**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ**

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

«До захисту допущена»
Зав. кафедри комп'ютерних наук
та прикладної математики
д.т.н., професор Турбал Ю.В.
«____» _____ 20_ р.

# КВАЛІФІКАЦІЙНА РОБОТА

Розробка інформаційної системи

«Смарт-агротеплиця»

Виконала: Осадець Олександра Романівна

Студентка навчально-наукового інституту кібернетики, інформаційних технологій та інженерії

група ІПЗ-41

_____
підпис

Керівник: к.т.н., доцент Мічута Ольга Романівна _____
підпис

Рівне-2025

**ЗМІСТ**

# РЕФЕРАТ

**Кваліфікаційна робота:** 66с., 26 рисунків, 6 таблиць, 17 джерел, 1 додаток.

**Мета роботи:** створити інформаційну систему для відстеження кліматичних умов всередині теплиці, керування пристроями для покращення стану кліматичних умов та отримання користувачем рекомендацій стосовно вирощування культур.

**Об`єкт дослідження:** інформаційна система «Смарт-агротеплиця».

**Предмет дослідження:** структура та функціонування інформаційної системи, що забезпечує збір, обробку та візуалізацію даних про стан теплиці, а також взаємодію між апаратною та програмною частинами системи.

**Методи** розробки базуються на проектуванні клієнт-серверної архітектури, використанні ASP.NET Core Web API, React Native, Arduino, базі даних Microsoft SQL Server.

Реалізовано інформаційну систему «Смарт-агротеплиця», яка складається з трьох основних компонентів: пристрою збору та передачі даних Arduino, серверної частини ASP.NET Web API, та мобільного клієнта React Native. Проведено тестування в реальних умовах та аналіз отриманих результатів.

**КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙН СИСТЕМА, АГРОТЕПЛИЦЯ, СМАРТ-АГРОТЕПЛИЦЯ, ДОДАТОК, ARDUINO, СЕРВЕР, БАЗА ДАНИХ.**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI / ШІ – штучний інтелект
API – інтерфейс програмування додатків
БД – база даних
Big Data – великі дані
Колбек – функція, передана як аргумент
DTO – об’єкт для передачі даних між шарами додатку
Ендпоінт – точка доступу до API
IoT – інтернет речей
JWT – стандарт токенів для аутентифікації
REST – архітектурний стиль для розробки веб-сервісів
UI – користувацький інтерфейс
UX – досвід користувача
JSON – формат обміну даними у вигляді тексту

# ВСТУП

У сучасних умовах активних кліматичних змін звичайне та й великомасштабне сільське господарство стикається з низкою викликів та труднощів, серед яких особливу увагу привертає необхідність підвищення ефективності вирощування сільськогосподарських культур за обмежених ресурсів. Розвиток технологій Інтернету речей створює нові можливості для автоматизації аграрних процесів. Використання сучасних інформаційних технологій та автоматизованих систем управління агро об'єктами є одним із перспективних та доцільних напрямів у вирішенні проблеми. Технологія «розумних» теплиць дозволяє створити оптимальні умови для росту рослин шляхом моніторингу параметрів навколишнього середовища та адаптивного керування мікрокліматом.

Станом на сьогодні на ринку існує низка комерційних рішень у сфері автоматизованого агровиробництва. Проте більшість з них є досить дорогими, складними у впровадженні та обслуговуванні, або ж не мають гнучкості для кастомізації під конкретні потреби невеликих фермерських господарств чи приватних теплиць. Тому актуальним є створення доступної, адаптивної та технологічно ефективної інформаційної системи, яка б дозволила автоматизувати процеси в теплиці за допомогою простих апаратних засобів та сучасних інструментів програмного забезпечення.

**Метою** кваліфікаційної роботи є розробка інформаційної системи «Смарт-агротеплиця», яка забезпечує моніторинг та часткове управління кліматичними параметрами в теплиці на основі даних, отриманих з мікроконтролера Arduino. Система складається з трьох основних частин: пристрій збору та передачі даних (Arduino), серверна частина (ASP.NET Web API), що обробляє та зберігає дані, та мобільний клієнт (React Native), який надає користувачеві доступ до інформації та керування.

**Об’єктом дослідження** є процес автоматизації контролю параметрів мікроклімату в тепличному господарстві. **Предметом дослідження** є структура та функціонування інформаційної системи, що забезпечує збір, обробку та візуалізацію даних про стан теплиці, а також взаємодію між апаратною та програмною частинами системи.

У процесі дослідження використовувались наступні **методи та засоби**: проектування клієнт-серверної архітектури, розробка RESTful API з використанням ASP.NET, реалізація мобільного застосунку за допомогою React Native, взаємодія з мікроконтролером Arduino через датчики, а також використання Microsoft SQL Server для збереження даних.

Результати дослідження було апробовано на студентській науково-практичній конференції студентів та аспірантів ННІ КІТІ 14-16 травня 2025 року, за матеріалами якої опубліковано тези доповіді на тему «Смарт-агротеплиця». Участь у конференції підтверджує наукову обґрунтованість отриманих результатів та їх актуальність у сучасних умовах розвитку смарт-технологій в агросекторі.

**Оригінальність дослідження** полягає у реалізації інформаційної системи, яка об’єднує апаратні та програмні компоненти для моніторингу агротехнічних параметрів з можливістю масштабування та подальшої автоматизації процесів. **Практична значущість** полягає в тому, що запропоноване рішення може бути використане в умовах малого аграрного бізнесу або приватних тепличних господарств у вигляді доступного та гнучкого засобу автоматизації.

Результати цієї роботи можуть бути використані як основа для подальших досліджень у сфері smart-агротехнологій, а також для створення подібних рішень в інших галузях автоматизації.

# РОЗДІЛ 1

# ДОСЛІДЖЕННЯ ІСНУЮЧИХ РІШЕНЬ

Сільське господарство є однією з найдавніших і найважливіших форм діяльності людини, що зародилась ще за ранніх цивілізацій. Галузь рослинництва сягає глибини тисячоліть. Людина розпочала розводити корисні рослини ще в кам'яному віці, а саме у пізньому палеоліті – близько 50 тис. років тому [2]. Спочатку збиралось те, що давала природа і що можна було використати в якості харчів – плоди, насіння та рослини. Пізніше почали колекціонувати окремі види дерев, чагарників чи трав, що давали їм їжу. Незабаром, коли люди почали сіяти — фактично розкидати — насіння корисних рослин і збирати врожай, виникло й перше примітивне землеробство [1]. Агросфера завжди знаходилась у стані динамічної еволюції.

## 1.1. Розвиток сільського господарства

Досвід людства накопичувався поступово на основі регулярної практики і передавався усно. Сільське господарство зіграло вирішальну роль у формуванні людських суспільств, а також і культур, тим самим забезпечуючи засоби для існування та сировину для широкого спектру людської діяльності.

Історію розвитку рослинництва можна умовно поділити на три етапи:

1. Перший період — первісне рослинництво виникло з поділом праці, коли людина почала вести осілий спосіб життя.
2. Другий період — рослинництво рабовласницько-античного суспільства.
3. Третій період розвитку характеризується бурхливим розвитком — він відбувся у час промислової революції XVIII–XIX століть.
4. Четвертий період започаткований "зеленим рухом" у 1900-х роках. Фактично він існує і дотепер.

5. П'ятий період — є логічним продовженням зеленої революції, проте грунтується на сучасних досягненнях генетики, біології, агрохімії, селекції, землеробства, генної та молекулярної інженерії [2].

Отже, з початком промислової революції аграрна сфера почала стрімко розвиватися, інтегруючи досягнення науки і техніки. З другої половини XX століття розпочався новий етап еволюції аграрного виробництва, пов'язаний з автоматизацією, розвитком комп’ютерних технологій і систем контролю.

У наш час сільське господарство активно впроваджує інновації у сфері біотехнологій, штучного інтелекту та автоматизації, що створює підгрунтя для розвитку сучасних смарт-рішень.

У контексті сучасних викликів особливої уваги набуває вплив кліматичних змін, що дедалі частіше проявляються у вигляді аномальних температур, нестабільності опадів, тривалих посух або навпаки — надмірної вологості. Такі коливання клімату ускладнюють вирощування традиційних культур у відкритому грунті, що зумовлює потребу в створенні контрольованих середовищ, зокрема у формі інтелектуальних теплиць. Смарт-рішення дають змогу адаптувати сільське господарство до умов нових реалій, зменшити залежність від непередбачуваних погодних умов та забезпечити стабільне виробництво продовольства.

### 1.2. Сучасний стан агросфери та основні проблеми

Сучасне сільське господарство стикається з низкою викликів, що загрожують стабільності роботи продовольчих систем. Зростання світового населення суттєво підвищує попит на продукти харчування, створюючи тиск на обмежені природні ресурси, зокрема землю, воду та енергію. Разом з тим, кліматичні зміни ускладнюють агровиробництво безпосередньо через підвищення температур, зміни режимів опадів та численні естримальні погодні явища, що призводять до зниження врожайності та погіршення якості вирощуваної продукції.

За визначенням Екодії (громадська організація, яка об’єднує експертів та активістів навколо ідеї збереження довкілля через вплив на прийняття рішень [4]), зміна клімату — це довгострокові зміни різних аспектів кліматичної системи Землі. Центр екологічних ініціатив вважає, що причиною кліматичних змін є парниковий ефект – це явище, під час якого парникові гази (CO2, CH4, N2O, O3) затримують сонячну енергію на поверхні Землі та в атмосфері і не дають їй повернутись назад у космос. Цей процес є цілком природним, та людство суттєво змінює концентрацію газів в атмосфері, спалюючи викопне паливо, як показано на рис.1.1 [5].

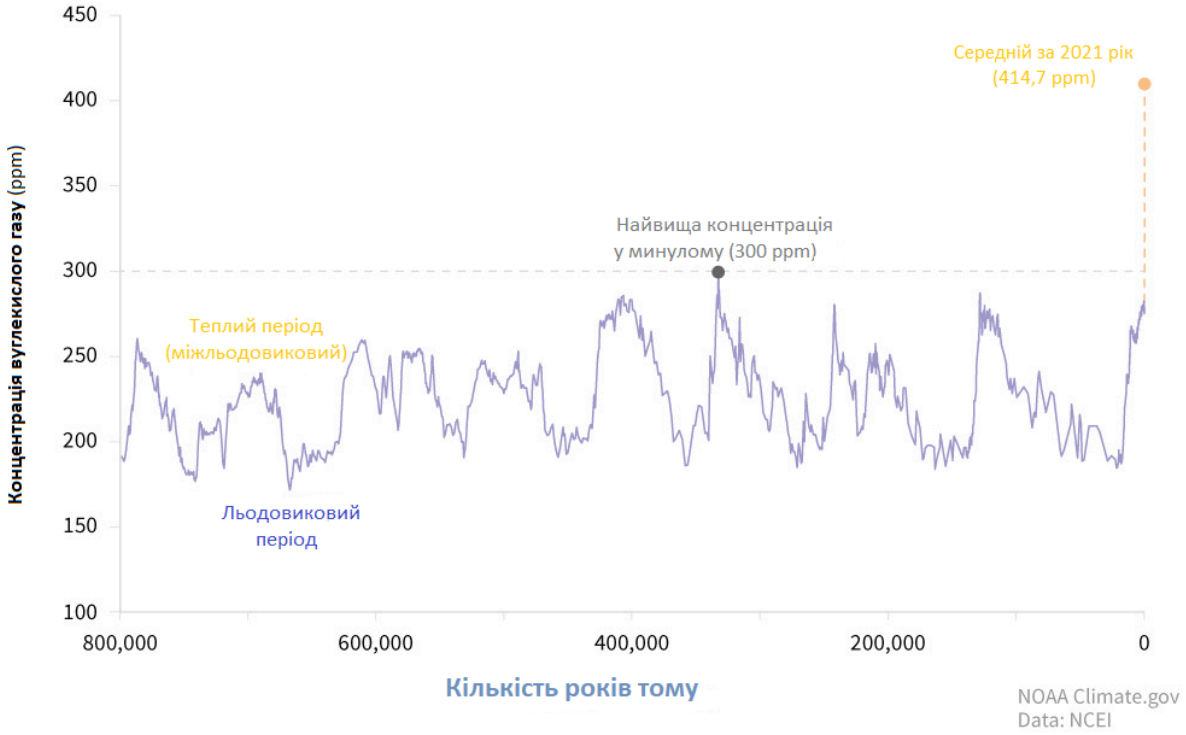



*Рис. 1.1. Графік зміни концентрації вуглекислого газу за крайні 800 років*

В Україні середня температура повітря зросла приблизно на 1.2°C за крайні 30 років. Результати наукових досліджень передбачають зниження рівню водності в Одеській області на 50% за наступні 10 років [3]. Все це так чи інакше впливає на вирощуваність культур.

Негативно впливають на ефективність сільського господарства й деградації та виснаження грунтів, зменшення біорізноманіття та зростання

стійкості шкідників до пестицидів. Людство поступово починає розуміти, що бажані нескінченні ресурси землі давно зникли.

У відповідь на всі перелічені виклики, аграрний сектор активно застосовує сучасні технології. Точне землеробство – це комплексна високотехнологічна система сільськогосподарського сектору, що включає в себе технології глобального позиціонування, географічні інформаційні системи, технології оцінки врожайності, технологію змінного нормування, технології дистанційного зондування землі і рішення технології "інтернет речей" [6].

### 1.3. Теоретичне підгрунтя смарт-агротехнологій

Смарт-агротехнології є інноваційним підходом до ведення сільського господарства, що базується на використанні цифрових технологій для підвищення ефективності, точності та стійкості виробництва. Дані технології допомагають зменшити витрати та підвищують урожайність. Вже на сьогодні, Нідерланди застосовують точне землеробство на найвищому рівні [7].

В основі концепції розумної агротехнології лежить інтеграція новітніх інформаційних технологій у всі частини агровиробництва. Одним із ключових елементів смарт-агроінфраструктури є сенсори — пристрої, що вимірюють температуру, вологість повітря та грунту, рівень освітлення, вміст $CO_2$ та інші параметри, важливі для оптимального росту рослин.

На відміну від традиційного фермерства, що базується переважно на емпіричних знаннях та ручній праці, смарт-фермерство орієнтується на дані та автоматизовані рішення. Наприклад, замість щоденного огляду теплиці фермер може переглядати інформацію з сенсорів у мобільному застосунку та отримувати автоматичні сповіщення про критичні відхилення. Це дозволяє значно знизити витрати, оптимізувати використання ресурсів та підвищити якість продукції.

### 1.4. Смарт-агротеплиці в Україні

Україна має потужний агропромисловий потенціал – аграрництво займає друге місце в структурі всього господарства держави, після індустріального виробництва. Загалом, сільськогосподарська продукція є найбільшою складовою українського експорту (рис. 1.2).

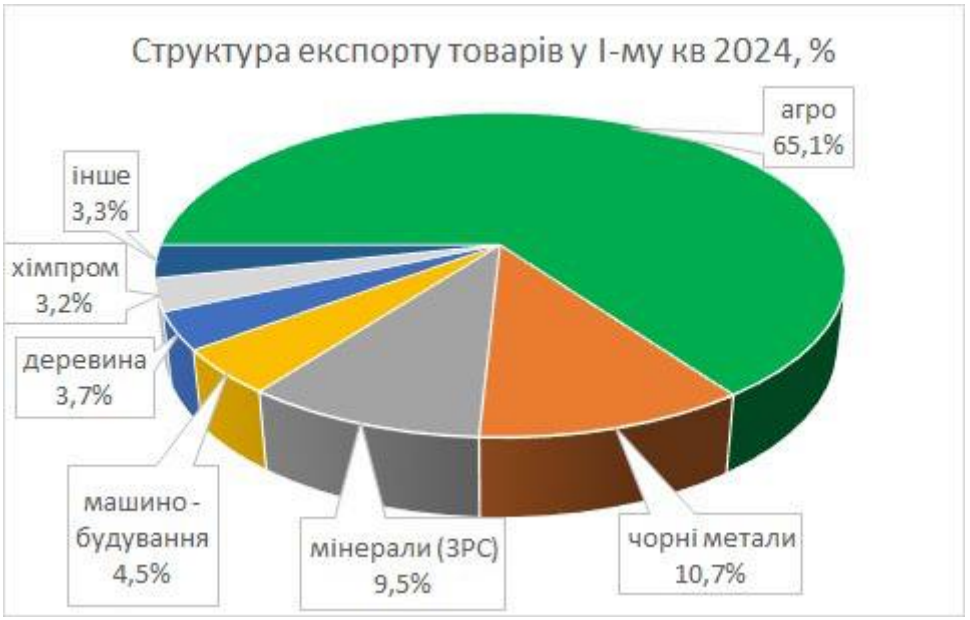



*Рис. 1.2. Експорт товарів в Україні 2024 р.*

Розвиток смарт-тепличних технологій поступово стає відповіддю на виклики сучасного сільського господарства, зокрема перелічені вище критерії. На даний момент, впровадження smart-технологій у тепличному господарстві в Україні відбувається повільніше, ніж в інших країнах ЄС чи США. Хоча, крайні роки демонструють нам зростання зацікавленості як з боку фермерських господарств, так і аграрних стартапів [8].

Наприкінці 2024 року Кабінет Міністрів України визначив основні напрями розвитку сільського господарства на період до 2030 року. Було затверджено кілька стратегічних цілей з метою вирішення наслідків змін клімату та адаптації до них. У розділі “Стратегічна ціль 6. Модернізація

аграрного сектору” описано напрямки цифровізації та інновацій, які є складовими смарт-агротехнологій [11].

Зокрема, стратегія передбачає:

- Цифровізацію сільського господарства
- Розвиток цифрових сільськогосподарських дорадчих послуг
- Інтеграцію цифрових систем

Смарт-агротеплиці поступово впроваджуються в агропромисловий сектор України як ефективний засіб боротьби з кліматичними викликами, дефіцитом води, високими витратами ресурсів та нестабільністю ринку. Одним із найкращих прикладів смарт-тепличних проектів в Україні є Житомирська розумна теплиця, створена на базі місцевого аграрного ліцею. Цей проект має освітньо-інноваційний характер і демонструє застосування мікроконтролерів Arduino для автоматизації температурного режиму, вологості грунту та освітлення. Створена система дозволяє учням вивчати IoT на практиці та одночасно вирощувати екологічно чисту продукцію з мінімальними витратами [9].

Інший приклад – смарт-ферма AquaSmart у Миколаївській області, яка спеціалізується на гідропонному вирощуванні зелені в умовах повністю закритого контролю середовища. В теплиці використовується централізована система моніторингу на базі ESP-модулів і контролерів, які регулюють температуру, рівень pH, електропровідність живильного розчину, рівень води та освітлення. Усі дані надсилаються до хмарного сховища, а керування теплицею здійснюється віддалено з телефону або комп’ютера. AquaSmart також впроваджує елементи штучного інтелекту для прогнозування врожайності на основі історичних даних та погодних трендів [10].

Ці проєкти засвідчують, що навіть за важких обставин українського бізнесу смарт-технології здатні вдало інтегруватися в аграрну галузь. До того ж, на Львівщині функціонує ініціатива "SmartGreen", що об'єднує невеликі

фермерські господарства, які використовують базові автоматизовані системи на базі Raspberry Pi, а на Київщині реалізується проект тепличного комплексу, де використовуються дрони для моніторингу стану культур та прогнозування збору врожаю.

### 1.5. Еволюція підходів до вирощування сільськогосподарських культур

Кліматичні зміни впродовж останніх десятиліть, значно вплинули на сільське господарство, зокрема на технології вирощування сільськогосподарських культур. У відповідь на загрози, що виникли внаслідок таких критичних змін, аграрна галузь пройшла кілька етапів покращення:

- Традиційне землеробство базувалося на ручній праці, природній вологості та відсутності точного контролю умов.
- Органічне виробництво, що набуло популярності з 1970-х років, було акцентоване на екологічній сталості, але також мало обмежену здатність протидіяти зміні клімату.
- Сьогодні аграрії дедалі частіше звертаються до смарт-технологій, що поєднують автоматизовані системи з контролю за кліматом, поливом, добривами, а також використовують сенсори, супутникові дані та аналітику.

Нижче представлено порівняльну таблицю 1.1, яка ілюструє основні відмінності між традиційними підходами та сучасними смарт-технологіями в агровиробництві.

*Таблиця 1.1*

*Порівняння аграрних технологій XX століття та смарт-агротехнологій сьогодення*

| № | Критерій | Традиційні методи (XX ст.) | Smart-агротехнології (XXI ст.) |
|---|---|---|---|
| 1 | Джерело інформації | Досвід фермера, спостереження | Сенсори, супутники, ШІ, Big Data |
| 2 | Система зрошення | Ручна або за графіком | Автоматизована, залежить від вологості |
| 3 | Контроль за кліматом | Відсутній | З інтеграцією IoT |
| 4 | Використання добрив | Загальне, без врахування потреб | Локалізоване, на основі аналізу грунту |
| 5 | Урожайність | Залежить від погоди та досвіду | Стабільна, прогнозована |
| 6 | Ризики | Високі | Мінімізовані за рахунок прогнозування |
| 7 | Екологічне навантаження | Високе | Зменшене, контрольований вплив |
| 8 | Управління | Ручне | Мобільне, через додатки або ПК |

### 1.6. Методологічні підходи до впровадження смарт-агротехнологій

Сучасні методи в галузі смарт-агротехнологій базуються на інтеграції новітніх цифрових інструментів для підвищення ефективності, зменшення людського втручання та оптимізації умов вирощування сільськогосподарських культур.

Ключові технології автоматизації в теплицях:

- IoT – інтернет речей дозволяє підключати датчики, контролери, реле та інші пристрої до єдиної системи моніторингу та керування в реальному часі;
- Big Data – масиви даних з багатьох джерел обробляються для виявлення закономірностей і прогнозування змін кліматичних або біологічних параметрів;
- AI – штучний інтелект використовується для прийняття рішень на основі аналітики даних, наприклад, автоматичне регулювання клімату в теплиці;
- Machine Learning – алгоритми машинного навчання можуть адаптуватися до змін зовнішнього середовища і вдосконалювати стратегії керування;.
- Системи моніторингу мікроклімату – збір інформації про температуру, вологість, освітлення, CO2 та інші параметри середовища.

Порівняння основних підходів автоматизації наведені в таблиці 1.2.

*Таблиця 1.2*

*Порівняння основних підходів автоматизації*

| № | Технологія | Переваги | Обмеження |
|---|---|---|---|
| 1 | IoT | Гнучкість, точний контроль, віддалений доступ | Потребує стабільного інтернет-з'єднання та живлення |
| 2 | Big Data | Великий аналітичний інструмент | Високі вимоги до зберігання та обробки великих обсягів інформації |
| 3 | ШІ | Автоматизація рішень, мінімальне втручання людини | Висока складність реалізації, потреба в даних для навчання |
| 4 | Машинне навчання | Адаптивність до змін, здатність самонавчання | Вимоги до великої кількості навчальних даних, складність реалізації |

У більшості розвинених країн агропромисловість активно впроваджує складні та вартісні смарт-рішення. В Україні ж, значна частина аграрного сектора складається з невеликих приватних теплиць в сільських регіонах, де побудова та встановлення сучасних масштабованих систем є недоречною. Більше того такі розумні агросистеми є дороговартісними та складними в встановленні та обслуговуванні, що унеможливлює їх використання серед простих громадян.

У зв’язку з цим, доцільно акцентувати увагу на простих, бюджетних, але ефективних рішеннях на базі IoT – наприклад, використання недорогих датчиків температури та вологості з можливістю дистанційного моніторингу та ручного регулювання параметрів через мобільний застосунок.

Таким чином завданням нашого проекту є:

1. Створити бюджетну IoT-систему для моніторингу параметрів у малій теплиці, адаптовану до реалій приватного сільського господарства.
2. Забезпечити інтуїтивно зрозумілий інтерфейс для користувача без спеціальної технічної освіти.
3. Забезпечити можливість мінімального керування пристроями теплиці.
4. Спроектувати архітектуру проекту таким чином, щоб вона могла бути розширена в майбутньому.

### 1.7. Критерії для вирощування культур

Для забезпечення високої продуктивності вирощування тепличних культур, надзвичайно важливо визначити і в подальшому підтримувати оптимальні умови мікроклімату, що відповідають всім біологічним потребам вирощуваних культур. Серед усіх факторів зовнішнього середовища, найбільший та найсуттєвіший вплив на ріст та розвиток врожаю рослин мають температура та вологість повітря і грунту, а також рівень освітлення. Вони виступають основними стимуляторами фізіологічних процесів, таких як

фотосинтез, транспірація, дихання, мінеральне живлення, розвиток генеративних (репродуктивних) органів тощо [12, 13].

**Температурний режим** визначає швидкість та якість обміну речовин, активність ферментації та темпи росту вегетативної маси. Підвищення температури до певної межі сприяє інтенсивному фотосинтезу та формуванню біомаси, однак перевищення критичних значень може призводити до пригнічення росту, порушення водного балансу і зниження урожайності. Недостатній рівень тепла, у свою чергу, уповільнює або повністю зупиняє проростання насіння та розвиток культур, що особливо критично на початкових фазах вегетації (період росту та розвитку рослини).

**Вологість повітря і грунту** є не менш важливим чинником. Вода необхідна для поглинання мінеральних речовин, транспортування асимілятів (органічні речовини, утворені в процесі фотосинтезу), терморегуляції рослинного організму та підтримки тургорного (внутрішнього) тиску.

**Освітлення** виступає основним джерелом енергії для фотосинтетичної активності. Його інтенсивність і тривалість світлового дня безпосередньо впливають на швидкість утворення хлорофілу. В умовах нестачі світла відбувається пригнічення фотосинтезу, що веде до сповільнення росту, витягування пагонів та формування менш повноцінного врожаю [12].

Підсумовуючи дану інформацію, можна зробити висновок, що для забезпечення якісного росту і розвитку рослин у теплиці враховуються наступні основні параметри клімату:

- Температура повітря – забезпечує оптимальний тепловий режим;
- Вологість повітря – важлива для запобігання пересихання та хвороб;
- Температура грунту – впливає на кореневу систему та обмін речовин;
- Вологість грунту – необхідна для правильного водопостачання рослин;
- Освітленість – ключовий фактор фотосинтезу.

Для кожної категорії культур у базі даних зберігаються рекомендовані межі кліматичних параметрів, що наведені у таблиці 1.3:

*Таблиця 1.3.*

*Таблиця оптимальних параметрів*

| Категорія | Температура повітря (°C) | Вологість повітря (%) | Температура грунту (°C) | Вологість грунту (%) | Освітленість (люкс) |
|---|---|---|---|---|---|
| Овочі | 18-26 | 60-80 | 18-24 | 65-75 | 10000-22000 |
| Зелень | 15-22 | 70-90 | 16-20 | 60-75 | 4000-10000 |
| Коренеплоди | 12-20 | 60-75 | 10-18 | 65-80 | 6000-12000 |
| Ягоди | 16-24 | 60-85 | 15-20 | 70-80 | 10000-18000 |
| Фрукти | 20-28 | 50-70 | 18-22 | 60-75 | 15000-20000 |
| Тепличні квіти | 18-25 | 60-80 | 16-22 | 55-70 | 8000-16000 |
| Гриби | 14-20 | 80-95 | 14-18 | 75-90 | 500-8000 |

Дані для таблиці 2.2 були взяті з інтернет джерел, наукових статей а також посібниках по рослинництву [2, 12, 14-15].

# РОЗДІЛ 2

# ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

У попередньому розділі було проаналізовано ключові стадії розвитку рослинництва в історичному контексті. На основі дослідження прикладів впроваджених смарт-теплиць визначено їхні переваги та недоліки, що дозволило окреслити головні напрями подальшої розробки інформаційної системи.

## 2.1. Формування вимог до системи

Інформаційна система “Смарт-агротеплиця” передбачає реалізацію адаптивної та доступної всім шарам суспільства розумної агротеплиці, що буде орієнтована на невеликі фермерські господарства чи приватні теплиці.

Система має надавати користувачам наступні функціональні можливості:

- Авторизація та автентифікація користувача;
- Перегляд списку власних телиць користувача з відображенням актуального статусу;
- Додавання нової теплиці з вказанням її параметрів;
- Редагування даних про власну теплицю;
- Видалення теплиці за необхідності;
- Отримання детальної інформації про обрану теплицю;
- Прив’язка теплиці до пристрою Arduino;
- Збір та відображення даних з датчиків у режимі реального часу;
- Розрахунок системою нормативних показників для обраної теплиці на основі її характеристик, обраних культур та сезону посадки;
- Керування виконавчими пристроями: вмикання/вимикання вентилятора, відкривання/закривання дверей;
- Отримання та редагування персональних даних користувача;
- Налаштування користувацьких параметрів системи;

- Вихід з профілю.

До нефункціональних вимог відносяться:

- Інтерфейс користувача – він має бути інтуїтивно зрозумілим, адаптованим до мобільних пристроїв;
- Безпека даних має бути забезпечена через механізми аутентифікації, авторизації та валідації запитів;
- Інтеграція – система повинна підтримувати обмін даними з пристроями на базі Arduino через HTTP-запити.

Безумовно, розроблена інформаційна система має певні обмеження та специфічні вимоги. В основному, це потреба у використанні мікроконтролера з підключеними сенсорами, стабільне інтернет-з’єднання, а також орієнтація виключно на малі приватні господарства або фермерські об'єднання.

### 2.2. Архітектура програмного продукту

Архітектура інформаційної системи “Смарт-агротеплиця” побудована за класичною та звичною клієнт-серверною моделлю з використанням багаторівневої архітектури.

Система складається з трьох основних частин:

1. Клієнтський рівень - реалізований у вигляді мобільного додатку на базі React Native з використанням Expo, що забезпечує зручний інтерфейс для користувача та взаємодію з сервером через HTTP-запити;
2. Серверна частина (Back-end) – побудована на основі ASP.NET Web API, що реалізує логіку обробки запитів, взаємодію з базою даних, авторизацію та автентифікацію користувачів, а також RESTful API для обміну даними між клієнтом і сервером;
3. Пристрій збору даних – мікроконтролер Arduino, до якого підключені сенсори температури, вологості, освітлення тощо. Він періодично відправляє зібрані дані на сервер через HTTP-запити у форматі JSON.

Система функціонує за принципом REST API, де клієнтська частина, тобто мобільний додаток, та пристрої взаємодіють із сервером через чітко визначені маршрути(ендпоінти) за допомогою різним методів HTTP.

На рис. 2.1 зображена архітектура системи, де чітко зображена логіка взаємодії всіх компонентів системи.

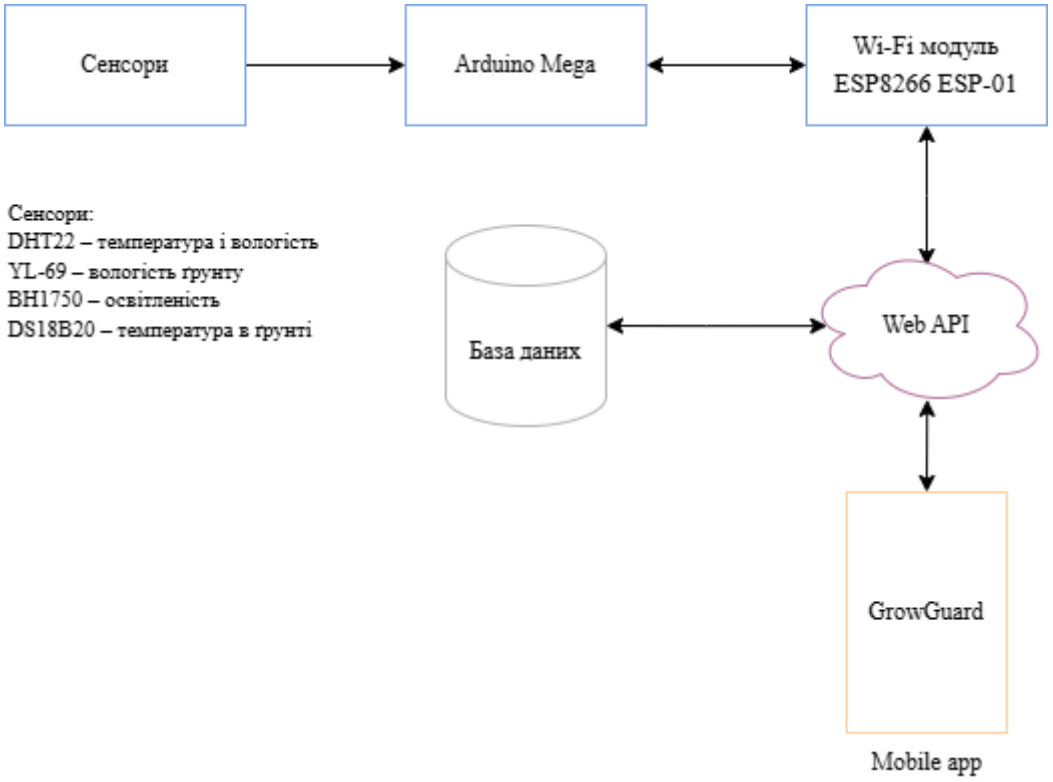



*Рис. 2.1. Архітектура системи*

Для зберігання інформації про користувачів, теплиці, культури, історію вимірювань та інші об'єкти використовується реляційна база даних SQL Server, яка забезпечує надійне, структуроване та ефективне зберігання даних. Зв’язок між Web API і базою даних реалізовано з використанням ORM Entity Framework Core, що дозволяє працювати з БД на рівні об’єктів.

### 2.3. Структура апаратної частини

Система моніторингу теплиці використовує мікроконтролер Arduino Mega як центральний елемент збору та обробки даних із сенсорів. Зібрана інформація передається через Wi-Fi модуль ESP8266 на сервер, що дозволяє здійснювати моніторинг у режимі реального часу. У системі також реалізовано функції керування середовищем — зокрема, вентиляцією та доступом у теплицю.

Перелік апаратних компонентів системи:

- Arduino Mega;
- Wi-Fi модуль ESP8266 (версія ESP-01);
- Сенсор температури та вологості повітря DHT22;
- Сенсор вологості грунту YL-69 (у комплекті з модулем HL-69);
- Сенсор освітленості BH1750;
- Сенсор температури грунту DS18B20;
- Сервопривод Futaba S3003(для імітації відкриття/закриття дверей);
- Система охолодження Raspberry Pi 4 (для керування вентиляцією);
- Макетна плата;
- Перемички;
- Резистори.

Дані компоненти забезпечують точний і надійний доступ користувача до інформації про температуру та вологість повітря, температуру та вологість грунту, а також рівень освітлення – ключові агротехнічні параметри, що впливають на якість і кількість врожаю.

#### 2.3.1. Обґрунтування вибору компонентів системи

Вибір апаратних компонентів системи було здійснено з урахуванням результатів досліджень та завдяки професійній підтримці старшого викладача Повшенюка Андрія Петровича, консультації з яким мали суттєве значення для формування технічного складу пристрою та успішного створення першої працюючої моделі системи.

**Мікроконтролер Arduino Mega** (рис. 2.2) обраний з розрахунком на достатню кількість вхідних та вихідних портів, обсягом оперативної пам’яті та компактний розмір. Саме цей компонент забезпечує взаємодію з усіма іншими частинами системи та забезпечує зв’язок із зовнішньою мережею. Він виконує роль центрального керуючого елемента системи.

*Рис. 2.2. Мікроконтролер Arduino Mega*

**Wi-Fi модуль ESP8266 (версія ESP-01)** (рис. 2.3) має вбудований Wi-Fi, що дає змогу підключити мікроконтролер до мережі Інтернет без додаткових пристроїв. Більше того, він легко інтегрується в любі компактні пристрої завдяки малим розмірам та низькому енергоспоживанню. Для забезпечення стабільного запуску і роботи ESP8266 було використано два підтягувальні резистори, які підключено через макетну плату до мікроконтролера Arduino.

*Рис. 2.3. Wi-Fi модуль ESP8266 (версія ESP-01)*

**Цифровий датчик температури та вологості повітря DHT22** (рис. 2.4) використовується для вимірювання двох критичних параметрів мікроклімату

теплиці — температури та відносної вологості повітря. DHT22 характеризується високою точністю(для температури ± 0,5°C, для вологості ±2–5%.), широким діапазоном вимірювання та стабільною роботою в умовах підвищеної вологості. Завдяки цифровому інтерфейсу він легко інтегрується з Arduino.

*Рис. 2.4. Цифровий датчик температури та вологості повітря DHT22*

**Грунтовий датчик вологості YL-69 / HL-69** (рис. 2.5) застосовується для вимірювання рівня вологості грунту шляхом аналізу його електропровідності. Цей елемент складається з сенсорної пластини та модуля керування.

*Рис. 2.5. Датчик вологості грунту YL-69 / HL-69*

**Датчик освітленості BH1750** (рис. 2.6) призначений для визначення інтенсивності рівня освітлення, вимірюється в одиницях виміру люкс. Він використовує цифровий інтерфейс I2C, що дозволяє передавати точні дані до мікроконтролера з мінімальними затримками. Завдяки високій чутливості та стабільності, BH1750 дозволяє точно вимірювати освітленість у широкому

діапазоні, що особливо важливо для моніторингу світла, необхідного для фотосинтезу в теплицях.


*Рис. 2.6. Датчик освітленості BH1750*

**Датчик DS18B20** (в пластиковій гільзі) (рис. 2.7) застосовується для точного контролю температури грунту, що є одним з важливих факторів для правильного розвитку кореневої системи рослин. Завдяки захищеному корпусу датчик стабільно працює у вологому середовищі грунту, забезпечуючи точність вимірювань для аналізу та автоматичного регулювання умов.

*Рис. 2.7. Датчик температури грунту*

**Сервопривод Futaba S3003** (рис. 2.8) використовується для механічного керування дверима теплиці. Його точне позиціонування дозволяє автоматично відкривати або закривати двері залежно від умов середовища, що контролюються сенсорами, а також виконувати команди користувача через мобільний додаток.

*Рис. 2.8. Сервопривод Futaba S3003*

**Система охолодження Raspberry Pi 4** (рис. 2.9) використовується для керування системою провітрювання теплиці.

*Рис. 2.9. Система охолодження Raspberry Pi 4*

Усі обрані сенсори та виконавчі пристрої мають функціональне значення для підтримки автоматизованого контролю мікроклімату в теплиці. Кожен компонент відповідає за вимірювання одного з ключових параметрів навколишнього середовища: температури повітря, вологості повітря й грунту, освітленості тощо. Відповідність між функціями компонентів і вимогами до системи дозволяє досягти високої точності збору даних і забезпечити керування в режимі реального часу. Усі використані апаратні елементи, їх функціональне призначення та причини вибору наведені в таблиці 2.1.

*Таблиця 2.1*

*Основні апаратні компоненти пристрою*

| № | Компонент | Функція | Переваги |
|---|---|---|---|
| 1 | Arduino Mega | Обробка сигналів від сенсорів | Велика кількість пінів, обсяг оперативної пам’яті |
| 2 | ESP8266 ESP-01 | Передача даних на сервер | Компактний, бюджетний, підтримує HTTP |
| 3 | DHT22 | Вимірювання температури та вологості повітря | Висока точність |
| 4 | DS18B20 | Вимірювання температури грунту | Точний цифровий сенсор із широким діапазоном |
| 5 | YL-69 / HL-69 | Вимірювання вологості грунту | Простий в інтеграції, доступний |
| 6 | BH1750 | Вимірювання освітленості | Має цифровий інтерфейс, зручний для Arduino |
| 7 | Raspberry Pi 4 | Керування вентиляцією | Висока обчислювальна здатність |
| 8 | Futaba S3003 | Механічне відкривання/закривання дверей | Надійність, точність |
| 9 | Макетна плата, перемички | З'єднання компонентів | Зручно для прототипування |
| 10 | Резистори (3 шт.) | Напруга для ESP8266 і сенсора грунту | Забезпечення стабільної роботи |

На рис. 2.10 наведено реальний приклад підключення основних апаратних компонентів до мікроконтролера, що відображає логіку побудови пристрою збору даних. До мікроконтролера підключено сенсори температури, вологості повітря та грунту, а також освітленості, модуль Wi-Fi для передачі даних на сервер, а також виконавчі пристрої (вентилятор і сервопривід для дверей).

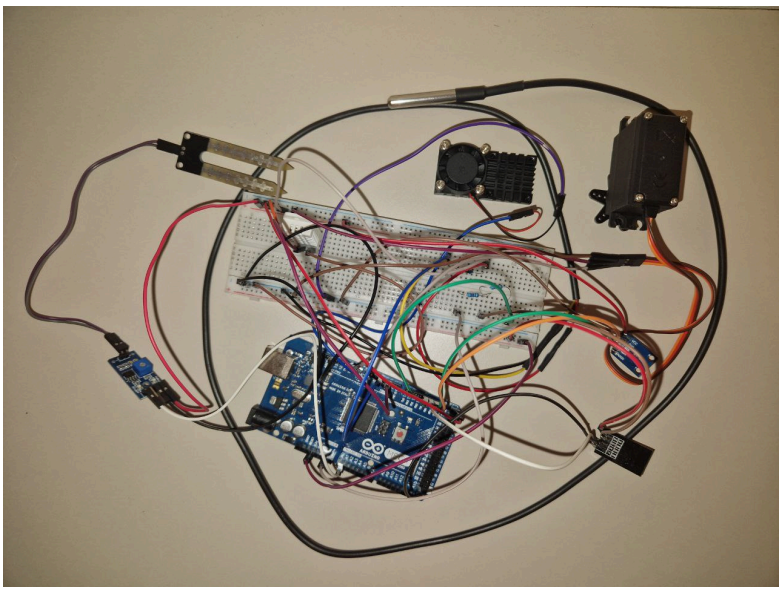


*Рис. 2.10. Підключення компонентів системи*

Зазначимо, що на схемі використано умовні або аналогічні компоненти, оскільки деякі реальні сенсори або модулі були недоступні у візуалізаційному середовищі. Проте схема достовірно ілюструє принцип підключення та взаємодії між елементами системи.

### 2.4. Архітектура серверної частини системи

Серверна частина інформаційної системи реалізована у вигляді веб-API, створеного з використанням платформи ASP.NET Core Web API. Її головна ціль полягає у забезпеченні централізованого збору, обробки та зберігання даних, надісланих від пристрою.

Ця частина інформаційної системи виступає посередником між мікроконтролером і базою даних (див. рис. 2.1) . Це забезпечує надійний обмін інформацією за допомогою HTTP запитів. Система отримує від Arduino оновлення показників мікроклімату теплиці у вигляді JSON-об’єктів, які обробляються відповідними методами контролера.

Запити, що надсилаються від Arduino, є POST запитами, що містять у тілі серіалізовані показники. API, у свою чергу, приймає ці запити, виконує базову

валідацію вхідних даних та передає їх у відповідні сервіси для подальшої обробки та збереження у базі даних.

### 2.5. Архітектура клієнтського застосунку

Клієнтський застосунок GrowGuard реалізований як мобільний застосунок, призначений для віддаленого моніторингу параметрів теплиці та керування пристроями (вентилятором і дверима) у режимі реального часу. Архітектура застосунку побудована з дотриманням принципів модульності, розділення відповідальностей та масштабованості.

Застосунок має екранну структуру, яка дозволяє користувачеві зручно перемикатися між основними функціональними розділами.

Основні функціональні області застосунку:

- Моніторинг теплиці – перегляд поточних показників температури, вологості, освітлення тощо.
- Норми та рекомендації – перегляд норм для клімату системи, обчислених на сервері.
- Керування теплицею – інтерфейс для зміни стану дверей та вентилятора.
- Інформація про теплицю – перегляд деталей і параметрів обраної теплиці.
- Налаштування теплиці – редагування встановлених оптимальних значень кліматичних умов.
- Форма додавання/оновлення теплиці – можливість додати або змінити дані теплиці.
- Реєстрація/авторизація користувача – екрани входу в систему або створення облікового запису.

Вся навігація між екранами організована в логічну структуру, яка дозволяє легко орієнтуватися в застосунку. Структура папок у проекті відображає розділення логіки за функціональними зонами, де кожен екран або функціональний блок винесений в окремий модуль.

Користувацький інтерфейс розроблено таким чином, щоб він був інтуїтивно зрозумілим та адаптованим до використання на мобільних пристроях. Для підтримки повторного використання та уніфікації стилів інтерфейсу використовуються спільні компоненти інтерфейсу.

Архітектура клієнта також враховує потребу в збереженні важливої інформації локально на пристрої користувача, а також у синхронізації з віддаленим сервером, зокрема для отримання показників з сенсорів теплиці та передавання команд на керування обладнанням.

### 2.6. Алгоритм функціонування системи

У цьому підрозділі описується загальна логіка функціонування мобільного застосунку GrowGuard, розробленого для дистанційного моніторингу стану теплиць та керування підключеними пристроями (вентилятором, дверима).

Ключові процеси, реалізовані у системі, включають:

- додавання нової теплиці користувачем;
- автоматичний збір даних із сенсорів;
- обробку даних з подальшим формуванням рекомендацій;
- керування пристроями на основі стану або вручну.

**Алгоритм додавання теплиці**

***Мета:*** надати користувачу можливість зареєструвати нову теплицю в системі для подальшого моніторингу.

***Послідовність дій:***

1. Користувач переходить до форми додавання теплиці;
2. Вводить назву теплиці, розміри (довжина, ширина, висота), сезон початку вирощування, розміщення, обирає культури, які планує вирощувати. Оскільки форма розташована на двох сторінках додатку, дані зберігаються в локальному сховищі для подальшого надсилання на сервер;

3. Якщо введені користувачем дані валідні – вони надсилаються на сервер, де проводиться розрахунок оптимальних значень для даної теплиці по її даних, відбувається штучна прив’язка до пристрою, оскільки в лабораторних умовах ми обмежені в лише одному девайсі для тестування, а також записуються перші дані про стан пристроїв керування;
4. Теплиця зберігається у базі даних, прив’язана до конкретного користувача.

На рис. 2.11 зображена детальна блок-схема, де прослідковується відповідний алгоритм створення нової теплиці.

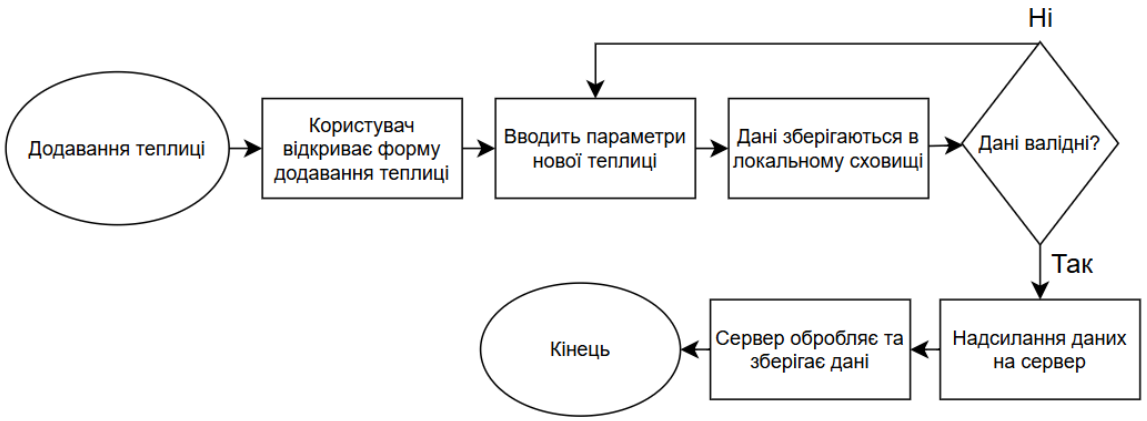



*Рис. 2.11. Блок-схема створення теплиці*

**Збір даних із сенсорів**

***Мета:*** автоматичне отримання актуальних кліматичних даних в реальному часі (температура, вологість, освітлення). ***Послідовність дій:***

1. Сенсори періодично зчитують дані;
2. Отримані значення конвертуються у відповідні типи, та надсилаються за допомогою Wi-Fi модуля на сервер;
3. Сервер отримує запит і зберігає показники в базу даних;
4. Мобільний застосунок періодично оновлює відображення даних використовуючи SignalR.

На рис. 2.12 зображена блок схема алгоритму отримання даних із сенсорів в реальному часі.


*Рис. 2.12. Блок-схема отримання даних із сенсорів*

**Отримання попереджень**

***Мета:*** надати користувачу рекомендації стосовно оптимізації кліматичних умов у теплиці.

***Послідовність дій:***

1. Після надходження нових сенсорних даних, система порівн.є їх з оптимальними межами для обраної культури;
2. Якщо виявлено відхилення – генерується відповідне повідомлення;
3. Користувачу надається повідомлення із переліком відхилень та рекомендацією звіритись з нормами.

На рис. 2.13 зображена блок-схема алгоритму отримання рекомендацій стосовно оптимізації кліматичних умов.


*Рис. 2.13. Блок-схема отримання рекомендацій*

**Керування пристроями**

***Мета:*** надати користувачу можливість вручну вмикати і вимикати окремі пристрої, що знаходяться в теплиці, такі як вентилятор або двері.

***Послідовність дій:***

1. Користувач відкриває інтерфейс керування у мобільному додатку;
2. Обирає дію, яку бажає виконати;
3. Застосунок надсилає команду на сервер;
4. Сервер обробляє повідомлення та надсилає відповідну команду Arduino через послідовний порт у вигляді текстової команди;
5. Arduino реагує на отриману команду та вмикає чи вимикає пристрої.

На рис. 2.14 зображено блок-схему алгоритму системи для реалізації керування пристроями.


*Рис. 2.14. Блок-схема алгоритму керування пристроями*

## 2.7. Логіка обчислення оптимальних параметрів

У цьому розділі детально описується логіка розрахунку системою оптимальних параметрів для підтримки комфортних для культур умов у смарт-агротеплиці з урахуванням потреб вирощуваних культур. В основі системи лежить база даних, що містить інформацію про оптимальні кліматичні умови для різних категорій рослин, які використовуються у проекті.

### 2.7.1. Алгоритм оцінки стану теплиці

Наш проект в якомусь плані виступає таким собі аграрним калькулятором, оскільки найголовнішою функцією системи є оцінка стану системи та підбір

рекомендацій.Оцінка поточного стану клімату у теплиці виконується на основі останніх показників сенсорів: температури повітря і грунту, вологості повітря і грунту, а також рівня освітленості. Логіка працює за наступним принципом:

1. Отримуються крайні доступні дані з сенсорів теплиці;
2. Дані порівнюються з оптимальними межеми для обраних користувачем культур;
3. Для кожного параметру визначається ступінь відхилення:
   - Критичне відхилення – якщо значення виходить за межі 10% від оптимального діапазону;
   - Незначне відхилення – якщо значення знаходиться поза межами, але менше ніж 10%;
4. Якщо присутні критичні відхилення — статус теплиці вважається “поганим”;
5. Якщо є лише незначні відхилення — статус стає “попереджувальним”;
6. Якщо всі параметри у межах норми — статус “в нормі”;
7. Освітленість оцінюється з урахуванням часу доби та сезону, з коригуванням за сезонним коефіцієнтом.

Для кожної теплиці користувач може встановити власні оптимальні параметри клімату, виходячи з особливостей культур, локального клімату та особистих уподобань. Це дає можливість адаптувати систему під індивідуальні потреби та отримати точні рекомендації.

# РОЗДІЛ 3

# ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

У даному розділі описано програмну реалізацію інформаційної системи “Смарт-теплиця”, яка забезпечує моніторинг і управління кліматичними параметрами тепличного середовища. Серверну частину системи реалізовано на основі платформи ASP.NET Core Web API, що забезпечує надійну обробку запитів та взаємодію з базою даних. Клієнтська частина розроблена із використанням React Native у поєднанні з фреймворком Expo, що дозволяє створювати кросплатформенний мобільний застосунок із сучасним інтерфейсом. Апаратна складова системи побудована на основі мікроконтролера з підключеними датчиками для збору кліматичних даних (температури, вологості тощо) та Wi-Fi модуля для передавання інформації. Програмування мікроконтролера здійснюється в середовищі Arduino IDE з використанням мови C++.

## 3.1. Логіка роботи апаратної частини

Прошивка для Arduino Mega реалізована мовою програмування C++ у середовищі Arduino IDE. Її основна мета — зчитування показників з підключених сенсорів, формування структурованих JSON-даних та передача їх на сервер за допомогою Wi-Fi модуля ESP8266 (версія ESP-01).

Для забезпечення цієї логіки використані такі бібліотеки:

- DHT.h - для роботи з DHT22;
- OneWire.h та DallasTemperature.h для зчитування DS18B20;
- Servo.h - для керування сервоприводом відкриття дверей;
- BH1750.h – для вимірювання рівня освітленості;

- SoftwareSerial.h - для зв’язку Arduino Mega з Wi-Fi модулем через AT-команди;
- Servo.h - для керування сервоприводом (відкриття/закриття дверей теплиці);

Основні етапи логіки роботи:

1. Функція setup() відповідає за ініціалізацію компонентів. У ній встановлюється зв’язок з датчиками та відбувається їх запуск, налаштовується підключення до мережі з wi-fi;
2. Повторюване зчитування та форматування даних у функції loop() запускається циклічно та обробляє два типи подій: отримання команд з сервера та періодичне зчитування даних;
3. Наступним кроком є формування даних у JSON рядок. Окрім показників середовища, він також включає серійний номер пристрою та айді теплиці. Ці дані поки є сталими незмінними оскільки в лабораторних умовах пристрій лише один. В подальшому планується допрацювання логіки прив’язування пристрою;
4. Передача даних на сервер відбувається за допомогою функції sendToServer(), де формується повноцінний HTTP POST-запит, який надсилається через AT-команди ESP8266 на вказану IP-адресу сервера.

На рис. 3.1 зображена блок-схема логіки роботи мікроконтролера.

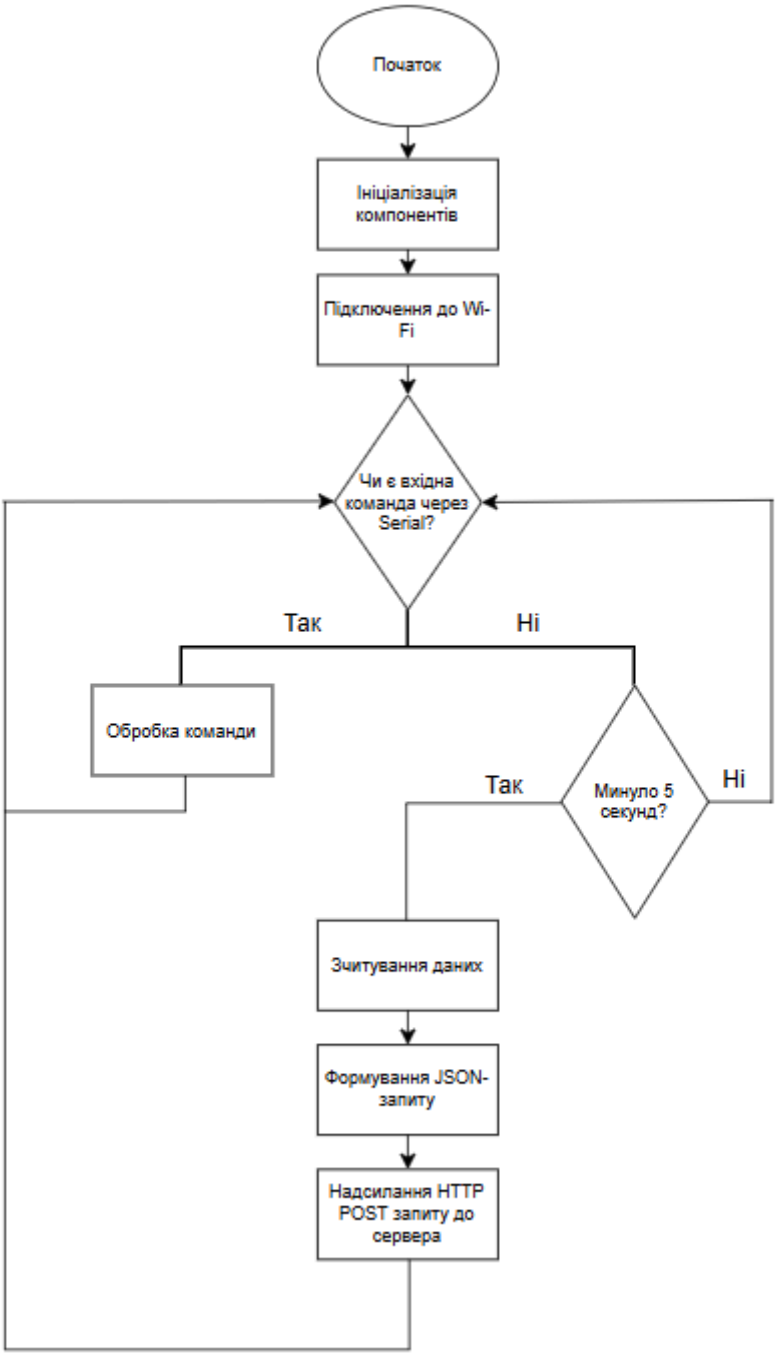



*Рис. 3.1. Блок-схема алгоритму роботи мікроконтролера*

Перелік основних функцій у коді можна переглянути в таблиці 3.1.

*Таблиця 3.1*

*Перелік основних функцій у коді*

| № | Функція | Призначення |
|---|---|---|
| 1 | setup() | Ініціалізація сенсорів, сервоприводу, порту та Wi-Fi з'єднання |
| 2 | loop() | Основний цикл зчитування, формування JSON і взаємодії з пристроями |
| 3 | sendToServer() | Створення та надсилання HTTP POST-запиту на API |
| 4 | connectWiFi() | Налаштування Wi-Fi-з'єднання через ESP8266 за допомогою AT-команд |
| 5 | flushSerial1() | Очистка серійного буфера UART1 |

### 3.1.1. Підключення до Wi-Fi

Однією з найнеобхідніший умов роботи системи є підключення мікроконтролера до мережі Інтернет. У даній кваліфікаційній роботі для підключення використовується модуль ESP8266, який підтримує роботу з мережами Wi-Fi за допомогою AT-команд. Логіка підключення до бездротової мережі винесена в окрему функцію connectWiFi().

```
void connectWiFi() {
  Serial1.println("AT+RST");
  delay(2000);
  flushSerial1();
  Serial1.println("AT+CWMODE=1");
  delay(2000);
  flushSerial1();
  Serial.print("Connecting to WiFi: ");
  Serial.println(ssid);
```

```
  Serial1.print("AT+CWJAP=\"");
  Serial1.print(ssid);
  Serial1.print("\",\"");
  Serial1.print(password);
  Serial1.println("\"");
  delay(15000);
  flushSerial1();
 }
```

Таким чином ця функція забезпечує підключення Wi-Fi модуля (ESP8266) до бездротової мережі за допомогою AT-команд. Всі затримки, які вимагає ця функція пояснюються необхідністю встановлення з’єднання. Безпосередньо для підключення використовується команда AT+CWJAP="SSID","PASSWORD" – з відповідно вказаними ім’ям та паролем Wi-Fi.

### 3.1.2. Обробка даних з датчиків

У системі важливо не лише зчитувати показники з датчиків, а й правильно їх обробляти, калібрувати та підготувати у форматі, придатному для надсилання на сервер. Ці дії реалізовано в основному циклі loop() у блоці, який виконується кожну хвилину (sensorInterval = 60000).

```
if (now - lastSensorRead >= sensorInterval) {
  lastSensorRead = now;
  float airTemp = dht.readTemperature();
  float airHum = dht.readHumidity();
  soilTempSensor.requestTemperatures();
  float soilTemp = soilTempSensor.getTempCByIndex(0);
  if (soilTemp == 85.0 || soilTemp == -127.0) {
   Serial.println("Очікуємо стабілізації датчика...");
   delay(500);
   soilTempSensor.requestTemperatures();
   soilTemp = soilTempSensor.getTempCByIndex(0);
```

```
}
int rawSoil = analogRead(SOIL_MOISTURE_PIN);
float soilHum = map(rawSoil, sensorMin, sensorMax, 100, 0);
float lightLevel = lightMeter.readLightLevel();
 }
```

Даний код демонструє особистий підхід до кожного датчика. Наприклад, датчик, що вимірює вологість грунту може бути дещо нестабільним на початку роботи, саме тому система розпізнає такі значення та очікує на стабілізацію DS18B20 для коректної передачі даних. Датчик вологості грунту, в свою чергу вимагає калібрування для передачі даних у відсотках. Функція map() дозволяє перетворити сирі дані в зрозумілі відсотки вологості.

Наступна частина коду, об’єднує всі отримані дані у JSON-рядок та передає створений об’єкт у функцію sendToServer(json).

```
String json = "{";
  json += "\"DeviceSerialNumber\":\"ARDUINO-001\",";
  json += "\"GreenhouseId\":1,";
  json += "\"AirTemp\":" + String(airTemp, 2) + ",";
  json += "\"AirHum\":" + String(airHum, 2) + ",";
  json += "\"SoilHum\":" + String(soilHum, 2) + ",";
  json += "\"SoilTemp\":" + String(soilTemp, 2) + ",";
  json += "\"LightLevel\":" + String(lightLevel, 2);
  json += "}";
  sendToServer(json);
```

Деякі дані, такі як: ідентифікатор теплиці та серійний номер пристрою задаються явно, оскільки ми працюємо в лабораторних умовах та не маємо можливості створити кілька девайсів для тестування. В подальшому ці дані обробляються на сервері та набувають коректних значень.

### 3.1.3. Надсилання даних на сервер

Для надсилання підготовлених даних на сервер використовується функція sendToServer(String json). З'єднання відбувається через модуль ESP8266 за допомогою коду надсилання команди:

```
String cmd = "AT+CIPSTART=\"TCP\",\"" + server + "\"," + String(port);
Serial1.println(cmd);
```

Після цього формується HTTP POST запит та відбувається спроба надіслати дані на сервер:

```
String request = "POST " + endpoint + " HTTP/1.1\r\n";
request += "Host: " + server + ":" + String(port) + "\r\n";
request += "Content-Type: application/json\r\n";
request += "Content-Length: " + String(json.length()) + "\r\n";
request += "Connection: close\r\n\r\n";
request += json;

Serial1.print("AT+CIPSEND=");
Serial1.println(request.length());
```

Після завершення передавання, з’єднання закривається через: AT+CIPCLOSE.

### 3.1.4. Обробка та виконання команд

Система передбачає можливість керування зовнішніми пристроями — зокрема, вентилятором та сервоприводом, — через серійний порт. Якщо в порт надходить текстова команда, вона зчитується і аналізується.

```
if (Serial.available()) {
  String command = Serial.readStringUntil('\n');
  command.trim();
  Serial.println("Отримано команду: " + command);
  if (command == "FAN_ON") {
```

```
    digitalWrite(fanPin, HIGH);
  } else if (command == "FAN_OFF") {
    digitalWrite(fanPin, LOW);
  } else if (command == "DOOR_OPEN") {
    myServo.write(180);
  } else if (command == "DOOR_CLOSE") {
    myServo.write(0);
  }
}
```

Підтримуються такі команди:

- FAN_ON — увімкнення вентилятора;
- FAN_OFF — вимкнення вентилятора;
- DOOR_OPEN — відкриття дверцят;
- DOOR_CLOSE — закриття дверцят;

Таким чином, реалізовано базове керування елементами теплиці.

### 3.2. Реалізація серверної частини

#### 3.2.1. Технології, бібліотеки та шаблони проектування

Для розробки серверної частини застосунку було обрано платформу .NET 8. Проект реалізовано у вигляді Web API з використанням REST-архітектури.

Основні технології та бібліотеки, що використовуються у проекті:

- Entity Framework Core – використано для роботи з базою даних, що забезпечує роботу через об’єктно-реляційне відображення, дозволяючи маніпулювати даними у вигляді об’єктів;
- Fluent API застосовується для налаштування моделі бази даних, зв’язків між таблицями, обмежень і поведінки при видаленні, що дозволяє більш гнучко і детально описати структуру БД, ніж це можливо через атрибути.
- AutoMapper – для автоматичного мапінгу між DTO та моделями;

- FluentValidation – дозволяє централізовано обробляти валідацію вхідних даних і повертати зрозумілі повідомлення про помилки.;
- JWT Bearer Authentication – для реалізації авторизації на основі токенів;
- ASP.NET Core Identity – для керування користувачами та ролями;
- SignalR – для реального часу комунікації (використовується для оновлення даних з сенсорів);
- Swagger – для документування API;
- System.IO.Ports – для взаємодії з Arduino через COM-порт.

Для організації бізнес-логіки в проекті застосовується паттерн “Repository”, що дозволяє абстрагуватися від конкретної реалізації збереження даних та не звертатись безпосередньо до контексту. Репозиторій реалізований у вигляді шаблонного інтерфейсу, що забезпечує гнучкість коду та тестованість.

Усі методи репозиторію підтримують асинхронність, що підвищує продуктивність системи, знижуючи рівень блокування потоків під час роботи з базою.

### 3.2.2. Структура проекту та основні компоненти

Серверна частина додатку реалізована з дотриманням принципів розділення відповідальностей та структури, характерної багаторівневій архітектурі. Це дозволяє легко масштабувати систему, покращує читабельність коду, його підтримку та тестування.

Проект побудований за допомогою багаторівневої архітектури, тому має:

- Презентаційний рівень: Controllers, Hubs;
- Рівень бізнес-логіки: Services, Helpers, Validation, Mapping, Interfaces;
- Рівень доступу до даних: Repositories, Configuration, Migrations;
- Рівень моделей: Entities, DTOs.

Зображення структури серверної частини системи можна побачити на рис. 3.2.

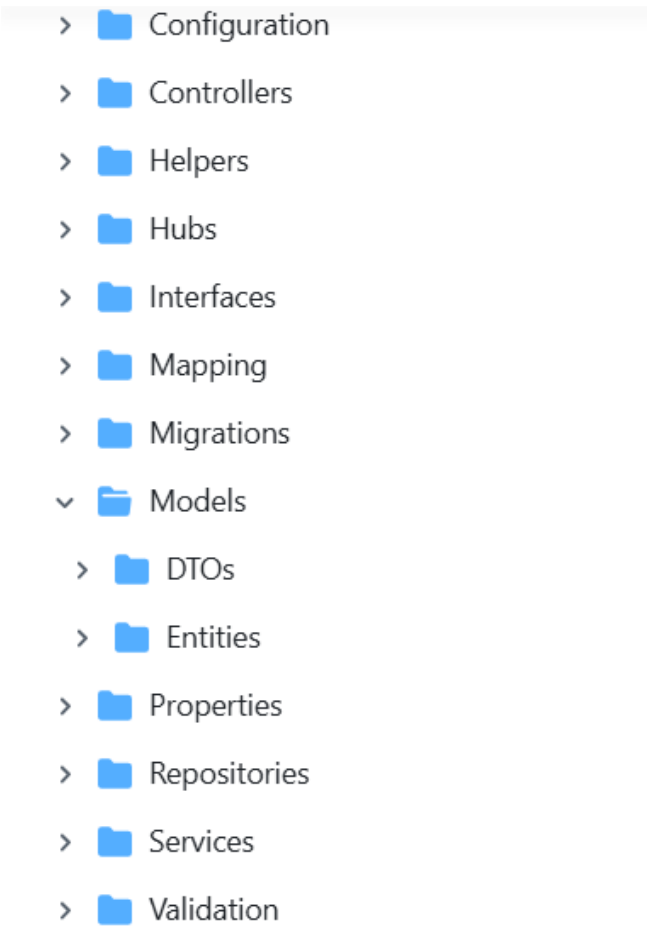



*Рис. 3.2. Структура серверної частини системи*

Така структура серверу забезпечує підтримуваність та гнучкість а також дозволяє масштабувати проект у майбутньому.

### 3.2.3. Реалізація патерну Repository

Паттерн репозиторій – це такий паттерн для проектування програмного забезпечення. Його основна ціль: ізолювати бізнес-логіку додатку від деталей реалізації доступу до даних, тим самим дотримуючись принципу розділення відповідальностей. Такий підхід дозволяє працювати з даними на рівні об'єктів, не прив'язуючись до конкретної технології збереження.

У проєкті реалізовано два рівні:

1. Універсальний репозиторій (Repository<T>) – забезпечує базову логіку для всіх сутностей;

2. Спеціалізовані репозиторії – реалізують інтерфейси з методами, специфічними для конкретних сутностей.

Базовий інтерфейс IRepository<TEntity> описує загальний набір CRUD-операцій (Create, Read, Update, Delete) для будь-якої сутності.

```
public interface IRepository<TEntity> where TEntity : class
{
    IEnumerable<TEntity> Get(
      Expression<Func<TEntity, bool>> filter = null,
            Func<IQueryable<TEntity>, IOrderedQueryable<TEntity>>
orderBy = null,
      params string[] includeProperties);

    TEntity GetById(object id);
    void Create(TEntity entity);
    void Update(TEntity entityToUpdate);
    void Delete(object id);
    void Delete(TEntity entityToDelete);
    void Save();
    Task AddAsync(TEntity entity);
    Task<List<TEntity>> GetAllAsync();
    Task SaveAsync();
}
```

Цей інтерфейс дозволяє працювати з будь-якою сутністю через універсальний підхід. До того ж, він підтримує асинхронні методи, що підвищує продуктивність при роботі з БД.

Спеціалізований інтерфейс IUserRepository створений спеціально для сутності користувача, щоб реалізувати специфічну логіку.

```
public interface IUserRepository
{
```

```
    Task<AppUser> GetByUserNameAsync(string userName);
    Task<AppUser> GetByIdAsync(int id);
    Task CreateRefreshTokenAsync(RefreshToken token);
    Task<RefreshToken> GetRefreshTokenAsync(string token);
    Task RevokeRefreshTokenAsync(RefreshToken token);
    Task SaveAsync();
}
```

Репозиторій для користувача містить методи, що стосуються аутентифікації та роботи з токенами оновлення, які не є універсальними для інших сутностей.

### 3.2.4. Fluent API

Fluent API — це один із двох використаних способів конфігурації моделей у Entity Framework Core (інший спосіб – через атрибути [DataAnnotations]). Fluent API дозволяє гнучко і детально налаштовувати структуру бази даних, зв’язки між таблицями, обмеження, типи даних, поведінку при видаленні.

Основні переваги Fluent API:

- повна підтримка складних налаштувань (наприклад: зовнішні ключі, каскадне видалення, типи колонок);
- винесення конфігурації в окремі класи, що дозволяє дотримуватись принципу Single Responsibility;
- легше масштабувати та змінювати налаштування без втручання в самі класи сутностей.

Усі конфігураційні файли винесені до папки Configuration, де кожен клас відповідний окремій сутності та реалізовує інтерфейс IEntityTypeConfiguration<TEntity>. Наприклад, код конфігурації для сутності Рослини:

```
public class PlantConfiguration : IEntityTypeConfiguration<Plant>
{
```

```
    public void Configure(EntityTypeBuilder<Plant> builder)
    {
        builder.ToTable("Plants");
        builder.HasKey(p => p.Id);
        builder.Property(p => p.Category)
            .IsRequired()
            .HasMaxLength(100);
        builder.Property(p=>p.ExampleNames)
            .HasMaxLength(500);
    }
}
```

У цьому прикладі вказується назва таблиці, первинний ключ, не нульовий рядок та обмеження на довжину рядка.

### 3.2.5. DTO, AutoMapper

DTO – це також один із шаблонів проектування, який зачасту використовують для забезпечення безпечної та структурованої передачі інформації між клієнтом і сервером. DTOs ізольовують внутрішню структуру моделей БД від зовнішнього представлення, тим самим забезпечують захист чутливої інформації та покращують контроль над тим, які саме дані надсилаються у запитах та відповідях.

У проекті DTO класи зберігаються в папці Models/DTOs, окремо від основних сутностей (Models/Entities), що також відповідає принципу розділення відповідальностей.

```
public class PlantExampleReadDto
{
    public int Id { get; set; }
    public string Category { get; set; }
    public List<string> ExampleNames { get; set; }
```

```
    }
```

Для автоматичного мапінгу між DTO та сутностями використовується бібліотека AutoMapper. Це дозволяє уникнути ручного присвоєння полів і мінімізує кількість шаблонного коду.

```
public class MappingProfile : Profile
    {
        public MappingProfile()
        {
            CreateMap<Plant, PlantExampleReadDto>();
        }
    }
```

У проєкті мапінги налаштовані в окремому профілі, який зберігається в папці Mapping.

### 3.2.6. Валідація з FluentValidation

Для перевірки коректності вхідних даних у проекті використовується бібліотека FluentValidation, яка надає зручний та гнучкий спосіб визначення правил валідації для моделей. Приклад валідатора для реєстрації користувача:

```
public class RegisterDtoValidator : AbstractValidator<RegisterDto>
{
    public RegisterDtoValidator()
    {
        RuleFor(x => x.UserName)
            .NotEmpty().WithMessage("Username is required.")
                .Length(3, 20).WithMessage("Username must be between 3 and 20 characters.");
        RuleFor(x => x.Email)
            .NotEmpty().WithMessage("Email is required.")
            .EmailAddress().WithMessage("Invalid email format.");
```

```
        RuleFor(x => x.Password)
            .NotEmpty().WithMessage("Password is required.")
                .MinimumLength(6).WithMessage("Password must be at least 6 characters long.");
    }
}
```

У цьому прикладі визначено валідатор для RegisterDto — моделі, яка використовується для реєстрації користувачів. Кожне поле проходить перевірку на заповненість, довжину та правильний формат.

У разі, якщо дані, передані на сервер, не відповідають встановленим правилам, API автоматично повертає повідомлення про помилки у зручному форматі:

```
{
  "errors": {
    "Email": [ "Invalid email format." ],
    "Password": [ "Password must be at least 6 characters long." ]
  }
}
```

### 3.2.7. Авторизація та аутентифікація

У системі реалізовано механізм аутентифікації та авторизації на основі JWT-токенів (JSON Web Token). Це сучасний стандарт для побудови безпечних RESTful API, який дозволяє уникнути збереження сесій на сервері.

Після успішної реєстрації або входу в систему користувач отримує два токени:

1. **Access Token** — короткоживучий токен, який містить інформацію про користувача у вигляді набору claims.

2. **Refresh Token** — довготривалий токен, що зберігається в базі даних і використовується для оновлення access токена без повторного входу користувача.

Генерація access токена відбувається у методі GenerateAccessToken:

```
var token = new JwtSecurityToken(
    issuer: _config["Jwt:Issuer"],
    audience: _config["Jwt:Audience"],
    claims: claims,
    expires: DateTime.Now.AddDays(30),
    signingCredentials: creds
);
```

### 3.2.8. Прийом даних та оновлення в реальному часі

Дана інформаційна система реалізовує комплексний механізм прийому, обробки та передачі даних від сенсорів, а також миттєве оновлення інтерфейсу користувача в реальному часі за допомогою SignalR.

Оскільки оновлення в реальному часі критично важливе для нашої інформаційної системи, повідомлення про оновлення надсилаються на додаток одразу після того, як Arduino з сенсорами відправляє показники (температура повітря, вологість повітря, вологість грунту, температура грунту, рівень освітленості) через HTTP POST запит на серверний API-ендпоінт. Вхідні дані спочатку потрапляють у сервіс _sensorService, де створюється запис про виміри у базі даних. Виконується зв'язок між серійним номером пристрою Arduino і конкретною теплицею, щоб коректно зберегти дані у відповідній групі. Метод сервісу AddSensorReading:

```
var device = _deviceRepository.Get
(d => d.SerialNumber == dto.DeviceSerialNumber).FirstOrDefault();
if (device == null)
    throw new ArgumentException("Пристрій не знайдено");
```

```
var reading = _mapper.Map<SensorReading>(dto);
reading.GreenhouseId = device.GreenhouseId;
reading.Timestamp = DateTime.UtcNow.ToLocalTime();
_repository.Create(reading);
_repository.Save();
```

Метод AddSensorReading у сервісі також повертає актуальний статус теплиці, який може містити інформацію про загальний стан або можливі попередження.

Після збереження нових даних сервер миттєво повідомляє клієнтів, які підписані на оновлення стану конкретної теплиці, використовуючи SignalR. Клієнти підключаються до SignalR Hub SensorHub і приєднуються до групи, яка відповідає конкретній теплиці. Сервер надсилає повідомлення у відповідну групу через метод SendAsync(), де передає оновлений статус та нові показники сенсорів.

### 3.2.9. Основні маршрути та взаємодія з клієнтами

Для забезпечення ефективної взаємодії між клієнтською частиною додатку та сервером реалізовано перелік основних ендпоінтів Web API, які відповідають принципам REST-архітектури. Ці ендпоінти дозволяють клієнтам, таким як веб-додаток або пристрої Arduino, надсилати та отримувати необхідні дані, що забезпечує надійний та двонаправлений обмін інформацією. Основні методи контролерів та їх призначення винесені в додаток А.

Перелічені кінцеві точки реалізовують повноцінний CRUD-інтерфейс для управління теплицями, а також взаємодію з датчиками та автентифікації користувачів. Система забезпечує валідацію даних, авторизацію, обробку помилок, оновлення даних датчиків в реальному часі, керування пристроями девайсу віддалено та розмежування відповідальностей. Таким чином, API є гнучким, безпечним і розширюваним рішенням, яке відповідає вимогам сучасних вебсервісів для розумного аграрного застосування.

Переглянути виконану роботу можна за посиланням у GitHub [16].

### 3.2.10. Робота з базою даних

Для зберігання та обробки даних інформаційної системи використовується система керування базами даних Microsoft SQL Server. Вона забезпечує надійність, масштабованість та підтримку складних зв’язків між сутностями.

У проекті реалізовано повноцінну взаємодію з базою даних за допомогою ORM-технології Entity Framework Core, яка забезпечує об'єктно-реляційне відображення даних та спрощує реалізацію CRUD-операцій.

Було обрано підхід для написання Code First, при якому структура бази даних формується на основі класів сутностей (entities) у коді. Для більш точного контролю за схемою таблиць використано Fluent API конфігурації, що дозволяє визначати імена таблиць, первинні та зовнішні ключі, обов’язковість полів, максимальні довжини текстових значень, типи видалення тощо.

Основні таблиці бази даних:

- Greenhouses, де зберігається інформація про теплиці;
- SensorReadings – в ній зберігаються дані зчитування показників сенсорів;
- DeviceStates – відповідає за збереження поточних та минулих станів пристроїв у теплицях (двері та вентилятор);
- UserSettings – налаштування, які задає користувач для кожної теплиці;
- Plants – список культур із прикладами назв та оптимальними параметрами клімату для кожної культури;
- RefreshTokens – токени оновлення для авторизації;
- Devices – список пристроїв;
- GreenhouseStatuses – записи про загальний статус теплиць;
- AppUser – користувачі системи.

Для поступового створення та оновлення структури баз даних використовувались міграції. Такий підхід забезпечив історію збереження стану бази даних без втрат.

На рис. 3.3 зображена діаграма бази даних, де можна чітко розпізнати зв’язки між таблицями та логіку їх взаємодій.


*Рис. 3.3. Діаграма бази даних*

### 3.3. Реалізація клієнтської частини

Клієнтська частина системи реалізована у вигляді додатку розробленому за допомогою React Native. Для забезпечення отримання оновлень у реальному часі використовується SignalR, який дозволяє ефективно синхронізувати дані між сервером та клієнтом без необхідності періодичних опитувань.

Для спрощення роботи з підключенням до SignalR створено кастомний React-хук useGreenhouseSignalR. Він відповідає за ініціалізацію з’єднання, підписку на оновлення конкретної групи (теплиці) та передачу отриманих даних до компонентів через callback-функції.

Коли на додаток приходить повідомлення про оновлення даних із сенсорів, кастомний хук useGreenhouseSignalR використовує створений SignalRProvider для встановлення та підтримки з'єднання з серверним SignalR-хабом. Хук підписується на конкретну групу теплиці за її greenhouseId, щоб отримувати лише відповідні повідомлення. При отриманні оновлення через подію GreenhouseStatusUpdated хук викликає передані колбеки для оновлення статусу теплиці та сенсорних даних у користувацькому інтерфейсі.

Переглянути програмну реалізацію проекту можна за посиланням у GitHub [17].

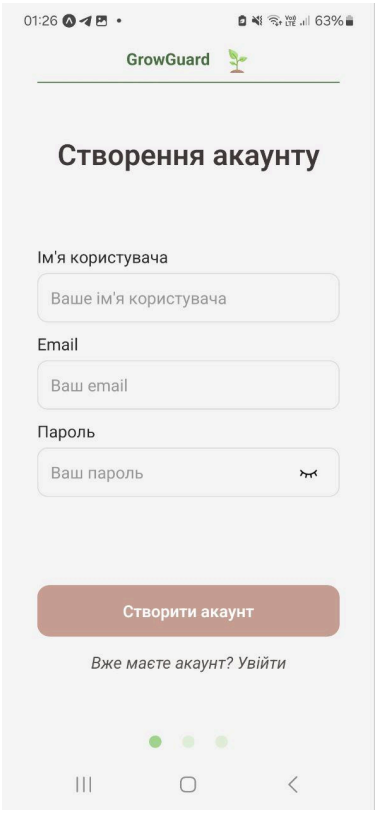
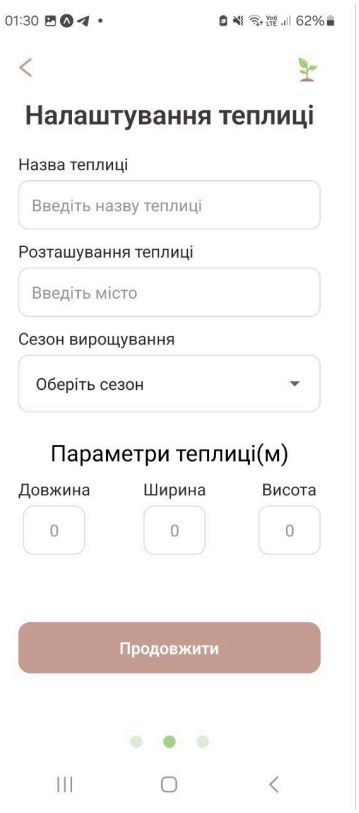
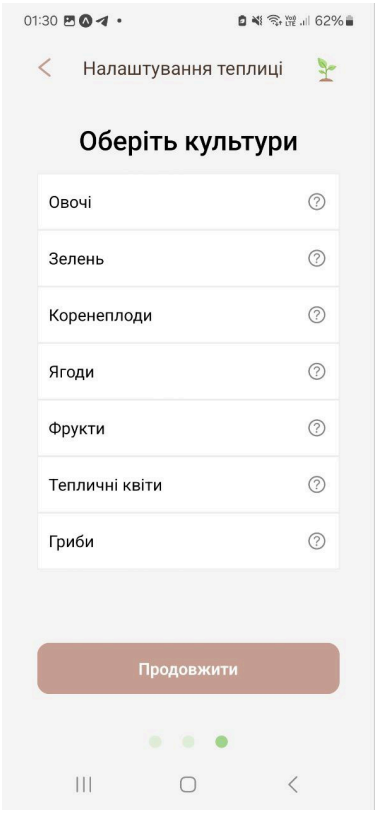
### 3.4. Інструкція користувача

У цьому розділі представлено інструкцію з користування розробленим мобільним додатком, який слугує основним інструментом взаємодії користувача з системою «Розумна агротеплиця». Додаток забезпечує зручний та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачу переглядати стан теплиці, налаштовувати параметри, контролювати підключені пристрої, а також отримувати актуальну інформацію про мікрокліматичні умови всередині теплиці.

Після першого запуску додатку користувача зустрічає форма реєстрації (табл. 3.2). Після успішної реєстрації система одразу пропонує створити першу теплицю, заповнивши форму з основними параметрами. Ці параметри можна буде згодом змінити. Усі форми в додатку оснащені валідацією введених даних, а також містять підказки, які допомагають користувачу пройти процес реєстрації, створення та редагування об'єктів без труднощів.

*Таблиця 3.2*

*Форма реєстрації та створення теплиці*

| **Форма реєстрації** | **Форма створення теплиці** | **Форма вибору культур** |
|---|---|---|
|  |  |  |

Після завершення початкового налаштування користувач переходить на головну сторінку додатку, де відображаються всі створені ним теплиці. Спочатку, після реєстрації, виводиться лише одна теплиця. Ліворуч від її назви розташований індикатор поточного стану. Внизу екрана знаходиться кнопка для додавання нової теплиці. Інтерфейс головної сторінки продуманий таким чином, щоб користувач легко орієнтувався в ньому (рис. 3.4).

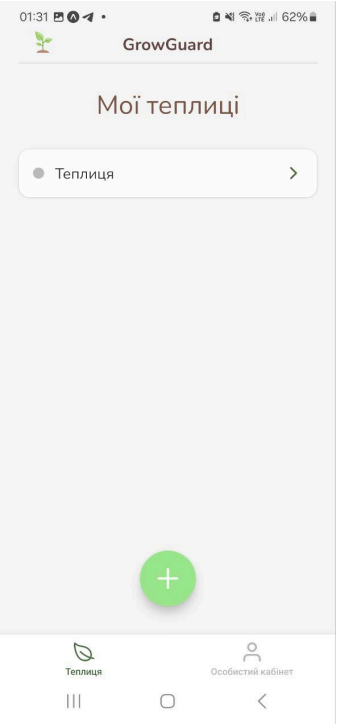



*Рис. 3.4. Головна сторінка додатку*

При переході до конкретної теплиці відкривається сторінка з детальною інформацією. Вона умовно поділена на дві частини.

У верхній частині розміщено великий індикатор стану теплиці, підпис до нього та інформацію про підключення до пристрою.

Теплиця може перебувати в чотирьох основних станах: “Все добре”, “Щось не так…”, “Критична помилка”, “Немає даних” (рис. 3.5).


*Рис. 3.5. Основні стани теплиці*

Нижня частина є основною інформаційною зоною та містить три ключові розділи навігації: Показники, Норми та Керування (рис. 3.6).

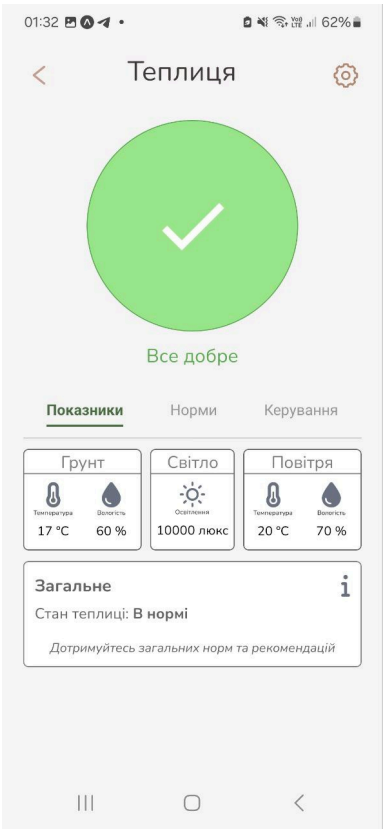



*Рис. 3.6. Екран детальної інформації теплиці*

Показники – відображають актуальні дані з сенсорів теплиці в режимі реального часу. Інформація оновлюється щогодини, що забезпечує безперервний моніторинг. Дані згруповано за категоріями (грунт, повітря, освітлення) та доповнено іконками для зручності сприйняття. Окремий блок надає попередження та поради щодо підтримки оптимальних умов у теплиці.

Норми – розділ із двома вкладками, що містять перелік норм та рекомендацій, сформованих системою на основі заданих параметрів та обраних культур (рис. 3.7).

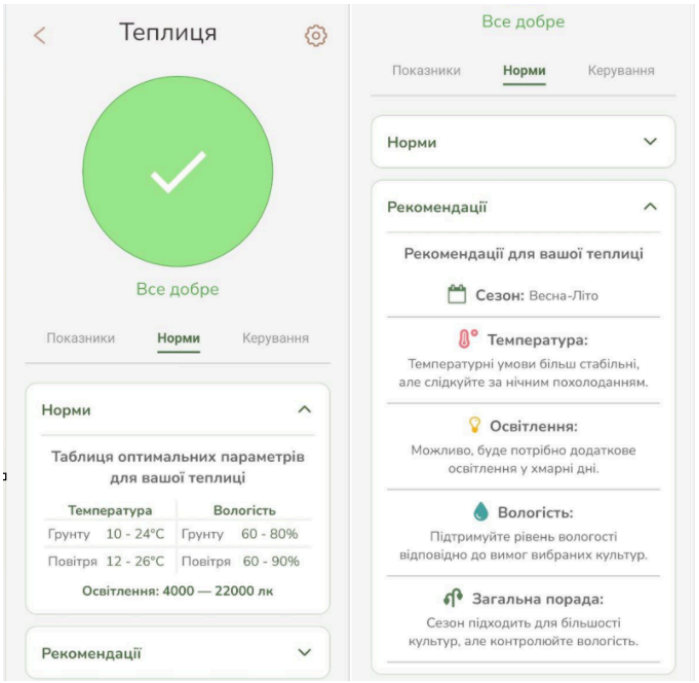



*Рис. 3.7. Екран додатку "Норми"*

Третя і остання секція – це блок керування пристроями. Він містить дві кнопки, які відповідають певним діям: увімкнути/вимкнути вентилятор, відкрити/закрити двері(рис. 3.8)

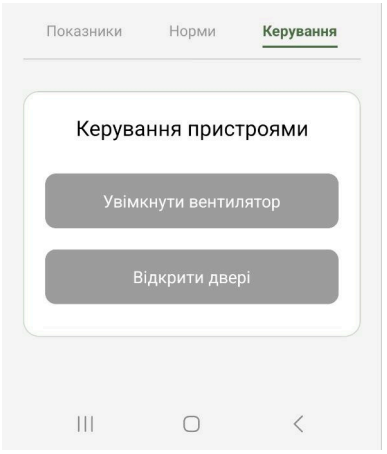



*Рис. 3.8. Блок керування пристроями*

У верхньому правому куті сторінки розташована іконка, яка відкриває меню для детальної роботи з обраною теплицею (рис. 3.9). Меню містить такі пункти:

- **Редагувати теплицю** – відкриває сторінку редагування, де користувач може змінити параметри теплиці (назву, розміри, обрані культури), а також налаштування оптимальних умов для вирощування.
- **Переглянути інформацію** – надає доступ до розширеної інформації про теплицю, включаючи її поточний стан, встановлені норми та всі введені параметри.
- **Переглянути аналітику теплиці** – відкриває сторінку з графіками, які відображають зміну показників датчиків за попередній день. Усього виводиться п’ять графіків. Кожен графік містить: зелену лінію з крапками, що відображає реальні значення сенсорних даних; дві червоні лінії, що показують верхню та нижню межі допустимих значень.
- **Видалити теплицю** – дозволяє повністю видалити теплицю разом з усіма пов’язаними даними. Перед видаленням система обов’язково запитує підтвердження дії.

- **Підключити до пристрою** – активує з’єднання теплиці з пристроєм Arduino. Після успішного підключення ця кнопка стає неактивною, що запобігає повторній ініціалізації.

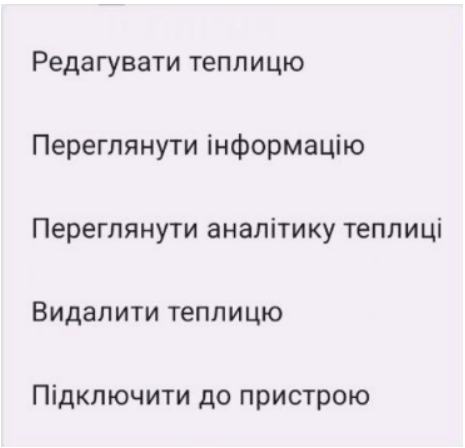



*Рис. 3.9. Меню дій із теплицею*

Додаток має дві основні вкладки (таби): перша – для роботи з теплицями, друга – для доступу до особистого кабінету користувача.

Особистий кабінет користувача містить блок із базовою персональною інформацією та списком усіх теплиць, створених користувачем (рис. 3.10). У цьому розділі також реалізовано можливість редагування особистих даних, включно зі зміною пароля, що забезпечує зручність у керуванні профілем та підвищує рівень безпеки.

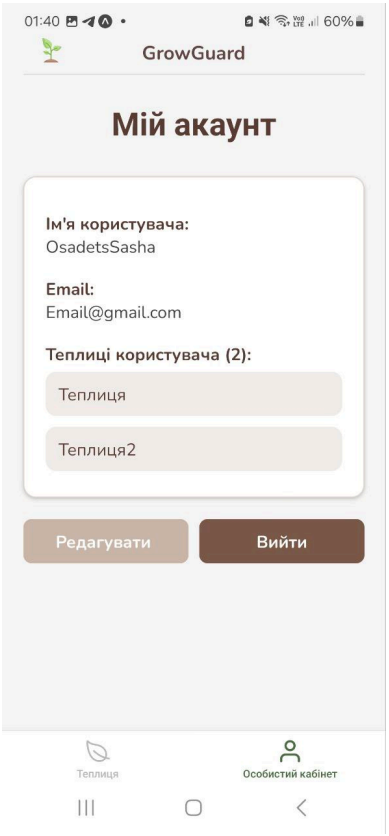



*Рис. 3.10. Особистий кабінет користувача*

Дизайн додатку виконано в м’яких, гармонійних тонах, що не перевантажують зір і створюють приємне візуальне середовище для користувача. Усі кольори інтерфейсу ретельно підібрані, щоб доповнювати одне одного та підтримувати візуальний баланс. Сучасний стиль оформлення та інтуїтивно зрозуміле розміщення елементів сприяють комфортному користуванню додатком навіть без попереднього ознайомлення. Такий підхід до дизайну робить взаємодію з системою простою, естетично привабливою та ефективною.

# ВИСНОВОК

У межах кваліфікаційної роботи було розроблено інформаційну систему «Смарт-агротеплиця», що забезпечує моніторинг та управління мікрокліматичними параметрами теплиці з використанням сучасних технологій Інтернету речей. Рішення поєднує апаратну частину на базі мікроконтролера Arduino, серверну частину (ASP.NET Web API) для обробки та збереження даних, а також мобільний клієнт (React Native), що надає користувачу зручний інтерфейс для взаємодії з системою.

У процесі розробки було проведено аналіз існуючих рішень у сфері автоматизації тепличного господарства, як в Україні так і світі. Було визначено як їхні сильні сторони – зокрема розширені функції моніторингу, – так і суттєві недоліки: висока вартість, складність у впровадженні, відсутність гнучкості для адаптації під потреби невеликих господарств. Ці результати стали ключовими для формування вимог розроблюваної системи. Крім того, було проведено дослідження кліматичних умов, необхідних для вирощування різних культур у тепличному середовищі. Це дозволило врахувати оптимальні параметри температури, вологості та освітлення, які використовуються системою як нормативні значення при моніторингу стану теплиці та формуванні рекомендацій для користувача.

У процесі розробки було реалізовано низку функціональних можливостей: авторизація та автентифікація користувачів, створення, редагування та перегляд теплиць, збір даних з датчиків, керування виконавчими пристроями, генерація оптимальних параметрів теплиці, генерація рекомендацій теплиці, створення графіків для реалізації звітування. Також було реалізовано оновлення даних, отриманих з сенсорів, в реальному часі. Окрема увага приділялась розробці інтуїтивно зрозумілого користувацького інтерфейсу.

Система була протестована в умовах, наближених до реальних, що підтвердило її працездатність, зручність використання та відповідність поставленим вимогам. Проведена апробація на науково-практичній конференції підтвердила актуальність теми та практичну цінність отриманих результатів.

Запропоноване рішення для реалізації сучасної смарт-агротеплиці у невеликих господарствах є оригінальним в поєднанні простих у реалізації апаратних засобів із сучасними програмними технологіями. Воно має високий потенціал для застосування в малому аграрному бізнесі си в індивідуальних тепличних господарствах. Система може слугувати основою для подальших досліджень у сфері смарт-агротехнологій, а також можу бути масштабованою для автоматизації більших процесів у сільському господарстві.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

**Електронні ресурси віддаленого доступу**


1. Нікончук Н.В., Ткачова Є.С., Дробітько А.В., Кузьома В.В., Біліченко О.С. Навчальний посібник Біолого-екологічні особливості овочевих культур. Миколаїв, МНАУ, 2020. 409 с. URL:https://dspace.mnau.edu.ua/jspui/bitstream/123456789/8376/1/%D0%91%D1%96%D0%BE%D0%BB%D0%BE%D0%B3%D0%BE-%D0%B5%D0%BA%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%87%D0%BD%D1%96%20%D0%BE%D1%81%D0%BE%D0%B1%D0%BB%D0%B8%D0%B2%D0%BE%D1%81%D1%82%D1%96%20%D0%BE%D0%B2%D0%BE%D1%87%D0%B5%D0%B2%D0%B8%D1%85%20%D0%BA%D1%83%D0%BB%D1%8C%D1%82%D1%83%D1%80.pdf (дата звернення: 05.05.2025).
2. Зінченко О.І., Салатенко В.Н., Білоножко М.А. Рослинництво: Підручник. Київ: Аграрна освіта, 2001. 591 с. URL: https://buklib.net/books/27142/ (дата звернення: 05.05.2025).
3. Наслідки зміни клімату — Центр екологічних ініціатив “Екодія”, 2018. URL: https://ecoaction.org.ua/diyalnist/klim-naslidky (дата звернення: 05.05.2025).
4. Центр екологічних ініціатив «Екодія». URL:https://ecoaction.org.ua/pro-nas (дата звернення: 05.05.2025).
5. Зміни клімату в Україні та світі: причини, наслідки та рішення протидії. Центр екологічних ініціатив «Екодія», 2020. URL: https://ecoaction.org.ua/zmina-klimatu-ua-ta-svit.html (дата звернення: 05.05.2025).
6. Дідур І., Пелех Л. Посібник «Точне землеробство».Вінниця: ВНАУ. 2023. 44 с. URL: https://socrates.vsau.org/b04213/html/cards/getfile.php/35029.pdf (дата звернення: 05.05.2025).

7. Шевніков М.Я. Світові агротехнології. Полтава, 2005. 179 с. URL:http://www.tsatu.edu.ua/rosl/wp-content/uploads/sites/20/lekcija-1.naukovi-osnovy-suchasnyh-ahrotehnolohij.pdf (дата звернення: 05.05.2025).
8. Панічев Р. Стаття: Смартрішення крокують агросвітом. IFarming, 2022. URL:https://ifarming.ua/itehnologii/selektsiya/smartrishennya-krokuyut-agrosvitom (дата звернення: 05.05.2025).
9. Остапчук І. Житомирська смарт-теплиця. Суспільне Житомир, 2020. URL:https://suspilne.media/zhytomyr/27756-u-zitomiri-zavilasa-smart-teplica-z-informacijnimi-tehnologiami/ (дата звернення: 06.05.2025).
10. Гапонов К. Стаття: На Миколаївщині фермер створив унікальну «розумну теплицю». 2025. URL: https://blik.ua/techno/3567-fermer-z-mikolayivshini-perevernuv-agrosvit-teplici-teper-mozhna-kontrolyuvati-zi-smartfona (дата звернення: 06.05.2025).
11. Розпорядження КМУ Про схвалення Стратегії розвитку сільського господарства та сільських територій в Україні на період до 2030 року. 2024 № 1163-р Київ.
URL: https://zakon.rada.gov.ua/laws/show/1163-2024-%D1%80#Text (дата звернення: 06.05.2025).
12. Мазур В.А., Поліщук І.С., Телекало Н.В., Мордванюк М.О. РОСЛИННИЦТВО. Навчальний посібник. Вінниця: Видавництво ТОВ «Друк». 2020. 352 с. URL: http://repository.vsau.org/getfile.php/27415.pdf (дата звернення: 08.05.2025)
13. Ковальов М.М. Вплив параметрів кліматозабезпечення на вирощування мікрозелені в умовах плівкової теплиці. Таврійський науковий вісник, 2022, 10 с. URL:https://tnv-agro.ksauniv.ks.ua/archives/126_2022/21.pdf (дата звернення: 08.05.2025)

14. Гіль Л.С., Пашковський А.І., Суліма Л.Т. Сучасні технології овочівництва закритого і відкритого грунту. Частина перша. Посібник Вінниця “Нова Книга” 2008. с 47-67. URL:https://moodle.osau.edu.ua/pluginfile.php/472/course/summary/%D0%A1%D1%83%D1%87%D0%B0%D1%81%D0%BD%D0%B0-%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F-%D0%BE%D0%B2%D0%BE%D1%87%D1%96%D0%B2%D0%BD%D0%B8%D1%86%D1%82%D0%B2%D0%B0-1%D1%87..pdf(дата звернення: 08.05.2025)
15. Коркін Ф. Які освітлення і температура мають бути в теплиці для гарного урожаю. РБК-Україна, 2024 URL:https://www.rbc.ua/rus/styler/ki-osvitlennya-i-temperatura-mayut-buti-teplitsi-1733819423.html (дата звернення: 08.05.2025)
16. GitHub - 0sadets/SmartGreenHouse_Server. GitHub. URL: https://github.com/0sadets/SmartGreenHouse_Server.git (дата звернення: 01.06.2025)
17. GitHub -0sadets/GrowGuard. GitHub. URL: https://github.com/0sadets/GrowGuard.git (дата звернення: 01.06.2025)

# ДОДАТКИ

Додаток А

*Ендпоінти системи та сценарії їх використання*

| № | Метод (HTTP) | Маршрут | Призначення |
|---|---|---|---|
| 1 | POST | /api/greenhouse/create | Створення нової теплиці з урахуванням оптимальних параметрів |
| 2 | GET | /api/greenhouse/{greenhouseId} | Отримання детальної інформації про теплицю за її ідентифікатором |
| 3 | PUT | /api/greenhouse/{greenhouseId} | Оновлення параметрів теплиці |
| 4 | DELETE | /api/greenhouse/{id} | Видалення теплиці та пов’язаних з нею сутностей |
| 5 | GET | /api/greenhouse/{id}/recommendation | Отримання рекомендацій по параметрах для конкретної теплиці |
| 6 | POST | /api/greenhouse/status | Збереження нового статусу теплиці та його оцінка |
| 7 | POST | api/sensor/readings | Додати показники сенсорів, надіслати статус у SignalR-групу |
| 8 | GET | api/sensor/latest/{greenhouseId} | Отримати останні сенсорні дані по теплиці |
| 9 | POST | api/auth/register | Реєстрація нового користувача |
| 10 | POST | api/auth/login | Вхід у систему |
| 11 | POST | api/auth/logout | Вийти з системи |