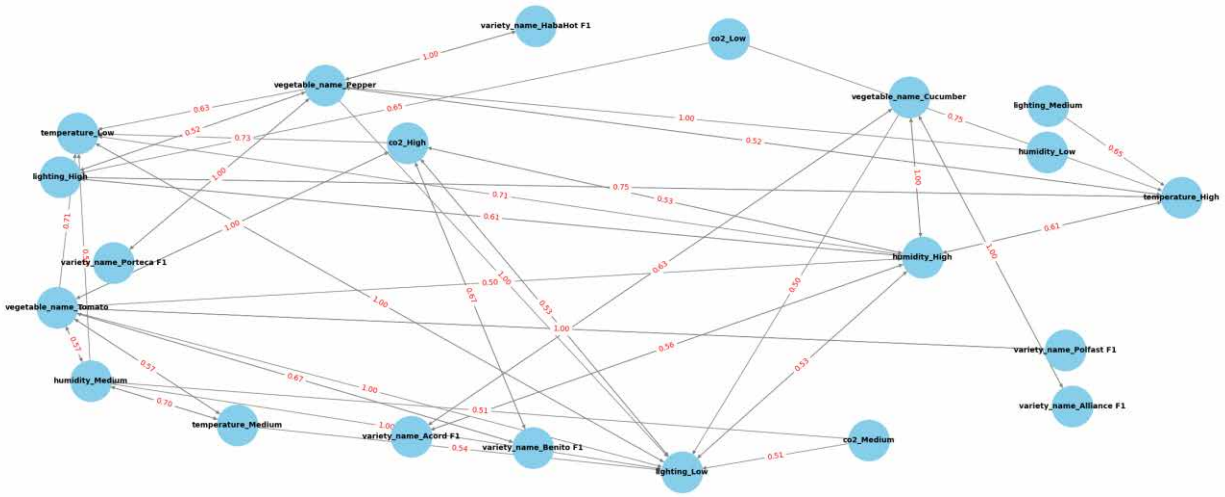


Рис. 14 Графічне відображення зв’язків між наборами елементів

Графічно відображає правила, знайдені на основі фактичних даних (support vs. confidence). Додає точки, що представляють ідеальну модель, де support і confidence дорівнюють 1, що є гіпотетично максимальними значеннями.

- Сині точки: Представляють фактичні правила з відповідними значеннями support та confidence.

- Червоні точки: Представляють ідеальні значення, що мають максимальний support і confidence.

Граф (орієнтований граф, DiGraph), де кожен вузол представляє елемент, а кожне ребро — зв’язок між цими елементами на основі знайдених правил асоціацій. Вузли представляють окремі елементи. Ребра з'єднують вузли, що утворюють правила асоціацій. Мітки на ребрах відображають значення впевненості для кожного правила.

Цей граф дозволяє візуально відобразити, які елементи (в даному випадку сенсори або характеристики рослин) часто зустрічаються разом і які взаємозв'язки можна виявити між ними на основі знайдених правил асоціацій.

У практичних умовах, створення моделі майже завжди супроводжується певними компромісами. Для цього може бути використано декілька метрик, які дозволяють оцінити, наскільки результат, отриманий реальним алгоритмом, наближається до результату, який можна було б отримати від ідеальної моделі. Ці метрики включають такі показники як точність, повнота, точність прогностичних висновків, обробка помилок, швидкість обчислень тощо. Вони дозволяють порівняти фактичну ефективність моделі з тим, що було б досягнуто в умовах без помилок або обмежень.

Точність моделі — одна з найбільш важливих характеристик, яка відображає загальний рівень відповідності передбачень до реальних результатів. Ідеальна модель не припускається жодної похибки, але реальні системи завжди мають похибки, і завдання полягає в тому, щоб мінімізувати ці похибки до мінімуму. Порівняння точності моделі з ідеальною допомагає з'ясувати, скільки різниць у прогнозах або класифікаціях можна знайти між реальною моделлю і тим, що ми б отримали, якби модель працювала без обмежень [18].

Оцінка повноти та точності моделі є ще однією важливою метою порівняння. Повнота вказує на здатність моделі виявляти всі значущі передбачення, навіть якщо деякі з них є помилковими. Точність же визначає

частку правильних передбачень серед всіх зроблених прогнозів. Ідеальна модель надає абсолютно правильні прогнози без помилок і завжди визначає найважливіші властивості та закономірності в даних. Порівняння реальної моделі з ідеальною в контексті повноти та точності дає можливість знайти найкраще співвідношення між цими двома характеристиками.

Особливо важливою є порівняльна оцінка в контексті класифікаційних моделей. Тут, порівняння з ідеальною моделлю дозволяє побачити, наскільки добре модель розділяє класи в навчальних та тестових наборах. У реальних умовах можуть бути випадки, коли модель не зможе чітко поділити об'єкти на класи через складність або нечіткість меж між ними. Ідеальна модель не має цих проблем, тому її результат є таким, що служить максимальним орієнтиром для оцінки ефективності реальних моделей.

Особливе значення порівняння моделі з ідеальною моделлю набуває при роботі з системами, що прогнозують. Прогнозні моделі застосовуються для визначення можливих подій на основі поточних або історичних даних, і їх точність критично важлива. У такому випадку ідеальна модель дає точне уявлення про майбутнє, але реальна модель завжди має певні недоліки, пов'язані з неповнотою даних, варіативністю майбутніх подій або обмеженнями в математичному моделюванні. Порівняння дає змогу зрозуміти, наскільки реальна модель наближається до цього ідеалу і де її прогнози можуть бути менш точними.

У багатьох випадках порівняння з ідеальною моделлю стає важливим етапом для оптимізації параметрів моделі. Оптимізація може включати в себе зміну архітектури моделі, використання нових методів навчання, зміну параметрів або введення додаткових змінних, що можуть покращити її точність. Якщо модель значно відрізняється від ідеальної, це може бути сигналом того, що необхідно здійснити додаткові кроки для поліпшення її роботи. Ці кроки

можуть включати в себе як технічні зміни в алгоритмі, так і покращення якості вхідних даних.

Порівняння моделі з ідеальною моделлю не є одноразовим процесом. Часто, особливо в складних і динамічних системах, порівняння проводиться на різних етапах розвитку моделі. Спочатку модель може бути дуже далека від ідеалу, але з часом, шляхом оптимізації і корекції, вона може наблизитися до ідеального стану. Водночас, завдяки порівнянню, можна визначити межі цього процесу: якими є технічні та практичні обмеження, що заважають досягненню ідеального результату.

# 4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

## 4.1 Вимоги до апаратного та програмного забезпечення

Ефективне функціонування експертної інформаційно-управляючої системи тепличного виробництва залежить від правильно підібраного апаратного та програмного забезпечення. Оскільки система має опрацьовувати великі обсяги даних, забезпечувати безперебійну роботу й підтримувати інтерактивну взаємодію з користувачами, до обладнання й програмного середовища пред’являються високі вимоги.

**Апаратне забезпечення**

Для реалізації системи необхідно передбачити мінімальні й рекомендовані технічні характеристики апаратного забезпечення, які визначаються масштабами діяльності та складністю обробки даних [19].

1. Сервер для бази даних

Оскільки PostgreSQL є центральною частиною системи, сервер, на якому вона працюватиме, повинен забезпечувати високу продуктивність і стабільність.

Основні вимоги:

- процесор: багатоядерний (рекомендовано від 8 ядер і більше) із частотою не менше 2.5 ггц;
- оперативна пам’ять (ram): мінімум 16 гб, рекомендовано – від 32 гб, особливо для складних аналітичних запитів і роботи з великими масивами даних;
- накопичувач: ssd-диск є обов’язковим для забезпечення швидкого доступу до даних; мінімальний обсяг – 1 тб, рекомендовано 2 тб і більше;

- мережева підключеність: висока пропускна здатність (1 гбіт/с і більше) для забезпечення швидкого доступу до сервера через локальну мережу або інтернет.

2. Робочі станції користувачів

Користувачі, які взаємодіють із системою через клієнтські додатки або веб-інтерфейс, потребують надійних робочих станцій:

- процесор: не менше 4 ядер, частота від 2.0 ггц;
- оперативна пам’ять: від 8 гб;
- накопичувач: ssd на 256 гб і більше;
- монітор: з роздільною здатністю full hd (1920×1080) або вище.

3. Сенсорні пристрої

Для збору даних із теплиці потрібні сенсори, здатні передавати інформацію про температуру, вологість, рівень освітлення й концентрацію CO2 у ґрунті. Вимоги до них:

- надійність і точність вимірювань;
- підтримка бездротового зв’язку або можливість інтеграції з локальною мережею;
- енергоефективність, якщо сенсори працюють автономно.

**Програмне забезпечення**

1. Система управління базами даних (СУБД)

PostgreSQL є основою для роботи з даними. Ця СУБД повинна бути встановлена на сервері з наступними умовами:

- операційна система: linux (рекомендовано, наприклад, ubuntu server або centos) або windows server;
- версія postgresql: остання стабільна версія для забезпечення доступу до нових функцій і виправлення помилок;

- модулі й розширення: додатково можуть знадобитися postgis для роботи з геопросторовими даними та pgadmin для зручного управління базою.

2. Серверний програмний стек

Для реалізації логіки системи, обробки даних і інтеграції із сенсорами використовується Python. Необхідно встановити:

- python 3.8 або вище;
- бібліотеки: sqlalchemy, pandas, numpy, scikit-learn (для аналітики), matplotlib (для візуалізації).

3. Клієнтське середовище

Користувацький інтерфейс розробляється як Windows Forms-додаток або веб-додаток:

- мова програмування: c# (для windows forms) або python (для веб-інтерфейсу за допомогою flask/django);
- інтеграція з базою даних: використання драйверів odbc або бібліотек для роботи з postgresql.

## 4.2 Тестування системи

Тестування є невід'ємною частиною процесу розробки будь-якої програмної системи, в тому числі і експертної інформаційно-управляючої системи для тепличного виробництва. Цей етап включає в себе перевірку функціональності системи, її стабільності, ефективності та здатності правильно виконувати завдання, які стоять перед користувачем. Метою тестування є виявлення можливих помилок або недоліків в програмному забезпеченні до його запуску в реальному середовищі, а також оцінка того, наскільки система відповідає вимогам і специфікаціям, які були визначені на етапі проектування.

Тестування експертної системи для тепличного виробництва було проведено на кількох етапах: функціональне тестування, тестування продуктивності, тестування на помилки, а також тестування користувацького інтерфейсу. Для кожного етапу тестування були визначені конкретні критерії, що дозволяли оцінити успішність виконання завдань. Вікно авторизації представлено рисунку 15.

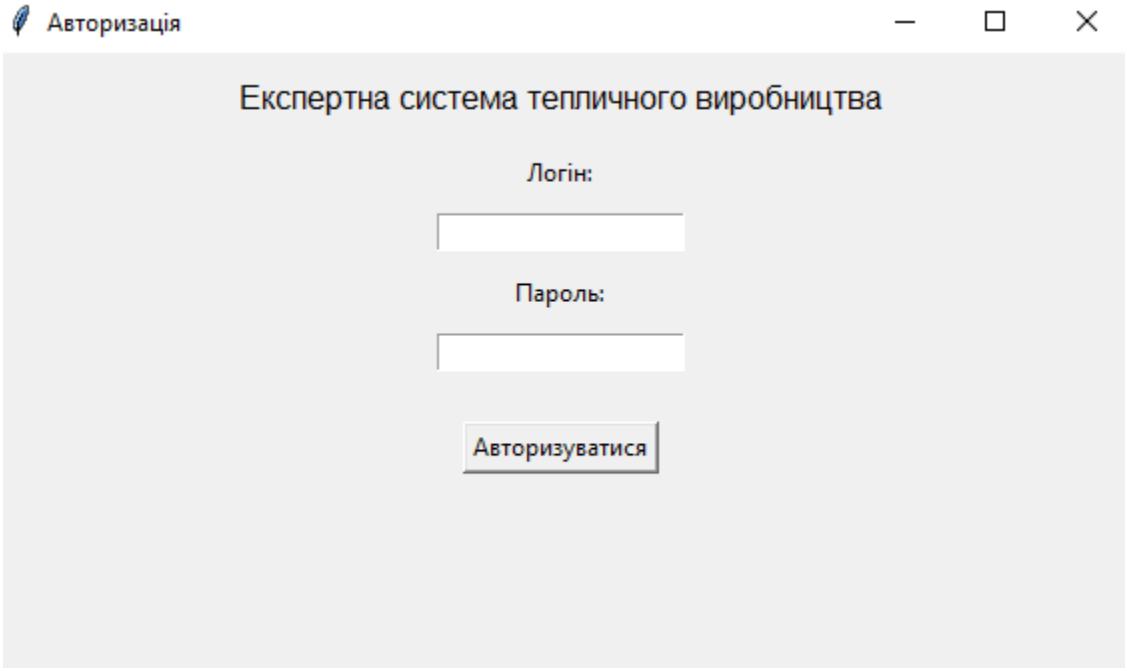


Рис.15 Сторінка авторизації

Це важливий етап для забезпечення безпеки даних та контролю доступу до системи. Вікно містить стандартні поля для введення логіну та паролю, що гарантує, що лише авторизовані користувачі можуть працювати з даними та управлінням тепличним процесом. Авторизація дозволяє системі ідентифікувати, хто саме використовує програму, чи є це агроном, відповідальний за прийняття рішень, чи аналітик, що обробляє дані. Такий підхід забезпечує належний рівень захисту інформації і дозволяє персоналізувати доступ до різних розділів та функцій програми. Головна сторінка представлена на рисунку 16.

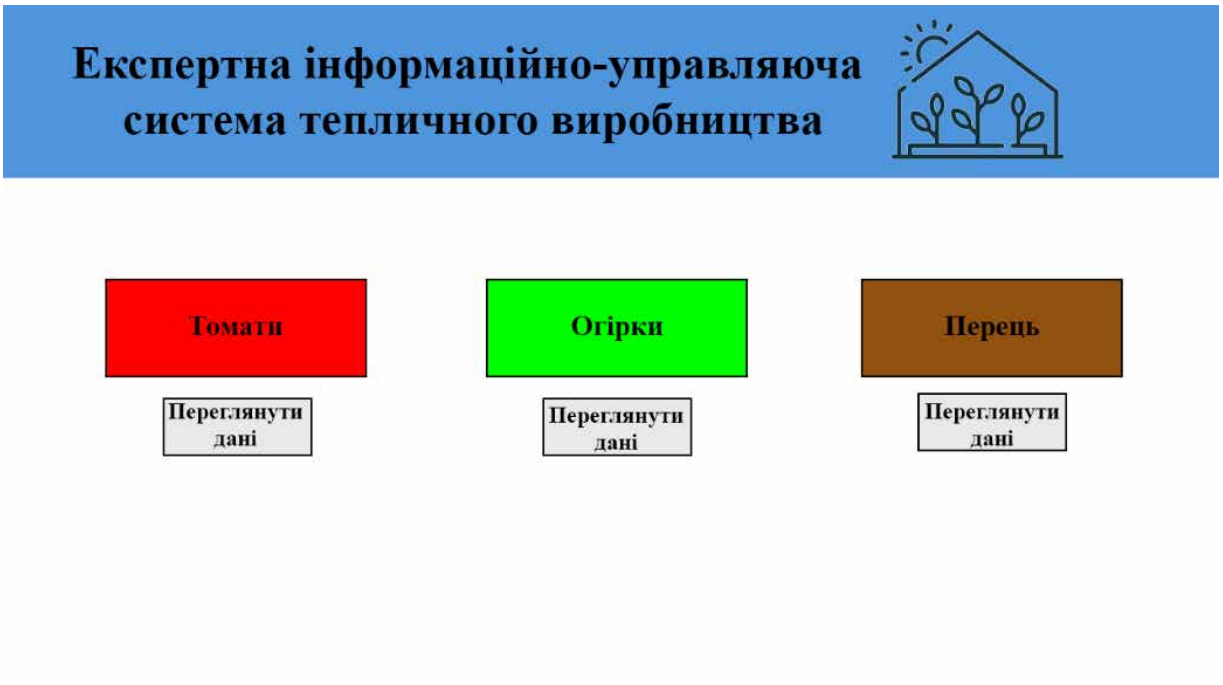


Рис. 16 Головна сторінка

Вона має зручний і інтуїтивно зрозумілий інтерфейс із трьома великими іконками овочів – томатів, огірків і перцю, які вирощуються в теплиці. Ці іконки символізують окремі категорії культур і слугують швидкими посиланнями для переходу до розширеної інформації про кожну з них. Поруч із кожною іконкою розташовані кнопки "переглянути дані", які дозволяють агроному переглядати детальну інформацію про стан кожного виду культури. Це спрощує навігацію і робить доступ до даних швидким та зручним, що є важливим для оперативного реагування на будь-які зміни у теплиці. Сторінка томатів та даних по ним представлена на рисунку 17.

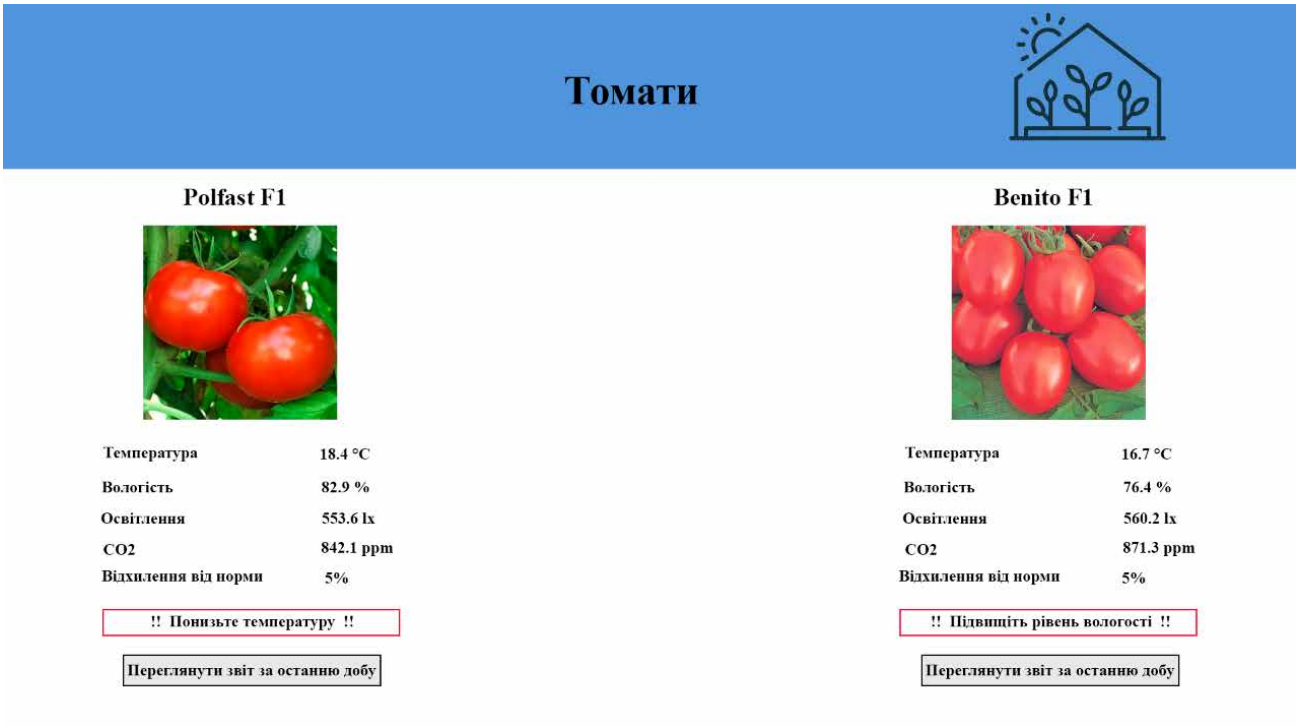


Рис.17 Сторінка перегляду даних про сорти

Тут система відображає поточні дані для двох сортів, надані у вигляді інформаційних блоків, які включають показники температури, вологості, рівня освітлення та вмісту CO2. Такий рівень деталізації дозволяє агроному в режимі реального часу контролювати параметри середовища для кожного сорту окремо. Під основною інформацією є рядок, який позначає відхилення показників від норми. Це дозволяє агроному швидко виявити, якщо показники вийшли за межі оптимальних значень, і зрозуміти, наскільки серйозною є ситуація. Ще нижче система відображає рекомендації на основі аналізу даних. Наприклад, якщо вологість опускається нижче норми, система може запропонувати "підвищіть рівень вологи", що дає змогу агроному вчасно вжити заходів для відновлення потрібних умов. Кнопка "переглянути звіт за останню добу" дозволяє отримати доступ до детальної інформації про зміну параметрів протягом останніх 24 годин, що допомагає бачити тренди та розуміти динаміку змін. Звіт відображений на рисунку 18.

**Benito F1**

**Звіт за останні 24 години**

| Овоч | Сорт | Дата та час | Температура | Вологість | Освітлення | CO2 |
|---|---|---|---|---|---|---|
| Tomato | Benito F1 | 2023-04-01 00:00:00 | 12.557872805167007 | 86.19335840915066 | 1.1954583956508 17 | 957.3949981746393 |
| Tomato | Benito F1 | 2023-04-01 01:00:00 | 17.82262463140159 | 72.30146978159338 | 19.810769074219415 | 907.8237523766181 |
| Tomato | Benito F1 | 2023-04-01 02:00:00 | 13.343733539995027 | 86.42156099870276 | 27.940049700070 68 | 971.5699152632375 |
| Tomato | Benito F1 | 2023-04-01 03:00:00 | 12.005663077486538 | 75.22295197488216 | 15.971935311164675 | 916.5791429786668 |
| Tomato | Benito F1 | 2023-04-01 04:00:00 | 15.170876527141843 | 72.30434113709175 | 94.98819279603956 | 914.6629588937315 |
| Tomato | Benito F1 | 2023-04-01 05:00:00 | 14.232851349735494 | 85.97021279818458 | 40.828901355474265 | 987.685664231596 |
| Tomato | Benito F1 | 2023-04-01 06:00:00 | 18.290875554012107 | 75.10064757457566 | 446.0941360161968 | 990.1846440730292 |
| Tomato | Benito F1 | 2023-04-01 07:00:00 | 21.686050989842784 | 79.38577475693725 | 797.7176845501208 | 835.2925553159854 |
| Tomato | Benito F1 | 2023-04-01 08:00:00 | 18.06691973653367 | 85.13458598287684 | 763.3818864352206 | 977.3068099892633 |
| Tomato | Benito F1 | 2023-04-01 09:00:00 | 24.332916576350108 | 82.24598089684066 | 549.708141533573 | 968.7679355778293 |
| Tomato | Benito F1 | 2023-04-01 10:00:00 | 19.469818632208888 | 86.34690344002453 | 438.5643973494083 | 998.4439794270829 |
| Tomato | Benito F1 | 2023-04-01 11:00:00 | 24.689576321032185 | 77.78517950093786 | 448.1891907219226 | 981.5757717392707 |
| Tomato | Benito F1 | 2023-04-01 12:00:00 | 19.647390755265477 | 86.5179313450979 | 702.6012255603473 | 947.1493200084279 |
| Tomato | Benito F1 | 2023-04-01 13:00:00 | 24.863659627005212 | 82.84470488275295 | 416.42959494846775 | 803.7510914762131 |
| Tomato | Benito F1 | 2023-04-01 14:00:00 | 19.339457104353347 | 74.28838440333806 | 729.0879370256262 | 911.342526999897 |
| Tomato | Benito F1 | 2023-04-01 15:00:00 | 21.658488935417463 | 75.69607081686247 | 557.0601655140928 | 998.8608309377687 |
| Tomato | Benito F1 | 2023-04-01 16:00:00 | 21.327114118864262 | 71.14967105970774 | 402.04333865571914 | 982.5132601894537 |
| Tomato | Benito F1 | 2023-04-01 17:00:00 | 20.023407502737616 | 71.45009264287847 | 427.63324462365 51 | 933.876960550163 |
| Tomato | Benito F1 | 2023-04-01 18:00:00 | 12.23276985156457 | 75.90837392987021 | 12.159876486270871 | 809.1909787949924 |
| Tomato | Benito F1 | 2023-04-01 19:00:00 | 12.586414846251479 | 82.75937563111572 | 3.8243104942110917 | 964.0698698824306 |
| Tomato | Benito F1 | 2023-04-01 20:00:00 | 17.456759198025942 | 70.21518187966254 | 97.626343738935 12 | 820.9681675038521 |
| Tomato | Benito F1 | 2023-04-01 21:00:00 | 13.33469342225918 | 87.1876164510468 | 1.7178860575911226 | 957.2099662252806 |
| Tomato | Benito F1 | 2023-04-01 22:00:00 | 17.303542270839444 | 70.87139182202542 | 73.048627407073 47 | 991.2830110625366 |
| Tomato | Benito F1 | 2023-04-01 23:00:00 | 17.969961639382223 | 81.11775640120389 | 80.710307183675 95 | 944.2935464575249 |

Рис.18 Звіт за останні 24 години

На четвертому фото представлена таблиця, що містить дані всіх сенсорів за останню добу. Ця інформація допомагає аналітику або агроному оцінювати, як змінювалися показники температури, вологості, освітлення та вмісту CO2 у теплиці протягом дня. Знання цих змін дає змогу проводити більш глибокий аналіз стану теплиці та впливу зовнішніх факторів на рослини. Завдяки можливості зберігання історичних даних у сховищі, експертна система може проводити аналіз не лише за останню добу, а й за триваліші періоди, щоб виявляти довготривалі тенденції. Це дозволяє агроному планувати дії заздалегідь і приймати рішення, що забезпечують максимальну врожайність і здоров'я рослин.

Описані можливості експертної системи дозволяють оптимізувати процеси вирощування та знижувати ризики. Система є не лише інструментом моніторингу, але й помічником, що пропонує практичні рекомендації для підтримки умов, які сприяють здоровому розвитку овочевих культур. Впровадження таких рекомендацій може суттєво вплинути на врожайність та якість продукції. Завдяки своєчасним сповіщенням та аналітичним звітам,

агрономи можуть оперативно реагувати на зміни у середовищі і запобігати потенційним проблемам, що можуть призвести до втрат.

Функціональне тестування є основною частиною перевірки коректності роботи системи. Це тестування того, чи правильно система виконує свої основні функції. Оскільки основна мета експертної системи полягає в тому, щоб надавати рекомендації агрономам на основі даних сенсорних систем, було проведено тестування наступних функцій:

Збір даних з сенсорів – перевірка того, чи правильно система отримує дані від сенсорів (температура, вологість, освітленість та інші параметри). Для цього були використані як реальні дані з сенсорів, так і синтетичні дані для тестування сценаріїв з різними умовами.

Обробка даних – перевірка алгоритмів обробки та аналізу отриманих даних. Включає в себе перевірку функцій кластеризації, виявлення аномалій та побудови моделей, які використовуються для генерації рекомендацій.

Генерація рекомендацій – тестування того, чи надає система адекватні рекомендації на основі проаналізованих даних. Було перевірено, чи рекомендації правильно відображаються в системі, чи відповідають реальним умовам теплиць [20].

Збереження і доступ до даних – перевірка того, чи правильно система зберігає результати вимірювань, рекомендацій і звітів у базі даних, і чи є можливість доступу до цих даних для подальшого аналізу чи використання.

Цей етап тестування дозволив виявити ряд проблем, зокрема пов'язаних з обробкою даних у випадках великих обсягів інформації. Однак після корекції алгоритмів та налаштувань система почала працювати стабільно, забезпечуючи правильну генерацію рекомендацій для агрономів.

Тестування продуктивності є важливим для оцінки здатності системи працювати в реальному середовищі, де вона повинна обробляти значні обсяги даних. Це включає в себе тестування швидкості обробки даних, а також

перевірку стабільності роботи при високому навантаженні. Для цього було створено тестові сценарії, які імітували збір і обробку великих обсягів даних за допомогою сенсорних систем. Тестування включало:

Швидкість обробки даних – скільки часу потрібно для обробки даних, що надходять від сенсорів, і для генерування відповідних рекомендацій. Оцінювалась можливість системи швидко реагувати на зміни в умовах теплиці.

Стабільність при високому навантаженні – перевірка того, чи система здатна працювати при постійному потоці даних без збоїв або значного сповільнення роботи. Це було критично важливо для системи, оскільки вимірювання та рекомендації мають бути доступні в режимі реального часу.

Використання ресурсів – оцінка ефективності використання апаратних ресурсів, таких як процесор, пам'ять та диск. Для цього використовувалися інструменти для моніторингу ресурсів під час навантажувальних тестів.

Результати тестування показали, що система здатна обробляти дані з кількох сенсорних систем без значних затримок, однак потребує додаткової оптимізації для роботи з великими обсягами даних на етапах збору і збереження інформації.

Цей етап тестування зосереджений на виявленні помилок та несправностей у системі, що можуть виникнути під час її роботи. Для цього використовувалися як ручні, так і автоматизовані методи тестування. Зокрема, було протестовано:

Помилки в алгоритмах – перевірка, чи алгоритми правильно працюють з різними вхідними даними, включаючи некоректні або неповні дані, які можуть надходити від сенсорів.

Несправності інтерфейсу – тестування користувацького інтерфейсу на наявність помилок, пов'язаних із його використанням. Це включало перевірку правильності відображення інформації, зручність взаємодії з системою.

Випадкові збої і відновлення – перевірка того, чи система здатна відновлюватися після випадкових збоїв або відключення, що є важливим аспектом для експертних систем, що працюють у реальному часі.

Цей етап тестування допоміг виявити декілька незначних багів, пов'язаних з некоректним відображенням даних в умовах високої навантаженості, але ці помилки були оперативно виправлені.

Тестування користувацького інтерфейсу (UI) стало важливим етапом, оскільки воно визначало зручність і ефективність використання системи для кінцевих користувачів — агрономів. В процесі тестування було оцінено:

Інтуїтивність інтерфейсу – чи легко агрономи можуть зрозуміти, як користуватися системою, чи є доступ до основних функцій, таких як перегляд даних сенсорів, аналіз рекомендацій і формування звітів.

Зручність виведення рекомендацій – наскільки просто агроному отримувати рекомендації і застосовувати їх на практиці. Був протестований процес генерації рекомендацій і їх відображення в системі.

Адаптивність до різних пристроїв – перевірка того, чи працює інтерфейс коректно на різних пристроях, таких як комп'ютери, планшети або смартфони, що може бути корисно для агрономів, які працюють на полях або в теплицях.

Після завершення тестування інтерфейсу були внесені кілька змін для покращення зручності користування та збільшення продуктивності користувачі

# ВИСНОВКИ

В ході реалізації експертної інформаційно-управляючої системи для тепличного виробництва було проведено всебічне дослідження, проектування, тестування та впровадження системи, здатної надавати рекомендації агрономам на основі аналізу даних, що надходять від сенсорних систем. Робота почалася з ретельного аналізу потреб тепличного господарства, зокрема агрономів, які мають потребу в точних і своєчасних рекомендаціях для оптимізації вирощування різних видів рослин у теплицях. За допомогою сенсорних систем, що збирають дані про температуру, вологість, освітленість та інші умови навколишнього середовища, було розроблено модель, яка дозволяє в реальному часі отримувати важливу інформацію для прийняття обґрунтованих рішень. Аналіз даних здійснюється через використання методів інтелектуального аналізу, зокрема, кластеризації, аналізу асоціативних правил та інших статистичних методів, що дозволяють знаходити закономірності в умовах вирощування і надавати рекомендації.

Вибір методу аналізу даних був одним з ключових етапів проекту, оскільки саме від цього залежала точність і корисність рекомендацій для кінцевого користувача – агронома. За допомогою класифікації та кластеризації агроном міг отримувати зведену інформацію про різні аспекти тепличного виробництва, від розподілу температури в різні пори доби до виявлення аномальних значень, які можуть впливати на врожай. Важливим елементом цієї системи стало також створення бази даних для зберігання результатів вимірювань, рекомендацій і аналітичних звітів, що дозволило забезпечити доступність і зручність роботи з інформацією.

Одним з найбільших досягнень цього проекту стало впровадження можливості для агронома отримувати рекомендації на основі реальних даних з сенсорів. Рекомендації допомагають оптимізувати використання ресурсів (води,

добрив, енергії) та підвищити врожайність шляхом налаштування умов для кожної рослини в теплиці. Також, система може автоматично виявляти аномалії, що дозволяє своєчасно вжити заходів для збереження врожаю.

Але разом з тим, процес тестування і реалізації показав, що в поточній версії системи є ще кілька аспектів, які потребують вдосконалення. Перш за все, це підвищення точності рекомендацій, оскільки деякі алгоритми можуть давати не зовсім точні результати при роботі з різними типами даних. Для покращення результатів рекомендується використати більш складні методи аналізу, такі як глибинне навчання або ансамблеві методи, що дозволяють покращити точність прогнозування і рекомендацій. Одним з важливих напрямків для подальшого розвитку є інтеграція системи з іншими датчиками та технологіями для збору даних, що дозволить збільшити обсяг даних, що використовуються для аналізу.

В результаті, можна зробити висновок, що реалізація експертної інформаційно-управляючої системи для тепличного виробництва є важливим кроком у напрямку автоматизації аграрного виробництва, яке дозволяє оптимізувати роботу теплиць, підвищити ефективність вирощування культур і знизити витрати ресурсів. Подальша робота над удосконаленням цієї системи дозволить розв'язати низку завдань, зокрема, щодо точності рекомендацій, обробки великих обсягів даних і інтеграції з іншими системами, що в кінцевому підсумку дозволить досягти максимальної ефективності в роботі тепличного виробництва.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Електронний ресурс (Python)https://docs.python.org/uk/3/tutorial/index.html

2. Dudnyk A. Features of Intelligent Control Systems of Biotechnological Objects //A. Dudnyk/ Proceedings of the International Scientific and Practical Conference

3. Information support of the remote nitrogen monitoring system in agricultural crops /Lysenko, V., Opryshko, O., Komarchuk, D., Pasichnyk, N., Zaets, N., Dudnyk, A. // International Journal of Computing

4. Lysenko V. Intelligent Control System of Biotechnological Objects with Fuzzy Controller and Information Channel Filtration Unit // V. Lysenko, A. Dudnyk, N. Zaets, D. Komarchuk, T.Lendel and I. Yakymenko // PIC S&T`2018 International Scientific and Practical Conference «Problems of Infocommunications. Science and Technology»

5. Електронний ресурс (UML) https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html

6. Електронний ресурс (Priva Context) https://danvan.dk/wp-content/uploads/220_en-priva-connext-brochure.pdf

7. Електронний ресурс (Огляд експертних систем) https://wiki.tntu.edu.ua/%D0%9E%D0%B3%D0%BB%D1%8F%D0%B4_%D0%B2%D0%B8%D0%B4%D1%96%D0%B2_%D0%B5%D0%BA%D1%81%D0%BF%D0%B5%D1%80%D1%82%D0%BD%D0%B8%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC_%D1%82%D0%B0_%D1%97%D1%85_%D0%BA%D0%BB%D0%B0%D1%81%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F

8. Електронний ресурс (Діаграма діяльності) https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html#use-case-diagram

9. Електронний ресурс (Діаграма активності) https://www.mindonmap.com/uk/blog/uml-activity-diagram/

10. Електронний ресурс (Діаграма послідовності) http://mmsa.kpi.ua/

11. Електронний ресурс (Діаграма прецедентів) https://www.wrike.com/blog/precedence-diagramming-method-project-management/

12. Електронний ресурс (OLAP) https://www.snowflake.com/guides/olap-vs-oltp/

13. Електроннинй ресурс (PostgreSQL) https://www.postgresql.org/

14. Електронний ресурс (1-Rule) https://www.geeksforgeeks.org/learn-one-rule-algorithm/

15. Електронний ресурс (Naïve Bayes) https://scikit-learn.org/1.5/modules/naive_bayes.html

16. Електронний ресурс (ErWin) https://www.erwin.com/

17. Електронний ресурс (Apriori algorithm) https://www.javatpoint.com/apriori-algorithm

18. Дудник А.О. Інформаційне забезпечення та методи розробки бази даних системи керування біотехнічним об’єктом. // Матеріали Міжнародної науково-практичної конференції студентів, аспірантів та молодих вчених "Інформаційні технології: економіка, техніка, освіта". 14-15 листопада 2017

19. Електронний ресурс (Кластеризація) https://unstop.com/blog/what-is-clustering-in-data-mining

20. Електронний ресурс (Класифікація) https://datascientest.com/en/classification-algorithms-definition-and-main-models

# Додаток А
# Фрагменти коду створення рокемандацій та виявлення аномалій

```
def generate_recommendations():
    conn = psycopg2.connect(**db_params)
    cursor = conn.cursor()

    cursor.execute("SELECT id_anomaly, id_variety, value, anomaly_type FROM anomalies")
    anomalies = cursor.fetchall()
    cursor.execute("SELECT id_variety, sensor_type, min_value, max_value FROM NormalRanges")
    normal_ranges = cursor.fetchall()



    normal_range_dict = {}
    for range_entry in normal_ranges:
        id_variety, sensor_type, min_value, max_value = range_entry
        normal_range_dict[(id_variety, sensor_type)] = (min_value, max_value)

    recommendations = []

    for anomaly in anomalies:
        id_anomaly, id_variety, value, anomaly_type = anomaly

        normal_range = normal_range_dict.get((id_variety, anomaly_type))

        if normal_range:
            min_value, max_value = normal_range

            if value < min_value:
                recommendations.append((id_anomaly, anomaly_type, f"Збільште значення
{anomaly_type} до рекомендованого діапазону. Поточне значення: {value}, мінімально допустиме:
{min_value}."))
            elif value > max_value:
```

```
                    recommendations.append((id_anomaly, anomaly_type, f"Зменшіть значення {anomaly_type} до рекомендованого діапазону. Поточне значення: {value}, максимально допустиме: {max_value}."))
                else:
                    recommendations.append((id_anomaly, anomaly_type, "Значення в межах нормального діапазону."))
            else:
                recommendations.append((id_anomaly, anomaly_type, "Нормальний діапазон не знайдено."))


        for recommendation in recommendations:
            id_anomaly, sensor_type, recommendation_text = recommendation
            cursor.execute("""
                INSERT INTO Recommendations (id_anomaly, sensor_type, recommendation_text)
                VALUES (%s, %s, %s)
            """, (id_anomaly, sensor_type, recommendation_text))

        conn.commit()

        cursor.close()
        conn.close()

    if __name__ == "__main__":
        generate_recommendations()



    def find_anomalies():
        conn = psycopg2.connect(**db_params)
        cursor = conn.cursor()


        cursor.execute("SELECT id_measurement, timestamp, temperature, humidity, lighting, co2 FROM measurement")
```

```
    measurements = cursor.fetchall()
    cursor.execute("SELECT id_variety, sensor_type, min_value, max_value FROM normalranges")
    normal_ranges = cursor.fetchall()

    for measurement in measurements:
        id_measurement, timestamp, temperature, humidity, lighting, co2 = measurement

        for range_entry in normal_ranges:
            id_variety, sensor_type, min_value, max_value = range_entry

            sensor_value = None
            if sensor_type == 'temperature':
                sensor_value = temperature
            elif sensor_type == 'humidity':
                sensor_value = humidity
            elif sensor_type == 'lighting':
                sensor_value = lighting
            elif sensor_type == 'co2':
                sensor_value = co2

            if sensor_value is not None and (sensor_value < min_value or sensor_value > max_value):
                # Запис аномалії в таблицю anomalies
                cursor.execute("""
                    INSERT INTO anomalies (id_variety, value, anomaly_type, timestamp)
                    VALUES (%s, %s, %s, %s)
                """, (id_variety, sensor_value, sensor_type, timestamp))

    conn.commit()

    cursor.close()
    conn.close()

if __name__ == "__main__":
    find_anomalies()
```

## Додаток Б

## Фрагмент коду з використанням алгоритму 1-Rule

```
average_co2= df.groupby('variety_name')['co2'].mean().reset_index()
average_co2.columns = ['variety_name', 'average_co2']

df = df.merge(average_co2, on='variety_name')

df['lower_than_avg'] = df['co2'] < df['average_co2']
df['higher_than_avg'] = df['co2'] > df['average_co2']

df['error'] = abs(df['co2'] - df['average_co2'])

df['success'] = df['higher_than_avg'].apply(lambda x: 'Висока' if x else 'Низька')

final_df = df.groupby('variety_name').agg(
   total_measurements=pd.NamedAgg(column='co2', aggfunc='count'),
   below_avg=pd.NamedAgg(column='lower_than_avg', aggfunc='sum'),
   above_avg=pd.NamedAgg(column='higher_than_avg', aggfunc='sum'),
   avg_co2=pd.NamedAgg(column='co2', aggfunc='mean'),
   total_error=pd.NamedAgg(column='error', aggfunc='sum'),
   success=pd.NamedAgg(column='success', aggfunc=lambda x: 'Висока' if (x == 'Висока').sum() >
(x == 'Низька').sum() else 'Низька')
).reset_index()
```