Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно–інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом

Виконав: студент IV курсу, групи СН–41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Багрій Д.П.
(підпис) (прізвище та ініціали)

Керівник _____ Марценко С.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Яцишин В.В.
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно–інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_14_» _травня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Багрію Дмитру Петровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом

Керівник роботи Марценко Сергій Володимирович, к.т.н. доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*14*» *травня* 2026 року № *4/9–239*

2. Термін подання студентом завершеної роботи 22 червня 2026р.

3. Вихідні дані до роботи Розробка програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналітична частина 1.1 Аналіз предметної області моніторингу IoT-пристроїв 1.2. Життєвий цикл IoT-пристроїв та його основні етапи. 1.3. Огляд існуючих програмних рішень для моніторингу IoT. 1.4. Аналіз протоколів і технологій обміну даними в IoT-системах. 1.5. Постановка задачі та формування вимог до програмного засобу. 1.6 Висновок до першого розділу. 2 Проектна частина. Проєктування програмного засобу 2.1. Проектування архітектури програмного засобу. 2.2. Проектування інформаційної моделі. та структури бази даних. 2.3. Розробка алгоритмів моніторингу стану IoT-пристроїв. 2.4. Проектування механізму управління життєвим циклом пристроїв. 2.5. Проектування користувацького інтерфейсу системи . 2.6 Висновок до другого розділу 3. Практична частина. Практичні аспекти реалізації програмнаого засобу. 3.1. Вибір інструментальних засобів розробки. 3.2. Реалізація серверної частини програмного засобу. 3.3. Реалізація модуля моніторингу та обробки телеметрії. 3.4. Реалізація користувацького інтерфейсу. 3.5. Тестування програмного засобу та аналіз результатів 3.6 Висновки до третього розділу 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки. 5. Перелік графічного матеріалу
1 Титульна сторінка. 2 Тема, Мета, Об'єкт, Предмет дослідження. 3 Завдання дослідження. 4 Актуальність дослідження. 5 Узагальнена структура IoT-системи моніторингу пристроїв 6 Архітектура програмного засобу 7 ER-діаграма бази даних 8 Блок-схема алгоритму 9 Діаграма станів життєвого циклу IoT-пристрою 10 Макет головної панелі моніторингу 11 Структура серверної частини програмного засобу 12 Послідовність обробки телеметричних даних 13 Інтерфейс сторінки моніторингу IoT-пристроїв 14 Інтерфейс картки окремого IoT-пристрою 15 Приклад відображення аварійного стану IoT-пристрою 16 Висновки. 17 Завершальний.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О. Я. к.т.н., доц. кафедри МТ	02.06.2026	05.06.2026

7. Дата видачі завдання 14 травня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	14.05.2026	*Виконано*
2.	Підбір джерел	14.05.2026-15.05.2026	*Виконано*
3.	Виконання дослідження згідно мети кваліфікаційної	16.05.2026-22.05.2026	*Виконано*
	роботи		
4.	Оформлення розділу «Аналітична частина»	23.05.2026-01.06.2026	*Виконано*
5.	Оформлення розділу «Проектна частина.	23.05.2026-01.06.2026	*Виконано*
	Проєктування програмного засобу»		
6.	Оформлення розділу «Практична частина.	23.05.2026-01.06.2026	*Виконано*
	Практичні аспекти реалізації програмнаого засобу»		
7.	Виконання завдання до підрозділу «Безпека	02.06.2026-05.06.2026	*Виконано*
	життєдіяльності»		
8.	Виконання завдання до підрозділу «Основи охорони	02.06.2026-05.06.2026	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	06.06.2026-10.06.2026	*Виконано*
10.	Нормоконтроль	11.06.2026-13.06.2026	*Виконано*
11.	Перевірка на плагіат	16.06.2026	*Виконано*
12.	Попередній захист кваліфікаційної роботи	17.06.2026	*Виконано*
13.	Захист кваліфікаційної роботи	22.06.2026	

Студент _____ Багрій Д.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марценко С.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом // Кваліфікаційна робота освітнього рівня «Бакалавр» // Багрій Дмитро Петрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно–інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН–41 // Тернопіль, 2026 // С.51 , рис. – 11, табл. – 10, кресл. – 17, додат. – 2, бібліогр. – 25.

Ключові слова: інтернет речей, моніторинг, телеметрія, життєвий цикл, вебзастосунок, база даних, програмний засіб.

Кваліфікаційна робота присвячена розробці програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом.

У першій частині роботи проаналізовано предметну область моніторингу IoT-пристроїв, їх життєвий цикл, наявні програмні рішення та протоколи обміну даними.

У другій частині спроектовано архітектуру програмного засобу, структуру бази даних, алгоритм моніторингу, механізм управління життєвим циклом і користувацький інтерфейс.

У третій частині обґрунтовано вибір технологій, описано реалізацію серверної частини, модуля обробки телеметрії, вебінтерфейсу та проведено тестування системи.

У четвертій частині розглянуто питання безпеки життєдіяльності та охорони праці під час розробки й експлуатації програмного засобу.

Об'єкт дослідження — процеси моніторингу, обліку та супроводу IoT-пристроїв у розподіленій інформаційній системі.

Предмет дослідження — методи, алгоритми та програмні засоби збору телеметрії, контролю станів, обробки подій і управління життєвим циклом IoT-пристроїв.

ANNOTATION

Development of a Software Tool for Monitoring IoT Device Status and Managing Their Lifecycle // Qualification work of the educational level «Bachelor» // Bahrii Dmytro Petrovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN–41 // Ternopil, 2026 // P.51, fig. – 11, tabl. – 10, chair. – 17, annexes. – 2, references – 25.

Keywords: internet of things, monitoring, telemetry, lifecycle, web application, database, software tool.

The qualification work is devoted to the development of a decision support system based on analytical indicators.

The first section of the thesis examines the theoretical foundations of decision support systems, analyses the subject area and analytical indicators, and formulates the system requirements.

The second section outlines the design of the system architecture, the database information model, the indicator processing algorithm, the recommendation module and the user interface.

In the third section justifies the choice of software tools, describes the implementation of the system's main modules, conducts testing and analyses the results of its operation.

The section 'Life safety, basics of occupational safety' highlights the specifics of safe working practices as well as safety requirements for working with computer systems.

The object of the study is the processes of monitoring, logging and managing IoT devices within a distributed information system.

The subject of the study is the methods, algorithms and software tools for collecting telemetry data, monitoring device statuses, processing events and managing the lifecycle of IoT devices.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API — Application Programming Interface, програмний інтерфейс прикладного програмування.

CoAP — Constrained Application Protocol, протокол прикладного рівня для пристроїв з обмеженими ресурсами.

CRUD — Create, Read, Update, Delete, базові операції створення, читання, оновлення та видалення даних.

DB — Database, база даних.

HTTP — Hypertext Transfer Protocol, протокол передавання гіпертексту.

HTTPS — захищена версія HTTP з використанням шифрування.

IoT — Internet of Things, Інтернет речей.

JSON — JavaScript Object Notation, текстовий формат обміну структурованими даними.

JWT — JSON Web Token, формат токена для авторизації користувачів.

MQTT — Message Queuing Telemetry Transport, легкий протокол обміну повідомленнями для IoT.

REST — Representational State Transfer, архітектурний стиль побудови API.

SQL — Structured Query Language, мова структурованих запитів до баз даних.

UI — User Interface, користувацький інтерфейс.

WebSocket — протокол двостороннього обміну даними між клієнтом і сервером у реальному часі.

ЗМІСТ

ВСТУП

Актуальність теми. Стрімке поширення технологій Internet of Things зумовило активне впровадження мережевих пристроїв у промислову автоматизацію, енергетику, логістику, транспорт, системи «розумного міста», побутову автоматизацію та інші сфери, де необхідне постійне збирання, передавання й аналіз даних. IoT-пристрої забезпечують отримання інформації з фізичного середовища, виконання керуючих дій, передавання телеметрії та взаємодію з інформаційними системами через мережеві протоколи. Водночас зростання кількості таких пристроїв у розподілених середовищах ускладнює їх облік, контроль працездатності, діагностику несправностей і супровід протягом усього періоду експлуатації [1], [2].

Актуальність теми бакалаврської кваліфікаційної роботи визначається потребою у створенні програмних засобів, які дають змогу централізовано контролювати стан IoT-пристроїв, своєчасно виявляти відхилення в їх роботі та забезпечувати керованість інфраструктури. У багатьох практичних випадках IoT-пристрої працюють автономно, мають обмежені обчислювальні ресурси, залежать від якості мережевого з’єднання, можуть втрачати зв’язок із сервером, передавати некоректні дані або переходити в аварійні стани. За відсутності систематичного моніторингу такі ситуації можуть призводити до втрати актуальності даних, несвоєчасного реагування на несправності, порушення технологічних процесів або зниження надійності всієї інформаційної системи.

Особливе значення має не лише поточний моніторинг IoT-пристроїв, а й управління їх життєвим циклом. Кожний пристрій проходить низку етапів: реєстрацію, ідентифікацію, налаштування, введення в експлуатацію, активну роботу, діагностику, оновлення, технічне обслуговування, деактивацію та виведення з експлуатації. Якщо ці етапи не відображаються в єдиній системі, ускладнюється контроль актуального стану пристроїв, історії їх змін, причин виникнення помилок і подальших дій адміністратора. Тому програмний засіб для моніторингу IoT-пристроїв має поєднувати функції приймання телеметрії,

відображення станів, ведення журналу подій, формування сповіщень і підтримки переходів між станами життєвого циклу [15], [16], [17].

Мета і задачі дослідження. Метою бакалаврської кваліфікаційної роботи є розробка програмного засобу для моніторингу стану IoT-пристроїв і підтримки управління їх життєвим циклом.

Для досягнення поставленої мети в роботі необхідно вирішити такі завдання: проаналізувати предметну область моніторингу IoT-пристроїв; розглянути основні етапи життєвого циклу IoT-пристрою; дослідити наявні програмні рішення для моніторингу; проаналізувати протоколи MQTT, HTTP/HTTPS, WebSocket і CoAP; сформувати функціональні та нефункціональні вимоги; спроектувати архітектуру системи; розробити інформаційну модель бази даних; запропонувати алгоритми приймання, перевірки й обробки телеметричних даних; розробити механізм управління життєвим циклом; реалізувати серверну частину, клієнтський вебінтерфейс і модулі обробки телеметрії; провести тестування програмного засобу [3], [4], [5], [6], [7].

Об’єктом дослідження є процеси моніторингу, обліку та супроводу IoT-пристроїв у розподіленій інформаційній системі.

Предметом дослідження методи, алгоритми й програмні засоби збору телеметрії, контролю станів, обробки подій і управління життєвим циклом IoT-пристроїв.

Практичне значення роботи полягає у створенні програмного прототипу, який може використовуватися для моніторингу невеликої IoT-інфраструктури, навчального стенду або демонстраційної системи керування пристроями. Розроблений програмний засіб може забезпечувати перегляд списку пристроїв, контроль їх поточного стану, приймання телеметричних даних, збереження історії показників, фіксацію подій, формування сповіщень і зміну статусів життєвого циклу.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перша частина бакалаврської кваліфікаційної роботи спрямована на дослідження предметної області моніторингу IoT-пристроїв, визначення особливостей їх функціонування, аналіз життєвого циклу та обґрунтування потреби у створенні спеціалізованого програмного засобу. У межах цього розділу розглядаються теоретичні основи побудови IoT-систем, типові технології передавання даних, наявні програмні рішення для моніторингу, а також вимоги, які повинна задовольняти розроблювана система.

Необхідність такого аналізу зумовлена тим, що IoT-пристрої є елементами розподіленої інформаційної інфраструктури, які взаємодіють із фізичним середовищем, генерують телеметричні дані та потребують постійного контролю працездатності. На відміну від традиційних програмних компонентів, IoT-пристрої часто мають обмежені ресурси, працюють у нестабільних мережевих умовах і можуть перебувати на віддалених об’єктах. Тому програмний засіб для їх моніторингу повинен не лише відображати поточні показники, а й забезпечувати облік пристроїв, фіксацію подій, формування сповіщень і підтримку управління життєвим циклом.

1.1 Аналіз предметної області моніторингу IoT-пристроїв

Internet of Things, або Інтернет речей, розглядається як концепція побудови інформаційних систем, у яких фізичні об’єкти оснащуються сенсорами, виконавчими механізмами, мікроконтролерами та засобами мережевої взаємодії. Такі об’єкти здатні збирати дані про навколишнє середовище, передавати їх до серверних або хмарних платформ, отримувати команди керування та взаємодіяти з іншими компонентами цифрової інфраструктури [1]. У сучасних IoT-системах пристрої можуть застосовуватися

для контролю температури, вологості, освітленості, енергоспоживання, стану обладнання, переміщення об’єктів, доступу до приміщень та інших параметрів.

Типова IoT-система складається з декількох рівнів. На нижньому рівні розміщуються фізичні пристрої: сенсори, виконавчі механізми, мікроконтролери, промислові контролери або вбудовані модулі. Вони забезпечують первинне збирання даних і виконання команд. Наступний рівень утворюють мережеві компоненти, які відповідають за передавання інформації: локальні мережі, бездротові канали, шлюзи, маршрутизатори та спеціалізовані протоколи обміну даними. Вище розміщується серверна або хмарна частина, у якій здійснюється приймання, збереження, обробка та аналіз телеметрії. Завершальним рівнем є користувацький інтерфейс, через який адміністратор або оператор отримує доступ до інформації про стан пристроїв і може виконувати керуючі дії [2].

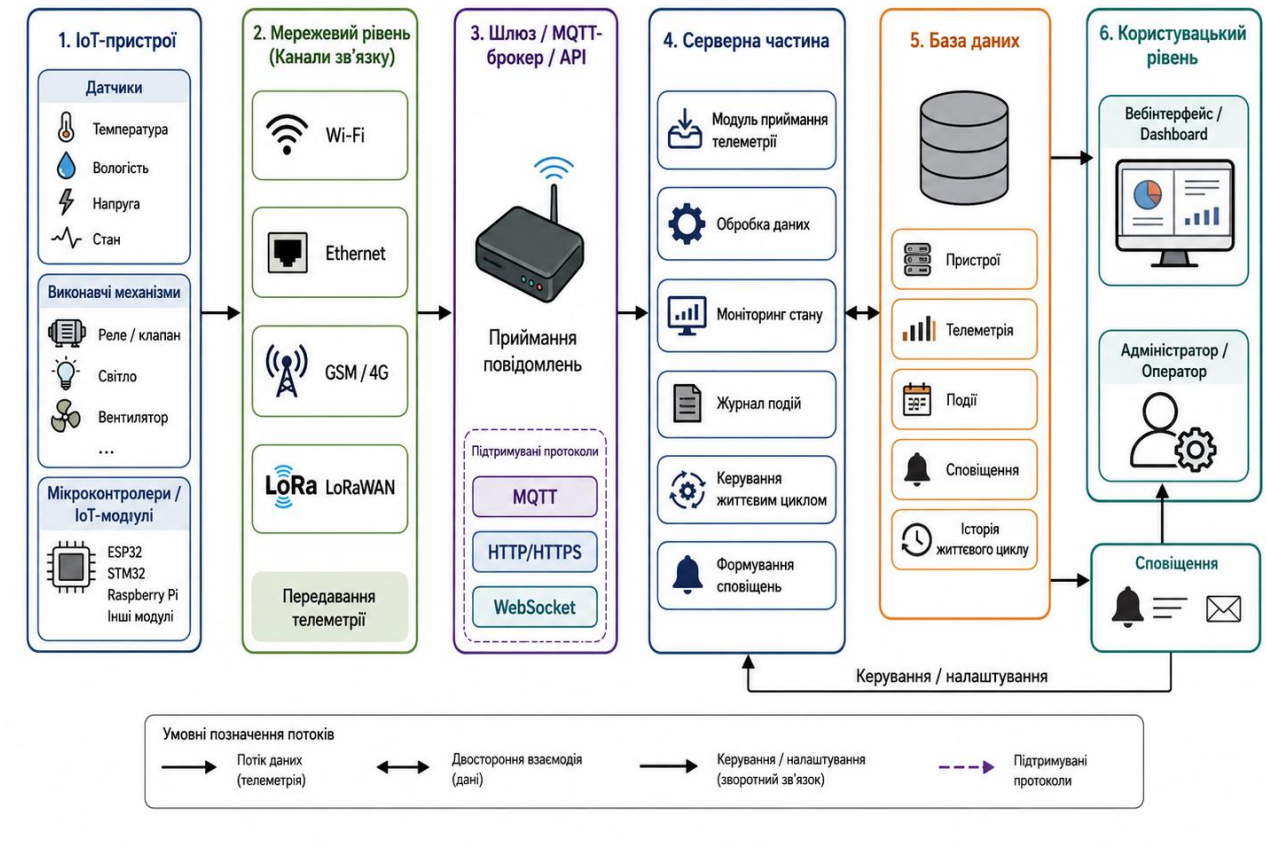

Рисунок 1.1 – Узагальнена структура IoT-системи моніторингу пристроїв

Предметна область моніторингу IoT-пристроїв охоплює процеси збирання, передавання, збереження, аналізу та відображення інформації про поточний стан пристроїв. Основними об’єктами моніторингу є не лише корисні вимірювальні показники, а й службові параметри, які характеризують працездатність самого пристрою. До таких параметрів належать час останнього з’єднання, рівень сигналу, напруга живлення, заряд акумулятора, версія програмного забезпечення, код помилки, статус підключення, інтенсивність передавання даних і кількість аварійних подій.

Моніторинг IoT-пристроїв має декілька важливих завдань. По-перше, він забезпечує контроль доступності пристрою. Якщо пристрій тривалий час не надсилає дані, система повинна виявити втрату зв’язку та зафіксувати відповідну подію. По-друге, моніторинг дає змогу контролювати коректність телеметричних даних. Значення, що виходять за допустимі межі, можуть свідчити про аварійний стан об’єкта, несправність сенсора або помилку вимірювання. По-третє, моніторинг підтримує діагностику пристроїв, оскільки збережена історія показників і подій дає змогу аналізувати динаміку змін і встановлювати причини відмов.

Особливістю IoT-пристроїв є те, що вони часто працюють автономно або напівавтономно. У промислових, транспортних чи побутових сценаріях оператор не має змоги постійно перевіряти кожний пристрій вручну. Саме тому програмний засіб моніторингу повинен автоматизувати контроль стану, виявлення проблем і повідомлення відповідальних користувачів. Наприклад, якщо пристрій передає критичне значення температури або не виходить на зв’язок упродовж визначеного часу, система має автоматично змінити його статус, створити запис у журналі подій і сформувати сповіщення.

1.2 Життєвий цикл IoT-пристроїв та його основні етапи

IoT-пристрій у межах інформаційної системи не є незмінним об’єктом. Від моменту додавання до системи до завершення експлуатації він проходить

певну послідовність станів, які утворюють його життєвий цикл. Управління життєвим циклом IoT-пристроїв передбачає контроль цих станів, збереження історії змін, фіксацію дій користувачів і забезпечення узгодженості між фактичним технічним станом пристрою та його статусом у програмній системі [15], [16]. Першим етапом життєвого циклу є реєстрація пристрою. На цьому етапі до системи вноситься базова інформація: унікальний ідентифікатор, назва, тип пристрою, місце встановлення, опис, відповідальний користувач і початковий статус. Реєстрація потрібна для того, щоб пристрій став відомим системі та міг бути пов’язаний із подальшими телеметричними повідомленнями. Під час експлуатації можуть виникати ситуації, які потребують діагностики або технічного обслуговування. Наприклад, пристрій може передавати нестабільні значення, мати низький рівень живлення, втрачати зв’язок або фіксувати внутрішню помилку. У таких випадках система повинна підтримувати зміну статусу пристрою на «потребує обслуговування», «попередження» або «аварійний стан». Це дає змогу відокремити нормально працюючі пристрої від тих, які потребують уваги адміністратора [17], [18].

Таблиця 1.1 – Основні етапи життєвого циклу IoT-пристрою

Етап	**Зміст**	**Основні дії**	**Результат**
Реєстрація	Внесення пристрою до системи	Створення запису, присвоєння ідентифікатора	Пристрій доступний для налаштування
Ідентифікація	Встановлення відповідності між фізичним пристроєм і записом у системі	Прив’язка ключа, типу та місця встановлення	Система розпізнає пристрій
Налаштування	Задання параметрів роботи	Встановлення порогів, інтервалів, правил сповіщення	Пристрій підготовлено до експлуатації
Експлуатація	Штатна робота пристрою	Передавання телеметрії, контроль стану	Система відображає актуальний стан
Обслуговування	Усунення несправностей або профілактичні дії	Переведення в сервісний режим	Відновлення штатної роботи
Виведення з експлуатації	Завершення використання пристрою	Архівування історії, блокування активності	Пристрій зберігається як історичний об’єкт

Управління життєвим циклом має важливе значення для підтримання порядку в IoT-інфраструктурі. Без такого управління система може містити пристрої з невизначеним статусом, застарілою конфігурацією або відсутньою історією обслуговування. Це ускладнює адміністрування, підвищує ризик помилок і знижує прозорість роботи системи.

1.3 Огляд існуючих програмних рішень для моніторингу IoT

ThingsBoard є платформою, орієнтованою на створення IoT-рішень, збір телеметрії, управління пристроями, побудову панелей моніторингу та формування правил обробки подій [8]. AWS IoT Core є хмарним сервісом для підключення, автентифікації, обміну повідомленнями та інтеграції IoT-пристроїв з іншими сервісами Amazon Web Services [9]. Azure IoT Hub забезпечує двосторонній обмін повідомленнями між IoT-пристроями та хмарними застосунками, підтримує керування пристроями та безпечну автентифікацію [10].

Node-RED є інструментом візуального програмування потоків даних, який часто використовується для прототипування IoT-рішень [11]. Prometheus є системою моніторингу та збирання метрик, яка широко використовується для контролю стану серверів, контейнерів і програмних сервісів [12]. Grafana є системою візуалізації, яка дає змогу створювати дашборди на основі різних джерел даних [13].

Zabbix є універсальною системою моніторингу мереж, серверів, сервісів і обладнання [14].

Таблиця 1.2 – Порівняльний аналіз програмних засобів моніторингу IoT-пристроїв

Рішення	**Призначення**	**Переваги**	**Обмеження**	**Придатність**
ThingsBoard	IoT-платформа для збору телеметрії та керування пристроями	Дашборди, правила обробки, підтримка IoT-протоколів	Значна складність, багато готової функціональності	Корисна як аналог
AWS IoT Core	Хмарне підключення та керування IoT-пристроями	Масштабованість, безпека, інтеграція з AWS	Залежність від хмари, складність налаштування	Приклад хмарного рішення
Azure IoT Hub	Хмарний сервіс взаємодії з IoT-пристроями	Двосторонній обмін, керування пристроями	Прив’язка до Azure	Приклад промислової платформи
Node-RED	Візуальне програмування потоків даних	Швидке прототипування, простота інтеграції	Обмеженість для складної системи життєвого циклу	Аналог для потоків
Prometheus	Збирання та аналіз метрик	Робота з часовими рядами, запити до метрик	Не орієнтований безпосередньо на IoT-життєвий цикл	Аналіз метрик
Grafana	Візуалізація даних і дашборди	Гнучкі панелі, підтримка багатьох джерел	Не приймає телеметрію самостійно	Приклад UI-моніторингу
Zabbix	Моніторинг інфраструктури та обладнання	Сповіщення, шаблони, контроль доступності	Менша гнучкість для довільної IoT-телеметрії	Аналог системного моніторингу

Проведений огляд показує, що наявні рішення можуть успішно використовуватися для промислових і корпоративних IoT-інфраструктур. Водночас вони не завжди відповідають завданням бакалаврського програмного проєкту, оскільки або є надмірно складними, або орієнтовані лише на окрему частину задачі: візуалізацію, збирання метрик, інтеграцію пристроїв чи хмарне керування. Тому в межах цієї роботи доцільно розробити власний програмний засіб, який реалізує базові, але цілісні функції моніторингу та управління життєвим циклом.

1.4 Аналіз протоколів і технологій обміну даними в IoT-системах

Одним із найбільш поширених протоколів для IoT є MQTT. Він побудований на моделі publish/subscribe, у якій пристрої публікують повідомлення в певні теми, а клієнти або серверні компоненти підписуються на ці теми через брокер повідомлень [4]. Така модель є зручною для телеметрії, оскільки пристрій не повинен безпосередньо знати адресу всіх отримувачів даних. HTTP і HTTPS використовуються переважно для взаємодії клієнтських застосунків із серверною частиною або для передавання даних за моделлю запит-відповідь [6]. У контексті розроблюваної системи HTTP/REST API доцільно застосовувати для операцій перегляду списку пристроїв, отримання детальної інформації, створення записів, зміни статусів життєвого циклу та роботи з журналом подій. WebSocket забезпечує двосторонній обмін даними між клієнтом і сервером у межах постійного з'єднання [7]. Ця технологія корисна тоді, коли потрібно оновлювати дані в інтерфейсі майже в реальному часі без постійного виконання HTTP-запитів. CoAP є протоколом прикладного рівня, орієнтованим на пристрої з обмеженими ресурсами та мережі з невисокою пропускною здатністю [5].

Таблиця 1.3 – Порівняння протоколів обміну даними в IoT-системах

Протокол	Модель взаємодії	Переваги	Обмеження	Доцільність використання
MQTT	Publish/subscribe через брокер	Легкість, асинхронність, зручність для телеметрії	Потребує MQTT-брокера	Основний варіант для приймання телеметрії
HTTP/HTTPS	Запит-відповідь	Простота, поширеність, підтримка REST API	Менш ефективний для частих дрібних повідомлень	Доцільний для вебінтерфейсу та API
WebSocket	Постійне двостороннє з'єднання	Оперативне оновлення даних, реальний час	Складніше керування з'єднаннями	Доцільний для live-оновлень
CoAP	Легка модель запит-відповідь	Придатність для обмежених пристроїв	Менша поширеність у веборієнтованих системах	Альтернатива для ресурсно обмежених пристроїв

Для розроблюваного програмного засобу найбільш доцільним є комбінований підхід.

MQTT може використовуватися для приймання телеметрії від IoT-пристроїв, оскільки він ефективно підтримує модель періодичного передавання повідомлень. HTTP/REST API доцільно застосовувати для взаємодії вебінтерфейсу із серверною частиною, оскільки такий підхід є зрозумілим, стандартизованим і зручним для реалізації CRUD-операцій. WebSocket може бути використаний для оновлення станів у реальному часі, а CoAP — розглядатися як можлива альтернатива для подальшого розвитку системи [3], [25].

1.5 Постановка задачі та формування вимог до програмного засобу

На основі аналізу предметної області, життєвого циклу IoT-пристроїв, існуючих програмних рішень і протоколів обміну даними можна сформулювати основну задачу бакалаврської роботи.

Вона полягає в розробці веборієнтованого програмного засобу, який забезпечує централізований облік IoT-пристроїв, приймання та збереження телеметричних даних, визначення поточного стану, ведення журналу подій, формування сповіщень і підтримку управління життєвим циклом пристроїв.

Крім наведених вимог, програмний засіб повинен мати логічно організовану структуру даних. Оскільки система працює з пристроями, телеметрією, подіями, сповіщеннями й історією життєвого циклу, доцільно передбачити окремі сутності для кожного з цих інформаційних об’єктів. Це забезпечить цілісність даних і спростить подальшу реалізацію функцій пошуку, фільтрації та аналізу..

Таблиця 1.4 – Функціональні та нефункціональні вимоги до програмного засобу

Тип вимоги	Вимога	Опис	Пріоритет
Функціональна	Реєстрація IoT-пристроїв	Додавання нового пристрою із зазначенням назви, типу, ідентифікатора та місця встановлення	Високий
Функціональна	Приймання телеметрії	Отримання даних від пристроїв через MQTT або HTTP	Високий
Функціональна	Збереження історії показників	Накопичення телеметричних записів у базі даних	Високий
Функціональна	Визначення статусу пристрою	Автоматичне оновлення стану на основі телеметрії, часу останнього з’єднання та помилок	Високий
Функціональна	Журнал подій	Фіксація інформаційних, попереджувальних і аварійних подій	Високий
Функціональна	Управління життєвим циклом	Переведення пристрою між станами: активний, обслуговування, деактивований	Високий
Нефункціональна	Надійність	Система повинна коректно обробляти помилкові або неповні повідомлення	Високий
Нефункціональна	Безпека доступу	Доступ до функцій керування має бути обмежений авторизованими користувачами	Високий
Нефункціональна	Масштабованість	Архітектура повинна передбачати додавання нових пристроїв і параметрів	Середній
Нефункціональна	Зручність інтерфейсу	Інтерфейс має забезпечувати швидке виявлення проблемних пристроїв	Високий

1.6 Висновок до першого розділу

У першому розділі бакалаврської кваліфікаційної роботи проаналізовано предметну область моніторингу IoT-пристроїв і визначено, що такі пристрої є елементами розподілених інформаційних систем, які потребують постійного контролю доступності, працездатності, коректності телеметрії та технічного стану. Розглянуто типову структуру IoT-системи, до складу якої входять сенсори, контролери, мережеві компоненти, серверна частина, база даних і користувацький інтерфейс.

Окремо досліджено життєвий цикл IoT-пристроїв, що охоплює реєстрацію, ідентифікацію, налаштування, введення в експлуатацію, моніторинг, діагностику, оновлення, обслуговування, деактивацію та виведення з експлуатації.

Проведено огляд існуючих програмних рішень для моніторингу IoT і суміжних задач, зокрема ThingsBoard, AWS IoT Core, Azure IoT Hub, Node-RED, Prometheus, Grafana та Zabbix. Також проаналізовано протоколи MQTT, HTTP/HTTPS, WebSocket і CoAP. На основі виконаного аналізу сформовано функціональні та нефункціональні вимоги до програмного засобу.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

Проектна частина бакалаврської кваліфікаційної роботи присвячена розробленню структурних, інформаційних та алгоритмічних рішень для програмного засобу моніторингу стану IoT-пристроїв і управління їх життєвим циклом. Якщо в аналітичній частині було визначено проблематику предметної області, особливості IoT-пристроїв, наявні програмні рішення, протоколи обміну даними та вимоги до системи, то в цьому розділі ці положення конкретизуються у вигляді архітектури, моделі даних, алгоритмів обробки телеметрії та логіки користувацького інтерфейсу.

Основним завданням проектування є створення цілісної моделі програмного засобу, яка забезпечує приймання телеметричних даних від IoT-пристроїв, збереження історії показників, визначення поточного стану пристроїв, фіксацію подій, формування сповіщень і підтримку переходів між станами життєвого циклу. При цьому система повинна бути достатньо простою для реалізації в межах бакалаврської роботи, але водночас мати модульну структуру, що дає змогу розширювати її функціональність у майбутньому.

2.1 Проектування архітектури програмного засобу

Архітектура системи підтримки прийняття рішень визначає склад її основних компонентів, їх функції, взаємозв’язки та порядок обміну даними. Для розроблюваної системи доцільно використати веборієнтовану клієнт-серверну архітектуру, яка забезпечує розділення користувацького інтерфейсу, серверної логіки, аналітичного модуля та бази даних. Такий підхід дозволяє структурувати програмне рішення, розмежувати відповідальність між модулями та забезпечити можливість розширення системи [9].

Веборієнтована архітектура передбачає, що користувач взаємодіє із системою через браузер. Клієнтська частина відповідає за подання інформації,

введення Архітектура програмного засобу моніторингу IoT-пристроїв має забезпечувати логічне розділення функцій приймання даних, їх обробки, збереження, відображення та управління життєвим циклом пристроїв. З огляду на сформовані вимоги, доцільно використовувати багаторівневу архітектуру, яка складається з рівня IoT-пристроїв, рівня обміну повідомленнями, серверного рівня, рівня збереження даних і клієнтського рівня.

Рівень обміну повідомленнями відповідає за передавання телеметрії від пристроїв до серверної частини. Для цього доцільно використати MQTT-брокер, оскільки MQTT підтримує модель publish/subscribe і є придатним для передавання невеликих телеметричних повідомлень від багатьох пристроїв [4]. IoT-пристрій публікує повідомлення у визначену тему, наприклад devices/{deviceId}/telemetry, а серверний модуль підписується на відповідні теми й приймає повідомлення для подальшої обробки.

Серверний рівень є центральною частиною програмного засобу. Він виконує приймання телеметрії, перевірку коректності повідомлень, взаємодію з базою даних, визначення статусу пристроїв, створення подій і сповіщень, а також надання REST API для клієнтського вебінтерфейсу. Доцільно передбачити окремі модулі серверної частини: модуль авторизації, модуль керування пристроями, модуль приймання телеметрії, модуль подій, модуль сповіщень, модуль життєвого циклу та модуль доступу до бази даних.

Рівень збереження даних забезпечує структуроване зберігання інформації про користувачів, пристрої, телеметричні записи, події, сповіщення, конфігурації та історію життєвого циклу. Для цього доцільно використати реляційну базу даних PostgreSQL, оскільки вона підтримує цілісність зв’язків між сутностями, транзакційність, індексацію та зручну роботу зі структурованими даними [22].

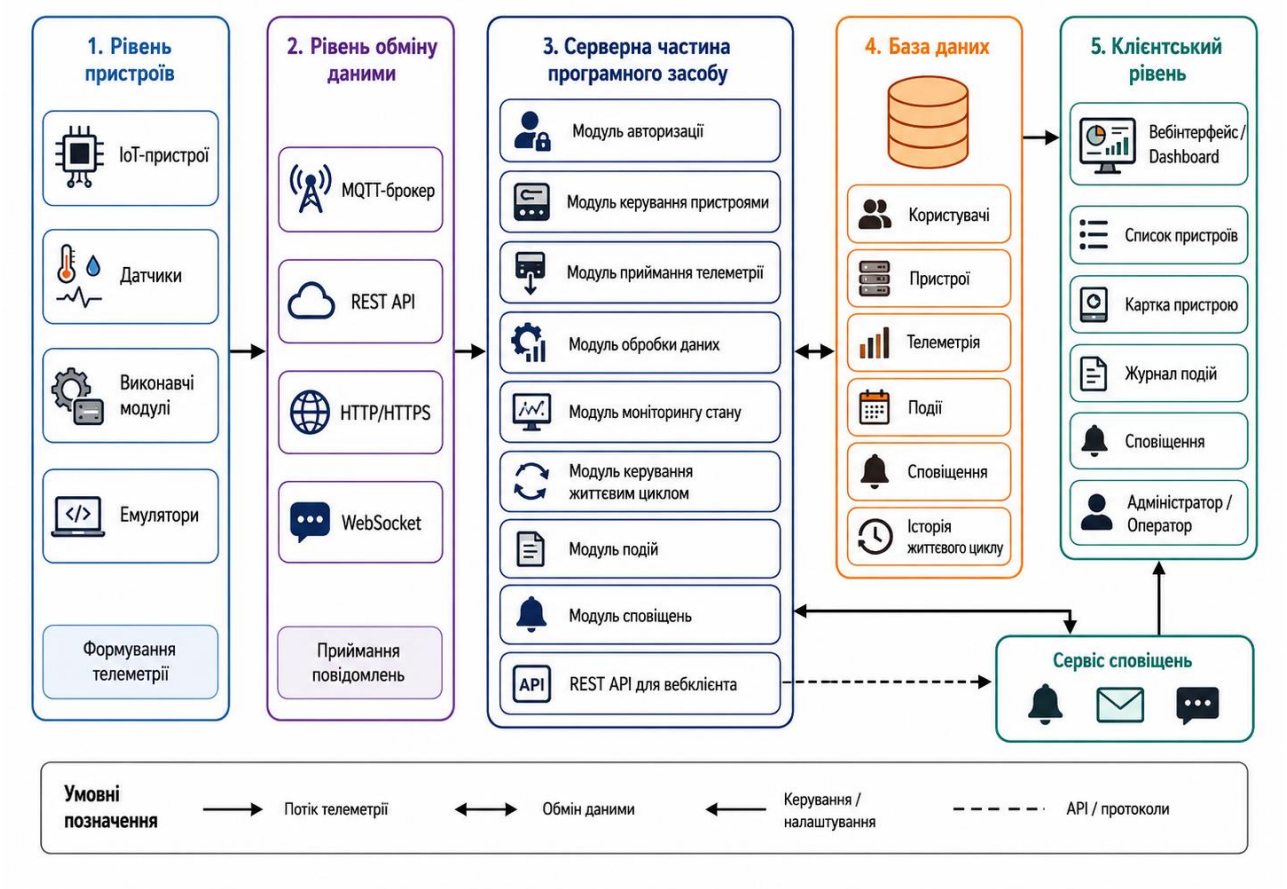

Рисунок 2.1 – Архітектура програмного засобу моніторингу IoT-пристроїв

2.2 Проєктування інформаційної моделі та структури бази даних

Інформаційна модель програмного засобу визначає, які дані зберігаються в системі, які сутності використовуються для опису IoT-пристроїв і як ці сутності пов’язані між собою. Оскільки розроблювана система повинна підтримувати моніторинг, журналювання подій і управління життєвим циклом, база даних має охоплювати не лише поточні характеристики пристроїв, а й історичні дані.

Основними сутностями системи є користувач, IoT-пристрій, тип пристрою, телеметричний запис, подія, сповіщення, конфігурація пристрою та запис історії життєвого циклу. Центральною сутністю є IoT-пристрій, оскільки саме навколо нього формуються телеметричні дані, події, статуси, сповіщення й історія змін.

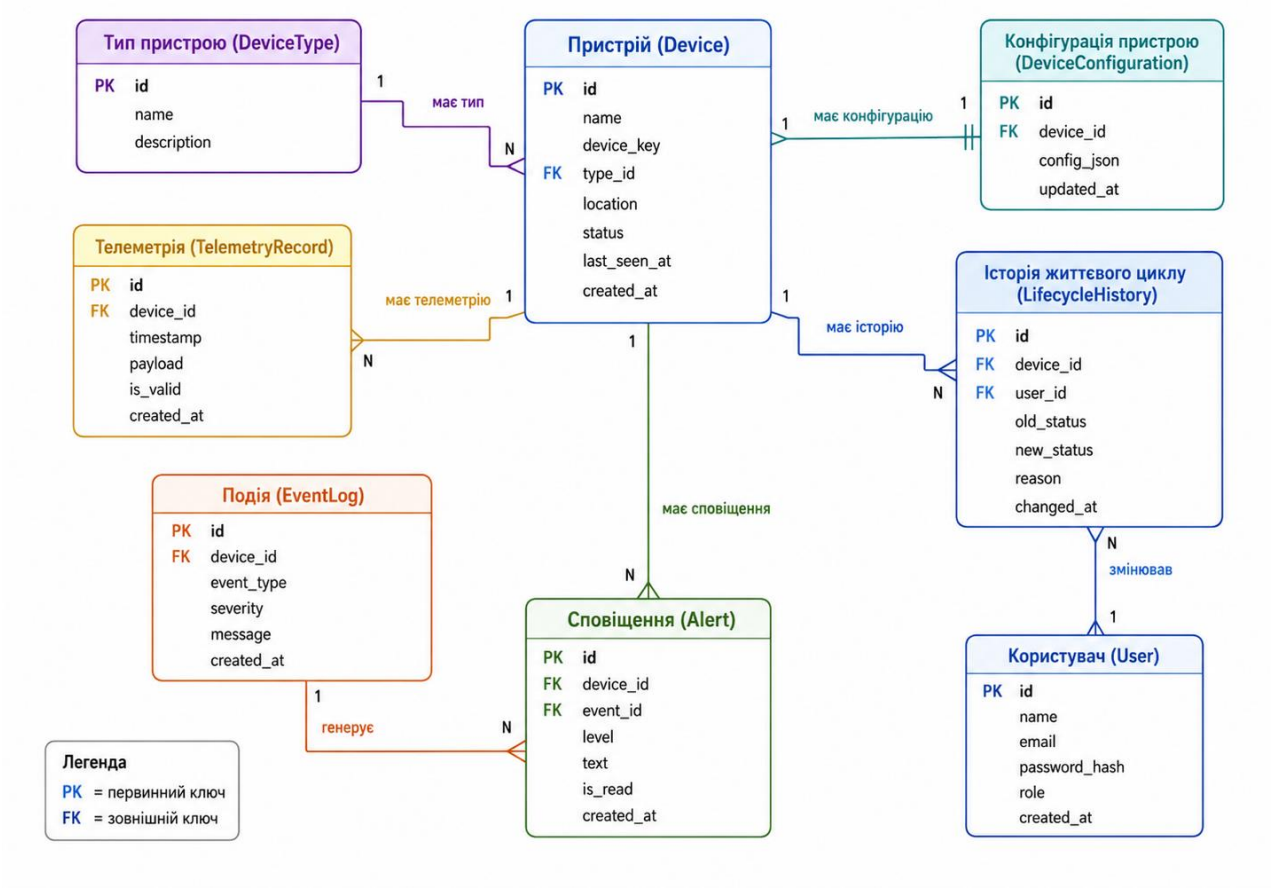

Рисунок 2.2 – ER-модель бази даних програмного засобу

Таблиця 2.1 – Опис основних таблиць бази даних

Сутність	Призначення	Основні атрибути	Зв’язки
User	Збереження даних користувачів	id, name, email, password_hash, role	Може створювати або змінювати пристрої та статуси
Device	Опис IoT-пристрою	id, name, device_key, type_id, location, status, last_seen_at	Має телеметрію, події, сповіщення, історію життєвого циклу
DeviceType	Класифікація пристроїв	id, name, description	Пов’язаний із багатьма пристроями
TelemetryRecord	Збереження телеметрії	id, device_id, timestamp, payload, is_valid	Належить одному пристрою
EventLog	Фіксація подій	id, device_id, event_type, severity, message	Пов’язаний із пристроєм і може формувати сповіщення
Alert	Збереження сповіщень	id, device_id, event_id, level, text, is_read	Пов’язаний із подією та пристроєм

Для забезпечення цілісності даних потрібно передбачити зовнішні ключі між таблицями. Наприклад, кожний телеметричний запис повинен належати

конкретному пристрою, а кожна подія має бути пов’язана з пристроєм, який її спричинив. Видалення пристрою з бази даних не є бажаним, оскільки разом із ним може бути втрачена історія телеметрії та подій. Тому доцільно використовувати логічне видалення або статус «виведений з експлуатації».

2.3 Розробка алгоритмів моніторингу стану IoT-пристроїв

Алгоритм моніторингу стану IoT-пристрою є одним із ключових елементів програмного засобу. Його завдання полягає не лише в прийманні телеметричних повідомлень, а й у перевірці їх коректності, оновленні поточного стану пристрою, виявленні відхилень, формуванні подій і сповіщень. Такий алгоритм має працювати автоматично після надходження кожного повідомлення або за розкладом для перевірки доступності пристроїв.

Першим етапом алгоритму є приймання телеметричного повідомлення. Повідомлення може надходити через MQTT-брокер або HTTP API. У разі використання MQTT серверний модуль підписується на теми пристроїв і отримує повідомлення у форматі JSON. У разі HTTP пристрій або емулятор надсилає POST-запит на спеціальний API-маршрут. Незалежно від каналу передавання дані повинні мати уніфіковану структуру, яка містить ідентифікатор пристрою, часову мітку та набір параметрів.

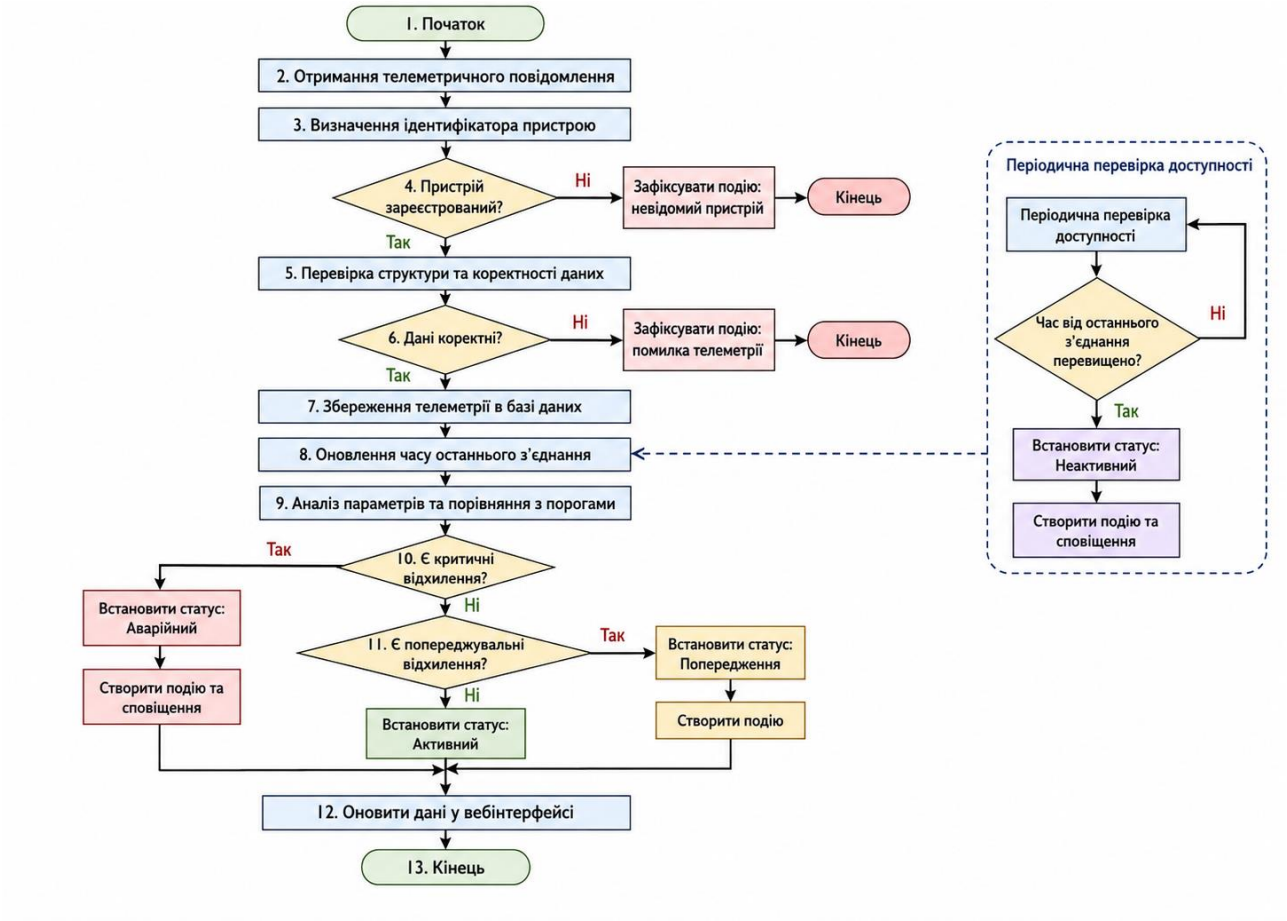

Рисунок 2.3 – Блок-схема алгоритму моніторингу стану IoT-пристрою

Таблиця 2.2 – Стани IoT-пристрою в системі моніторингу

Статус	**Умова встановлення**	**Опис**	**Дія системи**
Активний	Пристрій регулярно передає коректні дані	Штатний режим роботи	Оновлення телеметрії та часу останнього з'єднання
Неактивний	Відсутність телеметрії протягом визначеного часу	Пристрій не виходить на зв'язок	Створення події про втрату зв'язку
Попередження	Значення наближаються до критичних меж	Потрібна увага оператора	Формування попереджувального запису
Аварійний	Перевищено критичні пороги або отримано код помилки	Порушення штатної роботи	Створення аварійної події та сповіщення
На обслуговуванні	Статус встановлено користувачем	Пристрій тимчасово не бере участі в штатному моніторингу	Обмеження автоматичних аварійних сповіщень
Виведений з експлуатації	Пристрій більше не використовується	Історичний запис зберігається	Блокування приймання активної телеметрії

Алгоритм моніторингу повинен також підтримувати періодичну перевірку доступності пристроїв. Навіть якщо нові повідомлення не надходять, система має регулярно перевіряти поле last_seen_at для активних пристроїв. Якщо різниця між поточним часом і часом останнього з'єднання перевищує допустимий інтервал, створюється подія про втрату зв'язку, а статус пристрою змінюється на «неактивний».

2.4 Проєктування модуля підтримки прийняття рішень

Механізм управління життєвим циклом IoT-пристроїв повинен забезпечувати контрольовані переходи між станами пристрою, фіксацію історії змін і зв'язок між результатами моніторингу та управлінськими діями. На відміну від звичайного статусу працездатності, життєвий цикл відображає загальний етап існування пристрою в системі: від реєстрації до виведення з експлуатації.

У межах розроблюваного програмного засобу доцільно передбачити такі стани життєвого циклу: «зареєстрований», «налаштований», «активний», «попередження», «аварійний», «на обслуговуванні», «оновлюється», «деактивований», «виведений з експлуатації». Частина цих станів може встановлюватися користувачем, а частина — автоматично на основі результатів моніторингу.

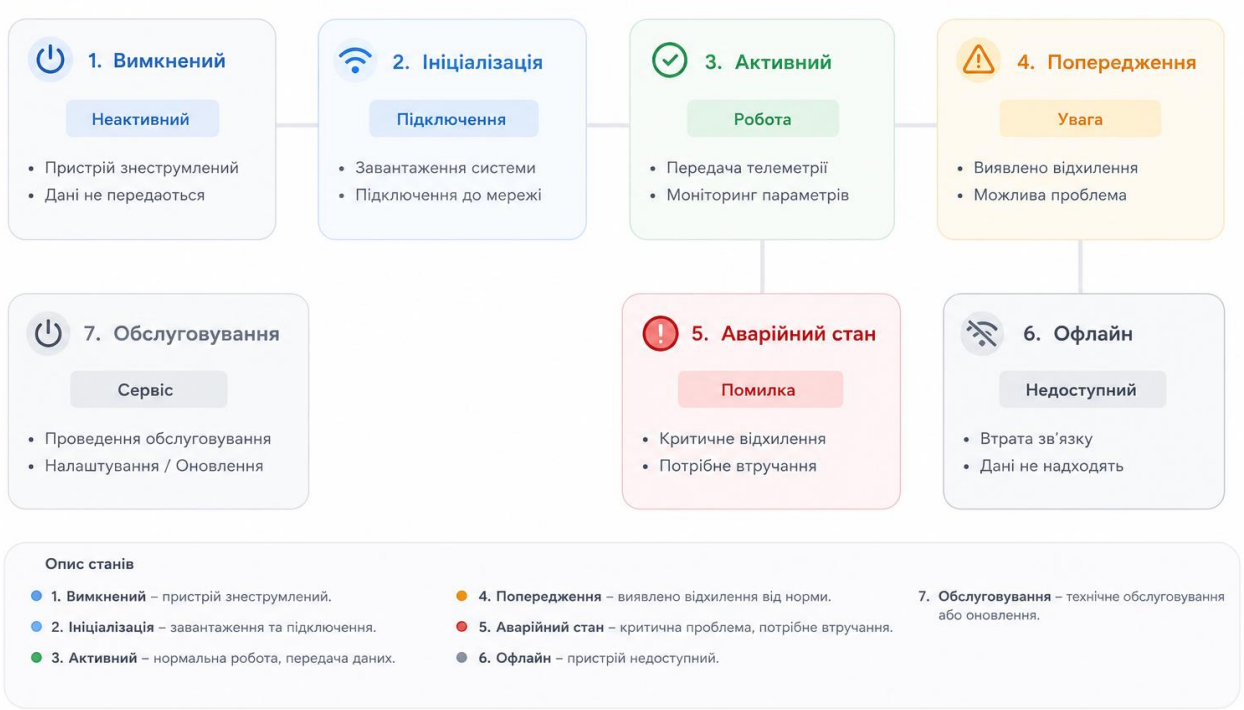

Рисунок 2.4 – Діаграма станів життєвого циклу IoT-пристрою

Для кожної зміни стану необхідно створювати запис у таблиці LifecycleHistory. Такий запис має містити старий статус, новий статус, дату зміни, причину та ініціатора переходу. Ініціатором може бути користувач або системний процес. Наприклад, якщо адміністратор переводить пристрій у режим обслуговування, ініціатором є користувач. Якщо система автоматично позначає пристрій як неактивний через відсутність телеметрії, ініціатором є системний механізм моніторингу.

2.5 Проєктування користувацького інтерфейсу системи

Користувацький інтерфейс програмного засобу є основним засобом взаємодії адміністратора або оператора із системою моніторингу IoT-пристроїв. Його призначення полягає в тому, щоб надати користувачу зручний доступ до інформації про стан пристроїв, телеметричні показники, події, сповіщення та дії життєвого циклу. Інтерфейс повинен бути інформативним, логічно структурованим і орієнтованим на швидке виявлення проблемних пристроїв.

Доцільно передбачити такі основні сторінки вебінтерфейсу: сторінка авторизації, головна панель моніторингу, список пристроїв, картка окремого пристрою, журнал подій, сторінка сповіщень, сторінка управління життєвим циклом і сторінка налаштувань. Така структура забезпечує розподіл функцій за логічними блоками та спрощує навігацію.

Головна панель моніторингу повинна містити узагальнену інформацію про стан IoT-інфраструктури. Доцільно відображати загальну кількість пристроїв, кількість активних пристроїв, кількість пристроїв у стані попередження, кількість аварійних і неактивних пристроїв, а також останні події. Для швидкого сприйняття станів можна використовувати кольорові індикатори, статусні мітки та короткі текстові повідомлення.

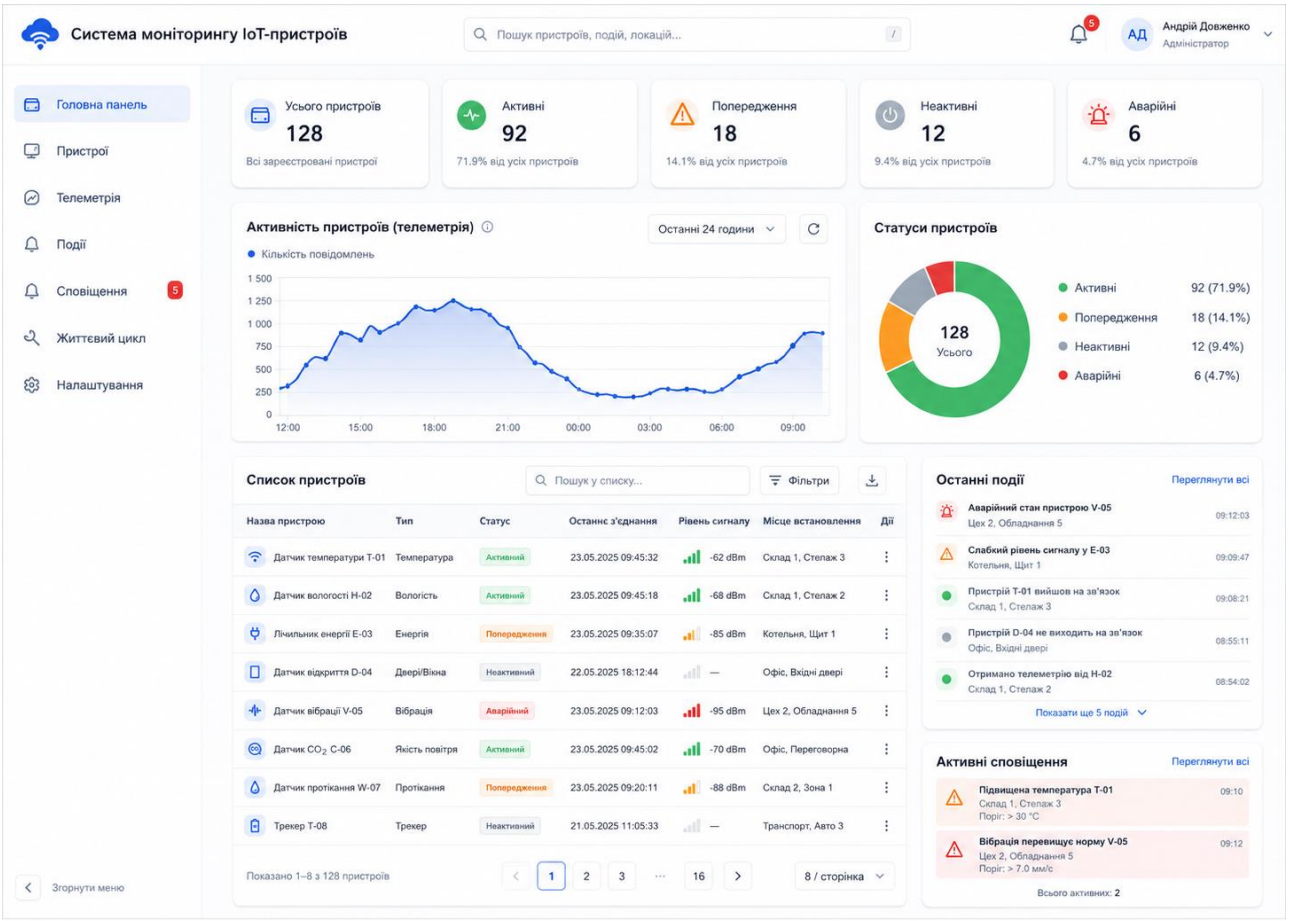

Рисунок 2.5 – Макет головної панелі моніторингу IoT-пристроїв

Рисунок 2.5 – Макет головної панелі моніторингу IoT-пристроїв

Сторінка списку пристроїв повинна відображати таблицю з основними відомостями про кожний пристрій: назва, тип, місце встановлення, поточний статус, час останнього з’єднання та наявність активних сповіщень. Для зручності роботи необхідно передбачити пошук, фільтрацію за статусом, фільтрацію за типом пристрою та сортування за часом останнього з’єднання.

Таблиця 2.3 – Основні API-операції програмного засобу

Метод	**Маршрут**	**Призначення**	**Результат**
POST	/api/auth/login	Авторизація користувача	Отримання токена або сесії
GET	/api/devices	Отримання списку пристроїв	Масив пристроїв зі статусами
POST	/api/devices	Створення нового пристрою	Новий запис пристрою
GET	/api/devices/:id	Отримання детальної інформації	Картка пристрою
PATCH	/api/devices/:id	Оновлення даних пристрою	Змінений запис пристрою
POST	/api/telemetry	Приймання телеметрії через HTTP	Створений телеметричний запис
GET	/api/events	Отримання журналу подій	Масив подій
GET	/api/alerts	Отримання сповіщень	Масив активних або архівних сповіщень
PATCH	/api/devices/:id/lifecycle	Зміна статусу життєвого циклу	Новий запис історії життєвого циклу

Проектований інтерфейс повинен відповідати основній меті системи — швидко надавати користувачу інформацію про стан IoT-пристроїв і забезпечувати можливість виконання дій щодо їх супроводу. Саме тому центральними елементами інтерфейсу є статуси, час останнього з’єднання, аварійні події, сповіщення та дії життєвого циклу.

2.6 Висновок до другого розділу

У другому розділі бакалаврської кваліфікаційної роботи виконано проектування програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом. Запропоновано багаторівневу архітектуру

системи, яка охоплює рівень IoT-пристроїв або емуляторів, рівень обміну повідомленнями через MQTT, серверну частину, базу даних і клієнтський вебінтерфейс. Така архітектура забезпечує логічне розділення функцій і створює основу для подальшої реалізації програмного засобу.

Розроблено інформаційну модель системи, у якій визначено основні сутності бази даних: користувачі, IoT-пристрої, типи пристроїв, телеметричні записи, події, сповіщення, конфігурації та історія життєвого циклу. Описано алгоритм моніторингу стану IoT-пристроїв, який передбачає приймання телеметрії, перевірку ідентифікатора пристрою, валідацію даних, збереження показників, оновлення часу останнього з’єднання, визначення статусу, створення подій і формування сповіщень. Також спроектовано механізм управління життєвим циклом і структуру користувацького інтерфейсу.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА. ПРАКТИЧНІ АСПЕКТИ РЕАЛІЗАЦІЇ ПРОГРАМНАОГО ЗАСОБУ

Третя частина бакалаврської кваліфікаційної роботи присвячена практичній реалізації програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом. У попередньому розділі було спроектовано архітектуру системи, інформаційну модель бази даних, алгоритми моніторингу, механізм управління життєвим циклом і структуру користувацького інтерфейсу. У цьому розділі ці проектні рішення конкретизуються через вибір інструментальних засобів, опис реалізації серверної частини, модуля обробки телеметрії, клієнтського інтерфейсу та тестування програмного засобу.

Розроблюваний програмний засіб доцільно реалізувати як веборієнтовану систему з клієнт-серверною архітектурою. Серверна частина забезпечує обробку запитів, взаємодію з базою даних, приймання телеметрії та формування подій. Клієнтська частина надає користувачу інтерфейс для перегляду стану IoT-пристроїв, журналу подій, сповіщень і виконання дій управління життєвим циклом. Для передавання телеметричних повідомлень від пристроїв використовується MQTT, а для взаємодії вебінтерфейсу із сервером — REST API.

3.1 Вибір інструментальних засобів розробки

Вибір інструментальних засобів розробки є важливим етапом реалізації програмного засобу, оскільки від нього залежать продуктивність, масштабованість, зручність супроводу та можливість подальшого розвитку системи. Для цієї бакалаврської роботи доцільно обрати технологічний стек, який є достатньо поширеним, має розвинену екосистему, підтримує веборієнтовану архітектуру та дає змогу реалізувати всі основні функції системи без надмірної складності.

Таблиця 3.2 – Обґрунтування вибору інструментальних засобів розробки

Засіб	**Призначення**	**Причина вибору**	**Альтернатива**
Node.js	Реалізація серверної частини	Асинхронна модель, зручність для подієвих систем	Python, Java
Express.js	Побудова REST API	Простота, гнучкість, підтримка middleware	FastAPI, Django
PostgreSQL	Збереження структурованих даних	Надійність, зв’язки, індекси, транзакції, JSON-поля	MySQL, MongoDB
React	Клієнтський інтерфейс	Компонентний підхід, динамічний UI	Vue.js, Angular
Eclipse Mosquitto	MQTT-брокер	Легкість, поширеність, publish/subscribe	EMQX, HiveMQ
REST API	Взаємодія клієнта із сервером	Зрозуміла структура ресурсів	GraphQL
OpenAPI	Документування API	Формалізація маршрутів і відповідей	Swagger-коментарі

Обраний стек технологій відповідає рівню бакалаврської роботи та дозволяє реалізувати повний програмний прототип. Його перевагою є те, що всі основні компоненти системи мають чітке призначення: Node.js і Express.js забезпечують серверну логіку, PostgreSQL — збереження даних, React — взаємодію з користувачем, Mosquitto — приймання IoT-повідомлень, а REST API — зв’язок між клієнтською та серверною частинами.

3.2 Реалізація серверної частини програмного засобу

Серверна частина програмного засобу є центральним компонентом системи, який забезпечує обробку запитів, приймання телеметрії, роботу з базою даних, управління пристроями, формування подій і сповіщень. Вона виконує роль проміжного рівня між IoT-пристроями, базою даних і клієнтським вебінтерфейсом.

З погляду програмної структури серверну частину доцільно поділити на декілька функціональних модулів: модуль авторизації, модуль керування пристроями, модуль приймання телеметрії, модуль подій, модуль сповіщень,

модуль життєвого циклу та модуль доступу до бази даних. Такий поділ забезпечує логічну ізоляцію функцій і спрощує супровід програмного коду.

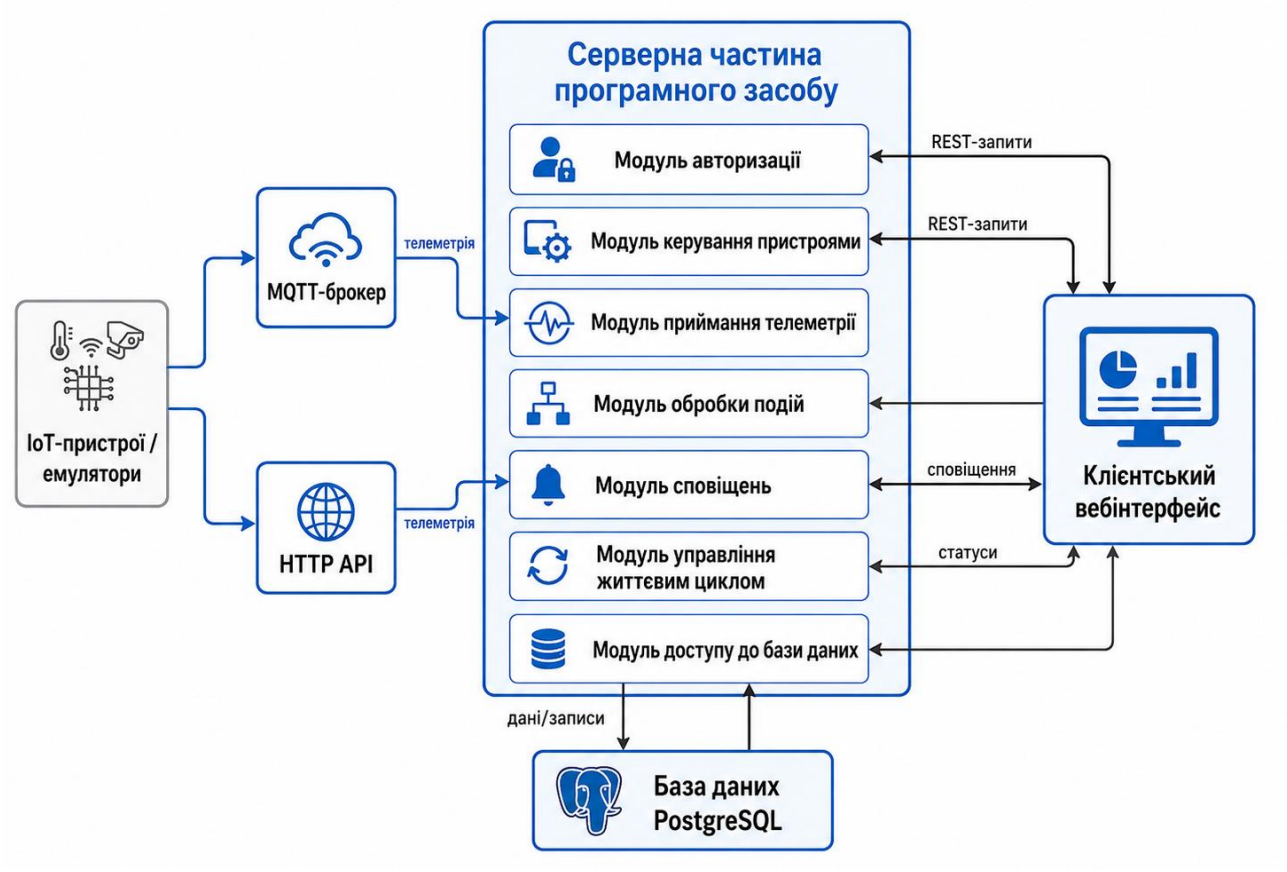

Рисунок 3.1 – Структура серверної частини програмного засобу

Модуль подій створює записи про значущі ситуації, що виникають у системі: реєстрацію пристрою, зміну статусу, надходження аварійної телеметрії, втрату зв’язку, помилку структури повідомлення, переведення пристрою в обслуговування або деактивацію. Модуль сповіщень формується на основі подій, які потребують уваги користувача. Модуль життєвого циклу відповідає за зміну станів пристрою та збереження історії таких змін.

3.3 Реалізація модуля моніторингу та обробки телеметрії

Модуль моніторингу та обробки телеметрії є ключовим функціональним компонентом розроблюваного програмного засобу. Його основне призначення

полягає в прийманні повідомлень від IoT-пристроїв, перевірці їх коректності, збереженні показників у базі даних, визначенні поточного стану пристрою та формуванні подій у разі виявлення відхилень.

Телеметричне повідомлення повинно мати уніфіковану структуру. Це спрощує обробку даних і дає змогу використовувати один механізм для різних пристроїв. У базовому варіанті повідомлення може містити такі поля: ідентифікатор пристрою, час формування повідомлення, температуру, вологість, напругу живлення, рівень сигналу, код помилки та службовий статус.

Таблиця 3.2 – Приклад структури телеметричних даних IoT-пристрою

Параметр	**Тип даних**	**Опис**	**Приклад значення**
deviceId	String	Унікальний ідентифікатор пристрою	sensor-001
timestamp	String / DateTime	Час формування повідомлення	2026-06-14T10:25:00
temperature	Number	Температура, °C	24.8
humidity	Number	Вологість, %	58
voltage	Number	Напруга живлення, В	3.7
signalLevel	Number	Рівень сигналу, % або dBm	82
errorCode	String / Null	Код помилки пристрою	null
status	String	Службовий статус пристрою	OK

Після надходження повідомлення модуль моніторингу виконує декілька послідовних дій. Спочатку перевіряється наявність обов’язкових полів. Якщо повідомлення не містить ідентифікатора пристрою або має некоректну структуру, воно не може бути оброблене як штатна телеметрія. У такому випадку система створює технічну подію про помилку приймання даних.

Рисунок 3.2 – Послідовність обробки телеметричних даних

Окремий механізм потрібен для виявлення втрати зв’язку. Він може працювати як фоновий процес, який через визначений інтервал часу перевіряє всі активні пристрої. Якщо поточний час перевищує last_seen_at більше ніж на допустиме значення, система змінює статус пристрою на «неактивний», створює подію про втрату зв’язку та формує сповіщення.

3.4 Реалізація користувацького інтерфейсу

Користувацький інтерфейс програмного засобу реалізується як вебзастосунок, що взаємодіє із серверною частиною через REST API. Його основне призначення полягає у відображенні стану IoT-пристроїв, телеметричних даних, журналу подій, сповіщень і засобів управління життєвим циклом. Інтерфейс повинен бути зрозумілим для адміністратора або оператора, забезпечувати швидке виявлення проблем і надавати доступ до основних функцій системи.

Клієнтську частину доцільно реалізувати на основі React. Компонентний підхід дає змогу поділити інтерфейс на незалежні елементи: навігаційну панель, картки статистики, таблицю пристроїв, індикатори стану, форму додавання пристрою, блок телеметрії, журнал подій і модальні вікна підтвердження дій [21].

Після авторизації користувач потрапляє на головну сторінку моніторингу. Вона повинна містити узагальнену статистику про IoT-інфраструктуру: загальну кількість зареєстрованих пристроїв, кількість активних пристроїв, кількість неактивних пристроїв, кількість пристроїв у стані попередження, кількість аварійних пристроїв і кількість активних сповіщень. Така інформація дозволяє користувачу швидко оцінити загальний стан системи.

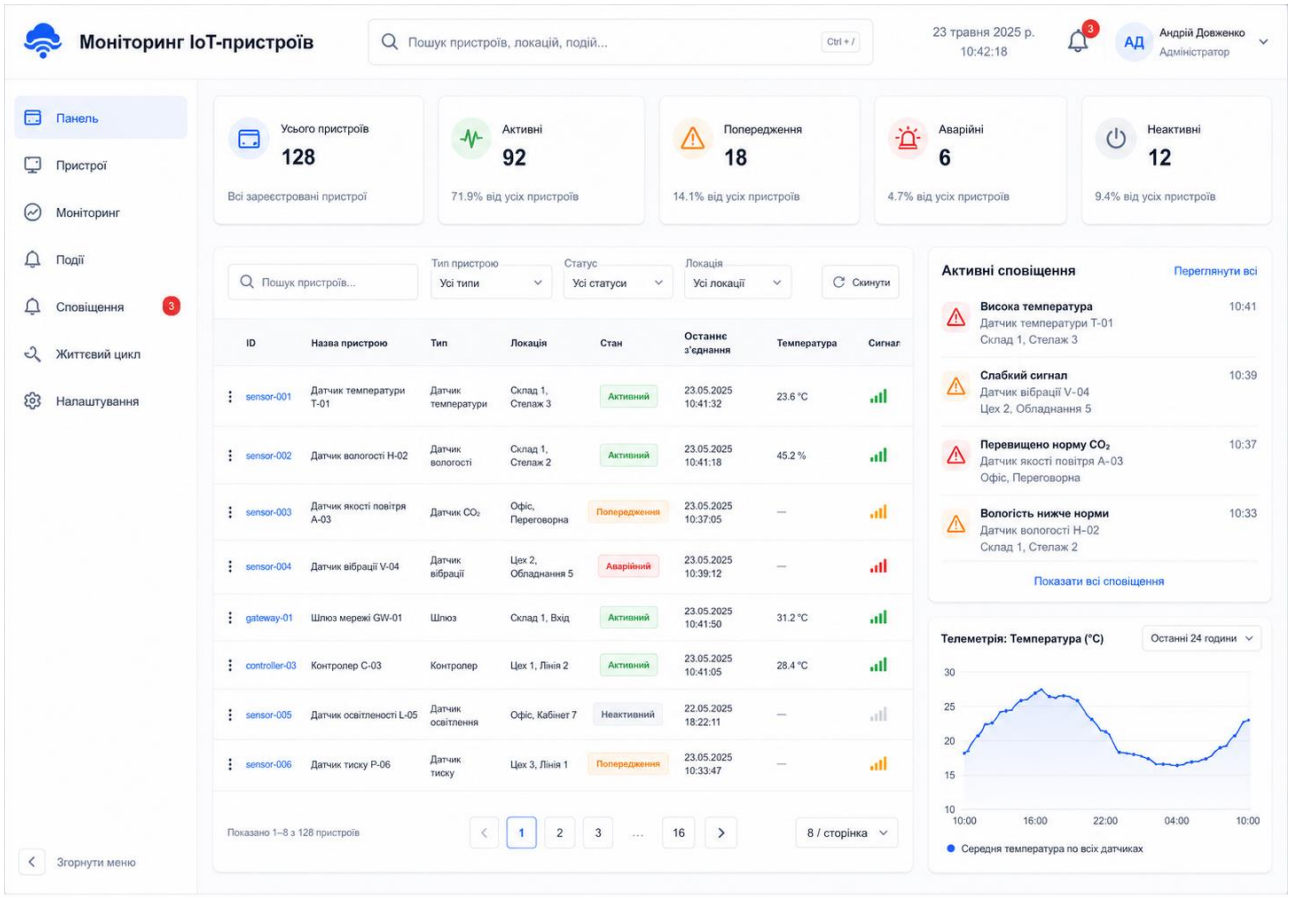

Рисунок 3.3 – Інтерфейс сторінки моніторингу IoT-пристроїв

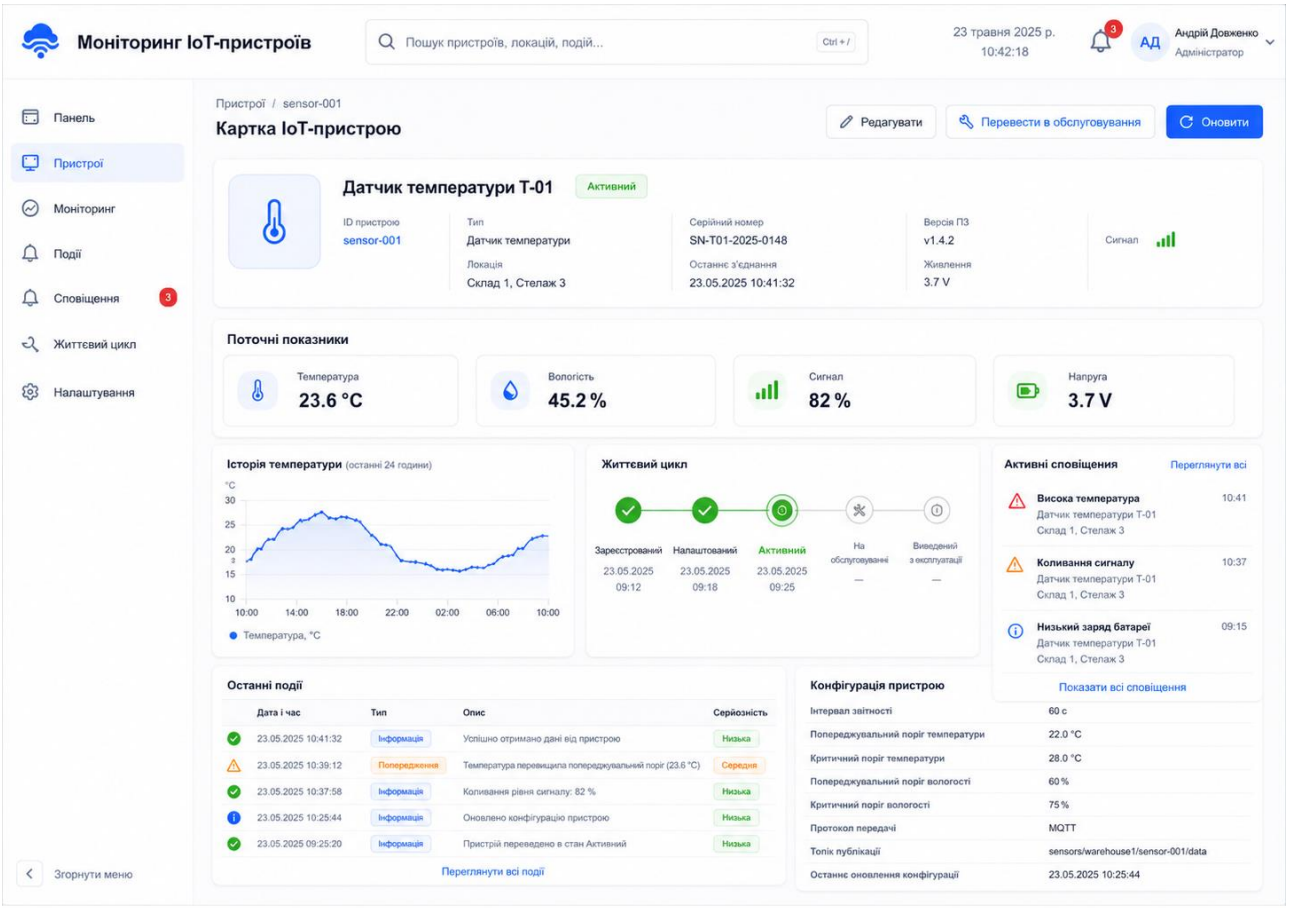

Рисунок 3.4 – Інтерфейс картки окремого IoT-пристрою

Рисунок 3.4 – Інтерфейс картки окремого IoT-пристрою

У картці пристрою потрібно передбачити блок життєвого циклу. У цьому блоці відображається поточний стан пристрою та доступні дії: активувати, перевести в обслуговування, повернути до активного стану, деактивувати або вивести з експлуатації. Під час виконання дії користувач має вказати причину зміни статусу. Ця причина зберігається в історії життєвого циклу.

3.5 Тестування програмного засобу та аналіз результатів

Тестування програмного засобу є необхідним етапом перевірки його працездатності та відповідності сформованим вимогам. Метою тестування є підтвердження того, що система коректно виконує основні функції: реєстрацію пристроїв, приймання телеметрії, перевірку даних, оновлення статусів, формування подій, створення сповіщень і підтримку змін життєвого циклу.

Таблиця 3.3 – Результати тестування програмного засобу

Тестовий сценарій	**Вхідні дані**	**Очікуваний результат**	**Фактичний результат**	**Висновок**
Реєстрація нового пристрою	Назва, тип, deviceId, місце встановлення	Пристрій створено та відображено у списку	Пристрій успішно додано	Виконано
Приймання коректної телеметрії	JSON із коректними параметрами	Дані збережено, статус оновлено	Телеметрію збережено, статус активний	Виконано
Некоректна структура повідомлення	JSON без deviceId	Дані відхилено, створено подію	Створено подію про помилку	Виконано
Повідомлення від невідомого пристрою	deviceId відсутній у базі	Повідомлення відхилено	Створено технічний запис	Виконано
Перевищення попереджувального порогу	Температура вище допустимого рівня	Статус «попередження», запис у журналі	Статус змінено, подію створено	Виконано
Перевищення критичного порогу	Критичне значення температури	Статус «аварійний», сповіщення	Створено аварійну подію і сповіщення	Виконано
Втрата зв’язку	Відсутність телеметрії понад допустимий час	Статус «неактивний», сповіщення	Статус оновлено, подію створено	Виконано
Переведення в обслуговування	Команда користувача із причиною	Статус змінено, історію збережено	Запис історії створено	Виконано
Деактивація пристрою	Команда деактивації	Пристрій не бере участі в активному моніторингу	Статус змінено на деактивований	Виконано

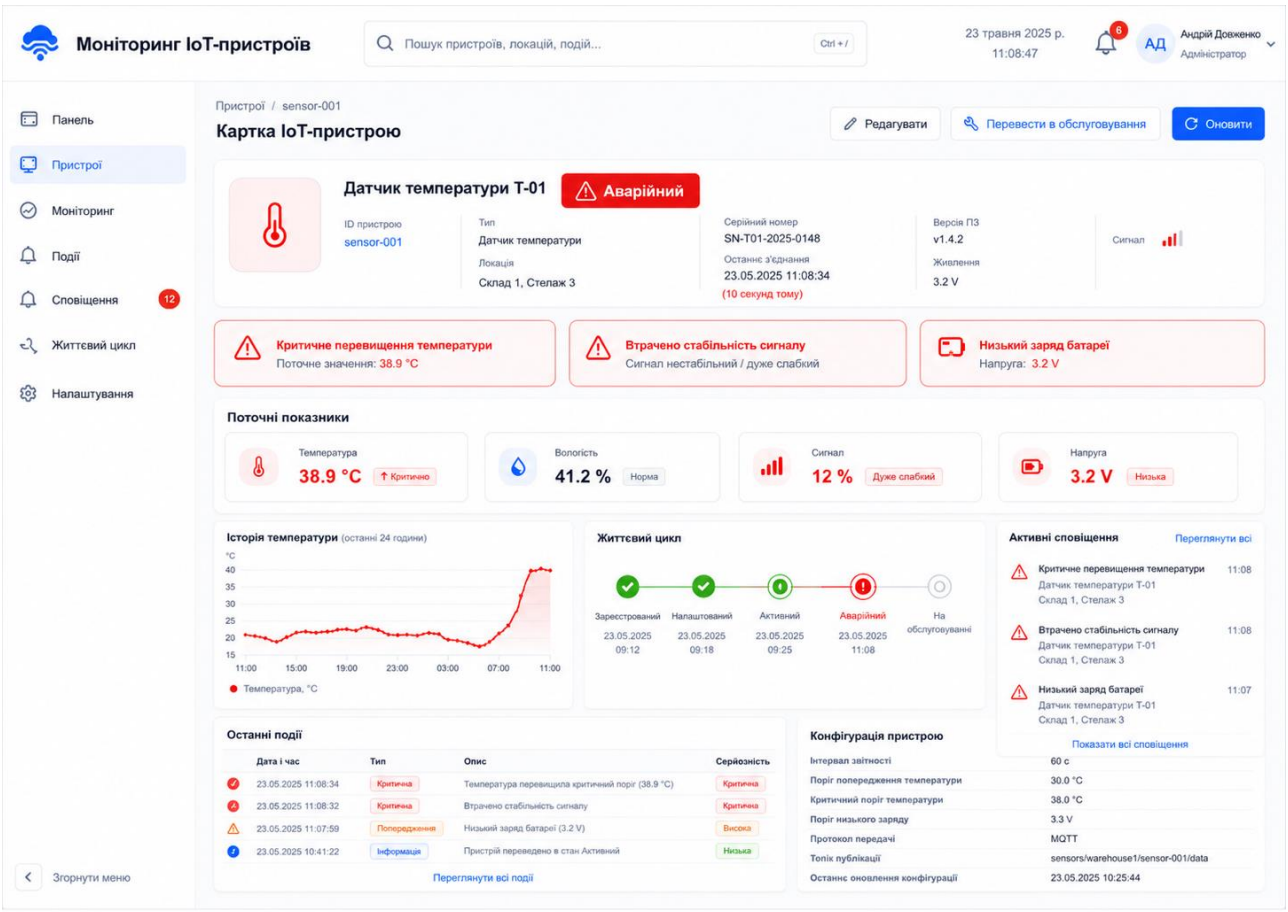

Рисунок 3.5 – Приклад відображення аварійного стану IoT-пристрою

Аналіз результатів тестування показує, що програмний засіб виконує основні функції, визначені на етапі формування вимог. Система коректно створює записи пристроїв, приймає телеметрію, перевіряє її структуру, зберігає історію показників, змінює статуси пристроїв, фіксує події та формує сповіщення. Особливо важливо, що система реагує не лише на вхідні повідомлення, а й на їх відсутність, що є суттєвим для моніторингу IoT-пристроїв.

3.6 Висновки до третього розділу

У третьому розділі бакалаврської кваліфікаційної роботи розглянуто практичну реалізацію програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом. Обґрунтовано вибір інструментальних

засобів розробки, зокрема Node.js, Express.js, PostgreSQL, React, Eclipse Mosquitto, REST API, JSON, JWT і OpenAPI. Обраний стек технологій забезпечує можливість реалізації клієнт-серверної системи з підтримкою приймання телеметрії, збереження даних, обробки подій і взаємодії користувача з вебінтерфейсом.

Описано реалізацію серверної частини програмного засобу, яка включає модулі авторизації, керування пристроями, приймання телеметрії, формування подій, сповіщень і управління життєвим циклом. Розкрито логіку роботи модуля моніторингу та обробки телеметрії. Також описано реалізацію користувацького інтерфейсу, який містить головну панель моніторингу, список пристроїв, картку окремого пристрою, журнал подій, сторінку сповіщень і блок управління життєвим циклом. Проведене тестування підтвердило, що програмний засіб коректно виконує основні функції, визначені у вимогах.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Питання щодо безпеки життєдіяльності

Безпека життєдіяльності є важливою складовою організації професійної діяльності фахівця у сфері комп'ютерних наук, оскільки робота з програмними засобами, комп'ютерною технікою, мережевим обладнанням та IoT-пристроями пов'язана з низкою потенційних ризиків. У межах розробки програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом питання безпеки життєдіяльності охоплюють як безпечну роботу користувача за комп'ютером, так і безпечну експлуатацію технічної інфраструктури, що включає серверне обладнання, мережеві пристрої, сенсорні модулі та джерела живлення.

Основними чинниками, які можуть негативно впливати на працівника під час роботи з програмним засобом, є тривале перебування за комп'ютером, зорове навантаження, статична поза, монотонність роботи, інформаційне перевантаження, електромагнітний вплив технічних засобів, шум від обладнання, ризик ураження електричним струмом і можливість виникнення пожежонебезпечних ситуацій. Крім того, у процесі обслуговування IoT-пристроїв можуть виникати ризики, пов'язані з підключенням сенсорів, мікроконтролерів, блоків живлення та комутаційного обладнання.

Робоче місце користувача програмного засобу має бути організоване відповідно до ергономічних вимог. Монітор необхідно розміщувати на відстані приблизно 50–70 см від очей користувача, а верхній край екрана має бути на рівні очей або трохи нижче. Клавіатура і миша повинні розміщуватися так, щоб зап'ястя не перебували в напруженому положенні. Стілець має забезпечувати підтримку спини, а висота робочої поверхні повинна давати змогу працювати без надмірного нахилу корпуса. Раціональна організація робочого місця зменшує ризик перевтоми, порушення постави та розвитку професійного дискомфорту.

Особливу увагу потрібно приділяти режиму праці та відпочинку. Тривала безперервна робота за комп’ютером може спричиняти зорову втому, головний біль, зниження концентрації уваги та загальне погіршення самопочуття. Для зменшення негативного впливу рекомендується робити короткі перерви через кожні 45–60 хвилин роботи, виконувати вправи для очей, змінювати положення тіла та уникати тривалого статичного навантаження. Під час роботи з системою моніторингу IoT-пристроїв оператор може одночасно аналізувати велику кількість показників, сповіщень і подій, тому важливо уникати надмірного інформаційного навантаження.

У контексті розроблюваного програмного засобу безпека життєдіяльності також пов’язана з правильним візуальним поданням інформації. Інтерфейс системи повинен бути зрозумілим, не перевантаженим зайвими елементами та мати чітке розмежування штатних, попереджувальних і аварійних станів. Надмірна кількість повідомлень або нечітке виділення критичних подій може призвести до помилок оператора, несвоєчасного реагування на несправності або неправильного трактування стану IoT-пристроїв. Тому в системі доцільно використовувати зрозумілі індикатори станів, пріоритетність сповіщень, фільтрацію подій і логічне групування інформації.

Важливою складовою безпеки є електробезпека. Комп’ютери, сервери, мережеве обладнання, маршрутизатори, блоки живлення та IoT-пристрої повинні експлуатуватися лише у справному стані. Забороняється використовувати пошкоджені кабелі, нестабільні джерела живлення, несправні розетки або саморобні з’єднання без належної ізоляції. Підключення обладнання повинно виконуватися з урахуванням допустимого навантаження електромережі. Усі пристрої бажано підключати через справні мережеві фільтри або джерела безперебійного живлення, особливо якщо система моніторингу використовується для безперервного контролю важливих параметрів.

Під час роботи з IoT-пристроями необхідно враховувати особливості їх розміщення. Пристрої можуть встановлюватися в технічних приміщеннях, на

складах, у виробничих зонах, серверних кімнатах або поблизу інженерного обладнання. У таких умовах потрібно дотримуватися правил безпечного доступу до обладнання, уникати контакту з відкритими струмопровідними частинами, не виконувати підключення у вологому середовищі та не проводити монтажні роботи без вимкнення живлення. Якщо пристрій розміщується в зоні підвищеної температури, вологості або запиленості, необхідно застосовувати відповідний корпус і захист від зовнішніх впливів.

Пожежна безпека також має важливе значення. Причинами пожежонебезпечних ситуацій можуть бути коротке замикання, перевантаження електромережі, перегрів блоків живлення, неправильна експлуатація акумуляторів, накопичення пилу в обладнанні або використання несертифікованих компонентів. Для зменшення ризику необхідно забезпечити справність електропроводки, не перевантажувати розетки, не залишати несправне обладнання без нагляду, регулярно перевіряти стан кабелів і блоків живлення. У приміщенні, де розміщено комп’ютерне або серверне обладнання, повинні бути доступні первинні засоби пожежогасіння.

Окремо слід зазначити роль програмного засобу в підвищенні рівня безпеки життєдіяльності. Система моніторингу IoT-пристроїв може своєчасно виявляти аварійні стани, втрату зв’язку, критичні значення параметрів, низький заряд батареї або нестабільність сигналу. Це дає змогу оператору швидше реагувати на потенційно небезпечні ситуації та запобігати їх розвитку. Наприклад, якщо IoT-пристрій контролює температуру обладнання або стан приміщення, своєчасне сповіщення про перевищення допустимих значень може запобігти перегріву, пошкодженню техніки або виникненню аварійної ситуації.

У процесі експлуатації програмного засобу важливо забезпечити надійність збереження даних і доступність інформації для відповідальних осіб. Втрата телеметричних даних або недоступність системи в критичний момент може негативно вплинути на безпеку об’єкта. Тому доцільно передбачати резервне копіювання бази даних, контроль працездатності серверної частини, захищений доступ користувачів і журналювання подій. Такі заходи мають не

лише технічне, а й безпекове значення, оскільки забезпечують збереження інформації про події, несправності та дії користувачів.

Отже, питання безпеки життєдіяльності у межах розробки та експлуатації програмного засобу моніторингу IoT-пристроїв охоплюють ергономіку робочого місця, безпечну роботу з комп'ютерною технікою, електробезпеку, пожежну безпеку, правильну експлуатацію IoT-обладнання та зменшення інформаційного навантаження на користувача. Дотримання цих вимог сприяє безпечній, стабільній і ефективній роботі системи, а також знижує ризики для користувачів, обладнання та об'єктів, на яких застосовуються IoT-пристрої..

4.2 Питання з основ охорони праці

Охорона праці є системою правових, організаційних, технічних, санітарно-гігієнічних і профілактичних заходів, спрямованих на збереження життя, здоров'я та працездатності працівника у процесі професійної діяльності. Для фахівця у сфері комп'ютерних наук, який займається розробкою, тестуванням і супроводом програмного забезпечення, питання охорони праці мають особливе значення, оскільки значна частина роботи виконується за комп'ютером, із використанням електронного обладнання, мережевих пристроїв і технічної документації.

У межах бакалаврської кваліфікаційної роботи, присвяченої розробці програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом, охорона праці стосується як процесу створення програмного продукту, так і умов його подальшої експлуатації. Розробник працює з комп'ютером, середовищами програмування, базами даних, серверними компонентами, емуляторами IoT-пристроїв і мережевими сервісами. Оператор або адміністратор системи використовує програмний засіб для контролю стану пристроїв, аналізу подій і реагування на сповіщення. Тому необхідно забезпечити такі умови праці, які мінімізують ризики для здоров'я працівника та сприяють стабільному виконанню професійних завдань.

Основними шкідливими та небезпечними чинниками під час роботи з комп’ютерною технікою є зорове навантаження, нервово-емоційне напруження, тривале статичне положення тіла, монотонність виконуваних операцій, недостатнє або надмірне освітлення, шум від технічного обладнання, несприятливий мікроклімат, а також ризик ураження електричним струмом. У разі роботи з IoT-пристроями додатково можуть виникати ризики, пов’язані з монтажем, підключенням, тестуванням сенсорних модулів, блоків живлення та комунікаційного обладнання.

Організація робочого місця розробника або оператора повинна відповідати ергономічним вимогам. Робочий стіл має забезпечувати достатній простір для розміщення монітора, клавіатури, миші, документації та іншого обладнання. Монітор потрібно встановлювати так, щоб уникати відблисків, надмірного нахилу голови та напруження очей. Освітлення має бути рівномірним, достатнім для читання документації й роботи з екраном, але не повинно створювати різких тіней або засліплення. Бажано використовувати природне освітлення в поєднанні зі штучним, а робоче місце розташовувати так, щоб світло не падало безпосередньо на екран.

Суттєве значення має правильна поза працівника. Тривале перебування в неправильному положенні може спричинити втому м’язів, біль у шиї, спині, плечах і кистях. Працівник повинен сидіти прямо, з опорою для спини, а ноги мають стояти на підлозі або спеціальній підставці. Клавіатура й миша повинні розміщуватися на зручній відстані, щоб руки не перебували в постійному напруженні. Робочий стілець має бути стійким, регульованим за висотою та забезпечувати підтримку поперекового відділу спини.

Для зменшення зорової втоми важливо дотримуватися раціонального режиму роботи. Під час тривалої роботи за комп’ютером рекомендується робити короткі перерви, змінювати фокус зору, виконувати прості вправи для очей і періодично відводити погляд від екрана. Особливо це важливо для розробників програмного забезпечення, які працюють із кодом, таблицями, схемами баз даних, інтерфейсами та технічною документацією. Висока

концентрація уваги протягом тривалого часу може призводити до помилок, зниження продуктивності й перевтоми.

Психофізіологічні чинники також є важливими для охорони праці. Розробка програмного забезпечення, тестування системи моніторингу, аналіз помилок і робота з аварійними повідомленнями можуть супроводжуватися значним інформаційним навантаженням. Для зменшення цього навантаження потрібно раціонально планувати робочий час, розподіляти складні завдання на окремі етапи, уникати тривалої безперервної роботи без відпочинку та використовувати зрозумілі інструменти контролю виконання завдань. У контексті розроблюваного програмного засобу важливо, щоб інтерфейс не створював додаткового стресу для оператора, а критичні повідомлення були чітко відокремлені від інформаційних.

Електробезпека є однією з основних вимог під час експлуатації комп'ютерного та IoT-обладнання. Усі технічні засоби повинні бути справними, мати надійне підключення до електромережі та використовуватися відповідно до їх призначення. Працівник не повинен самостійно ремонтувати блоки живлення, розбирати обладнання під напругою, використовувати пошкоджені кабелі або підключати пристрої до несправних розеток. Перед підключенням або заміною IoT-модулів необхідно переконатися у відсутності напруги на відповідних контактах, якщо це передбачено умовами роботи.

Під час тестування IoT-пристроїв потрібно враховувати особливості живлення мікроконтролерів, сенсорів і виконавчих модулів. Навіть якщо такі пристрої працюють із відносно низькою напругою, неправильне підключення може призвести до короткого замикання, пошкодження обладнання, перегріву компонентів або виникнення пожежонебезпечної ситуації. Тому необхідно використовувати справні з'єднувальні кабелі, стабілізовані джерела живлення, не перевантажувати макетні плати та контролювати температуру компонентів під час роботи.

Пожежна безпека під час роботи з комп'ютерною та мережевою технікою передбачає запобігання перегріву обладнання, короткому замиканню,

перевантаженню електромережі та накопиченню пилу. Робоче місце не повинно бути захаращене зайвими предметами, кабелі мають бути впорядковані, а вентиляційні отвори комп’ютерів і мережевого обладнання не повинні перекриватися. Забороняється залишати несправне обладнання увімкненим без нагляду. У разі появи запаху горілого, диму, іскор або нестабільної роботи електропристроїв потрібно негайно припинити роботу й вимкнути живлення.

Важливою складовою охорони праці є організаційні заходи. Працівник повинен бути ознайомлений із правилами безпечної роботи за комп’ютером, порядком дій у разі аварійної ситуації, правилами користування електрообладнанням і первинними засобами пожежогасіння. У приміщенні повинні бути забезпечені належні умови мікроклімату, вентиляції, освітлення та чистоти. Робочі проходи не повинні бути захаращені, а обладнання має бути розміщене так, щоб не створювати перешкод для евакуації. У контексті розроблюваного програмного засобу охорона праці також пов’язана з надійністю програмної системи. Некоректна робота системи моніторингу може призвести до несвоєчасного виявлення аварійного стану пристрою або неправильного інформування оператора. Тому під час розробки потрібно приділяти увагу тестуванню, перевірці вхідних даних, обробці помилок, журналюванню подій і захисту доступу. Надійне програмне забезпечення зменшує ризик помилкових дій користувача та сприяє безпечній експлуатації IoT-інфраструктури. Отже, охорона праці під час розробки й експлуатації програмного засобу моніторингу IoT-пристроїв охоплює комплекс технічних, організаційних, ергономічних і профілактичних заходів. До основних вимог належать правильна організація робочого місця, дотримання режиму праці та відпочинку, забезпечення електробезпеки, пожежної безпеки, зменшення зорового й психоемоційного навантаження, а також безпечна робота з IoT-обладнанням. Дотримання зазначених вимог сприяє збереженню здоров’я працівника, підвищенню ефективності роботи та надійній експлуатації розробленого програмного засобу.

Висновок до четвертого розділу

Забезпечення безпечних умов праці передбачає правильну організацію робочого місця, дотримання ергономічних вимог, раціональний режим праці та відпочинку, справність комп’ютерної техніки, виконання правил електробезпеки й пожежної безпеки. Дотримання цих вимог сприяє збереженню здоров’я виконавця, підвищенню працездатності, зменшенню ризику помилок під час програмної розробки та забезпеченню безперервності роботи над інформаційною системою.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі розв’язано задачу розробки програмного засобу для моніторингу стану IoT-пристроїв та управління їх життєвим циклом. У процесі виконання роботи досліджено предметну область Internet of Things, визначено особливості функціонування IoT-пристроїв у розподілених інформаційних системах і обґрунтовано потребу в централізованому контролі їх працездатності, доступності, телеметричних параметрів, подій і статусів життєвого циклу.

У роботі проаналізовано типову структуру IoT-системи, життєвий цикл IoT-пристрою, наявні рішення для моніторингу та протоколи MQTT, HTTP/HTTPS, WebSocket і CoAP. Визначено, що моніторинг IoT-пристроїв не обмежується відображенням вимірювальних значень, а має охоплювати контроль доступності, справності, часу останнього з’єднання, рівня живлення, помилок, аварійних подій і станів обслуговування.

У проектній частині спроектовано архітектуру програмного засобу, інформаційну модель бази даних, алгоритм моніторингу, механізм управління життєвим циклом і структуру користувацького інтерфейсу. Запропонована архітектура передбачає використання IoT-пристроїв або емуляторів, MQTT-брокера, серверної частини, бази даних і клієнтського вебінтерфейсу, що забезпечує логічне розділення функцій приймання телеметрії, обробки даних, формування подій і відображення стану пристроїв.

У практичній частині обґрунтовано вибір інструментальних засобів розробки: Node.js, Express.js, PostgreSQL, React, Eclipse Mosquitto, REST API, JSON, JWT і OpenAPI. Описано реалізацію серверної частини, модуля обробки телеметрії, користувацького інтерфейсу, журналу подій, сповіщень і механізму зміни статусів життєвого циклу. Проведене тестування підтвердило коректність виконання основних функцій: реєстрації пристроїв, приймання телеметрії, обробки некоректних повідомлень, виявлення аварійних станів, втрати зв’язку та зміни статусу пристрою.

У четвертій частині розглянуто питання безпеки життєдіяльності та охорони праці які є є невід’ємною складовою виконання бакалаврської кваліфікаційної роботи. Їх урахування дозволяє створити безпечні, раціональні та ефективні умови для розробки, тестування й подальшого використання системи підтримки прийняття рішень.

ПЕРЕЛІК ДЖЕРЕЛ

1 Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4), 2347–2376.

2 Sethi, P., & Sarangi, S. R. (2017). Internet of Things: Architectures, protocols, and applications. Journal of Electrical and Computer Engineering, 2017.

3 Salman, T., & Jain, R. (2019). A survey of protocols and standards for Internet of Things.

4 OASIS. (2019). MQTT Version 5.0. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

5 Shelby, Z., Hartke, K., & Bormann, C. (2014). The Constrained Application Protocol (CoAP). RFC 7252. Internet Engineering Task Force.

6 Fielding, R., Nottingham, M., & Reschke, J. (2022). HTTP Semantics. RFC 9110. Internet Engineering Task Force.

7 Fette, I., & Melnikov, A. (2011). The WebSocket Protocol. RFC 6455. Internet Engineering Task Force.

8 ThingsBoard. ThingsBoard documentation. Retrieved June 14, 2026, Джерело: https://thingsboard.io/docs/

9 Amazon Web Services. AWS IoT Core Developer Guide. Retrieved June 14, 2026, Джерело:https://docs.aws.amazon.com/iot/

10 Microsoft. What is Azure IoT Hub? Microsoft Learn. Retrieved June 14, 2026, Джерело: https://learn.microsoft.com/azure/iot-hub/

11 Node-RED. Node-RED documentation. Retrieved June 14, 2026, Джерело: https://nodered.org/docs/

12 Prometheus Authors. Prometheus documentation: Overview. Retrieved June 14, 2026, Джерело: https://prometheus.io/docs/introduction/overview/

13 Grafana Labs. Grafana documentation. Retrieved June 14, 2026, Джерело: https://grafana.com/docs/grafana/latest/

14 Zabbix LLC. Zabbix documentation. Retrieved June 14, 2026, Джерело: https://www.zabbix.com/documentation/

15 Fagan, M., Megas, K. N., Scarfone, K., & Smith, M. (2020). IoT device cybersecurity capability core baseline. NISTIR 8259A. National Institute of Standards and Technology.

16 Yousefnezhad, N., Malhi, A., & Främling, K. (2020). Security in product lifecycle of IoT devices: A survey. Journal of Network and Computer Applications, 171, Article 102779.

17 Miettinen, M., van Oorschot, P. C., & Sadeghi, A.-R. (2018). Baseline functionality for security and control of commodity IoT devices and domain-controlled device lifecycle management.

18 Sadhu, P. K., Yanambaka, V. P., & Abdelgawad, A. (2022). Internet of Things: Security and solutions survey. Sensors, 22(19), Article 7433. https://doi.org/10.3390/s22197433

19 Node.js. Node.js documentation. Retrieved June 14, 2026, Джерело: https://nodejs.org/docs/

20 Express.js. Express: Node.js web application framework. Retrieved June 14, 2026, Джерело: https://expressjs.com/

21 React. (n.d.). React documentation. Retrieved June 14, 2026, Джерело: https://react.dev/

22 PostgreSQL Global Development Group. PostgreSQL documentation. Retrieved June 14, 2026, from https://www.postgresql.org/docs/

23 MongoDB. MongoDB documentation. Retrieved June 14, 2026, Джерело: https://www.mongodb.com/docs/

24 Eclipse Foundation. Eclipse Mosquitto documentation. Retrieved June 14, 2026, from https://mosquitto.org/documentation/

25 OpenAPI Initiative. (2025). OpenAPI Specification Version 3.2.0. Retrieved June 14, 2026, Джерело: https://spec.openapis.org/oas/v3.2.0.html

ДОДАТКИ

Додаток А

Фрагменти програмного коду системи

Приклад телеметричного повідомлення IoT-пристрою

```
{
  "deviceId": "sensor-001",
  "timestamp": "2026-06-14T10:25:00",
  "temperature": 24.8,
  "humidity": 58,
  "voltage": 3.7,
  "signalLevel": 82,
  "errorCode": null,
  "status": "OK"
}
```

Приклад структури таблиць бази даних

```
CREATE TABLE devices (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    device_key VARCHAR(100) UNIQUE NOT NULL,
    type_id INTEGER,
    location VARCHAR(150),
    status VARCHAR(50) DEFAULT 'registered',
    last_seen_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE telemetry_records (
    id SERIAL PRIMARY KEY,
    device_id INTEGER REFERENCES devices(id),
    timestamp TIMESTAMP NOT NULL,
    payload JSONB NOT NULL,
    is_valid BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE event_log (
    id SERIAL PRIMARY KEY,
    device_id INTEGER REFERENCES devices(id),
    event_type VARCHAR(50),
    severity VARCHAR(30),
    message TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```