НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Комп’ютерних наук

_____ Голуб Б.Л.

“___” _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

Інформаційна система моніторингу показників мікроклімату теплиці

Спеціальність 122 – «Комп’ютерні науки»

Гарант освітньої програми

Д.е.н., професор _____ Руденський Р.А.

Керівник бакалаврської кваліфікаційної роботи

Кандидат педагогічних наук, доцент ____ Касаткін Д.Ю.

Виконав _____ Марченко М.О.

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

Комп'ютерних наук

_____ Голуб Б.Л.

"___" _____ 2025 р.

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту

Марченко Михайло Олексійович

Спеціальність 122 – «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи Інформаційна система моніторингу показників мікроклімату теплиці

затверджена наказом ректора НУБіП України від 24.04.2025р. №684с

Термін подання завершеної роботи на кафедру _____

Вихідні дані до бакалаврської кваліфікаційної роботи: _____

Перелік питань, які потрібно розробити:_____

Перелік графічних документів (за потреби)

_____

Дата видачі завдання "_____"_____20___ р.

Керівник бакалаврської кваліфікаційної роботи

Кандидат педагогічних наук, доцент _____ Касаткін Д.Ю.

Завдання прийняв до виконання _____ Марченко М.О.

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів виконання бакалаврської кваліфікаційної роботи	**Строк виконання етапів бакалаврської кваліфікаційної роботи**	**Примітка**
	Формулювання теми та цілей кваліфікаційної роботи	23.03.2025 – 27.03.2025	
	Аналіз предметної області та аналогічних систем	27.03.2025 – 01.04.2025	
	Проектування архітектури інформаційної системи	01.04.2025 – 07.04.2025	
	Розробка прикладогое програмного та апаратного забезпечення	07.04.2025 – 25.04.2025	
	Розробка інтерфейсу користувача (веб-частина)	25.04.2025 – 09.05.2025	
	Розробка рекомендацій щодо впровадження та експлуатації системи	09.05.2025 – 16.05.2025	
	Тестування, перевірка працездатності системи	16.05.2025 – 18.05.2025	
	Написання пояснювальної записки	18.05.2025 – 23.05.2025	

Студент _____ _Марченко М.О.

Керівник бакалаврської

кваліфікаційної роботи _____Касаткін Д.Ю.

ЗМІСТ

ВСТУП

Овочі є важливим джерелом вітамінів, а також мікро- і макроелементів, необхідних для нормального розвитку людського організму. Крім того, вони мають соціальне значення, оскільки часто слугують більш доступною альтернативою дорогим продуктам харчування для малозабезпечених верств населення.

Кліматичні умови України сприяють вирощуванню найпоширеніших овочевих культур у відкритому ґрунті впродовж 4–5 місяців – із кінця травня до першої половини осені. Проте попит на свіжі овочі існує протягом усього року. Якщо такі культури, як морква, капуста чи цибуля можна ефективно зберігати тривалий час, то помідори, огірки, перець і зелень потребують вирощування у тепличних умовах.

Згідно з офіційною статистикою, в Україні нараховується близько 2,5 тис. гектарів закритого ґрунту, призначених для овочівництва. Переважна частина цієї площі (приблизно 93%) використовується для вирощування огірків і томатів: огірки займають близько 1,4 тис. га, а томати – близько 1 тис. га.

Для отримання високих урожаїв у тепличних умовах необхідно чітко дотримуватись технологічних норм вирощування з мінімальними витратами. Основними чинниками, що забезпечують максимальну врожайність, є підтримання оптимальних температурних і вологісних режимів повітря та ґрунту під час вегетації, своєчасний посів і висаджування розсади, а також правильне внесення добрив.

Метою даної роботи є розробка онлайн-системи моніторингу, яка забезпечуватиме контроль параметрів мікроклімату в теплиці, зіставлення отриманих даних із нормативними значеннями, а також можливість автоматичного керування виконавчими елементами системи.

Сучасні агровиробники, які спеціалізуються на вирощуванні рослин, мають потребу в інформаційно-моніторингових системах для контролю

ключових параметрів, що впливають на розвиток рослин. Такі системи дають змогу оперативно реагувати на зміни температури, вологості, рівня освітлення, вмісту вуглекислого газу тощо, під час усього процесу вирощування. Крім того, вони дозволяють не лише виявляти відхилення від оптимальних умов, але й здійснювати відповідні дії, формувати моделі ідеальних мікрокліматичних умов для різних культур, а також приймати обґрунтовані рішення щодо внесення добрив та інших агротехнічних заходів.

Сучасні технології відкривають широкі можливості для створення подібних систем за допомогою платформи Arduino, що дозволяє інтегрувати різноманітні датчики, модулі бездротової передачі даних та інші компоненти, створюючи цілісну систему Інтернету речей (IoT).

Одним з важливих аспектів ефективності таких систем є зручне відображення моніторингових даних для кінцевого користувача. Веб-технології на сьогодні є одним із найефективніших інструментів для забезпечення доступу до даних у будь-який час, з будь-якого пристрою – комп’ютера, планшета чи смартфона – підключеного до Інтернету.

На нашу думку, веб-інтерфейс для моніторингу параметрів є найбільш зручним і ефективним способом взаємодії користувача із системою.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Завданням даної роботи є створення інформаційно-керуючої системи онлайн-моніторингу мікроклімату в тепличному середовищі. Розроблена система має автоматизувати процес збору даних і забезпечити можливість впливу на виконавчі механізми для коригування параметрів мікроклімату.

Ключові особливості запропонованої системи:

- Побудована за принципом клієнт-серверної архітектури з використанням веб-клієнта;
- Датчики, інтегровані з мікроконтролером Arduino, програмно налаштовуються на зчитування даних з певною періодичністю;
- Отримані значення зчитуються через COM-порт і обробляються за допомогою скриптів, написаних мовою програмування PHP, після чого дані у режимі реального часу зберігаються в базі даних SQL-сервера;
- Веб-додаток, реалізований на PHP, дозволяє візуалізувати показники в реальному часі, будувати графіки та виявляти відхилення від нормативних значень.

1.1 Постановка задачі

Головною метою є створення системи моніторингу мікроклімату теплиць, що є актуальною потребою сучасного тепличного господарства. Умови зростаючої конкуренції та прагнення до економічної ефективності зумовлюють необхідність оптимізації процесів у фермерських господарствах, агрокомпаніях та інших аграрних підприємствах. Впровадження подібної системи сприяє підвищенню прибутковості шляхом автоматизації контролю за умовами вирощування.

Поняття «розумна теплиця» набуло широкого поширення. Наприклад, компанія EPAM сформувала перелік основних функцій, які має включати така

система: контроль освітлення та зрошення, моніторинг температури й вологості, віддалене керування, автоматичне фотографування стану рослин, а також системи захисту від перебоїв в електропостачанні. Як апаратну платформу було обрано Raspberry Pi — компактний і функціональний мікрокомп’ютер, що сумісний з різними операційними системами та дозволяє підключення великої кількості датчиків. Спеціалісти EPAM навіть створили власний дистрибутив Linux для забезпечення повноцінної роботи проєкту, що дає змогу будь-кому встановити програмне забезпечення, підключити датчики та автоматизувати свою теплицю.

Використання системи моніторингу дозволяє аналізувати, як саме такі фактори, як температура та вологість, впливають на ріст рослин і врожайність. Запропонована система також спрощує роботу операторів, які слідкують за умовами мікроклімату, надаючи візуалізацію залежностей у вигляді графіків у режимі реального часу.

Для структурованого представлення функціональних можливостей та взаємозв’язків компонентів системи було застосовано метод CRC-карток (Class-Responsibility-Collaboration). Цей метод, запропонований Кентом Беком, використовується для виявлення класів, їхніх відповідальностей і взаємодій на ранніх етапах об'єктно-орієнтованого проєктування. У межах цієї роботи CRC-картки представлені в таблицях 1–4.

Таблиця №1

CRC картка №1

Проблема	Оперативне реагування на зміну мікроклімату
Відноситься до	Працівник
Її наслідком являється	Зменшення урожайності
Вирішення проблеми	Автоматизація процесу

Таблиця №2

CRC картка №2

Проблема	Управління системою віддалено
Відноситься до	Працівник
Її наслідком являється	Низька оперативність прийняття рішень
Вирішення проблеми	Веб-додаток системи моніторингу

Таблиця №3

CRC картка №3

Проблема	Управління системою у режимі реального часу
Відноситься до	Працівник
Її наслідком являється	Низька оперативність прийняття рішень
Вирішення проблеми	Передача даних на сервер веб-додатку у режимі реального часу

Таблиця №4

CRC картка №4

Проблема	Відхилення приросту продукції від нормативного показника
Відноситься до	Працівник
Її наслідком являється	Зменшення урожайності
Вирішення проблеми	Програмний модуль для опрацювання математичної моделі залежності приросту від параметрів мікроклімату

Опис користувачів системи

У системі передбачено декілька типів користувачів, кожен із яких має визначені повноваження та функціональні обов’язки.

1. Адміністратор це користувач, який відповідає за загальне управління

веб-ресурсом і базою даних. Адміністратор має повний доступ до системи, включаючи можливість створення, редагування та видалення облікових записів інших користувачів, а також надання або обмеження прав доступу.

2. Працівник це зареєстрований користувач, який має персональний профіль у системі. Працівник має доступ до всієї інформації, що стосується параметрів мікроклімату в теплиці. Він може переглядати дані в режимі реального часу, аналізувати інформацію, будувати графіки та приймати рішення щодо корекції умов вирощування.

Профілі користувачів

Нижче подано опис функціональних ролей кожного типу користувачів у вигляді профілів. Детальна інформація представлена в таблицях 5 та 6, де наведено характеристики облікових записів **Адміністратора** та **Працівника** відповідно.

Таблиця.5

Представник	Адміністратор
Опис	Додавання і видалення працівників до бази, редагування БД та контенту сайту
Тип	Користувач
Відповідальність	Коректне додавання нових матеріалів
Критерій успіху	Успішна робота

Таблиця.6

Представник	Працівник
Опис	Перегляд веб-додатку
Тип	Користувач
Відповідальність	Оперативне прийняття рішень
Критерій успіху	Максимальна урожайність

1.2 Огляд інформаційних джерел та існуючих рішень

Хоча венчурні інвестори визнають перспективність агротехнологій, активного фінансування цієї сфери поки що не відбувається. Водночас деякі українські аграрні корпорації вже інтегрують у свій бізнес рішення, створені місцевими IT-компаніями. AgTech — відносно новий, але неоднозначний напрям для України, що поступово розвивається попри обережність інвесторів.

Протягом тривалого часу аграрна галузь залишалася поза увагою IT-підприємців та венчурного капіталу. Основними бар'єрами для розвитку були консервативність власників агробізнесу та слабка інтеграція базових цифрових технологій у виробничі процеси. Однак ситуація почала змінюватися з 2014 року — на фоні перших успішних реалізованих проєктів та зростання інтересу з боку інноваційних гравців ринку.

Наприклад, лише за рік з моменту запуску стартапу Drone.ua компанія стала провідним постачальником аеророзвідки для агрохолдингу Kernel, одного з лідерів агросектору України. Вже з 2015 року Kernel застосовує безпілотні літальні апарати (п'ять квадрокоптерів і два дрони), отримуючи за допомогою IT-рішень команди Pixel Solutions важливі агродані — зокрема, індекси вегетації, інформацію про випаровування, рівень азоту та прогнози врожайності.

Інший приклад — стартап Petiole, заснований у 2014 році. Це «розумний» трекер росту рослин, який у 2015 році увійшов до списку 20 найгарячіших стартапів за версією американського телеканалу CNBC.

Попри те, що кількість «зіркових» AgTech-проєктів наразі поступається популярним нішам IT-індустрії, інтерес з боку підприємців у цій сфері стабільно зростає. Це пов'язано з тим, що агропромисловий комплекс становить вагому частину української економіки.

«Ми зосереджені на аграрному секторі, адже він стрімко розвивається і відкритий до впровадження технологій. Цей шанс не можна проґавити», — коментує засновник компанії Envi Logic Андрій Яремич. Його компанія створила Envy Sensor — систему моніторингу кліматичних умов і рівня вуглекислого газу

в теплицях та овочесховищах. Рішення базується на використанні графенового датчика, здатного фіксувати надзвичайно низькі концентрації CO_2.

Загалом українські IT-фахівці розглядають AgTech як експериментальний майданчик для тестування інновацій. Попри обіцянки великого потенціалу, інвестори поки що вичікують, намагаючись з'ясувати, наскільки вітчизняні агрохолдинги та фермерські господарства дійсно зацікавлені у впровадженні подібних технологій.

В цілому, аграрний сектор України нині переживає непрості часи. Незважаючи на стабільні обсяги експорту, ціни на аграрну продукцію продовжують знижуватися вже третій рік поспіль. Галузь стикається з численними викликами, що відкриває вікно можливостей для технологічних стартапів. Менш ефективні підприємства змушені залишати ринок, тоді як інші переходять до інтенсивного виробництва та точного землеробства за зразком розвинених країн.

Хоча аграрна та IT-галузі мають значний потенціал, наразі їхня взаємодія залишається слабкою. Аграрії стикаються з великою кількістю проблем, які можна ефективно вирішити за допомогою цифрових рішень. Проте часто IT-фахівці не до кінця розуміють специфіку потреб агросектору, а представники аграрного ринку не обізнані з сучасними технологіями та можливостями їх застосування.

Одним із прикладів вдалого поєднання технологій та агровиробництва є система автоматизації Genesis. Вона здійснює моніторинг і автоматичне керування такими параметрами:

- температура ґрунту;
- температура повітря;
- вологість ґрунту;
- вологість повітря;
- рівень освітлення;
- показники pH та електропровідності (EC) поливної води;

- ключові параметри роботи котельного обладнання та інші.

Система «Genesis» самостійно аналізує умови та керує виконавчими механізмами (насосами, клапанами, приводами, датчиками тощо) для підтримання оптимальних умов у автоматичному режимі. Такий підхід дозволяє знизити залежність від людського фактору та мінімізувати ризики, пов’язані з помилками персоналу. Автоматизація забезпечує стабільність процесів і підвищує ефективність сільськогосподарського виробництва.

1.3 Моделювання предметної області

1.3.1 Методології системного аналізу.

Системна методологія є найбільш структурованим і надійним підходом до управління складними, взаємопов’язаними процесами. Вона дає змогу детально досліджувати окремі компоненти системи, а також послідовно інтегрувати їх у єдину цілісну структуру. У межах системного підходу організація розглядається як система, кожен елемент якої має власні, чітко визначені та обмежені цілі.

Багато дослідників трактують суть системного підходу через такі ключові принципи:

1. Попереднє формулювання цілей і визначення їх ієрархії перед початком будь-яких управлінських дій, особливо перед прийняттям рішень;
2. Оптимізація шляхів досягнення цілей за допомогою порівняльного аналізу альтернативних варіантів і вибору найбільш ефективного при мінімальних витратах ресурсів;
3. Кількісна оцінка (квантифікація) цілей, методів та ресурсів на основі комплексного аналізу всіх потенційних результатів діяльності, а не лише обмежених показників.

Системний підхід — це напрям наукової методології, що базується на розгляді складних об’єктів як сукупності взаємопов’язаних елементів, між якими існують певні відносини та взаємозв’язки.

Система — це цілісне утворення, що складається з елементів, які взаємодіють між собою та разом формують нові властивості, не притаманні окремим її частинам. Залежно від типу соціальної системи, розрізняють три основні напрями менеджменту:

1. Соціально-політичний менеджмент (адміністративне управління);
2. Соціально-економічний менеджмент (управління у виробничій сфері);
3. Соціально-культурний менеджмент (управління у невиробничій сфері).

У межах системного підходу особливу вагу почали набувати дослідження, що отримали назву системного аналізу. Водночас деякі автори вважають за доцільне розрізняти системний підхід і системний аналіз. Їхня аргументація полягає в тому, що методологія системного аналізу, на відміну від загального системного підходу, обов’язково спирається на математичні моделі та подає результати у формалізованій, кількісній формі. Натомість системний підхід базується на ширших, не завжди формалізованих концепціях.

Системний аналіз виконує функцію інструменту для структуризації, аналізу та впорядкування складних проблем, і може застосовуватися як із залученням математичного апарату й обчислювальної техніки, так і без них. У цьому значенні поняття системного аналізу часто ототожнюється з термінами "системний підхід" або "системні дослідження", як це прийнято у деяких американських підходах. Такий аналіз ефективно застосовується для вирішення соціальних задач і розглядається як інструмент наукового підходу до пошуку оптимальних рішень з метою досягнення максимального результату.

Поняття системи дії описує складну структуру взаємопов’язаних дій та відносин між ними. Цей термін був введений Талкоттом Парсонсом у 1937 році в праці "Структура соціальної дії", разом із поняттям одиничної дії. За Парсонсом, система дії — це сукупність одиничних дій, поєднаних між собою через певну логіку взаємодії, яка формує своєрідну "координатну сітку" дії.

У системі дії можна виділити два ключові рівні відносин:

1. Взаємозв’язки між діями, згрупованими в більші організаційні одиниці — індивіди;
2. Відносини між індивідами як членами соціальних груп.

Крім того, координація дій передбачає чітке розрізнення між такими компонентами одиничної дії:

1. Ціллю;
2. Засобами досягнення цієї цілі;
3. Умовами, за яких реалізується дія.

1.3.2. Структурно-функціональне моделювання

Методологія SADT (Structured Analysis and Design Technique) є однією з найвідоміших концепцій для аналізу та проєктування систем. Її відмінність полягає в унікальній здатності відображати такі важливі характеристики, як керування, зворотний зв’язок і використання ресурсів. На відміну від багатьох інших структурних методів, які зосереджуються переважно на розробці програмного забезпечення, SADT створювалася як універсальна мова для опису функціонування систем будь-якого типу.

За допомогою методології SADT вирішуються основні завдання, незалежно від типу системи:

- аналіз функцій, які виконує система;
- формування специфікацій вимог і функціональностей майбутньої системи;
- проєктування загальної архітектури системи.

Упродовж перших десяти років SADT залишалась переважно «паперовою» методикою. Проте з появою персональних комп’ютерів з графічними можливостями у середині 1980-х років, методологія була адаптована для комп’ютерного використання. Одним із перших програмних інструментів, що реалізував структурно-функціональний аналіз за SADT, став пакет AUTOIDEF,

розроблений у межах програми ВПС США для створення інтегрованої автоматизованої системи управління виробництвом.

IDEF0 (Function Modeling) — це методологія функціонального моделювання та графічного представлення процесів, що використовується для формалізації та опису бізнес-процесів. Основною особливістю IDEF0 є акцент на ієрархічній структурі об'єктів, що спрощує розуміння предметної області. Метод зосереджується не на хронологічній послідовності дій (як у WorkFlow), а на логічних взаємозв’язках між функціями та сигналах управління, які визначають порядок і умови виконання робіт.

Цей підхід широко застосовується для моделювання як організаційних, так і адміністративних процесів, і вважається однією з найбільш сучасних та ефективних моделей управління бізнес-проєктами.

У результаті цих потреб і з’явилася методологія IDEF0. Від часу свого створення вона зазнала лише незначних змін, здебільшого рекомендаційного або обмежувального характеру. Остання офіційна редакція стандарту була опублікована в грудні 1993 року Національним інститутом стандартів і технологій США (NIST).

На рисунку 1.1 подано приклад FDD-діаграми, яка ілюструє процес моніторингу мікрокліматичних параметрів у теплиці.

SADT діаграма інформаційної системи аналізу продажу товарів відображає поєтапне використання ІС з вхідними/вихідними та керуючими даними на кожному з етапів роботи, діаграму приведено на рис.1.2.

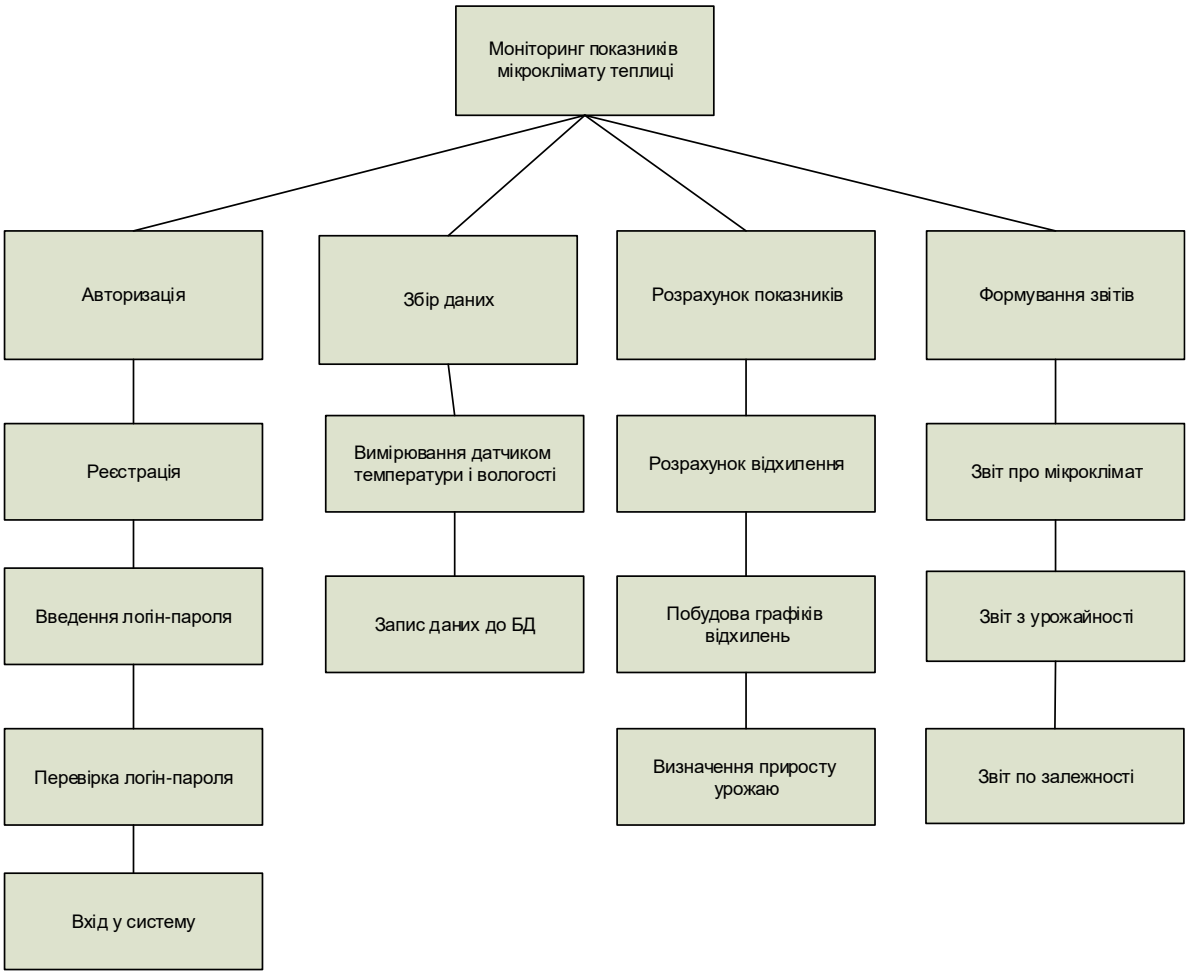


Рис.1.1 FDD діаграма «Моніторинг показників мікроклімату теплиці»

Рис. 1.2 Контекстна діаграма SADT «Моніторинг показників мікроклімату теплиці»

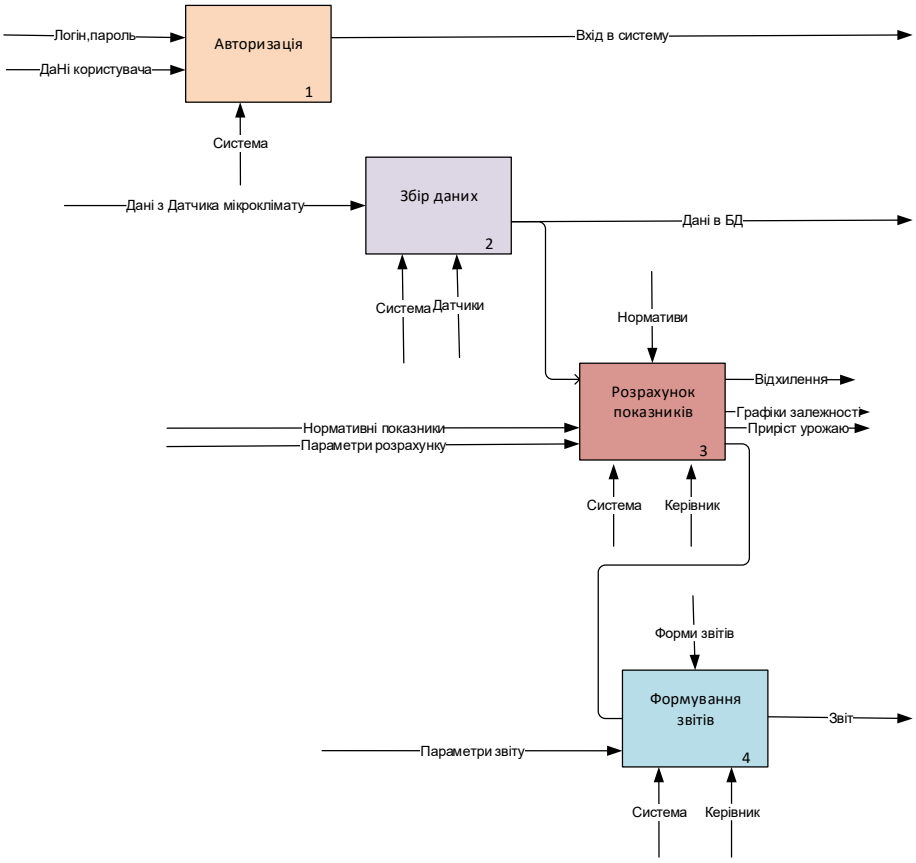


Рис. 1.3 Діаграма SADT 0 рівня «Моніторинг показників мікроклімату теплиці»

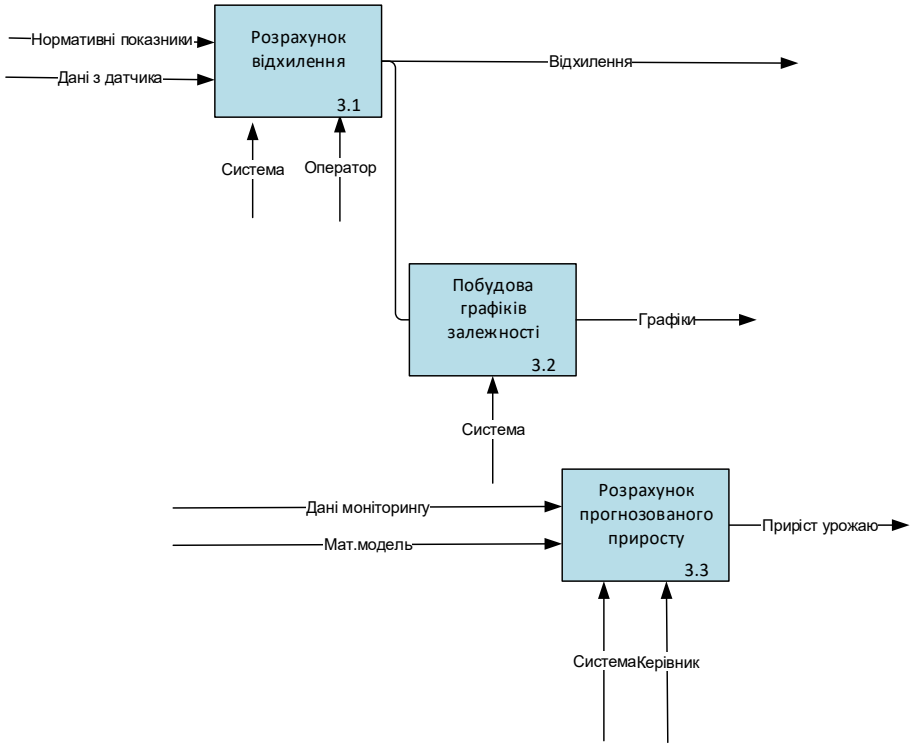


Рис.1.4 Діаграма SADT 1 рівня для процесу «Розрахунок показників»»

Діаграма потоків даних (DFD) — це візуальна модель, що застосовується для проєктування інформаційних систем і відображає, яким чином інформація циркулює всередині системи: як вона вводиться, обробляється, зберігається та передається. DFD часто використовують у структурному аналізі для наочної демонстрації логіки обробки даних.

Створення DFD зазвичай починається з контекстної діаграми, яка показує загальні взаємозв'язки системи з її зовнішнім оточенням. Далі діаграма деталізується — розкриваються внутрішні процеси та потоки даних, що дозволяє отримати більш чітке уявлення про внутрішню структуру системи.

У DFD використовуються чотири базові елементи:

- Процеси — позначають перетворення вхідної інформації у вихідну в межах системи;
- Сховища даних — місця зберігання інформації (репозиторії);
- Зовнішні сутності — об'єкти або особи поза межами системи, які взаємодіють з нею (наприклад, користувачі, постачальники, замовники, склади);
- Потоки даних — канали передачі інформації між процесами, сховищами та зовнішніми елементами.

Зовнішня сутність — це джерело або одержувач інформації, який не є частиною самої системи. На діаграмі вона зображується у вигляді прямокутника з тінню для візуального виділення. У процесі моделювання можливе перенесення зовнішніх сутностей до внутрішньої частини системи або навпаки — винесення внутрішніх процесів за її межі.

Процеси в DFD демонструють логіку обробки даних. Їх реалізація може бути різною — від програмного забезпечення чи організаційних підрозділів до апаратних засобів.

Сховища даних виступають абстрактними елементами, призначеними для збереження інформації, яку можна пізніше використати. У фізичному вигляді це

можуть бути бази даних, файли, картотеки, електронні таблиці тощо.

Потік даних описує переміщення інформації між компонентами системи. Це може бути передача по мережі, фізичне перенесення документів або будь-який інший спосіб передачі інформації.

Ієрархічна структура DFD дозволяє розбити складну систему на логічно впорядковані частини, кожна з яких уточнює попередній рівень. Такий підхід забезпечує зрозумілість, полегшує аналіз і подальше впровадження системи.

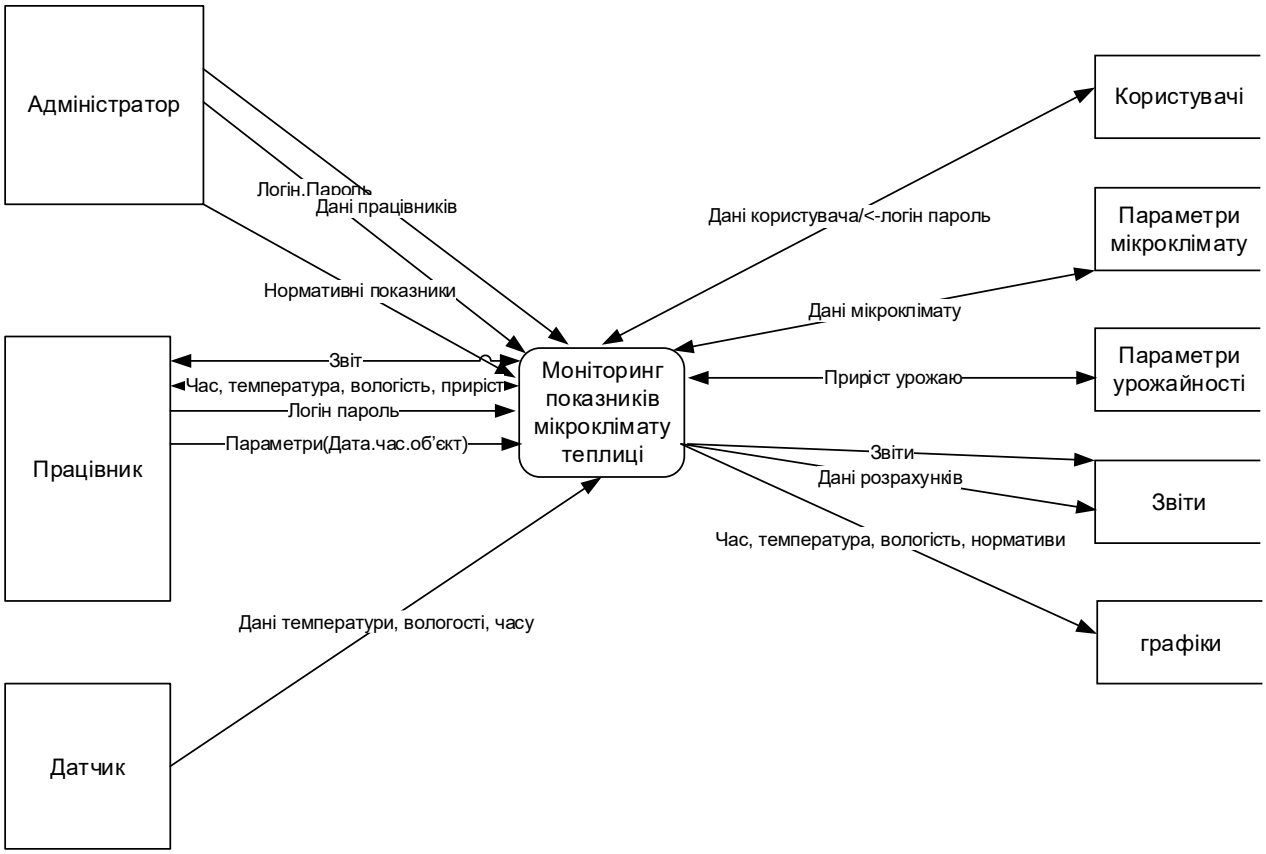


Рис 1.5 Контекстна діаграма DFD «Моніторинг показників мікроклімату теплиці»

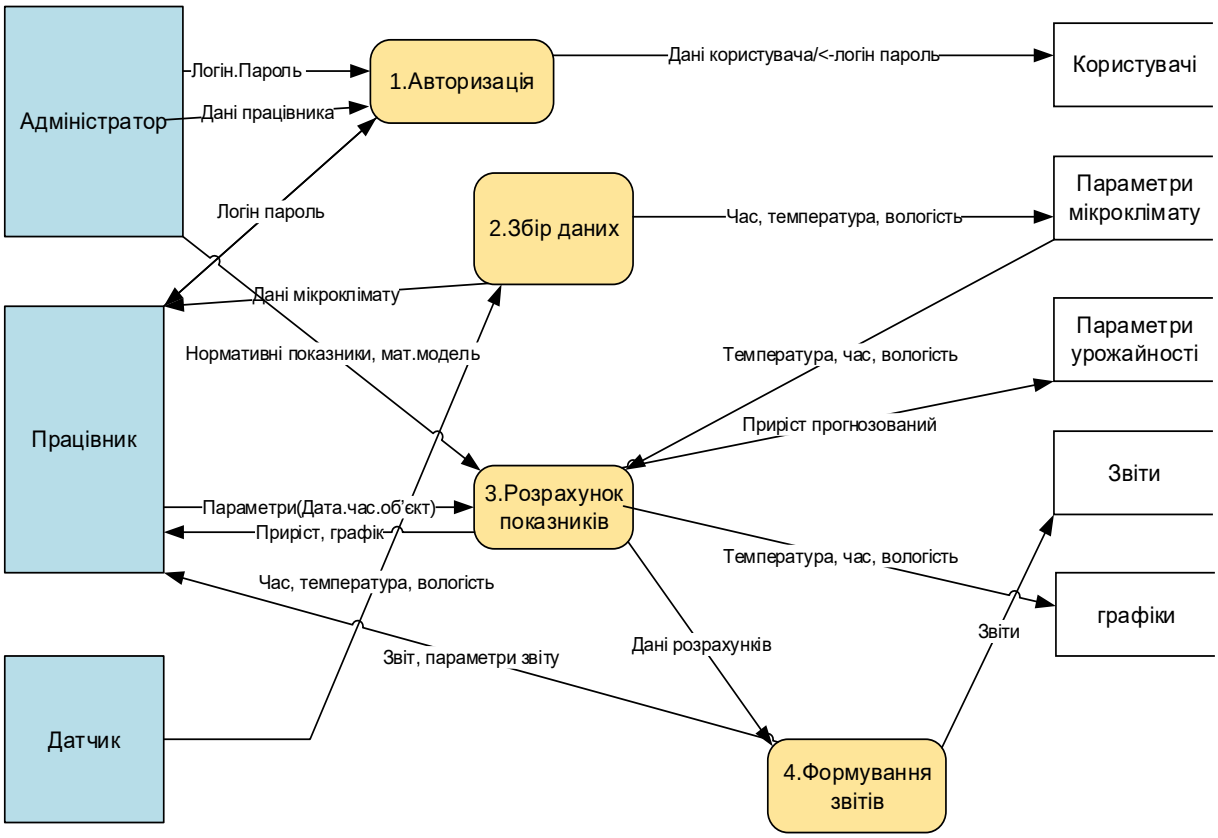


Рис 1.6 Діаграма потоків даних (DFD) 0-го рівня «Моніторинг показників мікроклімату теплиці»

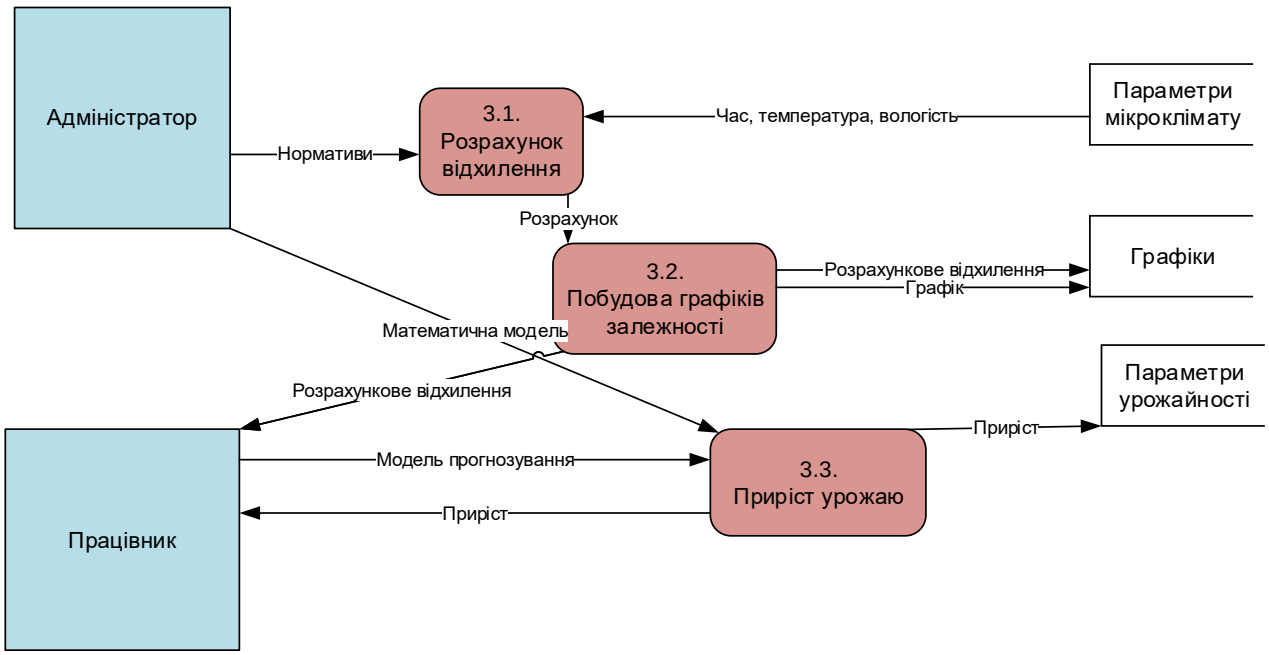


Рис.1.7 Діаграма потоків даних 1 рівня для процесу «Розрахунок показників»

1.3.3. Обєктно-орієнтоване моделювання

Діаграма прецедентів є складовою частиною UML (Unified Modeling Language — Уніфікована мова моделювання), що використовується в об'єктно-орієнтованому програмуванні для візуалізації та проєктування програмних систем. UML — це універсальний інструмент, який застосовується для побудови абстрактних моделей систем за допомогою графічних позначень. Створені UML-моделі допомагають легко трансформувати концептуальну схему системи у програмний код.

Діаграма прецедентів демонструє функціональність системи з точки зору користувачів (акторів), а також відображає їх взаємодію із системою через сценарії використання (прецеденти). Для опису взаємозв'язків між прецедентами використовуються два спеціальні типи зв’язків:

- «include» (включення) — означає обов’язкове виконання додаткової дії разом із основною. Наприклад, клієнт повинен пройти анкетування: система не дозволить надіслати анкету, поки користувач не заповнить усі необхідні поля (включаючи ім’я та email).
- «extends» (розширення) — позначає додаткову дію, яка виконується лише за певних умов. Наприклад, аналітик, проводячи аналіз даних, може за потреби продовжити збір інформації з анкет, якщо наявних даних недостатньо для прийняття рішення.

У розроблюваній системі визначено три ключових актори — Клієнт, Адміністратор та Аналітик, кожен з яких має своє коло завдань:

- Працівник (аналог клієнта) за допомогою веб-інтерфейсу отримує доступ до актуальних мікрокліматичних параметрів теплиці (температура, вологість, час), може переглядати візуалізацію даних та результати обчислень відхилень за вибраний період. Крім того, він може визначати прогнозований приріст продукції на основі фактичних умов.
- Адміністратор відповідає за управління користувачами системи, має

змогу реєструвати нових працівників, встановлювати нормативні параметри та математичні моделі для розрахунків.

- Датчик — технічний актор, який автоматично вимірює температурні показники та надсилає їх у систему з певною періодичністю.

Діаграма прецедентів забезпечує високий рівень абстракції та дозволяє легко зрозуміти структуру функціональності системи, як технічним спеціалістам, так і зацікавленим сторонам без технічної підготовки.

Ці задачі зображені на діаграмі прецедентів, що наведена на рис.1.8.

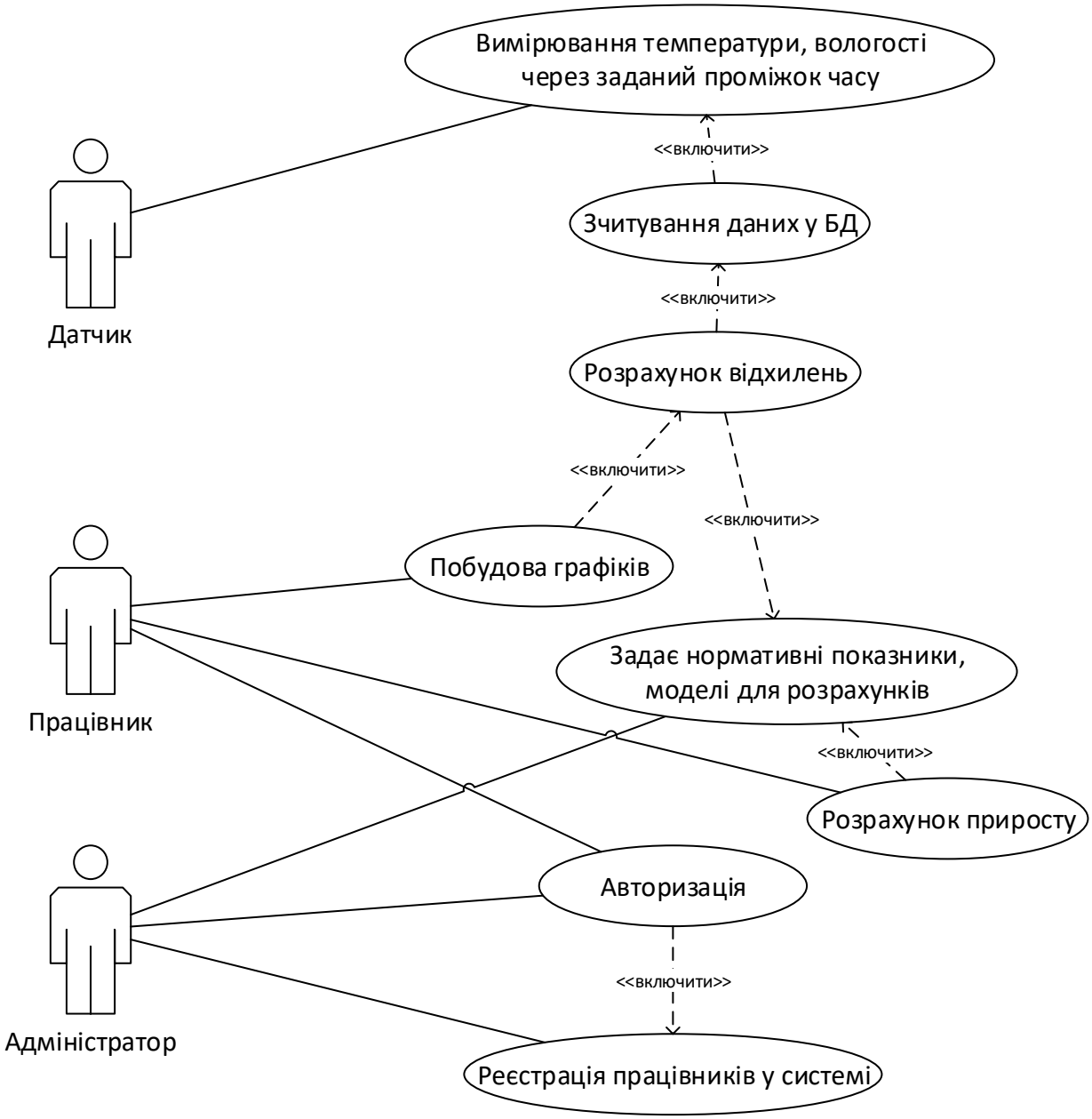


Рис.1.8. Діаграма прецедентів системи«Моніторинг показників мікроклімату теплиці»

Кожен прецедент, представлений на діаграмі, має зв'язок з відповідним актором, який бере участь у його реалізації. Наприклад, для виконання прецеденту «Розрахунок приросту продукції», адміністратор повинен попередньо внести в систему відповідну математичну модель, тоді як працівник задає часовий інтервал для виконання розрахунку.

Сценарій прецеденту «Розрахунок приросту продукції»:

- Передумова:

У систему попередньо введено модель, що дозволяє обчислити приріст на основі даних температури та вологості.

- Основний сценарій:

- Користувач обирає бажаний часовий інтервал;
- Система формує вибірку даних температури та вологості за вказаний період (A1);
- Визначаються відхилення кожного параметра від нормативного значення на відповідні моменти часу;
- Обчислюються середні відхилення температури та вологості за вибраний період;
- Виконується розрахунок приросту продукції на основі введеної моделі.

- Альтернативний потік:
  - (*A1*) — У базі даних відсутні показники за зазначений період часу.
- Результат:
  - Відбувається порівняння розрахованого приросту з фактичними даними.

Діаграма класів (Class Diagram) — це інструмент для відображення статичної структури системи. Вона показує взаємозв’язки між класами, типами даних, атрибутами та асоціаціями. Така діаграма може включати ієрархії пакетів, хоча її головна мета — представити модель у термінах об’єктно-орієнтованого підходу. Незважаючи на те, що деякі елементи поведінки можуть бути позначені, динамічні аспекти системи розкриваються через інші типи UML-діаграм

(наприклад, діаграми активностей або послідовностей).

Опис класів на діаграмі класів (рис. 1.9):

1. Класи-сутності:
   - Дані з датчиків — абстрактне представлення інформації, отриманої з сенсорів, із вказаними параметрами доступу;
   - Урожай — одиниця інформації, що репрезентує продукцію, отриману на основі обробки даних; об’єкти цього класу мають тимчасовий життєвий цикл у межах одного сеансу;
   - Показники — ключові об’єкти, які містять оброблені значення від сенсорних джерел;
   - Користувач — описує особу, що взаємодіє із системою; визначає її роль і рівень доступу;
   - Звіт — структурований документ, що містить оброблені дані, відібрані за певними критеріями та подіями.

Опис діаграми класів. В діаграмі класів (рис. 1.9) виділено такі класи:

2. Класи сутності:
   - Дані з датчиків – абстракція, яка представляє собою інформаційний ресурс та налаштування доступу для нього;
   - Урожай – цілісна одиниця інформації отримана від джерела; об’єкти цього класу «живуть» протягом сеансу обробки інформації отриманої від джерела;
   - Показники – робочі об’єкти системи; цілісна одиниця оброблені від джерела інформації;
   - Користувач – опис користувача в системі; визначає права користувача;
   - Звіт – вибрані та оброблені за вказаними критеріями дані на основі подій.

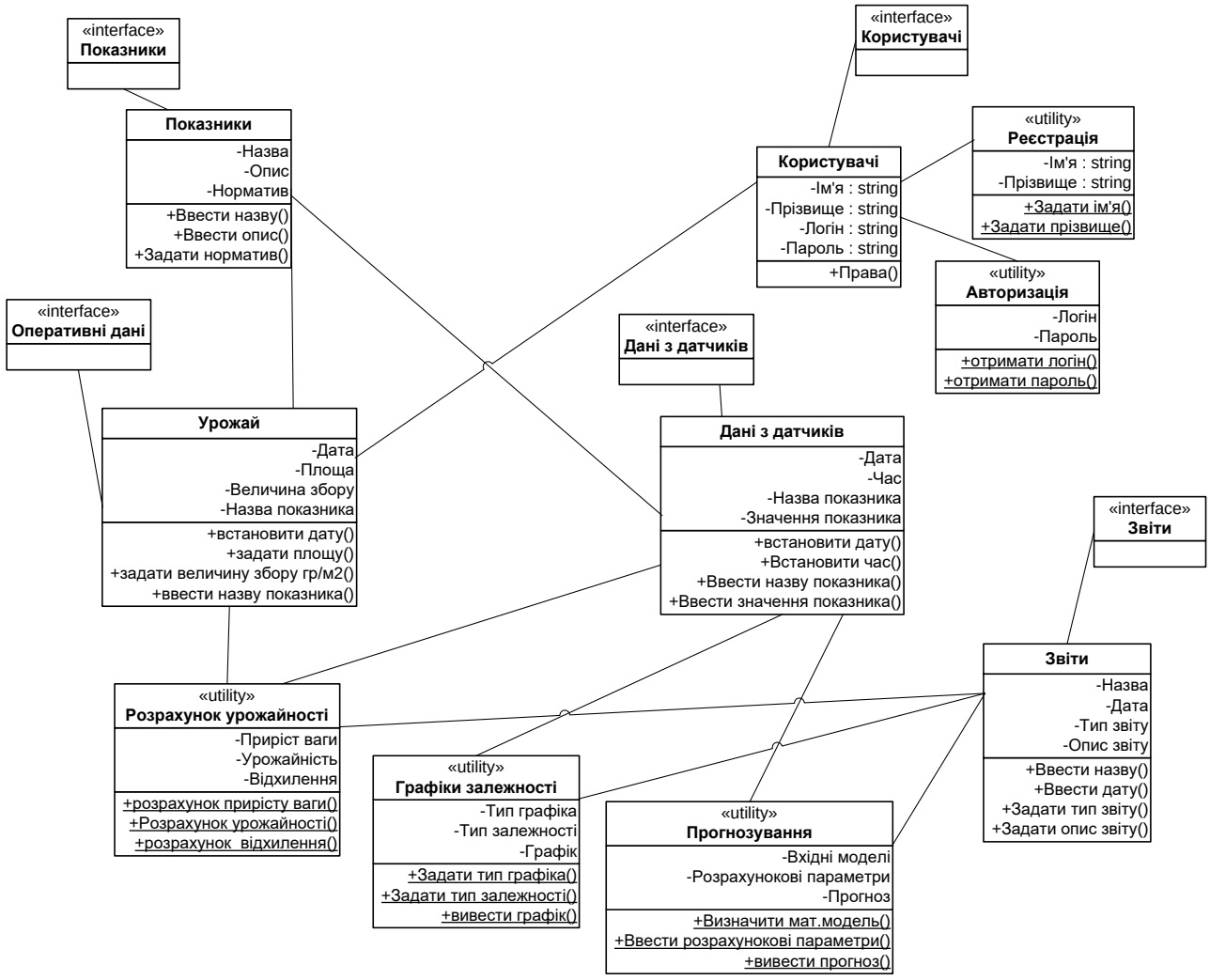


Рис.1.9 Діаграма класів системи

3. Класи контроллери:

- «прогнозування» даних – відповідає за опитування джерела та отримання від нього нових даних;
- Графіки – забезпечує перетворення отриманих від джерела даних в події;
- Розрахунки– перевіряє правильність введених користувачем даних (як при авторизації так і при редагуванні чи створенні користувача);
- формування звіту – відповідає за вибір по критерію даних та їхній подальший аналіз і формування звіту.

4. Класи інтерфейси:

- показники;

- авторизація;
- користувача;
- урожай;
- форма «Звіт».

Класи можуть перебувати у різних видах зв’язків між собою. Одним із таких типів є асоціація — це двосторонній взаємозв’язок між об’єктами різних класів, у якому обидва учасники мають рівний статус у відношенні. Для асоціацій зазвичай вказується мультиплікація — кількість об’єктів кожного класу, які можуть брати участь у зв’язку:

- 0 — відсутність об’єктів;
- 1 — один об’єкт;
- * — будь-яка кількість об’єктів;
- 0..* — відсутність або декілька об’єктів;
- 1..* — як мінімум один тощо.

На діаграмі класів (рис. 1.9) можна побачити такі асоціативні зв’язки між класами:

- Показники ↔ Дані з датчиків;
- Урожай ↔ Користувач;
- Дані з датчиків ↔ Прогнозування;
- Дані з датчиків ↔ Графіки;
- Користувач ↔ Формування звіту.

Діаграма послідовності (Sequence Diagram) описує обмін повідомленнями між об’єктами у певній послідовності відповідно до часу. Вона зосереджена на тому, в якій черговості і коли об’єкти надсилають один одному повідомлення, а не на структурних зв’язках між ними.

Існують два основні види таких діаграм:

- Дескрипторна — показує всі можливі сценарії взаємодії;
- Інстанційна (конкретна) — зображує один визначений сценарій виконання.

Хоча діаграми послідовності та кооперації передають однакову логіку взаємодії об’єктів, вони роблять це різними способами: перша — через акцент на часову вісь, друга — через структуру об’єктних зв’язків.

На рис. 1.10 наведено приклад діаграми послідовності.

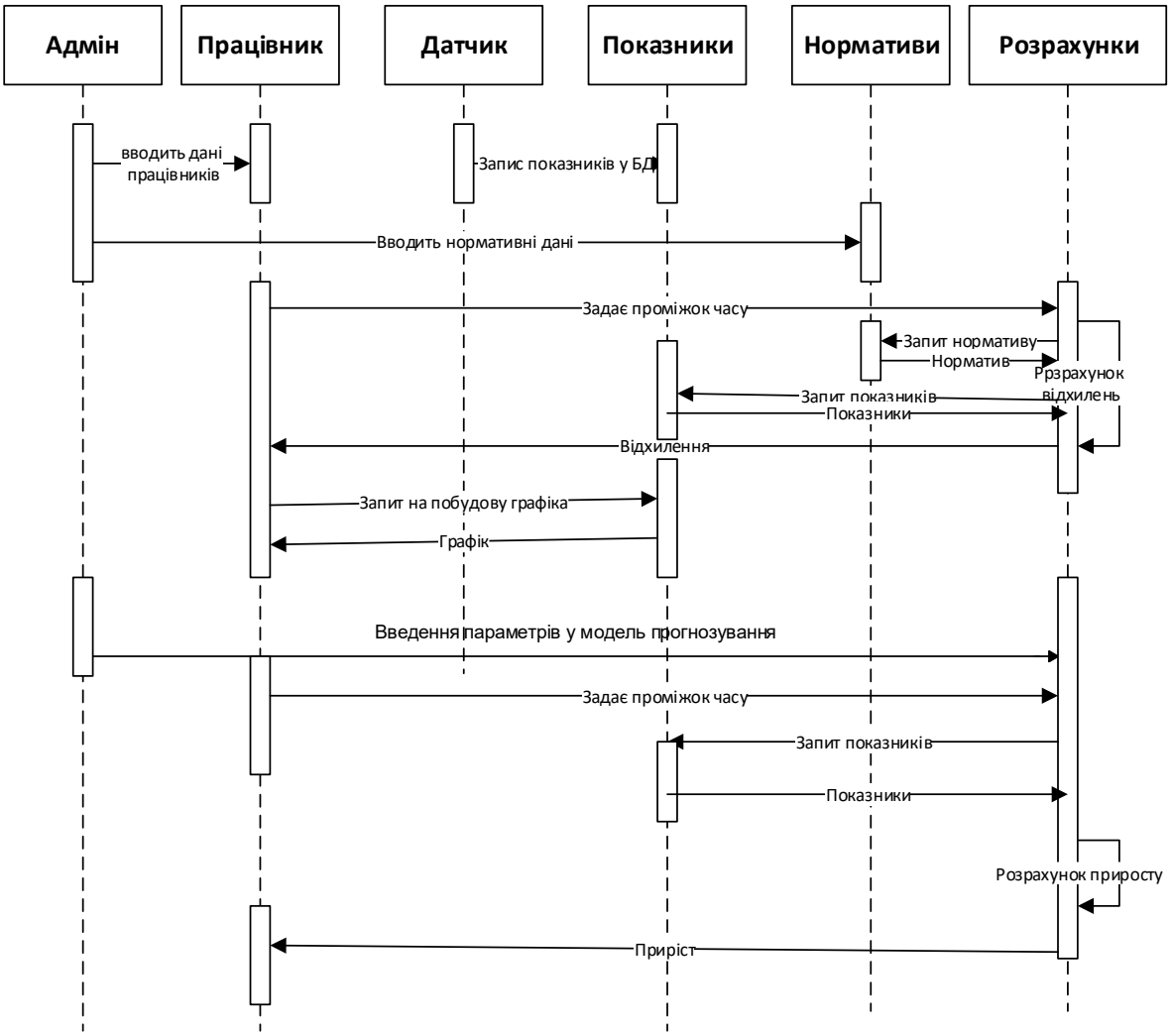


Рис.1.10 Діаграма послідовностей «Моніторинг показників мікроклімату теплиці»

Діаграма активності (Activity Diagram) є розвитком діаграми станів (Statechart Diagram) і використовується для відображення різних станів моделюваного об’єкта. Основне призначення такої діаграми — ілюструвати бізнес-процеси, що відбуваються в межах об’єкта.

Діаграма активності в UML являє собою графічне зображення послідовності дій (граф діяльностей). Цей граф є різновидом графа станів

скінченого автомата, де вершинами виступають окремі дії, а переходи між ними відбуваються після завершення відповідних дій.

Діаграми активності для процесу «Розрахунок показників» наведено на рис. 1.11 та 1.12.

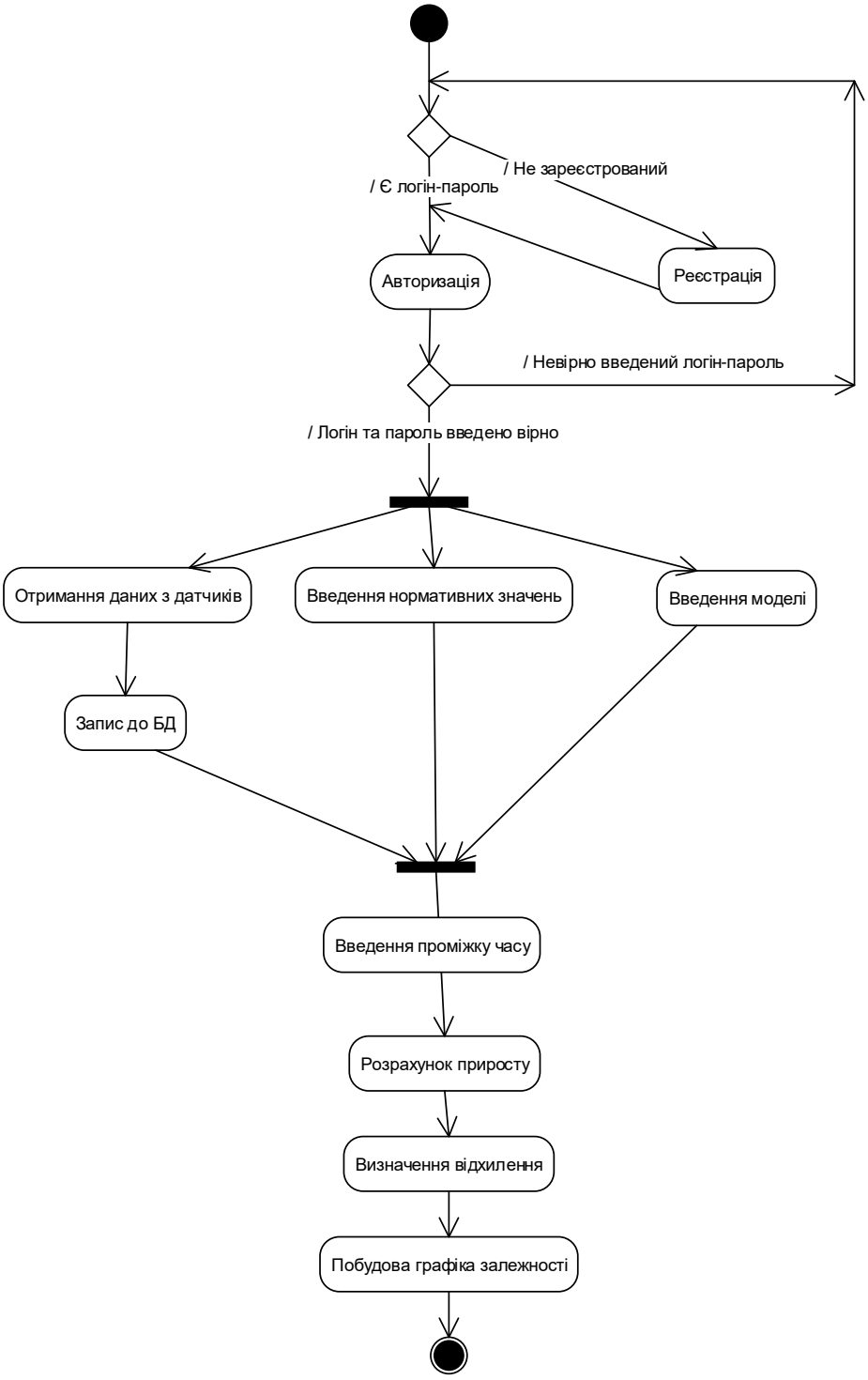


Рис 1.11. Загальна діаграма діяльності системи

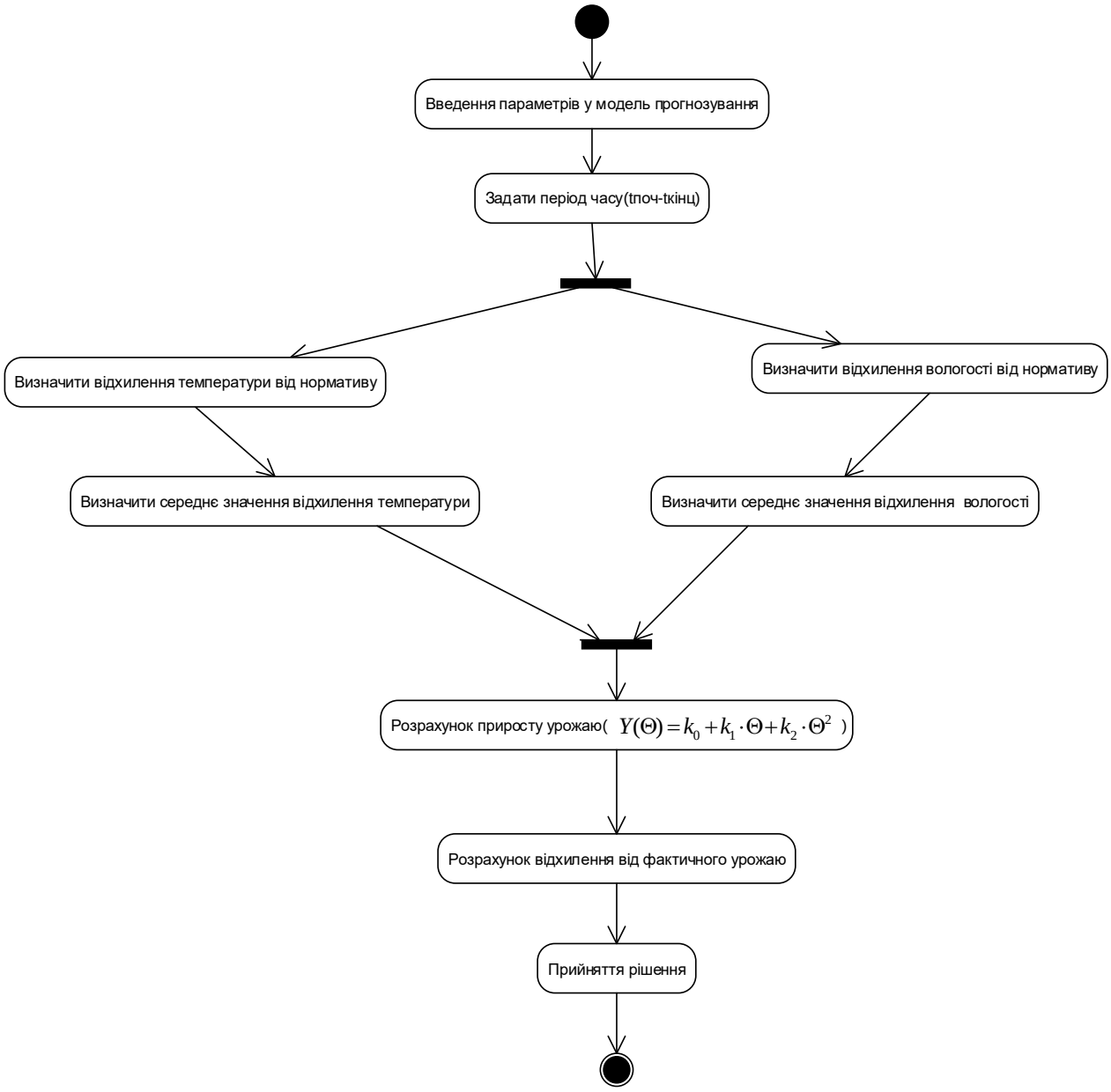


Рис 1.12. Діаграма діяльності «Розрахунок приросту»

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Побудова логічної моделі даних

2.1.1 Загальні відомості про ER-діаграму.

Модель «сутність-зв’язок» (Entity-Relationship, ER) забезпечує візуалізацію структури даних у базі даних. Її мета — створити концептуальну схему, яку легко відтворити в різних системах баз даних. На відміну від DFD, ER-модель фокусується саме на структурних зв’язках між даними.

ER-діаграми (ERD) використовуються для моделювання даних і визначення їхніх взаємозв’язків. Вперше нотацію запропонував Пітер Чен, але на практиці поширена модифікована нотація Баркера, наприклад, «Crow’s Foot». Вона включає сутності, зв’язки між ними та їх атрибути.

Сутність — це об’єкт (реальний чи уявний), що має значення для предметної області і про який зберігається інформація. Кожна сутність зображується прямокутником із унікальним ім’ям і переліком атрибутів. Атрибути описують характеристики сутності.

Тип сутності — це множина об’єктів з однаковими властивостями, які можуть бути фізичними або концептуальними. Унікальний екземпляр типу — це сама сутність. Сутності групуються у класи, а типи поділяються на сильні (незалежні) та слабкі (залежні від інших сутностей). Слабкі сутності також називають дочірніми або підлеглими, а сильні — батьківськими або домінантними.

Атрибут — це характеристика сутності, важлива для аналізованої предметної області, яка служить для ідентифікації, класифікації, кількісної оцінки або опису стану сутності. Він відображає властивості, пов’язані з групою об’єктів (людей, місць, подій, ідей тощо).

Конкретний атрибут має тип і значення. В ER-діаграмах атрибути прив’язуються до сутностей, і кожен екземпляр сутності має одне визначене

значення для кожного атрибута. Серед атрибутів виділяють потенційний ключ — унікальний набір, що ідентифікує кожен екземпляр. З них один обирається як первинний ключ, решта — альтернативні.

Взаємозв’язок — це іменована асоціація між двома сутностями, важлива для предметної області. Для позначення ідентифікації через зв’язок використовують спеціальні позначки (риски або суцільні лінії).

Правила моделювання ER:

- Кожен елемент (сутність, атрибут, зв’язок) має унікальне ім’я в межах моделі.
- Сутність містить один або кілька атрибутів, власних або успадкованих через зв’язки.
- Атрибути всередині сутності мають унікальні імена.
- Атрибути з різних сутностей можуть мати однакові імена, але з однаковим значенням і тлумаченням.
- Сутність має принаймні один атрибут або набір атрибутів, що унікально ідентифікують її екземпляри.
- Кожна сутність може бути пов’язана з іншими через один або кілька зв’язків.

2.1.2 Побудови ER-діаграми.

Для створення ER-діаграми використано програму MySQL WorkBench.

Розглянемо основні сутності логічної моделі даних:

1. Сутність «Норматив» — містить нормативні значення параметрів мікроклімату в різний час доби і має неідентифікуючий зв’язок із сутністю «Культура». Атрибути:

- id нормативу (числовий, первинний ключ);
- час доби (тип дата-час);
- температура (числовий);
- вологість (числовий);

- id культури (числовий, зовнішній ключ).

2. Сутність «Культура» — опис тепличних культур. Атрибути:

- id культури (числовий, первинний ключ);
- назва (символьний);
- опис (символьний).

3. Сутність «Працівники» — містить інформацію про співробітників:

- індивідуальний номер (числовий, первинний ключ);
- прізвище, ім'я по батькові, посада, логін, пароль (символьні атрибути).

4. Сутність «Показники» — зберігає дані мікроклімату:

- час (дата-час, первинний ключ);
- температура (числовий);
- вологість (числовий).

5. Сутність «Розрахунки» — фіксує виконані розрахунки за періоди часу, має три неідентифікуючі зв’язки. Атрибути:

- id розрахунку (числовий, первинний ключ);
- інд. номер працівника (числовий, зовнішній ключ);
- id нормативу (числовий, зовнішній ключ);
- час початку і час кінця (дата-час);
- приріст, відхилення температури, відхилення вологості (числові).

Модель сутність-зв’язок наведена на рис. 1.13.

2.2 Вибір інструментарію для створення БД та роботи з нею

2.2.1 Вибір СУБД.

Для управління інформацією (введення, пошук, оновлення, видалення) використовують спеціальні програми — системи управління базами даних (СУБД). Сучасні СУБД активно розвиваються, підтримують нові типи даних (мультимедіа, геоінформація) та нові технології, як-от клієнт-серверна архітектура і розподілені бази, що забезпечують широкий доступ через Інтернет.

Інформація у базах даних зберігається у таблицях, де стовпці — це поля, а

рядки — записи.

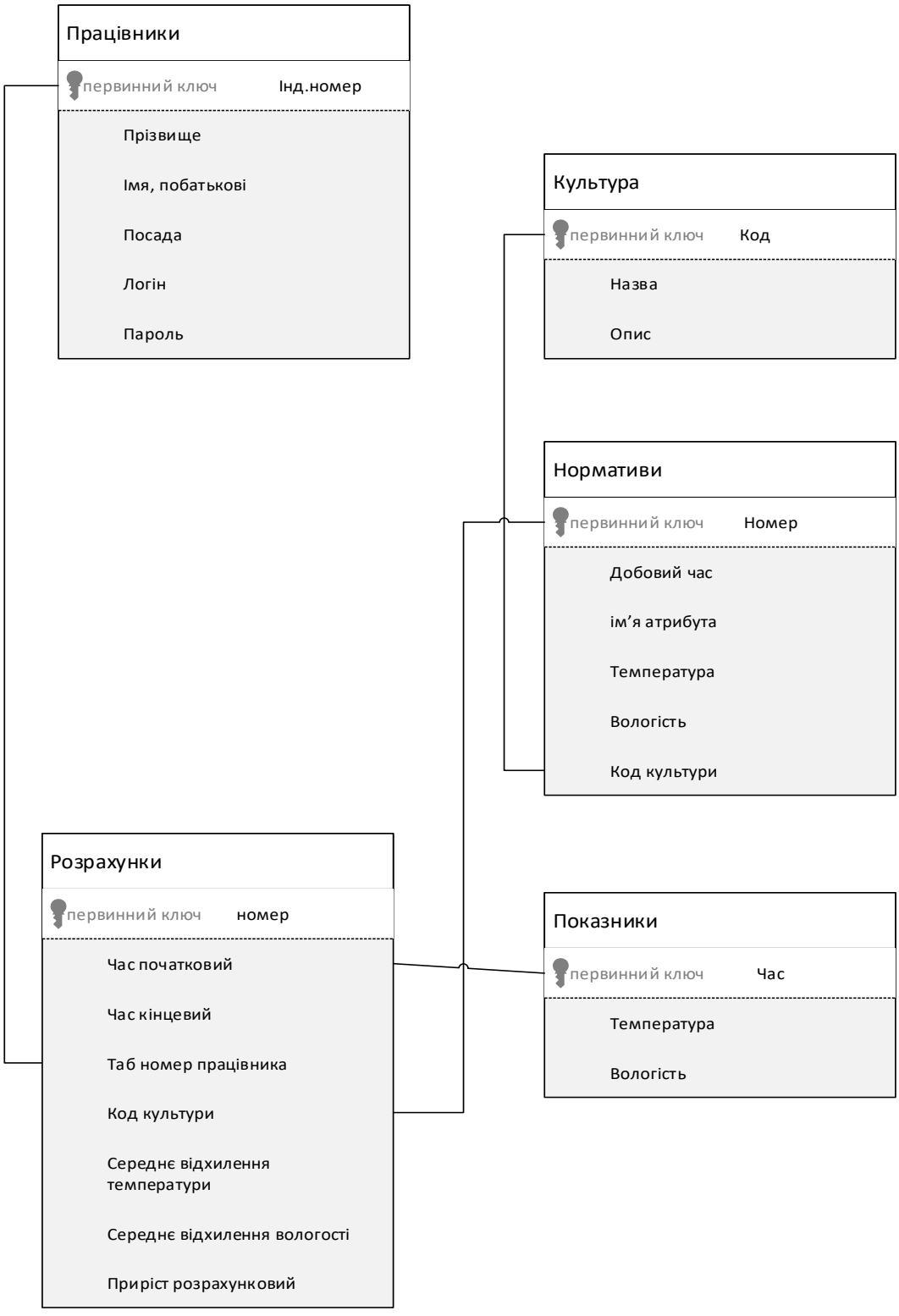


Рис.1.13 Модель «сутність-зв'язок»

Для цієї роботи обрали MySQL через простоту установки, швидкість та портативність. Система не прив'язана до конкретної СУБД і може працювати з

будь-якою реляційною базою.

MySQL — це популярна СУБД з відкритим кодом, розроблена компанією «ТсХ» як альтернатива комерційним системам. Вона широко використовується для динамічних веб-сайтів і підтримує багато мов програмування.

MySQL має подвійне ліцензування: безкоштовну GPL-ліцензію, що вимагає відкритого коду для програм з використанням бібліотек MySQL, та комерційну ліцензію Oracle для закритих проектів із сервісною підтримкою.

Назва «MySQL» походить або від префікса «My» у розробках компанії, або від імені дочки одного з розробників. Логотип у вигляді дельфіна на ім'я «Sakila» обрали зі списку, запропонованого користувачами.

Клієнт-серверна система складається з двох окремих процесів — клієнта і сервера, які можуть працювати на різних комп'ютерах і обмінюються даними через мережу. Сервер надає певні послуги (наприклад, файлові або бази даних), а клієнт звертається до сервера з запитами і чекає відповіді.

У системі моніторингу матеріальних витрат на основі СУБД використовується саме клієнт-серверна архітектура.

Для порівняння розглянемо файл-серверну систему, де дані зберігаються на файловому сервері, а обробка виконується на робочих станціях за допомогою настільних СУБД (Microsoft Access, FoxPro тощо). У такій системі додаток на робочій станції відповідає за весь функціонал: інтерфейс, логіку і маніпуляції з даними. Сервер лише відкриває, закриває і змінює файли, а не працює з базою даних напряму.

Це призводить до того, що обробкою даних займаються кілька незалежних процесів, а також весь обсяг даних має передаватися по мережі на робочу станцію, що знижує ефективність.

У клієнт-серверній системі функції розділені: сервер баз даних (наприклад, MySQL) зберігає і обробляє дані, а клієнт відповідає за інтерфейс і формує SQL-запити до сервера. Сервер виконує ці запити і повертає результати клієнту. Також логіку обробки можна розподіляти між клієнтом і сервером.

Завдяки цьому обробка даних відбувається безпосередньо на сервері, де вони зберігаються, що значно зменшує трафік і покращує продуктивність системи.

Сервер баз даних працює з транзакціями, які забезпечують:

- Атомарність — всі операції транзакції виконуються повністю або зовсім не виконуються;
- Ізольованість — транзакції різних користувачів не заважають одна одній;
- Стійкість — результати транзакції зберігаються навіть після збоїв.

Цей механізм ефективніший, ніж у настільних СУБД, бо сервер централізовано контролює транзакції. У файл-серверних системах збій на робочій станції може призвести до втрати даних, тоді як у клієнт-серверній — збій клієнта не впливає на цілісність і доступність даних.

Масштабованість системи дозволяє працювати з великою кількістю користувачів і великими обсягами даних, збільшуючи продуктивність апаратури без зміни програмного забезпечення.

Настільні СУБД обмежені кількістю користувачів (5-7) і розміром (30-50 Мб), і ці обмеження не усуваються збільшенням апаратної потужності. Серверні СУБД можуть підтримувати тисячі користувачів і сотні гігабайтів даних.

Сервер баз даних також забезпечує гнучкий захист даних — права доступу можна налаштувати навіть на рівні окремих полів таблиць. Для додаткового захисту можна заборонити прямий доступ до таблиць, використовуючи уявлення та збережені процедури.

У додатках виділяють три логічних шари:

- інтерфейс користувача;
- бізнес-логіку (правила обробки);
- управління даними.

У файл-серверних системах всі шари поєднані в одному додатку на робочій станції, що ускладнює оновлення і підтримку системи.

2.2.2 Мова маніпулювання даними SQL.

Зі збільшенням обсягу даних та потребою їх зберігання і обробки виникла необхідність у створенні універсальної мови баз даних, яка могла б працювати на різних типах комп'ютерних систем. Така мова дає змогу користувачам оперувати даними незалежно від того, чи це персональний комп'ютер, мережева робоча станція чи потужна ЕОМ.

Після розробки логічної моделі та вибору платформи для фізичної реалізації, у нашому випадку — Oracle MySQL, переходять до створення фізичної бази даних на обраному сервері. Сервер, на якому розміщується база, може бути розташований як на одному комп'ютері, так і на іншому. На сервері спочатку створюється основна база, а в ній — необхідні таблиці.

У MySQL базу даних і таблиці можна створити двома способами:

- Через графічний інтерфейс MySQL Workbench;
- За допомогою мови SQL.

SQL (Structured Query Language) — це структурована мова запитів, що дозволяє створювати і обробляти дані у реляційних базах. Її незалежність від конкретних комп'ютерних технологій та широке визнання в галузі зробили SQL стандартом для роботи з базами даних. До появи реляційних БД існували мови, орієнтовані на опрацювання даних у вигляді записів файлів, що вимагало глибокого розуміння організації зберігання даних.

SQL працює з даними у вигляді логічно пов'язаних таблиць-відношень, зосереджуючись на кінцевому результаті запиту, а не на деталях його виконання. Вона сама вибирає оптимальний спосіб отримання даних, індекси та порядок операцій, тому користувач не має вказувати ці нюанси.

Основні переваги SQL:

- Стандартність — визнана міжнародними організаціями.
- Незалежність від СУБД — однаковий синтаксис в різних системах.
- Переносимість — підтримка різних обчислювальних платформ від локальних ПК до великих мереж.
- Реляційна модель — простота й зрозумілість через табличну структуру.

- Інтерактивні запити — швидкий доступ до інформації без написання складного коду.
- Програмний доступ — однакові оператори для інтерактивної роботи і вбудованих додатків.
- Гнучке представлення даних — різні користувачі можуть бачити різні уявлення таблиць, що покращує безпеку і персоналізацію.
- Динамічні зміни структури БД — можна змінювати структуру бази без переривання роботи додатків.
- Підтримка клієнт-серверної архітектури — SQL виступає посередником між користувачем і сервером бази, забезпечуючи ефективну взаємодію.

Зазвичай користувачі працюють із заздалегідь створеними адміністраторами баз даних структурами, не створюючи таблиць самостійно.

В якості системи керування базами даних (СКБД) було обрано MySQL з таких причин:

- MySQL є провідною СКБД у сфері веб-розробки;
- завдяки простоті використання MySQL забезпечує високу продуктивність;
- MySQL є портативною і підтримується на багатьох платформах, таких як Linux, MacOS, Windows, Solaris;
- для PHP існує спеціальний модуль для роботи з MySQL;
- більшість хостинг-провайдерів підтримують MySQL;
- MySQL є вільною (open-source) реляційною системою управління базами даних.

2.3 Побудова структури БД

Для побудови баз даних на веб-ресурсах широко використовується PhpMyAdmin — інструмент, який містить увесь необхідний функціонал для створення таблиць і роботи з ними за допомогою SQL-запитів. Для створення

бази даних відкриваємо вікно сервера з локальною базою даних та вибираємо пункт PhpMyAdmin, як показано на рисунку 2.1.

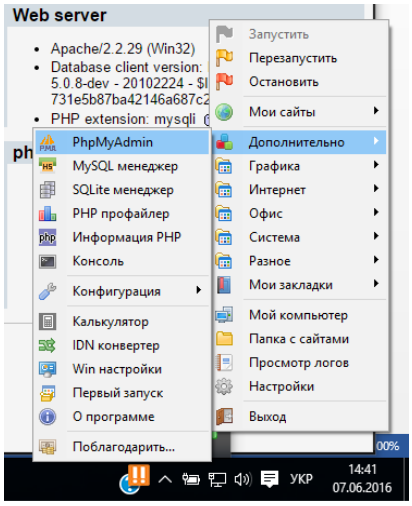


Рис.2.1 Меню створення нової БД

Далі заповнюємо поля з назвою БД та обираємо кодування, в якому буде відбуватися вся подальша робота з базою. В данному випадку я обрав utf8_unicode_ci як зображено на рис.2.2, оскільки за такого кодування в усіх існуючих браузерах де працюватиме система – буде корректно відображено зміст таблиць.

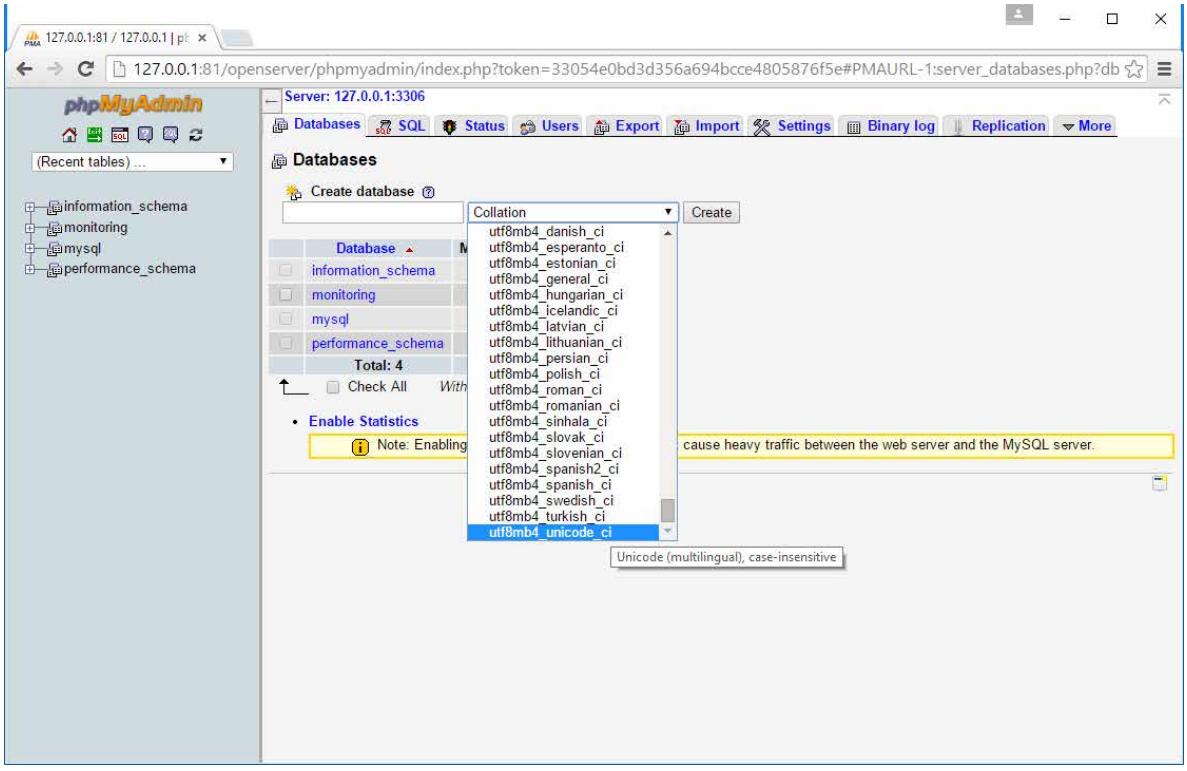


Рис.2.2 Створення нової БД

Створення таблиці з використанням інтерфейсу БД представлено на рис.2.3

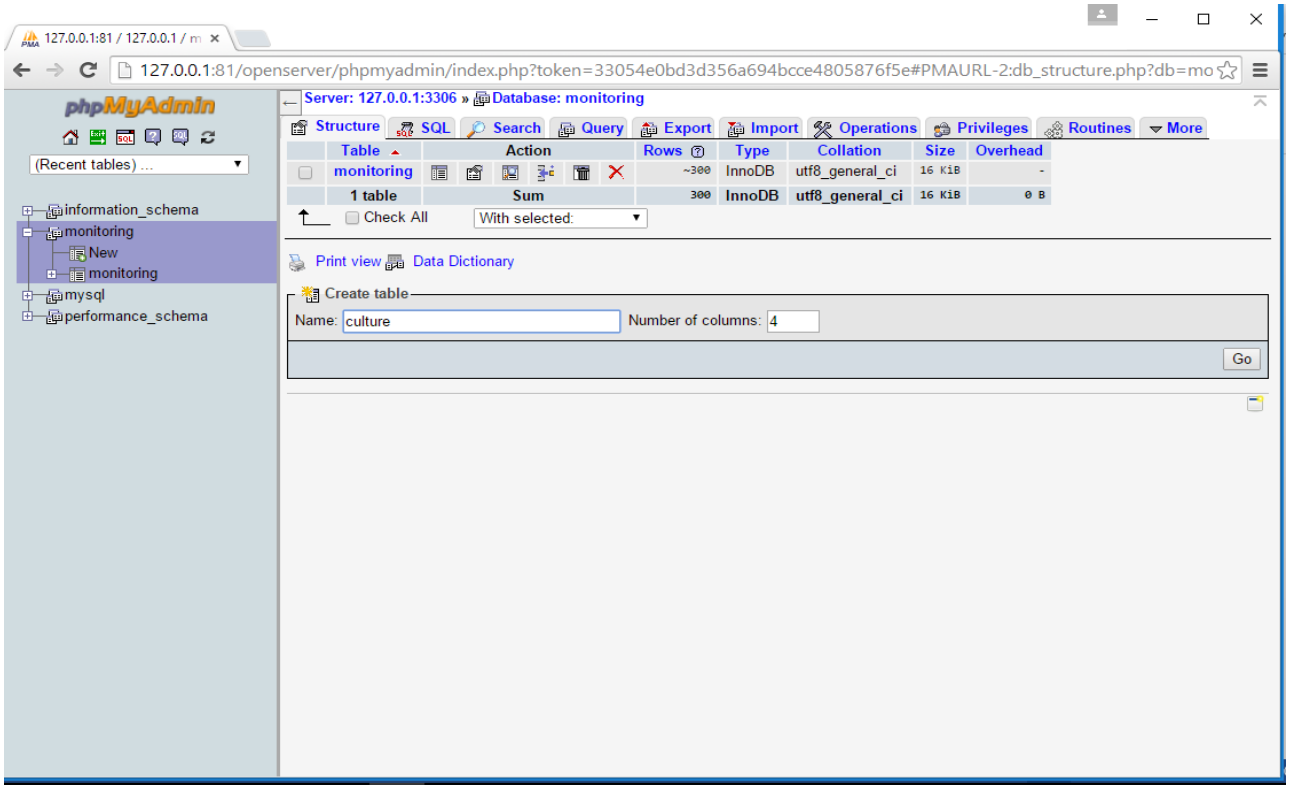


Рис.2.3 Інтерфейс створення таблиці

3 ПРИКЛАДНЕ ПРОГРАМНЕ ТА АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Обгрунтування технічного забезпечення системи

Інформаційна система моніторингу мікроклімату в теплиці складається з кількох складових, які включають технічне обладнання, програмне забезпечення та інформаційну базу даних із показниками моніторингу. Система побудована за клієнт-серверною архітектурою і має веб-клієнт (рис. 3.1).

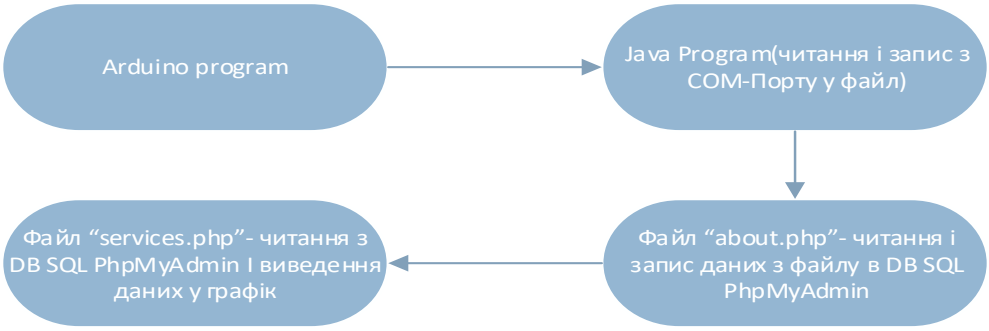


Рис.3.1 Структурно-функціональна схема роботи компонентів системи

Датчики, інтегровані в плату Arduino, програмно налаштовані на зчитування параметрів через задані інтервали часу. Модуль для зчитування даних із COM-порту, реалізований на Java, передає інформацію у текстовий файл, який за допомогою PHP-скриптів обробляється, і дані у реальному часі записуються до бази даних на SQL-сервері.

Веб-додаток, розроблений на PHP, дозволяє виводити моніторингові показники в режимі реального часу, будувати графіки та визначати відхилення від нормативних значень. За необхідності функціонал системи може бути розширений.

Для вимірювання параметрів мікроклімату використовувалась плата Arduino — апаратна платформа з мікроконтролером Atmel AVR та набором входів/виходів. Arduino підтримує програмування на основі мови, схожої на C/C++ (Processing/Wiring), та може працювати автономно або в зв'язці з ПЗ на комп'ютері (наприклад, Adobe Flash, Processing).

Плата містить мікроконтролер з вбудованим завантажувачем, що

дозволяє програмувати її через USB або послідовний порт без додаткових програматорів. Вона має стабілізатор живлення (+5В або +3,3В) і працює на кварцовому резонаторі з частотою 8 або 16 МГц.

Arduino забезпечує велику кількість цифрових і аналогових входів/виходів, а також підтримує підключення розширень — «shields» — через спеціальні штирові роз'єми, що дає змогу додавати різноманітні датчики та виконавчі пристрої. Існують різні форм-фактори плат, включно зі зменшеними (Nano, Lilypad) та спеціалізованими для робототехніки.

Інтегроване середовище розробки Arduino — кросплатформенний додаток на Java, що включає редактор коду, компілятор і засіб завантаження програм у плату. Мова програмування базується на C++ з додатковими бібліотеками, спрощеними для новачків.

Програми для Arduino пишуться на C або C++. Середовище розробки Arduino містить бібліотеку «Wiring», яка спрощує виконання стандартних операцій вводу/виводу. Для створення циклічної програми користувач має визначити лише дві основні функції.

Arduino Mega 2560 базується на мікроконтролері ATmega2560 і має 54 цифрових входи/виходи (15 з них підтримують ШІМ), 16 аналогових входів, 4 апаратних UART, кварцовий резонатор на 16 МГц, USB-роз’єм, роз’єм живлення, ICSP для програмування та кнопку скидання. Для роботи пристрою достатньо підключити його до комп’ютера через USB або живлення від адаптера чи батарейки. Mega сумісний із більшістю плат розширень Arduino Duemilanove та Diecimila.

Датчик (давач) — це пристрій, який перетворює вимірювану фізичну величину (температуру, тиск, вологість тощо) у зручний для обробки сигнал (електричний, оптичний тощо). Він використовується у системах вимірювання, керування та автоматизації, сигналізуючи про стан об’єкта.

DHT11 — цифровий датчик температури та вологості з каліброваним виходом. Він поєднує резистивний датчик вологості і NTC-термістор для виміру

температури, забезпечуючи точні та швидкі показники завдяки вбудованому 8-бітному мікроконтролеру.

3.2 Вибір інструментарію для створення програмного забезпечення

Модуль, що відповідає за зчитування даних із COM-порту плати Arduino, реалізований із застосуванням мови програмування Java. Java є об'єктно-орієнтованою мовою, розробленою компанією Sun Microsystems у 1995 році і служить ключовим компонентом платформи Java. На сьогодні розвитком та підтримкою мови займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксис Java має багато спільного з мовами C та C++, що полегшує адаптацію програмістів, знайомих з цими мовами. У стандартній реалізації Java-програми компілюються у проміжний байт-код, який під час виконання інтерпретується спеціальною віртуальною машиною Java (JVM), що адаптує код під конкретну платформу.

Компанія Oracle надає офіційний компілятор Java та віртуальну машину, які відповідають стандартам Java Community Process і поширюються під ліцензією GNU General Public License. Мова Java значною мірою запозичила свій синтаксис із C та C++, зокрема взяла за основу об'єктну модель останньої, однак внесла модифікації, які дозволили уникнути багатьох потенційних помилок програміста і значно спростили розробку об'єктно-орієнтованих застосунків. Деякі задачі, які в C/C++ виконуються програмістом вручну, у Java делеговані віртуальній машині, що підвищує надійність коду.

Java була спроектована перш за все як мова, незалежна від апаратної архітектури, тому вона має обмежений доступ до низькорівневих ресурсів апаратного забезпечення. Проте, за потреби, вона дозволяє звертатися до підпрограм, написаних іншими мовами, щоб забезпечити більш тісну взаємодію з обладнанням.

Історично Java вплинула на створення мови J++, розробленої компанією Microsoft. Однак через судовий позов від Sun Microsystems розробка J++ була

припинена, оскільки ця мова була модифікацією Java. Пізніше Microsoft у рамках платформи .NET випустила мову J#, яка допомагала розробникам J++ і Java переходити на нову платформу. Згодом основною мовою платформи стала C#, яка запозичила багато концепцій з Java. Остання підтримка J# була включена в Microsoft Visual Studio 2005. Важливо зазначити, що мова сценаріїв JavaScript, попри схожу назву і синтаксис, не має прямого зв'язку з Java.

Під поняттям «платформонезалежність» Java розуміє можливість запуску програм на будь-якій підтримуваній апаратній та системній платформі без необхідності змін у вихідному коді чи його повторної компіляції. Це досягається шляхом компіляції початкового Java-коду у байт-код — проміжний формат, що є набором спрощених машинних інструкцій. Виконання цієї програми можливе на будь-якій системі, яка має встановлену віртуальну машину Java, котра інтерпретує байт-код і адаптує його до специфіки конкретної операційної системи та процесора. Сьогодні JVM доступні для більшості процесорних архітектур і операційних систем.

Стандартні бібліотеки Java забезпечують уніфікований доступ до таких платформозалежних функцій, як обробка графіки, багатопотокове виконання і робота з мережами. У деяких реалізаціях для підвищення продуктивності JVM здатна компілювати байт-код у машинний код безпосередньо перед або під час виконання програми (технологія JIT-компіляції).

Головною перевагою використання байт-коду є висока портативність програм. Водночас, інтерпретація байт-коду призводить до додаткових накладних витрат, через що інтерпретовані програми зазвичай виконуються повільніше, ніж нативні, скомпільовані безпосередньо у машинний код. Саме це сприяло появі репутації Java як «повільної» мови програмування. Однак останні версії JVM впровадили ряд оптимізацій, які суттєво звузили розрив у продуктивності між Java та нативним кодом.

Одним із ключових методів підвищення продуктивності виконання Java-програм є використання технології Just-In-Time-компіляції (JIT). Вона полягає в

тому, що байт-код Java перетворюється на машинний код безпосередньо під час першого виконання програми, після чого результат зберігається в кеші для подальшого використання. Це дозволяє запускати програму швидше, ніж у разі використання лише інтерпретації, хоча вимагає додаткових ресурсів для виконання компіляції в процесі роботи.

Більш складні реалізації віртуальних машин застосовують динамічну рекомпіляцію, яка включає аналіз виконання програми в реальному часі. Віртуальна машина спостерігає за поведінкою застосунку, визначає найбільш часто використовувані ділянки коду (так звані "гарячі точки", англ. hot spots), зокрема внутрішні цикли, й оптимізує їх, рекомпілюючи вибрані частини байт-коду. Такий підхід забезпечує вищий рівень оптимізації, ніж звичайна статична компіляція, оскільки компілятор має змогу враховувати умови виконання та особливості завантажених класів.

Існує також технологія під назвою ahead-of-time (AOT) компіляція, або статична компіляція, яка передбачає компіляцію Java-програми у машинний код до її виконання, ще на етапі створення. Цей підхід схожий на класичні компілятори та забезпечує хорошу продуктивність порівняно з інтерпретацією. Проте він значно обмежує портативність, оскільки скомпільовану програму можна запускати лише на цільовій платформі, для якої вона була зібрана.

Продуктивність офіційної віртуальної машини Java істотно зросла з моменту випуску її перших версій. Сучасні тести демонструють, що продуктивність JIT-компіляторів у ряді випадків наближається до ефективності традиційних компіляторів, які генерують машинний код. Однак варто враховувати, що ефективність компілятора не завжди прямо впливає на швидкість роботи застосунку. Лише всебічне тестування дозволяє точно оцінити, наскільки швидко й ефективно виконується конкретна програма у певному середовищі.

На відміну від C++, Java вважається більш строго об'єктно-орієнтованою мовою. Усі дії та дані в ній структуровані у вигляді об'єктів і класів. Винятком є

лише примітивні типи даних (наприклад, int, float тощо), які використовуються з метою забезпечення високої продуктивності. Це було свідомим кроком розробників мови, тому Java не вважається повністю об'єктно-орієнтованою, на відміну від, наприклад, мови Smalltalk.

У Java всі об’єкти є нащадками базового класу Object, від якого вони успадковують стандартні методи й властивості. Крім того, на відміну від C++, у якому дозволено множинне успадкування, Java підтримує лише одинарне успадкування класів. Такий підхід унеможливлює конфлікти між методами чи змінними, які можуть виникати при спадкуванні від кількох батьківських класів.

Початковою метою проєктувальників Java було створити мову, яка б стала безпечною, надійною та простою альтернативою C++. Вони проаналізували слабкі сторони C++ — особливості, що найчастіше призводять до помилок, — і виключили їх з Java. Таким чином, Java стала сучасною мовою з мінімізованим ризиком критичних помилок.

Ще однією особливістю Java є наявність системи обробки винятків, яка дозволяє ефективно реагувати на неочікувані ситуації під час виконання програми. Прикладами таких ситуацій є:

- звернення до елементів масиву за межами його розміру або до неініціалізованого елемента;
- спроба читання з недоступного каталогу або неправильної адреси URL;
- введення користувачем некоректних або недопустимих даних.

Завдяки цій системі Java дозволяє будувати більш стабільні програми, здатні коректно реагувати на помилки й уникати аварійного завершення роботи.

Одна з переваг віртуальної машини Java полягає в тому, що помилки (винятки) не спричиняють аварійного завершення роботи всієї програми. Існують спеціальні інструменти, які підключаються до середовища виконання та автоматично реєструють інформацію про винятки, полегшуючи процес налагодження.

Керування пам’яттю у Java виконується автоматично за допомогою збирача сміття. Програміст створює об’єкти, а віртуальна машина сама звільняє пам’ять, коли об’єкт більше не використовується. Якщо об’єкт залишається доступним через посилання, навіть коли він більше не потрібен, може виникнути витік пам’яті. Збирання сміття запускається в будь-який момент, часто під час простою програми, або примусово при нестачі вільної пам’яті, що іноді спричиняє короткочасні затримки. Існують спеціалізовані версії JVM для систем реального часу з оптимізованими збирачами сміття.

На відміну від C/C++, Java не підтримує вказівники, що забезпечує безпечніше керування пам’яттю та дозволяє переміщення об’єктів у пам’яті збирачем сміття.

Java-програми складаються з класів та інтерфейсів. Класи включають змінні, методи та конструктори. Дані можуть бути примітивними (наприклад, int, char, boolean) або посиланнями на об’єкти. Java — мова з суворою статичною типізацією: тип змінної відомий під час компіляції.

Java використовує кодування UTF-16 і підтримує Unicode, що дозволяє використовувати символи з різних мов. Програми можуть записуватись у Unicode, але більшість елементів використовують символи ASCII або їх Unicode-коди.

Типи в Java поділяються на:

- Примітивні: логічний (boolean), цілі (byte, short, int, long), дійсні (float, double), символи (char);
- Посилальні: об'єкти класів, масиви, інтерфейси. Наприклад, рядки є об'єктами класу String.

Java Development Kit (JDK) — це безкоштовний набір інструментів для розробки Java-застосунків, який надається компанією Oracle (раніше Sun Microsystems). JDK містить компілятор javac, стандартні бібліотеки, утиліти, документацію, приклади та виконувальне середовище Java (JRE). Однак до JDK не входить інтегроване середовище розробки (IDE), тому при його використанні

доводиться писати код у текстовому редакторі й запускати програми через командний рядок.

Сучасні IDE, такі як Eclipse, NetBeans, IntelliJ IDEA та Android Studio, використовують JDK для компіляції програм і зазвичай або постачаються з ним, або потребують його попереднього встановлення. Вихідний код JDK, включаючи код компілятора javac, доступний для вільного перегляду.

Open Server — це портативна серверна платформа, орієнтована на веб-розробників. Вона включає набір серверного ПЗ (наприклад, Apache, MySQL, PHP), має зручний інтерфейс і широкі можливості адміністрування. Використовується для розробки, тестування й запуску локальних веб-додатків. Незважаючи на те, що компоненти Open Server спочатку не були створені для спільної роботи, у комплексі вони забезпечують стабільну роботу, подібну до Linux-серверів, але на Windows.

PHP — це популярна скриптова мова програмування з відкритим кодом, призначена для створення динамічних HTML-сторінок на боці сервера. PHP-код виконується на сервері, а клієнт отримує вже готовий HTML. Це підвищує безпеку, хоча й обмежує інтерактивність у порівнянні з JavaScript. PHP дозволяє вбудовувати код прямо в HTML за допомогою тегів <?php ... ?>, а завдяки великій кількості вбудованих функцій дозволяє спростити написання програм. PHP активно підтримується більшістю хостинг-провайдерів і залишається одним із найпоширеніших інструментів у веб-розробці.

Основні особливості PHP

Підтримка багатьох СУБД: PHP має вбудовані бібліотеки для роботи з такими базами даних, як MySQL, PostgreSQL, Oracle, InterBase, Sybase та іншими. Також доступне підключення через стандарт ODBC, що забезпечує сумісність із більшістю СУБД за наявності відповідного драйвера.

Знайомий синтаксис: PHP запозичив елементи з мов C та Perl, тому він інтуїтивно зрозумілий для програмістів з досвідом у різних мовах програмування. Це знижує поріг входу та прискорює навчання.

Відкритий код і безкоштовність: PHP розповсюджується за ліцензією Open Source, що дозволяє кожному вільно користуватись, змінювати й поширювати його код. Глобальна спільнота користувачів активно підтримує проект, пропонуючи поради та вирішення проблем у спільнотах і форумах.

Ефективність у веб-розробці: PHP є інтерпретованою мовою, що дозволяє швидко обробляти сценарії. Хоча він поступається компільованим мовам у швидкості, його продуктивності цілком достатньо для створення повноцінних веб-додатків, особливо невеликих і середніх за масштабом.

Усі PHP-сценарії оформлюються у вигляді окремих блоків коду. Ці блоки можуть бути вбудовані безпосередньо в HTML-код, але обов'язково повинні бути відокремлені спеціальними маркерами. Для розміщення PHP-коду всередині HTML-документа застосовуються теги <?php для відкриття блоку та ?> для його закриття. Альтернативно можна використовувати теги <script language="php"> та </script>, однак рекомендованим варіантом є саме використання <?php ?>, оскільки цей формат забезпечує сумісність із XML-документами.

Існують також альтернативні скорочені стилі вставки PHP-коду — наприклад, <? ?>, однак для їх використання потрібно активувати відповідну опцію у конфігураційному файлі php.ini, встановивши параметр short_open_tag у значення On. Інший варіант — використання тегів у стилі ASP: <% %>. Для цього також потрібне увімкнення відповідної опції в php.ini — змінна asp_tags повинна бути встановлена у On.

PHP виконує лише той код, який розміщений усередині таких обмежувачів, як <?php ?>. Усі елементи, що знаходяться поза межами цих блоків, інтерпретуються як звичайний HTML і виводяться без змін. Таким чином реалізується інтеграція PHP-коду в HTML-сторінки.

Основою програмування в PHP, як і в багатьох інших мовах, є змінні. На відміну від багатьох мов, PHP (як і деякі скриптові мови UNIX) не вимагає

попереднього оголошення змінних. У PHP існує кілька способів оформлення змінних:

- Короткий стиль: змінні записуються у вигляді $variable і є найпоширенішим варіантом. Такий синтаксис використовується для створення змінних у коді. У деяких випадках (наприклад, при ввімкненому параметрі register_globals у php.ini) ці змінні можуть автоматично заповнюватися даними з HTML-форми.
- Середній стиль: використовується для отримання даних з HTML-форм за допомогою суперглобальних масивів: $_POST['variable'], $_GET['variable'], $_REQUEST['variable']. Залежно від способу передачі даних (POST чи GET), застосовується відповідний масив. Ці змінні є глобальними.
- Довгий стиль: передбачає використання застарілих глобальних масивів, таких як $HTTP_POST_VARS['variable'] або $HTTP_GET_VARS['variable']. Це найрозлогіший, але найменш уживаний варіант. Починаючи з версії PHP 5.0.0, ці змінні можуть бути вимкнені, а з версії PHP 6 — повністю недоступні. Використання такого стилю не рекомендується з міркувань сумісності з новими версіями мови.

PHP є динамічно типізованою мовою програмування. Це означає, що немає потреби в явному оголошенні типу змінної — інтерпретатор самостійно визначає її тип на основі контексту. За потреби можна примусово привести змінну до бажаного типу за допомогою відповідних механізмів мови. Це важливо, оскільки залежно від типу значення змінна може поводитися по-різному. PHP також дозволяє змінювати тип змінної під час виконання програми. Слід враховувати, що імена змінних є чутливими до регістру.

До основних типів даних у PHP належать:

- Булеві значення (Boolean): використовуються для представлення істинності (true/false).

- Цілі числа (Integer): можуть бути записані у десятковій, вісімковій або шістнадцятковій формі. Розрядність залежить від платформи, зазвичай — 32 біти. Беззнакові числа не підтримуються.
- Числа з плаваючою комою (Float): допускають запис у десятковій або експоненційній формі.
- Рядки (String): можуть містити текст будь-якої довжини.

Для кожної змінної за потреби можна вказати або змінити тип вручну, що забезпечує гнучкість у роботі з даними.

```
# перший вид надання змінній типу даних
$var1 = true;
$var2 = '1abc';
$var3 = 'abc1';
settype($var1, 'string'); // видасть рядок 1
settype($var2, 'integer'); // видасть число 1
settype($var3, 'bool'); // видасть true

# другий тип надання змінній типу даних
$var1 = (int);
$var1 = '1abc';
echo $var1; // видасть число 1
```

Рядки у PHP поділяються на два типи: ті, що аналізуються інтерпретатором (з підстановкою змінних та обробкою керівних символів), і ті, що не підлягають аналізу. Кожен символ рядка має значення від 0 до 255, однак підтримується і робота з багатобайтовими символами. До символів рядка можна звертатися як до елементів масиву.

PHP має велику кількість функцій для обробки рядків, зокрема пошуку та заміни тексту. Реалізована підтримка регулярних виразів двох типів — Perl-сумісних та POSIX-сумісних, які різняться синтаксисом та поведінкою.

Оператори виконують дії над операндами і бувають унарними, бінарними та тернарними. Їм властиві пріоритет і асоціативність. Булеві оператори порівняння поділяються на ті, що враховують тип, і ті, що не враховують (з примусовим приведенням). Числа в PHP завжди округлюються вниз. Є також оператори виконання, обробки помилок і перевірки типу.

Функції є контейнерами для коду та можуть містити інші функції чи класи. Це дозволяє створювати умовні функції, однак у такому разі виклик функції можливий лише після її оголошення. Перевизначення функцій не підтримується. Функція може повертати лише одне значення, але це обмеження можна обійти, використовуючи масиви або посилання.

У PHP реалізована підтримка посилань, що дозволяє створювати кілька псевдонімів на один і той самий сегмент пам’яті. Після завершення роботи сценарію або видалення всіх посилань пам’ять звільняється автоматично. PHP не підтримує перевантаження функцій, натомість використовує динамічну обробку аргументів — кількість та значення параметрів визначається під час виклику. Можна задавати значення аргументів за замовчуванням.

Збірка сміття очищує пам’ять автоматично після виконання сценаріїв, хоча доступні і функції ручного очищення. Програми в PHP можна будувати модульно: код розділяється на частини, які підключаються за потреби. Підключені файли можуть повертати значення, а сам процес підключення може бути умовним.

HTML (від англ. HyperText Markup Language — мова розмітки гіпертекстових документів) — це стандартна мова для створення веб-сторінок у мережі Інтернет. Переважна більшість сучасних веб-сторінок створюється з використанням HTML або його розширення XHTML. HTML-документ обробляється веб-браузером і відображається на екрані у формі, зручній для сприйняття користувачем.

HTML є похідною мовою від SGML, успадкувавши від неї концепцію структурної розмітки тексту та механізм визначення типу документа. Хоча

HTML є формальною комп’ютерною мовою, вона не вважається мовою програмування.

Разом із HTML використовуються каскадні таблиці стилів (CSS) та вбудовані скрипти — ці три технології складають основу сучасної веб-розробки.

HTML забезпечує такі можливості:

- створення структурованого документа через позначення логічних частин тексту — заголовків, абзаців, списків, таблиць, цитат тощо;
- отримання інформації з Інтернету за допомогою гіперпосилань;
- створення інтерактивних форм;
- вставляння мультимедійних елементів — зображень, аудіо, відео та інших об’єктів — у текст документа.

CSS (англ. Cascading Style Sheets, або каскадні таблиці стилів) — це спеціалізована мова, яка призначена для оформлення вигляду веб-сторінок, створених на основі мов розмітки, таких як HTML або XHTML. Хоча CSS найчастіше застосовується саме з цими мовами, вона також може використовуватись для форматування інших типів XML-документів.

Розробкою та підтримкою специфікацій CSS займається Консорціум Всесвітньої павутини (W3C). CSS існує в різних версіях або рівнях, які позначаються як CSS1, CSS2, CSS3 тощо. Кожен наступний рівень базується на попередньому, розширюючи або вдосконалюючи функціональність. Також існують так звані профілі CSS — набори правил, адаптовані під конкретні типи пристроїв або інтерфейсів, наприклад, для мобільних пристроїв або принтерів.

CSS-презентація (каскадна або блочна верстка) поступово замінила застарілу табличну верстку, яка раніше використовувалась для побудови структури сторінки. Основною перевагою сучасного підходу є чітке відокремлення вмісту сторінки від її зовнішнього вигляду.

CSS дає змогу розробникам та користувачам задавати кольори, шрифти, розташування елементів і багато інших параметрів візуального оформлення. Однією з головних переваг є можливість змінювати вигляд документа, не

змінюючи сам вміст, що значно полегшує розробку та обслуговування веб-сайтів.

Таке відокремлення змісту й стилю:

- покращує доступність і зручність сприйняття контенту;
- дає більше контролю над зовнішнім виглядом на різних пристроях;
- дозволяє зробити контент структурованішим і легшим для підтримки;
- зменшує дублювання коду.

CSS також забезпечує адаптивність веб-контенту під різні пристрої: комп'ютери, смартфони, планшети, принтери, телевізори, а також спеціальні засоби — наприклад, голосові браузери чи пристрої для людей із вадами зору, включаючи підтримку шрифту Брайля.

SQL (від англійського Structured Query Language — мова структурованих запитів) — це декларативна мова програмування, призначена для взаємодії користувачів із базами даних. Вона широко використовується для формування запитів, оновлення даних, керування реляційними базами даних, створення та зміни структури бази, а також для реалізації систем контролю доступу.

Окрім базових можливостей для роботи з даними, стандарт SQL також містить засоби для опису змін, перевірки правильності даних і забезпечення їх безпеки. SQL є діалоговою мовою, яка дозволяє здійснювати запити до бази даних, вносити зміни до даних і керувати базами даних загалом. Більшість сучасних СКБД реалізує підтримку SQL із розширеннями, які виходять за рамки базового стандарту.

Основу мови SQL складають команди, що дають змогу виконувати основні операції з даними — пошук, вставлення, оновлення та видалення. Крім цього, SQL містить інтерфейс CLI (Call Level Interface), який забезпечує можливість віддаленого доступу до баз даних і керування ними.

Історично SQL виникла в лабораторіях IBM на початку 1970-х років. Перша версія мала назву SEQUEL і була створена для роботи з реляційною базою

даних IBM System R. Згодом мову було стандартизовано Американським національним інститутом стандартів (ANSI) у 1986 році. Хоча початково SQL була мовою запитів і управління даними, з часом постачальники СКБД розширили її функціональність, додавши елементи процедурного програмування, команди управління потоком виконання та інші розширення. Із появою стандарту SQL:1999 ці доповнення були офіційно інтегровані в мову під назвою Persistent Stored Modules (SQL/PSM).

MySQL — це вільна система керування реляційними базами даних із відкритим вихідним кодом. Вона була розроблена компанією TcX із метою забезпечення високої швидкості обробки великих обсягів даних. Спочатку MySQL була схожа на іншу систему — mSQL, однак з часом вона значно розширила свої можливості й на сьогодні є однією з найпопулярніших СКБД у світі.

MySQL часто застосовується для створення динамічних веб-сторінок, завдяки хорошій сумісності з різними мовами програмування. Це компактний, багатопотоковий сервер баз даних, який відзначається високою продуктивністю, надійністю та простотою у використанні.

Серед основних можливостей MySQL варто відзначити:

- простоту встановлення та експлуатації;
- підтримку необмеженої кількості одночасних користувачів;
- здатність працювати з таблицями, що містять до 50 мільйонів рядків;
- високу швидкість виконання SQL-команд;
- наявність зручної та ефективної системи безпеки.

MySQL є особливо ефективним рішенням для малих і середніх проєктів. Вихідні коди сервера можна скомпілювати на багатьох операційних системах, а повний потенціал системи розкривається в середовищі UNIX, де підтримка багатопоточності суттєво підвищує загальну продуктивність.

phpMyAdmin — це веб-застосунок із відкритим вихідним кодом, написаний мовою PHP, який надає графічний інтерфейс для адміністрування

серверів баз даних MySQL. Завдяки phpMyAdmin, адміністрування можна здійснювати безпосередньо через веб-браузер: виконувати SQL-запити, переглядати й редагувати таблиці, змінювати структуру бази та керувати нею.

Цей інструмент користується великою популярністю серед веб-розробників, оскільки дозволяє працювати з базами даних без потреби вручну вводити SQL-команди або встановлювати додаткове програмне забезпечення. Доступ до phpMyAdmin можливий з будь-якого пристрою, підключеного до Інтернету.

phpMyAdmin розповсюджується на умовах ліцензії GNU General Public License, що дозволяє іншим розробникам вільно інтегрувати його до своїх проєктів, як це зроблено, наприклад, у середовищах XAMPP або Denwer.

Засновником phpMyAdmin є німецький розробник Тобіас Ратшіллер (Tobias Ratschiller). Основою для створення цього інструменту послужив інший веб-застосунок — MySQL-Webadmin, який з’явився ще у 1997 році. Проєкт phpMyAdmin локалізовано більш ніж на 50 мов світу, що сприяє його широкому використанню в усьому світі.

3.3 Алгоритмізація та розробка програмного забезпечення

Розроблена інформаційна система складається з двох основних компонентів: прикладного програмного забезпечення та інформаційного забезпечення.

Прикладне програмне забезпечення має тримодульну структуру, кожен з модулів виконує окремі функції в межах загальної архітектури системи.

Перший модуль відповідає за реалізацію користувацького інтерфейсу. Він забезпечує взаємодію користувача із системою, надаючи доступ до всіх функціональних можливостей, які необхідні для виконання завдань. За допомогою цього модуля користувач може інтуїтивно та зручно здійснювати навігацію, вводити дані, переглядати результати тощо.

Другий модуль забезпечує зв’язок із базою даних. Він виконує низку ключових функцій:

- підключення до бази даних із використанням логіна та пароля, які вводить користувач;
- здійснення введення нових даних, а також редагування вже наявної інформації;
- пошук потрібних даних за заданими критеріями та виведення результатів у зручному форматі.

Третій модуль, який часто називають бекендом (back end), відповідає за реалізацію логіки обробки запитів користувача. Він виконує обробку даних, отриманих від користувача, здійснює необхідні операції відповідно до внутрішніх алгоритмів системи, та формує відповіді, які повертаються у структурованому та коректному вигляді.

Завдяки такому розподілу функцій між модулями забезпечується чітке розмежування обов’язків, що сприяє покращенню гнучкості, масштабованості та підтримуваності інформаційної системи.

1.3.1 Модуль управління вимірюванням даних моніторингу

Для програмування контролера Arduino Mega 2560 використовується офіційне середовище розробки Arduino версії 1.6.8, а мова програмування — C++.

Алгоритм роботи контролера наведений на рисунку 3.13. Відповідно до цього алгоритму, система здійснює перевірку стану порту. Якщо порт виявляється вільним, програма передає дані з підключеного датчика через COM-порт.

Програма починається з підключення бібліотеки для роботи з датчиком DHT11, що забезпечує коректну взаємодію з цим датчиком, який підключається до плати контролера.

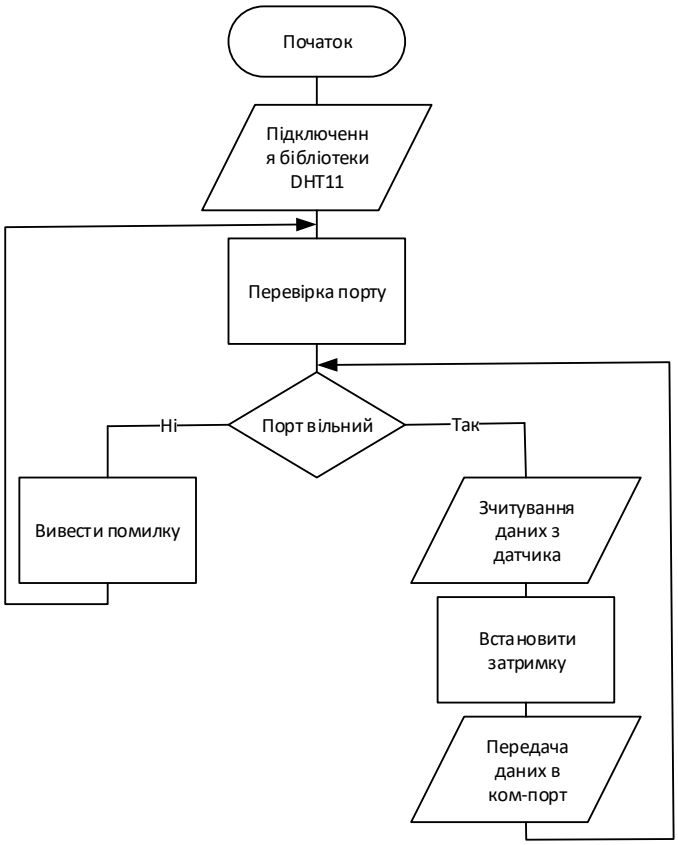


Рис.3.13 Алгоритм роботи програми ардуіно

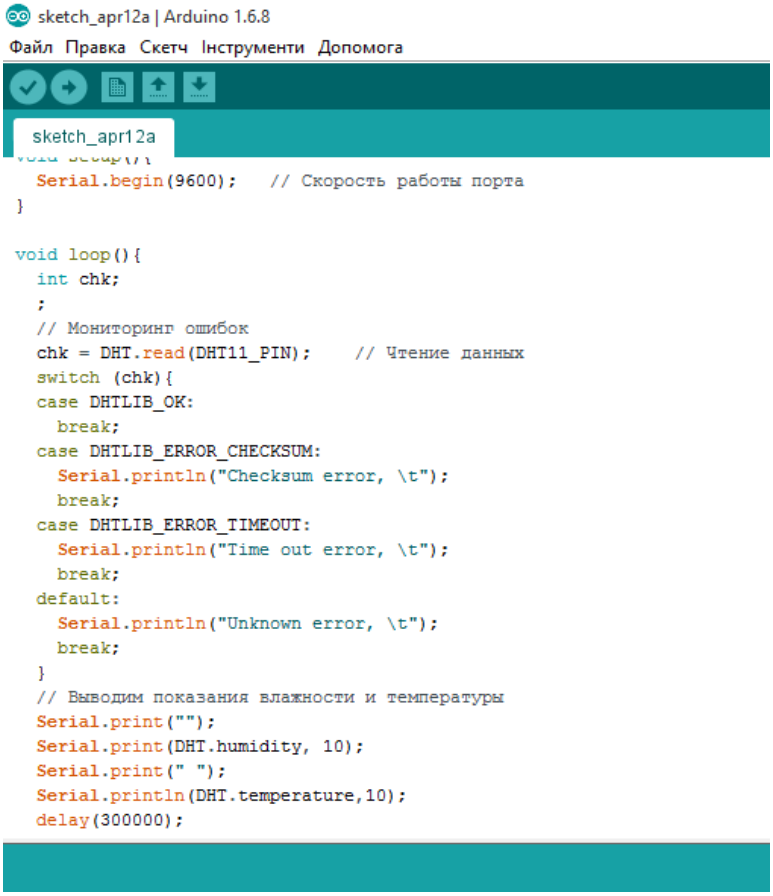


Рис. 3.14 Робоче вікно середовища програмування Arduino 1.6.8

1.3.2 Модуль зчитування даних в базу даних

Зчитування даних із COM-порту та запис їх у базу даних реалізується за допомогою різних технологій. Зокрема, для отримання даних із COM-порту плати Arduino вони зберігаються у файл формату TXT відповідно до алгоритму, показаного на рисунку 3.15.

Код програми написаний на мові Java. Фрагмент коду наведено на відповідному рисунку. Програма розроблена в середовищі NetBeans IDE версії 8.1. Для коректної роботи коду використовуються спеціалізовані бібліотеки, які забезпечують зчитування та запис даних через COM-порт.

gnu.io.CommPort;

import gnu.io.CommPortIdentifier;

import gnu.io.NoSuchPortException;

gnu.io.PortInUseException;

gnu.io.SerialPort;

gnu.io.SerialPortEventListener;

gnu.io.UnsupportedCommOperationException;

java.io.BufferedReader;

java.io.FileInputStream;

java.io.FileOutputStream;

java.io.IOException;

java.io.InputStream;

java.io.InputStreamReader;

java.io.OutputStream;

java.io.OutputStreamWriter;

java.io.Writer;

java.util.Date;

java.util.TooManyListenersException.

Зокрема, ми використали клас CommPortIdentifier, у якому можуть бути

змінні і методи класу, наведені нижче.

Змінні класу:

- PORT_SERIAL public static final int PORT_SERIAL RS-232 послідовний порт
- PORT_PARALLEL public static final int PORT_PARALLEL IEEE 1284 паралельний порт.

Методи Класу:

- public static Enumeration getPortIdentifiers (). Метод повертає об'єкт типу enumeration, який містить об'єкти типу CommPortIdentifier для кожного порту системи.
- public String getName (). Повертає ім'я порту. Наприклад, "COM1" і "COM2" на PC;
- public int getPortType (). Метод повертає тип порту PORT_SERIAL або PORT_PARALLEL.
- public synchronized CommPort open (String appname, int timeout) throws PortInUseException. Відкриває порт. Повертає об'єкт типу CommPort. параметри: appname - Ім'я програми викликає даний метод. (Довільний рядок); timeout - час в мілісекундах, протягом якого, блокується доступ до порту для його відкриття.

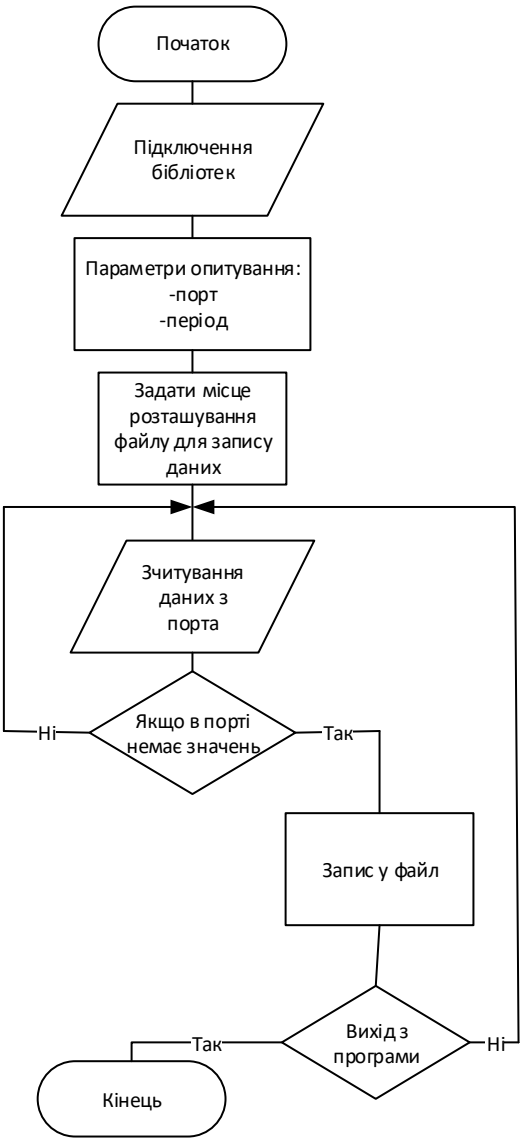
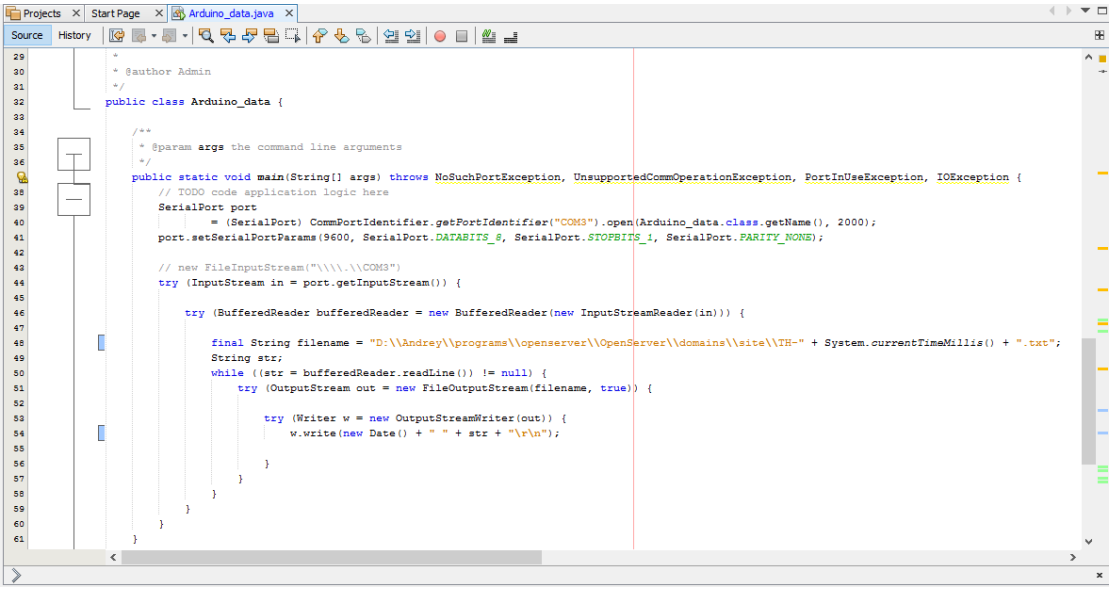


Рис. 3.15. Алгоритмізація процесу зчитування даних з COM порту



```
 *
 * @author Admin
 */
public class Arduino_data {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws NoSuchPortException, UnsupportedCommOperationException, PortInUseException, IOException {
        // TODO code application logic here
        SerialPort port
                = (SerialPort) CommPortIdentifier.getPortIdentifier("COM3").open(Arduino_data.class.getName(), 2000);
        port.setSerialPortParams(9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

        // new FileInputStream("\\\\.\\COM3")
        try (InputStream in = port.getInputStream()) {

            try (BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(in))) {

                final String filename = "D:\\Andrey\\programs\\openserver\\OpenServer\\domains\\site\\TH-" + System.currentTimeMillis() + ".txt";
                String str;
                while ((str = bufferedReader.readLine()) != null) {
                    try (OutputStream out = new FileOutputStream(filename, true)) {

                        try (Writer w = new OutputStreamWriter(out)) {
                            w.write(new Date() + " " + str + "\r\n");

                        }
                    }
                }
            }
        }
    }
```

Рис. 3.15 Фрагмент коду програми зчитування даних з COM порту

Результатом роботи коду буде текстовий файл, рядками якого будуть текстові дані з показниками часу, температури і вологості.

1.3.3 Програмування веб-додатку

Запис даних в MySQL Server здійснюється відповідно до алгоритму поданого на Рис. 3.16.

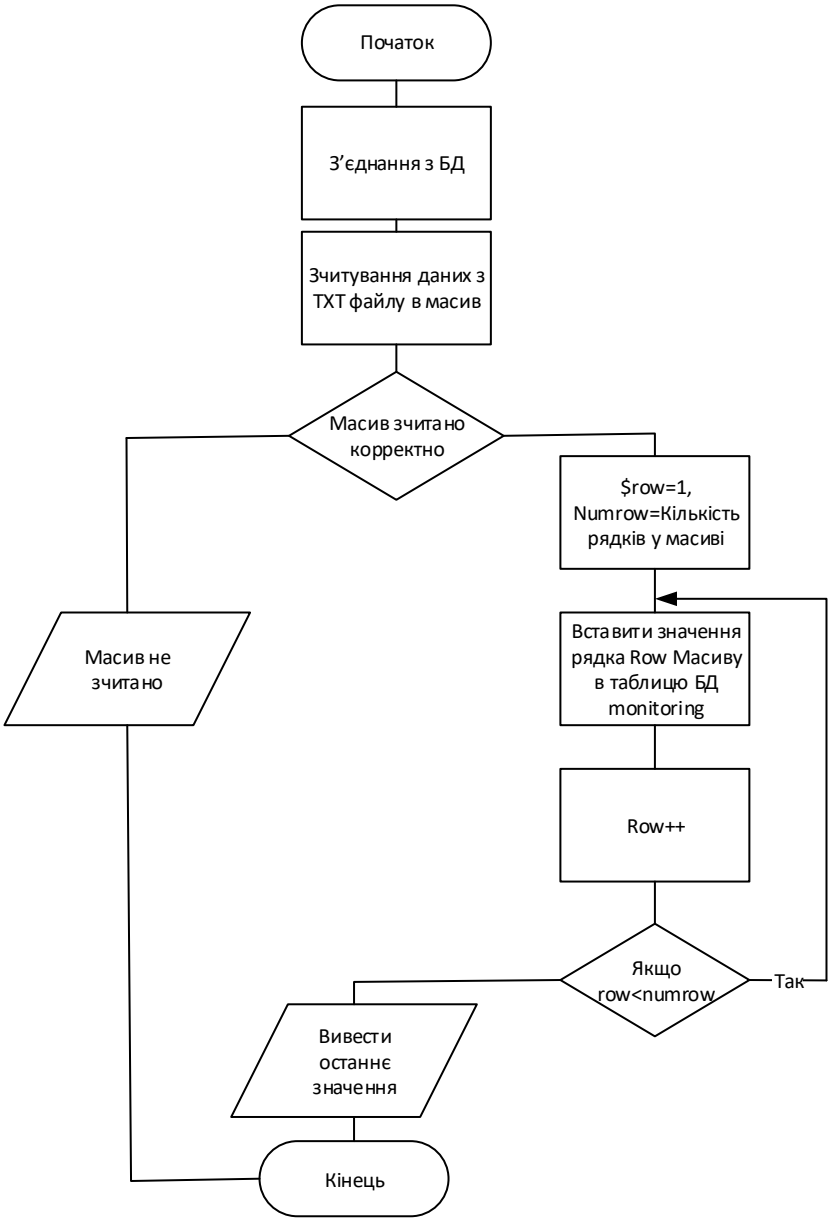


Рис. 3.16 Алгоритм запису даних до БД

Дані з файлу формату TXT зчитуються у масив, після чого інформація з кожного рядка переноситься у відповідні поля таблиці з назвою «monitoring».

Інтерфейс є дуже важливою частиною програми. Можна навіть сказати, що

це одна з найважливіших складових, оскільки більшість користувачів при виборі програмного продукту звертають насамперед увагу саме на якість інтерфейсу, і лише потім оцінюють переваги функціональності.

Весь користувацький інтерфейс, зокрема веб-сайт, було створено за допомогою мови гіпертекстової розмітки — HTML. За допомогою мови каскадних таблиць стилів CSS елементам інтерфейсу надається унікальний зовнішній вигляд, а також впорядковується розташування кнопок, текстових полів і зображень на сторінці так, щоб інтерфейс був інтуїтивно зрозумілим та зручним для користувача.

Для програмування веб-сайту застосовувалась мова PHP у середовищі розробки PHP Expert Editor версії 4.3. Зокрема, на мові PHP реалізовано операції зчитування та запису в реальному часі параметрів мікроклімату, отриманих із датчика, у базу даних, а також відображення останнього актуального значення на сторінці «Показники» веб-сайту. Крім того, реалізовано побудову графіків вологості та температури за вибраний період часу, порівняння цих даних з еталонними показниками та розрахунок приросту урожаю за заданою моделлю. Фрагмент коду, що виконує розрахунок приросту урожаю, разом із результатом його роботи наведено на рисунку 3.17.

```
Приріст урожаю на одиницю продукції при даній температурі: 151.1 грам за добу
Еталонний приріст урожаю : 152 грам на добу при температурі 25°C
Втрата приросту продукції з однієї рослини : 0.9 грам на добу
```

```
117
118  id=base_connect();
119  y = "SELECT round(AVG(temperature),1 )  AS sertemp FROM monitoring WHERE hour(data) BETWEEN 9 AND 18";
120  lts=  mysql_query ($query,$con_id);
121  =mysql_fetch_object($results);
122  sertemp=$row->sertemp;
123
124
125  k1=-158.38;
126  k2=24.97;
127  k3=-0.5037;
128  $y=$k1+$k2*$sertemp+$k3*$sertemp*$sertemp;
129  $y=round($y,1);
130  $r=152-$y;
131  "Приріст урожаю на одиницю продукції при даній температурі: ". $y ." грам за добу" . "<br> Еталонний приріст урожаю : 152 грам на
132  "<br> Втрата приросту продукції з однієї рослини : " .$r . " грам на добу";
```

Рис. 3.17 Фрагмент коду розрахунку приросту урожаю та результат його виконання

4.РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Тестування системи є надзвичайно важливим етапом, адже система представляє собою веб-сайт, яким користуватиметься велика кількість людей, що використовують різні операційні системи та браузери. Основне завдання тестування полягає у перевірці коректної роботи сайту та правильного відображення в різних браузерах і на різних ОС. Для цього було застосовано веб-сервіс browsershots.org, який дає можливість переглянути, як виглядає сайт у вибраних операційних системах та браузерах, включно з різними версіями браузерів.

Під час розробки використовувалися кросбраузерні стилі та компоненти, а також динамічний дизайн, що базується на відсоткових стилях. Завдяки цьому система легко пройшла тестування, і розробники отримали впевненість, що будь-який користувач побачить сайт саме так, як це було задумано і реалізовано.

Для початку роботи з системою користувачу потрібно підключити плату Arduino Mega 2560 через USB-порт до персонального комп’ютера. Після цього необхідно запустити програму для зчитування даних з USB-порту, як показано на рисунку 4.1.

Найпопулярнішим веб-браузером на сьогоднішній день є Google Chrome, тому на його прикладі і було проведено тестування в першу чергу. З головної сторінки «Управління» користувач може обрати культуру і період часу зчитування даних температури та вологості з датчика. На рис.4.2 зображено головну сторінку веб-сайту звідки і відбувається перехід до основних розділів сайту.

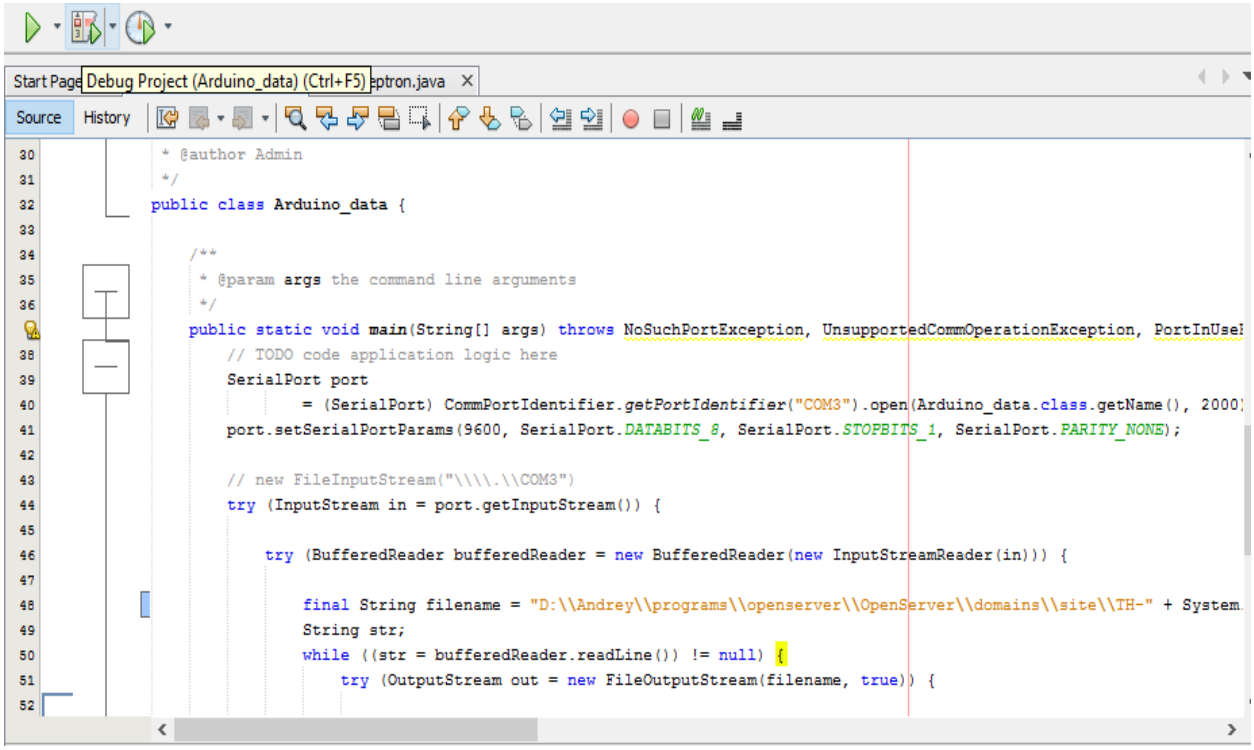


Рис.4.1 Програма зчитування даних з датчика.

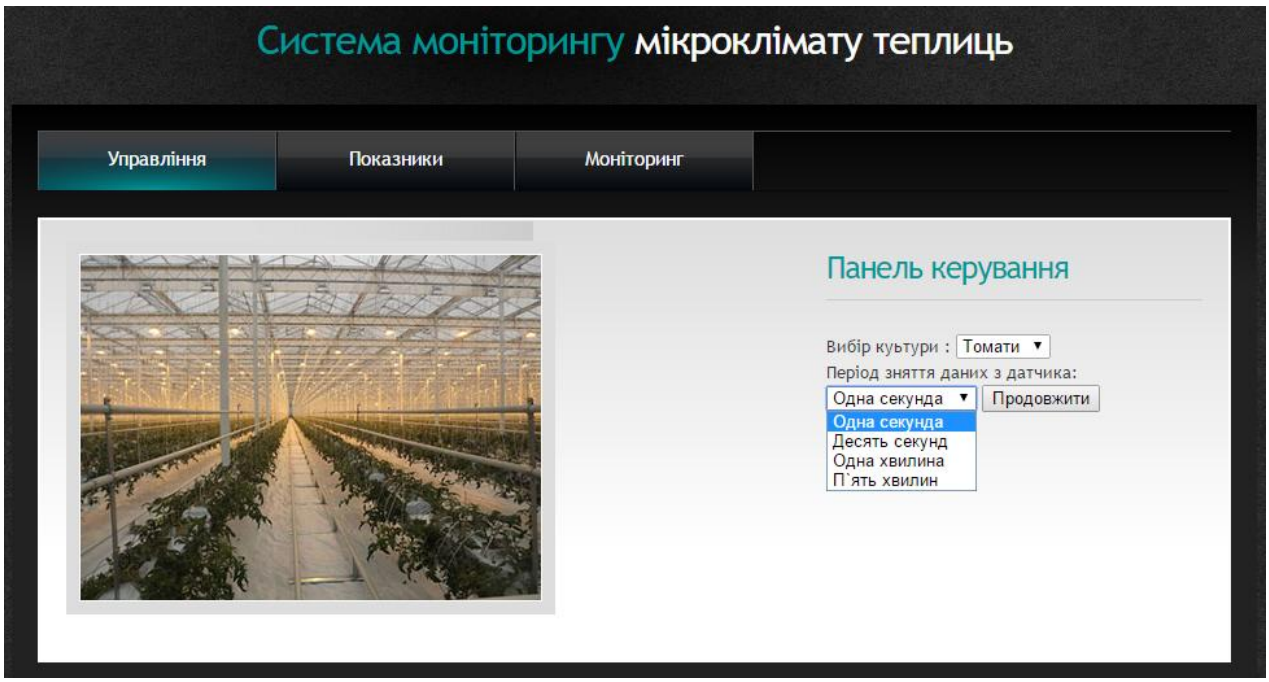


Рис.4.2 Сторінка «Упрвління»

Далі користувач обирає вікно «Показники» , де спостерігає за вхідними параметрами і реакцією системи на їх зміну. Ця сторінка зображена на рис.4.3.

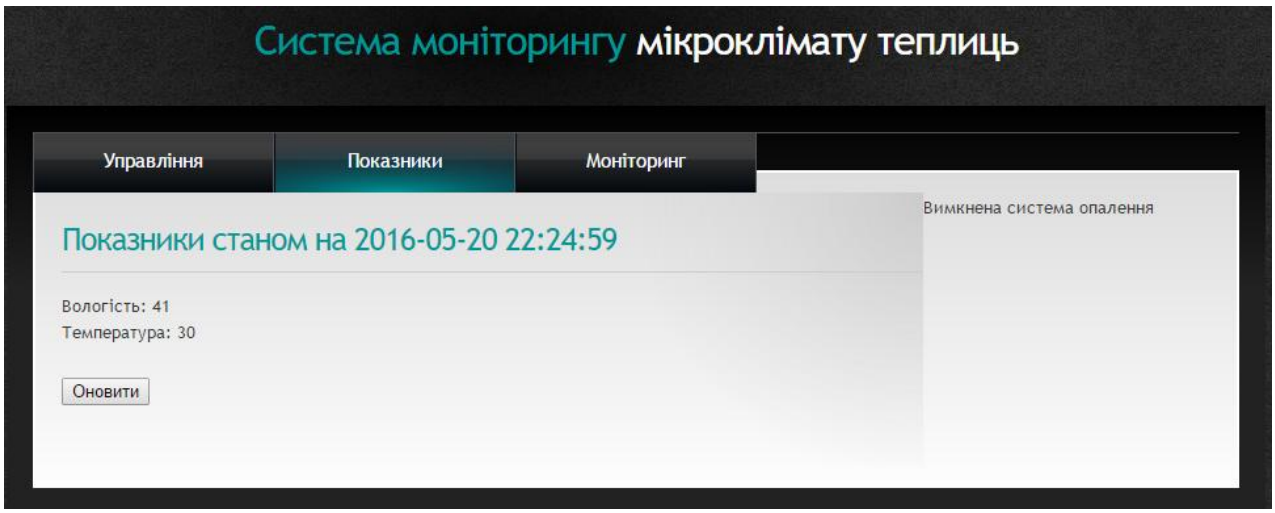


Рис.4.3 Вікно «Показники».

Для перегляду змін показників параметрів мікроклімату теплиці користувач переходить на вкладку «Моніторинг» що зображена на рис.4.4.

Рис.4.4 Вікно «Моніторинг».

На даній вкладці користувач має можливість обирати дату та період для побудови графіку і спостерігати за його динамікою. Також користувач може переглянути приріст грам урожаю на одиницю продукції і порівняти його з

еталонним значенням для культури. яку він обрав у вікні «Управління».

Сторінка з формую для реєстрації зображена на рис.4.5.

Рис.4.5 Сторінка реєстрації користувача

Зареєстрований та авторизований користувач отримує безпосередньо роль «Користувач» і тепер має змогу продивлятися лекції, тести та власні успіхи. Ці сторінки зображені на рисунках

ВИСНОВКИ

Завдяки використанню сучасних технологій та підходів розробка інформаційних систем стає простою, зрозумілою та послідовною.

Під час виконання роботи було освоєно системний підхід і методології, що допомагають розглядати проблеми та проектувати системи для їх ефективного вирішення. Застосування таких підходів надає роботі рис проекту, зокрема чітке формулювання цілей, завдань і визначення проблеми.

Окрім того, методи SADT і UML суттєво спрощують розуміння всієї системи — як у цілому, так і по її окремих частинах — завдяки чіткій візуалізації компонентів системи.

Для регулювання температури повітря та ґрунту в теплиці було обрано мікропроцесорну плату Arduino Mega 2560. В цій платі встановлено модуль інтерфейсу RS-485, який дає змогу керувати пристроєм за допомогою комп’ютера, а також отримувати інформацію про стан регулюючої системи безпосередньо на ПК.

У процесі роботи успішно розроблено інформаційно-управляючу систему електронного навчання. Розробка вважається успішною за такими критеріями:

- система повністю функціонує без логічних помилок і критичних збоїв;
- відповідає всім визначеним вимогам;
- ефективно виконує поставлені задачі;
- зручна у користуванні та управлінні;
- є універсальною та легко масштабованою.

Слід також підкреслити, що подібні системи за своїм призначенням вже користуються великим попитом і мають значний потенціал, при цьому продовжуючи активно розвиватися.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Усаков Д. Ю. електронний посібник “Як створити свій сайт” [Електронний ресурс] http://myrusakov.ru.
2. Адміністрування MySQL для початківців, основи SQL [Електронний ресурс]. Безкоштовний електронний курс для студентів http://www.intuit.ru/studies/courses/989/165/info.
3. Довідник [Електронний ресурс] вирішення проблемних ситуацій з БД та php частиною веб-сайту http://www.cyberforum.ru/database/
4. Історія та призначення ER-діаграм [Електронний ресурс] https://ru.wikipedia.org/wiki/ER-модель_данных
5. Никсон. Р. Создаем динамические веб приложения на Java — Питер, 2009 г, —297с.
6. Лендер С.Adobe Photoshop CS с нуля! / Лендер С., Нечаев И — Лучшие Книги, 2005 г, —315с.
7. Линн Бейли Изучаем PhP и MySQL : підруч. [для розробників веб-контенту] / Майкл Моррисон — Канада : 2010. — 800с
8. Мартиненко І. І., Лисенко В. П.”Проэктирование систем автоматики ” Москва, “Агропромиздат” 1990 г.
10. Мартиненко І.І., Лисенко В.П., Тищенко Л.П., Болбот І.М., Олійник П.В. Проектування систем електрифікації та автоматизації АПК Київ, Інтас. - 2008. – 305 с.