НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Приладобудівний факультет
Кафедра автоматизації та систем неруйнівного контролю

«До захисту допущено»
В. о. завідувача кафедри
_____ Галина БОГДАН

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерно-інтегровані системи та технології в приладобудуванні»

спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

на тему: «Мережева система виявлення загрози займання»

Виконав:
студент IV курсу, групи ПМ-11
Совгуть Богдан Васильович _____

Керівник:
доцент, к.т.н.
Богдан Галина Анатоліївна _____

Рецензент:
доцент, к.т.н. Козир Олег Васильович _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2025 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4	ДП 11.13. 00.000 ПЗ	Пояснювальна записка		
2	А1	ДП 11.13. 01.001 Е1	Схема структурна		
3	А1	ДП 11.13. 05.005	Алгоритм		

				ДП ПК11.13.1760.00.000		
	ПІБ	Підп.	Дата			
Розробник	Совгуть Б.В.			Відомість дипломного проекту	Лист	Листів
Керівник	Богдан Г.А.				1	67
Консульт.					КПІ ім. Ігоря Сікорського Каф. АСНК	
Н/контр.						
В.о.зав. каф.	Галина Богдан				Гр. ПМ-11	

Пояснювальна записка
до дипломного проєкту
на тему: «Мережева система виявлення загрози займання»

Київ – 2025 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра автоматизації та систем неруйнівного контролю

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп’ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютерно-інтегровані системи та технології в приладобудуванні»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Галина БОГДАН

«___»_____2025 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Совгуть Богдан Васильович

1. Тема роботи «Мережева система виявлення загрози займання», керівник роботи Богдан Галина Анатоліївна, кандидат технічних наук, доцент, затверджені наказом по університету від «26» травня 2025 р. №1765-с

2. Термін подання студентом проєкту 1.06.2025

3. Вихідні дані до проєкту: розробка мережевої системи виявлення загрози займання.

4. Зміст пояснювальної записки

1. Вступ
2. Аналітичний огляд
3. Розробка
4. Висновки
5. Список використаних джерел

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) ? креслиників ? плакати

1 – Схема електрична структурна

2 – Алгоритм

3 – Плакат 1

4 – Плакат 2

5 – Презентація

6. Дата видачі завдання 01.02.2025_____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Формулювання завдання проекту	01.02.2025	
2	Проведення аналітичного огляду	18.05.2025	
3	Розробка	21.05.2025	
4	Розробка алгоритму роботи	28.05.2025	
5	Подача диплому	01.06.2025	

Студент Богдан СОВГУТЬ

Керівник Галина БОГДАН

Анотація

Дана кваліфікаційна робота присвячена розробці сучасного web-застосунку для мережевої системи виявлення загроз займання, що функціонує в режимі реального часу. У роботі акцентовано увагу на актуальності проблеми пожежної безпеки в умовах ускладнення інженерних мереж, підвищеного навантаження на електросистеми та зростання ризику виникнення пожеж. Традиційні системи зазвичай реагують на вже наявні ознаки пожежі, такі як дим або підвищення температури, тоді як запропоноване рішення орієнтоване на превентивне виявлення загрозових станів шляхом постійного моніторингу критичних параметрів: температури, вологості, стрибків напруги, концентрації газів тощо.

У теоретичній частині проаналізовано фізичні принципи горіння, підходи до обробки сигналів датчиків, сучасні методи фільтрації помилкових спрацювань, а також архітектурні особливості інтелектуальних систем безпеки. Дослідження підтверджує ефективність поєднання традиційних засобів детекції з інноваційними підходами, зокрема використанням алгоритмів машинного навчання, багатопараметричного аналізу та адаптивної фільтрації.

З технічного боку реалізовано гнучку та масштабовану клієнт-серверну архітектуру. Фронтенд побудований із використанням сучасного стеку технологій: Vue 3, Nuxt 4, Tailwind CSS і Nuxt Charts. Це забезпечило високий рівень реактивності інтерфейсу, автоматичне оновлення даних, кольорову індикацію станів та генерацію сповіщень. Серверна частина реалізована на Node.js із використанням Hono.js та PostgreSQL з ORM Drizzle для зберігання даних. Використання Docker забезпечило легке розгортання та стабільність роботи в різних середовищах.

Особливу увагу приділено модульності системи: реалізовано окремі компоненти для збору даних, їхньої обробки, аналізу, формування повідомлень та керування інтерфейсом. Алгоритми аналізу базуються на порогових значеннях та статистичному аналізі з використанням ковзних вікон і градієнтної оцінки. Управління станом системи реалізоване через Pinia, а

завдяки використанню VueUse composables досягнуто зменшення дублювання коду та спрощення логіки інтерфейсу.

Система пройшла тестування в умовах, максимально наближених до реального застосування. Було перевірено стабільність усіх функціональних модулів, точність обробки даних, відображення графіків, сповіщення про загрози та здатність адаптації до нових типів сенсорів. Результати демонструють високу точність виявлення загроз та мінімізацію хибних спрацювань.

Розроблена система має значний потенціал для впровадження у сфері безпеки, як у житлових, так і промислових об’єктах. Завдяки централізованому web-застосунку, забезпечується зручний доступ до системи з будь-якого пристрою, можливість інтеграції з іншими системами автоматизації будівель і розширення відповідно до потреб користувача. Це відповідає сучасним тенденціям розвитку технологій у рамках концепції «розумного дому» (Smart Home) та промислового Інтернету речей (IIoT).

Робота складається з вступу, 3 розділів, висновків, 19 рисунків, списку використаних джерел із 24 позицій та додатків. Загальний обсяг роботи – 66 сторінок, з яких основна частина викладена на 54 сторінках.

Ключові слова: автоматизація, web застосунок, пожежа, сенсори.

Annotation

This qualification thesis is dedicated to the development of a modern web application for a network-based fire threat detection system operating in real-time. The work emphasizes the relevance of fire safety issues amid the increasing complexity of engineering infrastructure, growing loads on electrical systems, and the rising risk of fire outbreaks. Traditional systems typically react to the physical presence of fire indicators such as smoke or elevated temperature, whereas the proposed solution focuses on preventive detection by continuously monitoring critical parameters: temperature, humidity, voltage surges, gas concentrations, and more.

The theoretical part analyzes the physical principles of combustion, approaches to sensor signal processing, modern methods for false alarm filtering, and architectural features of intelligent safety systems. The research confirms the effectiveness of combining traditional detection methods with innovative approaches, including the use of machine learning algorithms, multiparametric analysis, and adaptive filtering.

From a technical perspective, a flexible and scalable client-server architecture was implemented. The frontend is built using a modern technology stack: Vue 3, Nuxt 4, Tailwind CSS, and Nuxt Charts, ensuring high interface responsiveness, automatic data updates, state color indication, and alert generation. The backend is implemented with Node.js using Hono.js, and data is stored in PostgreSQL using Drizzle ORM. Docker was used to simplify deployment and ensure system stability in various environments.

Particular attention was paid to the modularity of the system: separate components were developed for data collection, processing, analysis, notification generation, and interface control. The analytical algorithms are based on threshold values and statistical methods using sliding windows and gradient evaluation. System state management is implemented with Pinia, and the use of VueUse composables helped reduce code duplication and simplify the interface logic.

The system was tested in conditions closely approximating real-world operation. All functional modules were verified for stability, data processing

accuracy, graph rendering, threat notifications, and adaptability to new sensor types. The results demonstrate high accuracy in threat detection and minimal false alarms.

The developed system has significant potential for implementation in safety systems, both in residential and industrial environments. The centralized web application enables convenient access from any device, integration with other building automation systems, and scalability according to user needs. This aligns with modern technological trends under the Smart Home and Industrial Internet of Things (IIoT) concepts.

The thesis consists of an introduction, three chapters, conclusions, 19 figures, a list of 24 references, and appendices. The total length of the work is 66 pages, with the main content presented on 54 pages.

Keywords: automation, web application, fire, sensors.

ЗМІСТ

` ВСТУП

Забезпечення пожежної безпеки є пріоритетним напрямком техногенної та екологічної безпеки на об’єктах різного призначення, починаючи від виробничих потужностей і закінчуючи житловими будівлями. В сучасних умовах, коли інженерні мережі стають все більш складними, а навантаження на електричні системи збільшується, значно зростає ризик виникнення пожеж. Це обумовлює необхідність впровадження ефективних систем моніторингу та раннього виявлення загроз займання.

Традиційні пожежні системи, як правило, реагують вже на факт виникнення пожежі, за допомогою датчиків диму чи температури. Проте новітні підходи зосереджені на попередженні загоряння, шляхом постійного аналізу параметрів, які можуть вказувати на потенційну небезпеку, наприклад перегрів, стрибки напруги, підвищення рівня вологості або концентрації газів.

У зв'язку з цим, актуальною є розробка мережевої системи виявлення загрози займання, здатної в режимі реального часу відстежувати показники з датчиків, аналізувати їх за певними критеріями та сповіщати відповідальних осіб про можливу небезпеку. Важливою частиною такої системи є web-застосунок, який забезпечує зручну візуалізацію даних, керування сповіщеннями та доступ з будь-якого пристрою.

Застосування web-застосунку дає можливість централізувати контроль, забезпечувати дистанційний доступ до системи, а також інтегрувати її з іншими компонентами автоматизації будівлі. Це робить її особливо ефективною в рамках концепції «розумного будинку» та промислового Інтернету речей (IIoT).

Використання web-застосунку дозволяє централізувати контроль, забезпечити віддалений доступ до системи, а також інтегрувати її з іншими елементами автоматизації будівлі. Це робить її особливо ефективною в контексті концепції «розумного дому» та промислового Інтернету речей (IIoT).

Метою даної кваліфікаційної роботи є розробка web-застосунку для мережевої системи виявлення загрози займання, який забезпечує збір, обробку

та виведення даних із сенсорів у реальному часі, а також реалізація програмної логіки оцінки загрозових станів на основі заданих порогів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМ ВИЯВЛЕННЯ ПОЖЕЖНИХ ЗАГРОЗ

У наш час ми живемо в умовах стрімкого розвитку технологій і безперервного зростання міст, що безпосередньо впливає на потребу у вдосконаленні різних аспектів безпеки, зокрема пожежної. З кожним роком все більше уваги приділяється тому, як запобігати пожежам і швидко реагувати на можливі загрози. Завдяки активному впровадженню сучасних інформаційних технологій, а також розвитку мережевих інструментів, з'являється все більше рішень для створення «розумних» систем, які можуть автоматично відстежувати ситуацію і сигналізувати про потенційну небезпеку. У зв’язку з цим досить важливим є вивчення як теоретичних аспектів, так і практичних методів побудови ефективних систем, здатних своєчасно виявляти займання та зменшувати ризики.

1.1 Аналіз сучасного стану систем пожежної безпеки

Пожежна безпека, як галузь, зазнала значної трансформації за останні десятиліття. Якщо раніше це були доволі прості й обмежені в можливостях механічні пристрої, то сьогодні мова йде вже про цілісні та технологічно розвинені системи, що функціонують на основі складних алгоритмів і мережевих рішень. Завдяки досягненням сучасної науки та техніки стало можливим створювати такі системи, які не лише здатні швидко виявити займання ще на початкових етапах, а й можуть аналізувати ризики, передбачати ймовірність виникнення пожежі та навіть автоматично здійснювати відповідні дії для мінімізації шкоди.[1]

1.1.1 Еволюція систем пожежної сигналізації

Якщо поглянути на те, як розвивалися технології пожежної безпеки, то можна побачити справді захоплюючу історію інновацій. Все почалося ще в 1800-х роках, коли інженери винайшли перші механічні пристрої для виявлення вогню. Ці ранні детектори працювали на досить примітивних

принципах - наприклад, коли температура підвищувалася, метал розширювався і це спричиняло спрацьовування сигналізації. Звичайно, такі системи мали багато недоліків: вони повільно реагували і часто давали помилкові сигнали.[2]

Справжній прорив відбувся в першій половині минулого століття, коли технології електрики стали більш доступними. Саме тоді почали з'являтися термоелектричні сенсори, які вже могли передавати інформацію про небезпеку на значні відстані. Для того часу це було неймовірним досягненням! Але найбільше все змінилося в 60-ті роки, коли електроніка почала бурхливо розвиватися. Тоді з'явилися набагато більш чутливі та швидкі детектори, які могли краще розпізнавати реальні загрози.[3]

Сучасні системи - це вже зовсім інший рівень. Завдяки комп'ютерним технологіям та мережевим з'єднанням тепер існують "розумні" комплекси, які одночасно відстежують температуру, дим, гази та інші параметри. Особливо цікаво те, що використовуються алгоритми штучного інтелекту для підвищення точності - система "вчиться" розрізняти справжню пожежу від звичайного пару чи пилу. Крім того, ці системи легко інтегруються з іншими засобами безпеки будівлі, створюючи комплексний захист.

Окремо варто відзначити адресні технології - це була справжня революція в галузі. До їх появи було складно точно зрозуміти, де саме виникла проблема. Тепер же система може вказати конкретний датчик, який зафіксував загрозу, що дозволяє швидко та ефективно реагувати на ситуацію. Це кардинально змінило підходи до проектування та обслуговування протипожежних систем.[4]

1.1.2 Класифікація існуючих рішень виявлення займання

На даний момент у нас є досить широкий вибір детекторів для виявлення пожежної небезпеки. Власне, їх можна групувати за різними критеріями, але найчастіше розділяють саме за способом детектування загроз. Основні категорії включають температурні сенсори, детектори диму, змішані моделі та деякі спеціальні пристрої.

Температурні детектори - це такі, що реагують або на загальне підвищення температури в кімнаті, або на різке збільшення температури за дуже короткий час. Вони досить практичні в використанні та мають прийнятну вартість. Але є одне "але" - вони не дуже швидко спрацьовують, що іноді може бути критично важливим.

Щодо детекторів диму, то тут все трохи складніше. Їх розділяють на оптичні та іонізаційні моделі. Оптичні спрацьовують тоді, коли дим заважає проходженню світлового променя - частинки диму його розсіюють, і датчик це "помічає". А іонізаційні працюють по-іншому: вони відстежують, як змінюється здатність повітря проводити електрику в спеціальній камері з іонізованим повітрям.

Кожен тип має свої сильні сторони, але і обмеження теж є. Головне - це те, що ефективність залежить від того, який саме матеріал горить, бо різні речовини дають різний дим за своїми характеристиками. Тому при виборі конкретного детектора треба враховувати особливості приміщення та те, що там може загорітися.

Рис. 1.1 Класифікація датчиків пожежної сигналізації за принципом детекції

Під час вивчення систем пожежної безпеки виявляється, що їх можна розділити на групи не тільки за принципом роботи, але й за структурою мережі - тобто топологією системи. Якщо говорити про основні види, то можна виділити радіальні, кільцеві та комбіновані варіанти.

Розглядаючи радіальні системи, слід зазначити, що тут усе досить зрозуміло - є один центральний блок управління, від якого залежить уся мережа. Це дуже спрощує налаштування та встановлення такої системи. Але є одна велика проблема: якщо цей головний елемент ламається, то вся система припиняє працювати. Це як з деревом - якщо пошкодити стовбур, то всі гілки перестануть отримувати живлення.

Кільцеві системи влаштовані інакше. Тут немає одного головного центру, а всі елементи з'єднані у замкнуте коло. Така конструкція складніша, але й надійніша. Якщо десь обривається зв'язок, сигнал може пройти іншим шляхом через кільце. Тому навіть при пошкодженні однієї ділянки система продовжує функціонувати.

Комбіновані системи - це коли поєднуються обидва підходи. Це дає можливість підлаштувати систему під конкретні потреби будівлі чи об'єкта.
Але топологія - це не єдиний спосіб класифікації. Дуже важливо також розуміти, наскільки "розумною" є система. Тут є три основні типи: аналогові, адресні та інтелектуальні.

Аналогові системи передають інформацію постійним сигналом від датчика до пульта управління. Вони прості у використанні, але коли спрацьовує сигнал тривоги, не завжди можна швидко зрозуміти, де саме виникла проблема.

Адресні системи вже краще - вони можуть "сказати", який конкретно датчик подав сигнал. Це дуже полегшує пошук місця, де почалася пожежа.

Найсучасніші - це інтелектуальні системи. Вони вміють не просто передавати дані, а й самостійно їх аналізувати, перевіряти свою роботу на несправності та пристосовуватися до змін в оточенні.

Варто окремо розглянути аспіраційні системи - це особливий тип, який працює за принципом активного відбору повітря. Система постійно "всмоктує" повітря з приміщення через спеціальні трубки та аналізує його в детекторі. Це дозволяє виявляти дим на дуже ранніх стадіях з високою точністю. Але є нюанс - для такої системи потрібна складна мережа труб, що робить монтаж та обслуговування досить складними.[5]

1.1.3 Порівняльний аналіз комерційних продуктів

Сучасний ринок протипожежних систем характеризується великою різноманітністю пропозицій від провідних світових виробників. Особливо цікавими є рішення від Honeywell, зокрема їхня лінійка NOTIFIER, яка вирізняється надійністю та стабільністю функціонування. Що мене вразило у цих системах - це їхня здатність легко поєднуватися з різними інженерними мережами будівлі завдяки підтримці протоколів BACnet та Modbus, що дозволяє створювати комплексні системи автоматизації.[6]

Не менш цікавою є продукція Bosch Security Systems, а саме серія FPA-5000.[7] Модульна побудова цих систем дуже вражає - можна легко налаштувати конфігурацію як для невеликого офісу, так і для великого промислового комплексу. Особливо корисною вважаю можливість підключення майже 4000 адресованих пристроїв до однієї панелі управління, а вбудовані засоби діагностики значно полегшують технічне обслуговування.[8]

Окремо хочу відзначити рішення від Siemens - систему Cerberus PRO. Її головна перевага полягає у використанні розумних алгоритмів, що мінімізують помилкові спрацювання - проблему, з якою часто стикаються користувачі традиційних систем. Технологія ASAtechnology особливо цікава тим, що датчики самостійно підлаштовуються під умови навколишнього середовища, забезпечуючи більш точне виявлення загроз.

Компанія Edwards (тепер частина Carrier) також заслуговує на увагу завдяки своїй системі EST3, яка особливо популярна у Північній Америці. Головними плюсами цього рішення є інтуїтивне налаштування та програмування, а також підтримка різноманітних комунікаційних протоколів для інтеграції з іншими системами безпеки.[9]

Аналізуючи представлені на ринку комерційні протипожежні системи, можна помітити, що базовий функціонал у них досить схожий. Проте кожен виробник має свої унікальні особливості: одні роблять акцент на можливостях розширення системи, інші - на інтелектуальну обробку даних або зручність експлуатації.

Варто також зазначити, що сьогодні багато систем оснащені веб-інтерфейсами для віддаленого управління та моніторингу. Проте аналітичні можливості та функції прогнозування в таких системах поки що розвинені недостатньо і потребують значного покращення у майбутньому.

1.2 Теоретичні основи виявлення пожежних загроз

Якість роботи сучасних пожежних детекторів напряму залежить від того, наскільки добре ми розуміємо фізичні явища під час горіння та знаємо способи їх розпізнавання. Сьогодні це вже не просто реакція на дим або спалах температури - технології стали набагато складнішими.

Нинішні способи розпізнавання вогню базуються на одночасному вивченні декількох фізичних параметрів за допомогою комплексного підходу. Застосовуються спеціальні алгоритми для аналізу отриманих даних, які достатньо складні для того, щоб своєчасно помітити небезпеку та водночас знизити кількість хибних сигналів. Такий метод забезпечує більш точну та результативну роботу порівняно з застарілими системами попередніх поколінь.[10]

1.2.1 Фізичні принципи горіння та детекції

Процес горіння являє собою набагато більше, ніж просто появу полум'я - це комплексна хімічна взаємодія окислення. Під час неї активно вивільняється тепло, світло та з'являються різні речовини згоряння, зокрема гази й дрібні частинки. Для запуску цього процесу потрібна наявність трьох обов'язкових компонентів. Спочатку - матеріал, здатний горіти (твердий, рідкий чи газоподібний). Далі - окислювач, найчастіше це кисень з атмосфери. Нарешті - енергетичне джерело для початку реакції. Ці три складові формують відому концепцію «пожежного трикутника», що пояснює основи виникнення займання.

У самому початку пожежі зазвичай відбувається тління або ледь помітне горіння без яскравого полум'я. Цей ранній період відзначається помірною

температурою, проте супроводжується активним димоутворенням. Оскільки вогонь ще не досяг стадії інтенсивного горіння з відкритим полум'ям, саме цей момент є оптимальним для розпізнавання загрози. Швидке реагування на цьому етапі дозволяє уникнути катастрофічних наслідків, які неодмінно настають при подальшому поширенні пожежі. Отже, своєчасне виявлення небезпеки в початковій фазі має критичне значення для захисту людей та майна.

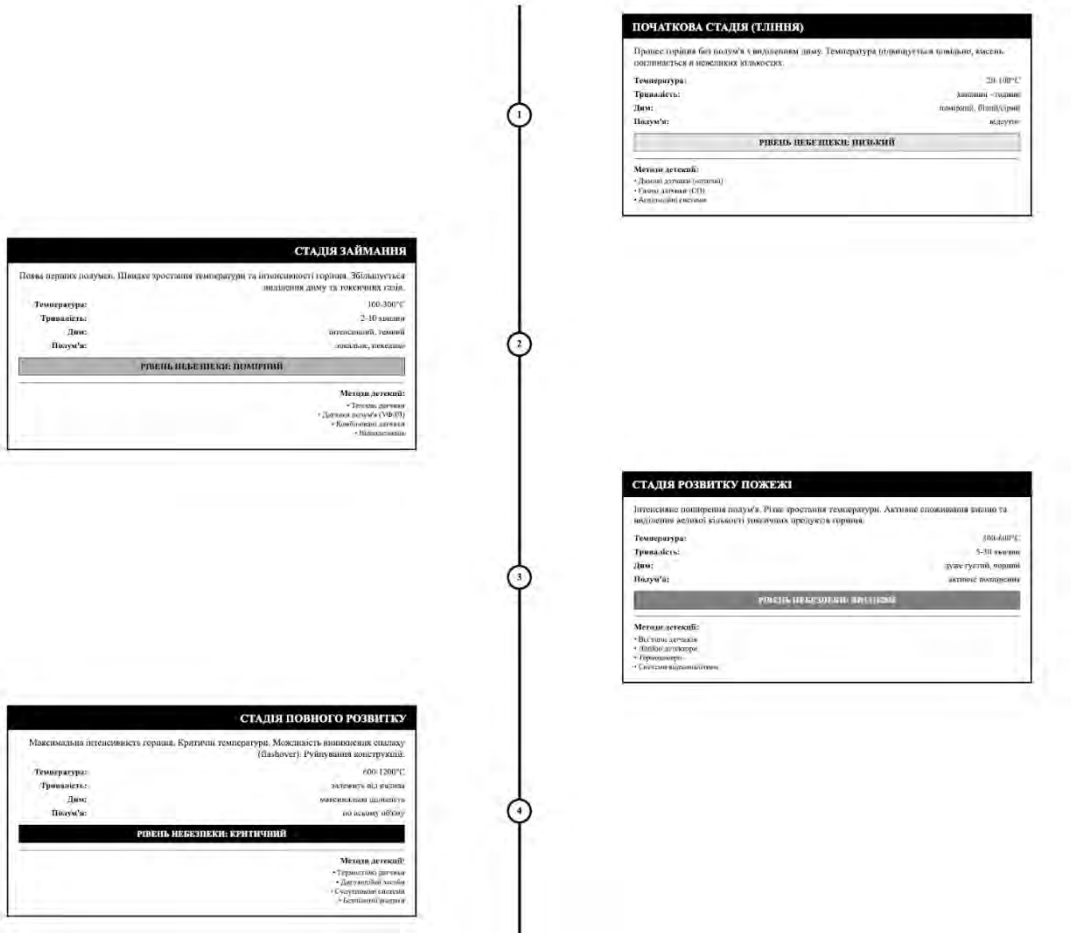

Рис. 1.2 Стадії розвитку пожежі

Під час горіння виникають різні теплові явища, з яких найважливішими є радіаційний і конвекційний способи передачі тепла. Завдяки конвекції, коли тепло переноситься через переміщення повітряних потоків, формуються висхідні струмені розігрітого повітря, що рухаються від вогнища догори. Ці потоки можна зафіксувати, використовуючи термічні детектори, які чутливо реагують на температурні коливання довкола. Радіаційне випромінювання

являє собою інший ключовий спосіб поширення тепла - воно розходиться в різних напрямках від центру займання. Це тепло розповсюджується як інфрачервоне випромінювання та фіксується ІЧ-детекторами в протипожежних системах.

Оптичні властивості продуктів згоряння суттєво відрізняються залежно від типу матеріалу, що горить, та специфіки самого процесу. Якщо тліють природні речовини (деревина, папір), то з'являються мікроскопічні димові частинки розміром приблизно 0,01-1 мкм. При інтенсивному горінні з полум'ям формуються більші частинки - до 10 мкм. Ця різниця в розмірах визначає характер взаємодії диму зі світловими променями, що впливає на принципи роботи оптичних детекторів.

Важливим фактором є також хімічний склад газів, що виділяються при згорянні. Целюлоза при горінні дає переважно вуглекислий і чадний гази. Проте синтетичні матеріали можуть продукувати значно шкідливіші речовини, включаючи отруйні гази типу хлороводню або синильної кислоти. Для розрізнення таких випадків сучасні протипожежні комплекси часто використовують багатофункціональні сенсори, які не лише фіксують наявність займання, а й досліджують повітряне середовище, визначаючи тип пожежі за хімічними показниками.

1.2.2 Алгоритми обробки сигналів датчиків

Одним із ключових чинників для точної детекції загроз і мінімізації хибних спрацювань є якісна обробка сигналів, що надходять із сенсорів. Сучасні підходи до аналізу таких сигналів використовують різні математичні інструменти як у часовому, так і в частотному вимірах. Найпростішими серед них є порогові методи, де значення сигналу порівнюється з попередньо встановленим рівнем. Однак цей підхід має обмеження, оскільки не враховує динаміку зміни параметрів, що може призводити до хибних сповіщень при поступових змінах у навколишньому середовищі.

Для більшої точності застосовуються алгоритми, які аналізують темп змін, тобто градієнт сигналу. Наприклад, при роботі з температурними сенсорами може оцінюватися швидкість зростання температури, що дозволяє виявити потенційно небезпечні ситуації ще до того, як температура досягне критичних значень. Крім того, у системах дедалі частіше використовуються статистичні методи - зокрема, обчислення дисперсії, кореляцій та спектрального аналізу. Аналіз дисперсії допомагає помітити коливання сигналу, характерні для початкової стадії пожежі, а кореляція - зіставляти дані з кількох сенсорів і виявляти просторові взаємозв’язки.

Окрему групу складають адаптивні методи, які самостійно налаштовуються відповідно до змін у середовищі. Вони базуються на алгоритмах машинного навчання і здатні створювати модель звичайного стану об'єкта, помічаючи найменші відхилення. Все більшого поширення набувають нейронні мережі та глибоке навчання, які дозволяють ідентифікувати складні нелінійні зв’язки між параметрами. Такі алгоритми можуть розпізнавати шаблони, що залишаються невидимими для класичних методів аналізу, роблячи систему детекції більш чутливою та інтелектуальною.[11]

1.2.3 Методи фільтрації хибних спрацьювань

Хибні спрацювання залишаються однією з головних проблем у роботі систем пожежної сигналізації, адже вони знижують рівень довіри користувачів до системи і можуть спричинити ігнорування реальних сигналів тривоги. Основними факторами, що викликають хибні спрацьовування, є зовнішні впливи, несправності техніки та некоректне налаштування обладнання.

Темпоральна фільтрація базується на вивченні часових характеристик сигналу. Справжні пожежі, як правило, проявляються поступовим збільшенням інтенсивності сигналу, в той час як перешкоди часто мають короткочасний, імпульсний характер. Завдяки застосуванню ковзних вікон та цифрових фільтрів можна ефективно відсікати короткочасні перешкоди. Просторова кореляція використовує сигнали від кількох сенсорів для

підтвердження справжньої тривоги. Якщо тривожний сигнал надходить лише від одного датчика, а сусідні не фіксують аномалій, це може свідчити про хибне спрацьовування. Алгоритми просторової кореляції враховують фізичні закономірності розповсюдження продуктів горіння.

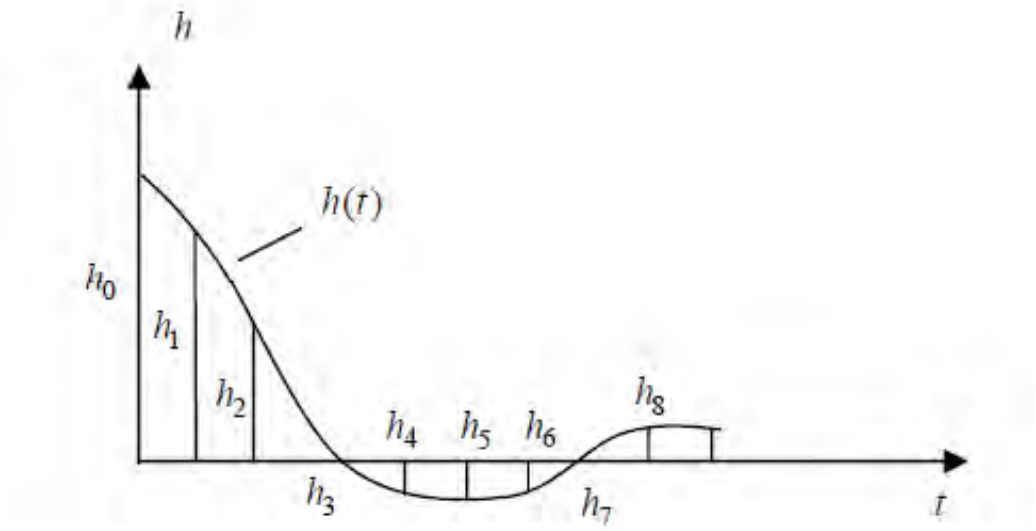

Рис. 1.3 Цифрова фільтрації сигналів від мережі датчиків

Багатопараметричний аналіз базується на обробці даних від різних типів датчиків для підвищення точності визначення пожежі. Наприклад, одночасне виявлення підвищення температури та концентрації диму значно підсилює достовірність сигналу у порівнянні з інформацією від одного сенсора.

Контекстний аналіз бере до уваги додаткові параметри, такі як час доби, день тижня, погодні умови та режим функціонування приміщень. Це дозволяє точніше налаштовувати алгоритми виявлення залежно від конкретних експлуатаційних умов і знижувати кількість помилкових тривог.

Моделі машинного навчання здатні автоматично розпізнавати справжні пожежі та відрізняти їх від хибних тривог, спираючись на історичні дані. Такі системи підлаштовуються під унікальні умови конкретного об’єкта, з часом покращуючи точність детекції.

1.3 Архітектурні підходи до побудови мережевих систем моніторингу

Сучасні системи моніторингу пожежних загроз є складними розподіленими системами, які вимагають ретельного проектування архітектури для забезпечення надійності, масштабованості та ефективності. Вибір архітектурного підходу значною мірою визначає функціональні можливості системи та її здатність до розвитку.[12]

1.3.1 Клієнт-серверна архітектура у системах безпеки

Клієнт-серверна архітектура є традиційним підходом до побудови систем моніторингу безпеки. У такій архітектурі центральний сервер відповідає за збір, обробку та зберігання даних від датчиків, а клієнтські додатки забезпечують інтерфейс для користувачів та адміністраторів системи.

Основною перевагою клієнт-серверної архітектури є централізоване управління та можливість забезпечення високого рівня безпеки даних. Усі критичні операції виконуються на сервері під контролем адміністратора, що спрощує аудит та забезпечення відповідності нормативним вимогам.

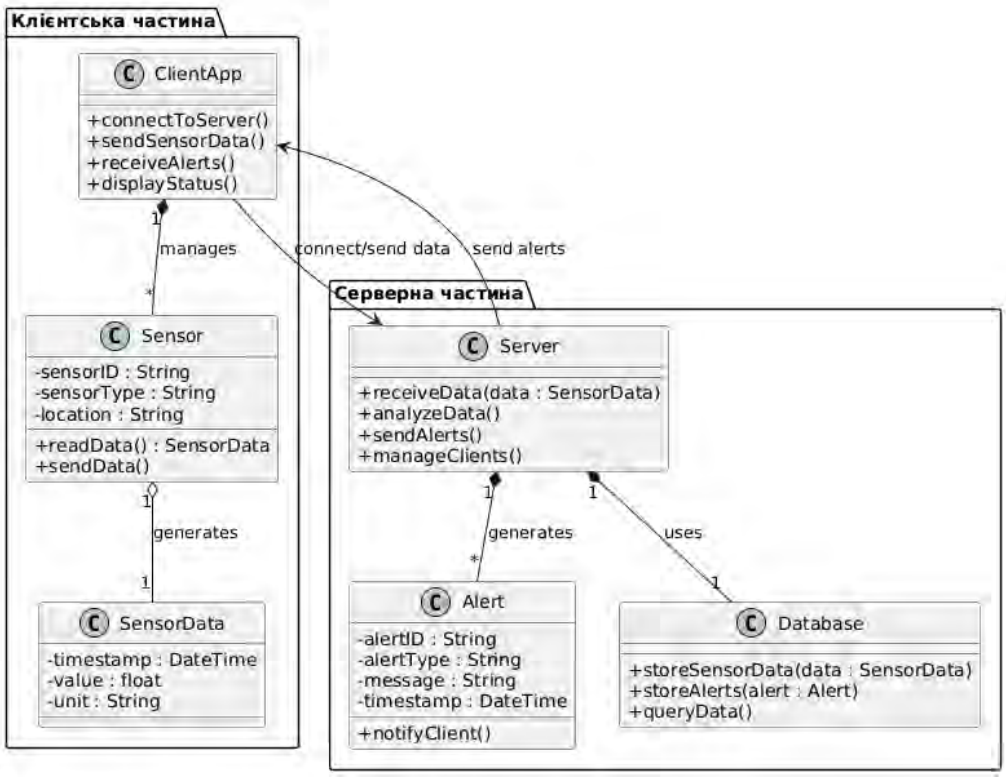

Рис. 1.4 Схема клієнт-серверної архітектури системи пожежного моніторингу

Сервер у системі пожежної безпеки зазвичай реалізується як багатопоточний додаток, здатний одночасно обслуговувати множину клієнтів та датчиків. Він включає підсистеми збору даних, їх обробки, зберігання та розповсюдження. База даних сервера містить конфігурацію системи, історичні дані та журнали подій.

Клієнтські додатки можуть бути реалізовані як настільні програми, веб-додатки або мобільні застосунки. Веб-клієнти мають переваги у вигляді незалежності від платформи та можливості доступу з будь-якого пристрою з веб-браузером. Настільні клієнти можуть забезпечувати більш багатий функціонал та кращу продуктивність.

Протоколи зв'язку між клієнтом та сервером повинні забезпечувати надійну передачу даних та підтримувати механізми автентифікації та шифрування. Широко використовуються протоколи HTTP/HTTPS для веб-клієнтів та спеціалізовані протоколи для настільних додатків.[13]

Недоліками клієнт-серверної архітектури є єдина точка відмови у вигляді сервера та потенційні проблеми з масштабованістю при зростанні навантаження. Для вирішення цих проблем використовуються кластерні рішення та системи резервного копіювання.

1.3.2 Мікросервісна архітектура для розподілених систем

Мікросервісна архітектура представляє сучасний підхід до побудови розподілених систем, при якому додаток розбивається на множину невеликих, незалежно розгортуваних сервісів. Кожен мікросервіс відповідає за конкретну бізнес-функцію та може розроблятися, тестуватися та розгортатися незалежно від інших компонентів.

У контексті систем пожежної безпеки мікросервісна архітектура дозволяє виділити окремі сервіси для збору даних від датчиків, їх обробки, генерації сповіщень, управління користувачами та ведення звітності. Така декомпозиція підвищує гнучкість системи та спрощує її підтримку.[14]

Сервіс збору даних відповідає за взаємодію з фізичними датчиками та первинну обробку сигналів. Він може підтримувати різні протоколи зв'язку та забезпечувати буферизацію даних при тимчасових збоях у мережі. Цей сервіс також відповідає за валідацію вхідних даних та їх нормалізацію.

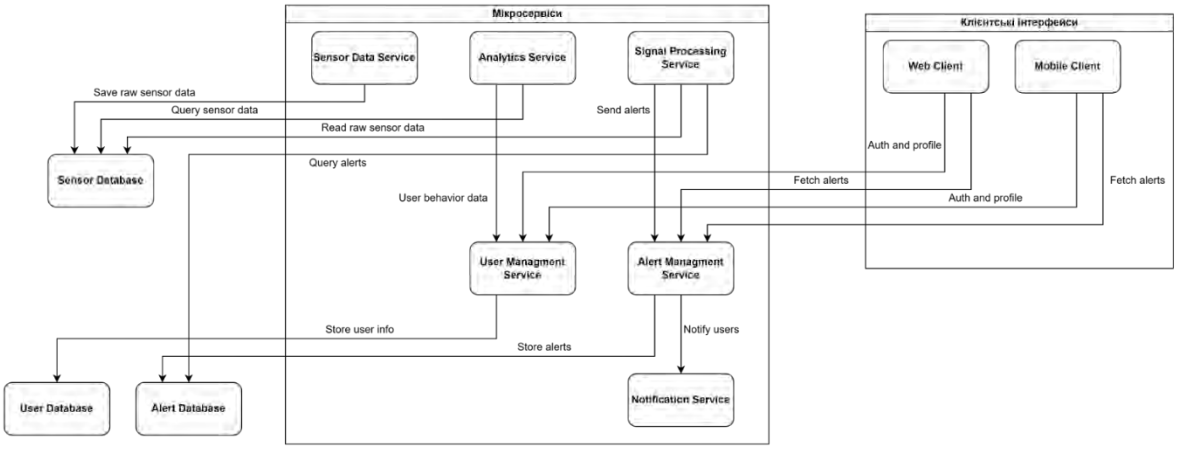

Рис. 1.5 Мікросервісна архітектура системи виявлення пожежних загроз

Сервіс аналізу даних реалізує алгоритми виявлення пожежних загроз та генерації сповіщень. Він може використовувати складні алгоритми машинного навчання та мати можливість горизонтального масштабування для обробки великих обсягів даних. Результати аналізу передаються іншим сервісам через API або системи повідомлень.

Сервіс сповіщень відповідає за доставку тривог користувачам через різні канали зв'язку, такі як електронна пошта, SMS, push-повідомлення або інтеграція з зовнішніми системами. Він також веде журнал доставки повідомлень та може реалізовувати політики ескалації.[15]

Перевагами мікросервісної архітектури є висока відмовостійкість, оскільки відмова одного сервісу не впливає на роботу інших, можливість використання різних технологій для різних сервісів та спрощення горизонтального масштабування. Однак така архітектура також має складності у вигляді підвищеної складності розгортання та моніторингу.

1.3.3 Патерни проектування для систем реального часу

Системи виявлення пожежних загроз мають жорсткі вимоги до часу відгуку, що робить критичним використання відповідних патернів проектування для систем реального часу. Ці патерни допомагають забезпечити передбачувану продуктивність та надійність системи навіть при високих навантаженнях.

Патерн Publisher-Subscriber є одним з найважливіших для систем моніторингу. Він дозволяє датчикам публікувати свої дані без знання про те, які компоненти системи їх використовують. Підписники можуть динамічно підписуватися на потрібні їм типи даних та отримувати оновлення в реальному часі.

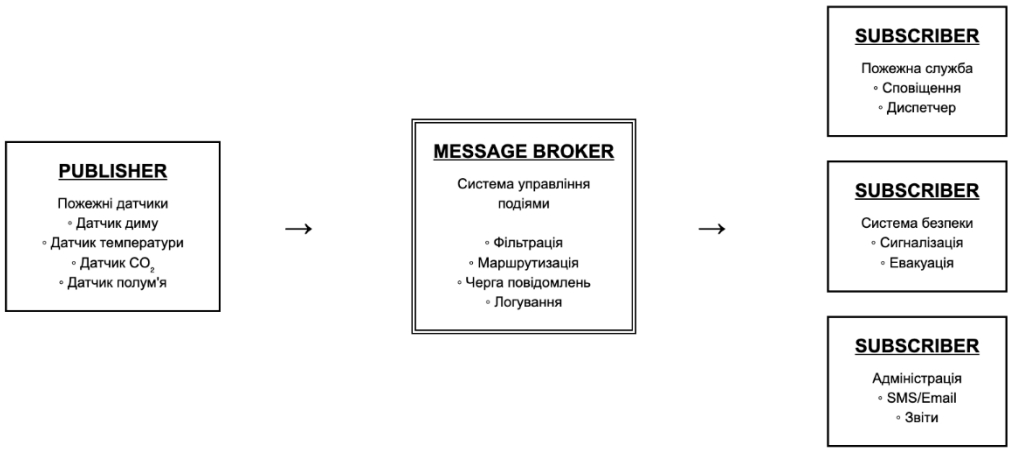

Рис. 1.6 Реалізація патерну Publisher-Subscriber у системі пожежного моніторингу

Патерн Event Sourcing забезпечує надійне збереження всіх подій у системі та можливість відтворення стану системи на будь-який момент часу. Це особливо важливо для систем безпеки, де необхідно вести детальний аудит всіх дій та мати можливість аналізу причин інцидентів.

Патерн Circuit Breaker використовується для захисту системи від каскадних відмов. Якщо певний компонент системи стає недоступним або працює з помилками, Circuit Breaker тимчасово блокує звернення до нього, дозволяючи системі продовжувати роботу в обмеженому режимі.

Патерн Command Query Responsibility Segregation дозволяє розділити операції читання та запису, що підвищує продуктивність системи. Команди

модифікації стану обробляються оптимізованими для запису компонентами, тоді як запити на читання обслуговуються спеціалізованими службами з оптимізованими для читання структурами даних.

Патерн Saga використовується для управління розподіленими транзакціями у мікросервісній архітектурі. Він забезпечує консистентність даних між різними сервісами без використання глобальних транзакцій, які можуть значно знижувати продуктивність системи.

Патерн Bulkhead ізолює критичні ресурси системи від некритичних, запобігаючи впливу проблем у некритичних компонентах на основну функціональність. Наприклад, сервіс генерації звітів може бути ізольований від сервісу обробки тривог, щоб проблеми зі звітністю не впливали на виявлення пожеж.[16]

ВИСНОВКИ ДО РОЗДІЛУ 1

Проведений аналіз предметної області та теоретичних основ систем виявлення пожежних загроз дозволяє зробити ряд важливих висновків щодо сучасного стану та перспектив розвитку цієї галузі. Еволюція систем пожежної сигналізації демонструє поступовий перехід від простих механічних пристроїв до складних інтелектуальних мережевих комплексів. Сучасні системи характеризуються високою чутливістю, здатністю до самодіагностики та можливістю інтеграції з іншими системами безпеки будівлі. Аналіз комерційних продуктів показав, що провідні виробники активно впроваджують цифрові технології та алгоритми машинного навчання для підвищення ефективності детекції.

Дослідження фізичних принципів горіння виявило складність процесів, що відбуваються на різних стадіях розвитку пожежі. Початкова стадія тління характеризується відносно низькими температурами та помірним виділенням диму, що робить її найбільш сприятливою для раннього виявлення. Аналіз алгоритмів обробки сигналів датчиків показав переваги багатопараметричних підходів над простими пороговими методами. Використання статистичних методів, аналізу градієнтів та адаптивних алгоритмів значно підвищує точність детекції. Особливо перспективними є методи машинного навчання, які дозволяють системам адаптуватися до специфічних умов експлуатації.

Проблема хибних спрацювань залишається однією з найбільш актуальних у галузі пожежної безпеки. Ефективними методами її вирішення є темпоральна та просторова фільтрація сигналів, багатопараметричний аналіз та контекстний підхід, який враховує додаткову інформацію про стан об'єкта. Аналіз архітектурних підходів виявив переваги клієнт-серверної архітектури для централізованого управління та мікросервісної архітектури для забезпечення масштабованості та відмовостійкості. Патерни проектування для систем реального часу, такі як Publisher-Subscriber, Event Sourcing та Circuit Breaker, є критично важливими для забезпечення надійної роботи системи.

Встановлено, що сучасні системи виявлення пожежних загроз повинні поєднувати традиційні методи детекції з інноваційними підходами,

включаючи веб-технології, системи реального часу та алгоритми інтелектуального аналізу даних. Це створює основу для розробки нового покоління систем, здатних забезпечувати високу ефективність виявлення загроз при мінімальній кількості хибних спрацьювань.

Результати аналізу показують, що розвиток мережевих технологій та веб-додатків відкриває нові можливості для створення гнучких, масштабованих та зручних у використанні систем пожежної безпеки. Такі системи можуть забезпечувати дистанційний моніторинг, інтуїтивно зрозумілі інтерфейси користувача та інтеграцію з сучасними екосистемами розумних будівель.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ ПОЖЕЖНИХ ЗАГРОЗ

Розробка сучасної системи виявлення пожежних загроз вимагає комплексного підходу до вибору технологій та архітектурних рішень. Ефективна система повинна забезпечувати надійний збір даних від датчиків, їх обробку в реальному часі, генерацію сповіщень та зручний інтерфейс для моніторингу. Проектування такої системи передбачає ретельний аналіз функціональних вимог, вибір оптимального технологічного стеку та створення масштабованої архітектури.

2.1 Архітектура та технологічний стек системи

Архітектурні рішення системи базуються на принципах розділення відповідальностей та модульності. Система побудована за клієнт-серверною архітектурою з чітким розділенням на фронтенд та бекенд компоненти. Такий підхід забезпечує гнучкість розробки, спрощує тестування та дозволяє незалежно масштабувати різні частини системи.

2.1.1 Вибір технологій frontend розробки Vue.js, Nuxt

Для розробки клієнтської частини системи було обрано Vue.js 3 як основний фреймворк, що забезпечує реактивність інтерфейсу та ефективне управління станом.[17] Vue.js відзначається високою продуктивністю, зручністю розробки та потужною екосистемою бібліотек. Використання Composition API дозволяє створювати більш гнучкі та переносні компоненти.[18]

Поверх Vue.js використовується фреймворк Nuxt 4, який надає додаткові можливості для створення повнофункціональних веб-додатків. Nuxt забезпечує серверний рендеринг, автоматичну маршрутизацію, оптимізацію коду та вбудовані засоби для роботи з API. Це особливо важливо для систем моніторингу, де критична швидкість завантаження та SEO оптимізація.

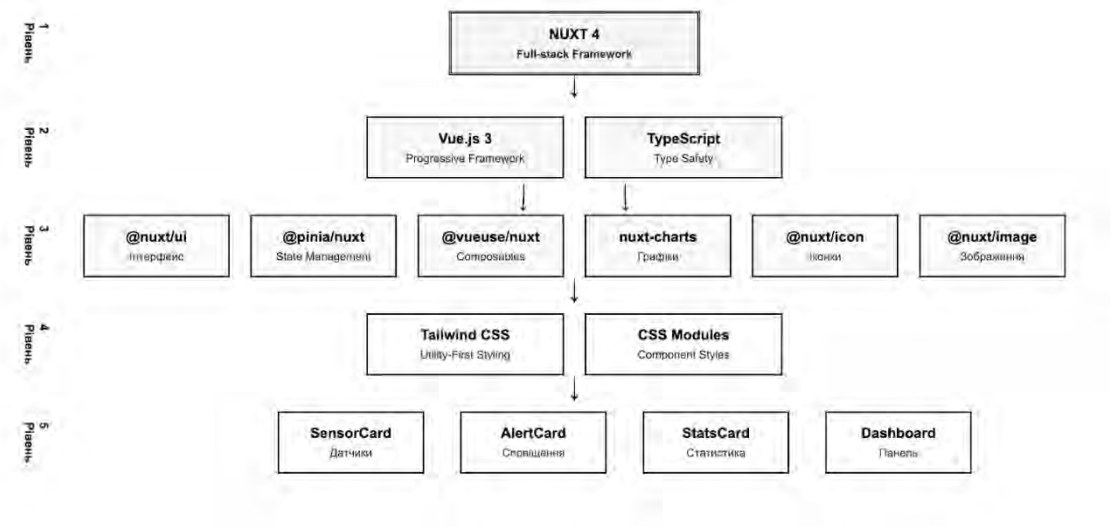

Рис. 2.1 Архітектура фронтенд додатку на базі Vue.js та Nuxt

TypeScript було вибрано як основну мову розробки для забезпечення статичної типізації та підвищення надійності коду. Типізація особливо важлива в системах реального часу, де помилки типів можуть призвести до некоректного відображення критичної інформації про стан датчиків.

Для створення користувацького інтерфейсу використовується Nuxt UI в поєднанні з Tailwind CSS. Nuxt UI надає готові компоненти, які оптимізовані для accessibility та мають консистентний дизайн. Tailwind CSS забезпечує гнучкість стилізації через utility-first підхід, що дозволяє швидко створювати адаптивні інтерфейси.

Система іконок реалізована через Nuxt Icon з підтримкою бібліотек MDI та Line MD. Це забезпечує широкий вибір іконок для різних типів датчиків та станів системи. Nuxt Image використовується для оптимізації зображень та графіків, що важливо для швидкодії системи моніторингу.

2.1.2 Серверна частина та API архітектура

Серверна частина системи розроблена на Node.js з використанням фреймворку Hono.js. Вибір Node.js обумовлений його високою продуктивністю для I/O операцій, що критично важливо для систем збору

даних в реальному часі. Hono.js забезпечує мінімальні накладні витрати та підтримує сучасні веб-стандарти.

API побудовано за RESTful принципами з використанням HTTP методів для різних операцій. Структура API включає ендпоінти для роботи з датчиками, показаннями, сповіщеннями та статистикою системи. Всі відповіді API мають консистентний формат JSON з відповідними HTTP статус кодами.

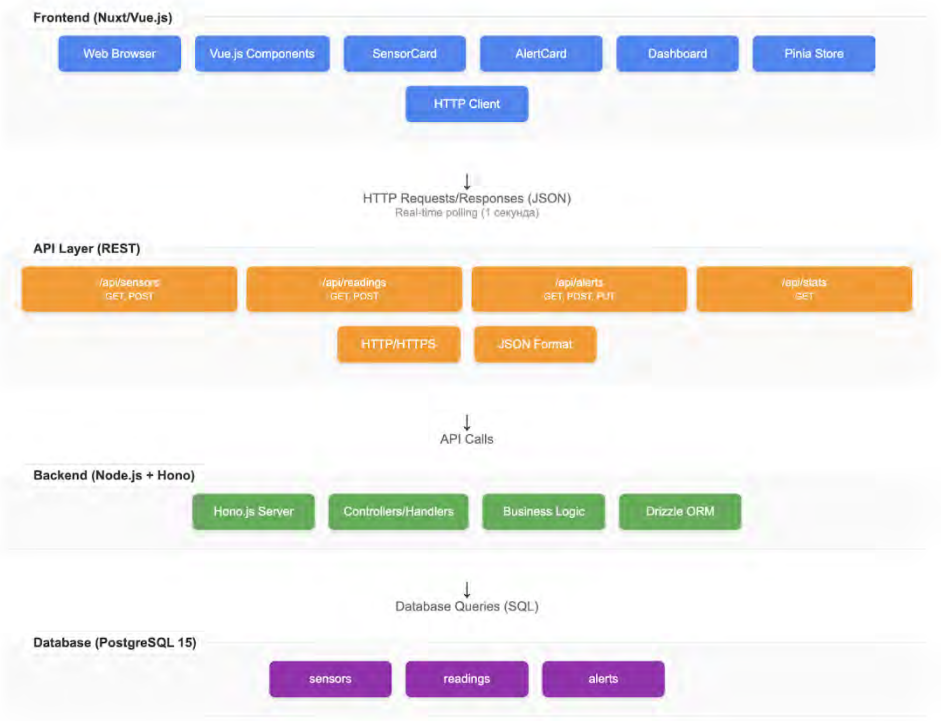

Рис. 2.2 Схема API архітектури та взаємодії компонентів

База даних PostgreSQL 15 використовується для зберігання всіх даних системи. Вибір PostgreSQL обумовлений його надійністю, підтримкою ACID транзакцій та потужними можливостями для роботи з часовими рядами даних. Це особливо важливо для зберігання історичних показань датчиків та аналізу трендів.

Drizzle ORM забезпечує type-safe взаємодію з базою даних та автоматичну генерацію типів TypeScript з схеми бази даних. Це дозволяє уникнути помилок при роботі з даними та забезпечує консистентність між схемою бази даних та кодом додатку.

Контейнеризація системи реалізована через Docker, що забезпечує консистентність середовища розробки та продакшну. Docker Compose використовується для оркестрації сервісів та спрощення процесу розгортання. Це дозволяє легко відтворювати робоче середовище на різних машинах.

2.1.3 Протоколи передачі даних та мережева взаємодія

Взаємодія між клієнтом та сервером здійснюється через HTTP/HTTPS протокол з використанням JSON формату для передачі даних. HTTP був обраний як стандартний та широко підтримуваний протокол, що забезпечує сумісність з різними браузерами та мережевою інфраструктурою.

Для забезпечення роботи в реальному часі використовується polling механізм з автоматичним оновленням даних кожну секунду. Такий підхід забезпечує актуальність інформації про стан датчиків без надмірного навантаження на сервер. Інтервал оновлення є конфігурованим та може бути адаптований під специфічні вимоги різних типів об'єктів.

API забезпечує підтримку CORS для безпечної взаємодії між фронтендом та бекендом, що дозволяє розгортати їх на різних доменах. Реалізовані механізми обробки помилок та retry логіка для забезпечення стійкості до тимчасових мережевих збоїв.[19]

Система підтримує кешування на рівні HTTP заголовків для оптимізації продуктивності. Статичні дані кешуються на тривалий час, тоді як динамічні дані мають короткий час життя кешу. Це дозволяє знизити навантаження на сервер та прискорити відгук системи.

Валідація даних здійснюється на рівні API з використанням схем, що забезпечує цілісність даних та запобігає обробці некоректної інформації. Всі API ендпоінти мають детальну документацію з описом параметрів та можливих відповідей.

2.2 Основні компоненти системи

Система складається з декількох ключових компонентів, кожен з яких відповідає за конкретні функції. Модульна архітектура дозволяє незалежно розвивати та тестувати різні частини системи, що підвищує її надійність та спрощує підтримку.

2.2.1 Модуль збору та обробки даних датчиків

Модуль збору даних реалізований як сервіс, що відповідає за прийом показань від різних типів датчиків. Система підтримує три основні типи датчиків: температурні, димові та вологості. Кожен тип датчика має специфічні характеристики та пороги спрацювання.

Обробка показань включає валідацію вхідних даних, нормалізацію значень та розрахунок статусу датчика на основі заданих порогів. Для температурних датчиків нормальний діапазон становить менше 25°C, попередження 25-35°C, критичний стан понад 35°C. Димові датчики мають пороги: нормальний стан менше 100ppm, попередження 100-300ppm, критичний понад 300ppm.

надається за запитом до авторів

Рис. 2.3 Алгоритм обробки даних від різних типів датчиків

Система автоматично розраховує статус датчика на основі поточних показань та зберігає історичні дані для подальшого аналізу. Реалізовано

механізм детекції аномалій, який враховує не лише абсолютні значення, але й швидкість зміни параметрів.

База даних оптимізована для зберігання часових рядів з індексами по часу та ідентифікаторам датчиків. Це забезпечує швидкий доступ до історичних даних та ефективне виконання аналітичних запитів. Реалізовано автоматичне архівування старих даних для підтримки продуктивності системи.

Симуляція роботи датчиків реалізована через спеціальні скрипти, які генерують реалістичні дані з урахуванням добових циклів та поступових змін. Це дозволяє тестувати систему в умовах, наближених до реальних, та демонструвати її можливості.

2.2.2 Система сповіщень та алертів

Система сповіщень автоматично генерує алерти при перевищенні порогових значень датчиків. Реалізовано три рівні критичності: нормальний, попередження та критичний. Кожен рівень має відповідне візуальне оформлення та prioritet обробки.

Для запобігання спаму реалізовано механізм дедуплікації сповіщень, який блокує генерацію повторних алертів від одного датчика протягом 10 хвилин. Це дозволяє уникнути надмірної кількості сповіщень при коливаннях показань навколо порогових значень.

Кожне сповіщення містить детальну інформацію про датчик, тип загрози, поточне значення та часову мітку. Алерти можуть бути позначені як оброблені користувачем, що дозволяє відстежувати стан реагування на інциденти.

Система підтримує фільтрацію сповіщень за рівнем критичності, типом датчика та часовим періодом. Це дозволяє операторам швидко знаходити найбільш важливі події та відстежувати тренди в роботі системи.

Реалізовано REST API для управління сповіщеннями, що дозволяє інтегрувати систему з зовнішніми сервісами моніторингу та автоматизації. API підтримує операції створення, читання, оновлення та видалення сповіщень.

2.2.3 Веб-інтерфейс моніторингу та управління

Користувацький інтерфейс побудований за принципами responsive design та забезпечує зручну роботу на різних пристроях. Головна панель dashboard надає загальний огляд стану системи з ключовими метриками та візуалізацією даних.

Компонент SensorCard відображає інформацію про окремий датчик з поточними показаннями, статусом та мініграфіком змін. Картки датчиків автоматично оновлюються та змінюють колір залежно від рівня загрози. Це забезпечує швидке візуальне сприйняття стану системи.

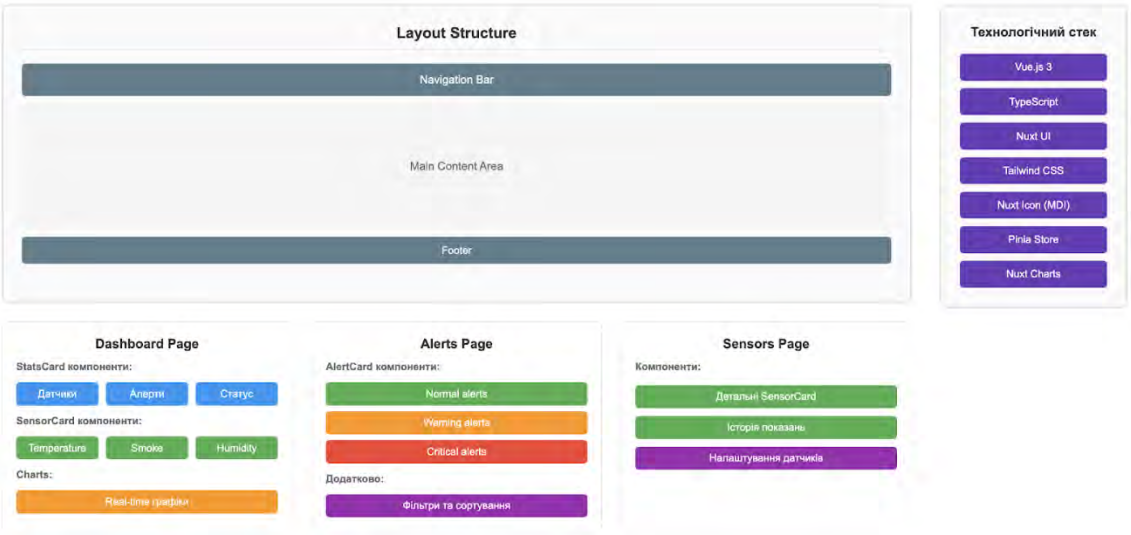

Рис. 2.4 Компоненти користувацького інтерфейсу системи моніторингу

AlertCard компонент призначений для відображення сповіщень з можливістю фільтрації та сортування. Реалізовано різне візуальне оформлення для різних рівнів критичності, що дозволяє швидко ідентифікувати найбільш важливі події.

StatsCard забезпечує відображення статистичної інформації у зручному форматі з підтримкою анімацій та переходів. Компоненти мають консистентний дизайн та підтримують dark/light теми.

Навігація реалізована через маршрутизацію Nuxt з підтримкою deep linking та breadcrumbs. Це дозволяє користувачам легко переміщатися між різними розділами системи та повертатися до попередніх сторінок.

2.3 Алгоритмічні рішення та патерни проектування

Система використовує сучасні патерни проектування та алгоритмічні підходи для забезпечення ефективності та надійності роботи. Вибір конкретних рішень базується на специфічних вимогах систем реального часу та необхідності обробки великих обсягів даних.

2.3.1 Алгоритми виявлення аномалій у показниках датчиків

Виявлення аномалій базується на комбінації порогових алгоритмів та статистичного аналізу. Основний алгоритм порівнює поточні показання з заданими пороговими значеннями для кожного типу датчика. Це забезпечує швидке виявлення критичних ситуацій без складних обчислень.

Додатково реалізовано алгоритм аналізу градієнта, який відстежує швидкість зміни показань. Різке зростання температури або концентрації диму може свідчити про початок пожежі навіть при значеннях, що не досягли критичних порогів.

Система використовує ковзне вікно для згладжування короткочасних коливань показань. Це дозволяє відфільтровувати шум та зменшити кількість хибних спрацювань. Розмір вікна адаптується залежно від типу датчика та його характеристик.

Реалізовано простий алгоритм детекції аномалій на основі стандартного відхилення від середнього значення за певний період. Якщо поточне значення відхиляється більш ніж на два стандартних відхилення, система генерує попереджувальне сповіщення.

Для кожного датчика ведеться статистика нормальної роботи, що дозволяє адаптувати пороги під конкретні умови експлуатації. Це особливо важливо для датчиків, розташованих у приміщеннях з різними температурними режимами.

2.3.2 Реалізація патерну Observer для системи сповіщень

Система сповіщень реалізована з використанням патерну Observer, що дозволяє автоматично оповіщувати зацікавлені компоненти про зміни стану датчиків. Це забезпечує loose coupling між модулем обробки даних та системою сповіщень.

При надходженні нових показань система автоматично перевіряє необхідність генерації сповіщень та оповіщує всіх підписників про зміни. Це дозволяє легко додавати нові типи сповіщень без модифікації основного коду обробки даних.

Реалізовано механізм підписки різних компонентів системи на події зміни стану датчиків. Веб-інтерфейс автоматично оновлюється при генерації нових сповіщень, а система логування записує всі критичні події.

Патерн Observer також використовується для оновлення кешу статистичних даних. При зміні показань датчиків автоматично перераховуються агреговані метрики та оновлюється dashboard.

Система підтримує пріоритизацію сповіщень, що дозволяє обробляти критичні події в першу чергу. Реалізовано чергу сповіщень з можливістю batch обробки для оптимізації продуктивності.

2.3.3 State Management та реактивність у веб-додатку

Управління станом фронтенд додатку реалізовано через Pinia, що забезпечує централізований та типізований доступ до даних. Основний store sensors містить інформацію про датчики, показання, сповіщення та статистику системи.

Pinia store організований за модульним принципом з окремими actions для різних операцій: завантаження даних датчиків, отримання сповіщень, оновлення статистики. Це забезпечує чіткий розподіл відповідальностей та спрощує тестування.

Реактивність забезпечується через Vue.js Composition API з використанням computed properties та watchers. Компоненти автоматично

оновлюються при зміні даних у store, що забезпечує синхронізацію інтерфейсу з актуальним станом системи.

Реалізовано кешування API запитів на рівні store для зменшення навантаження на сервер. Кеш автоматично інвалідується при оновленні даних або через заданий інтервал часу.

VueUse composables використовуються для реалізації загальної функціональності, такої як auto-refresh таймери, обробка HTTP запитів та управління локальним станом компонентів. Це забезпечує консистентність коду та зменшує дублювання.

2.4 Забезпечення надійності та масштабованості

Надійність та масштабованість є критичними характеристиками для систем безпеки. Реалізовано комплекс заходів для забезпечення стабільної роботи системи навіть при високих навантаженнях та відмовах окремих компонентів.

2.4.1 Обробка помилок та відмовостійкість системи

Система обробки помилок реалізована на всіх рівнях архітектури. На рівні API всі ендпоінти мають обгортки для перехоплення винятків та повернення консистентних відповідей про помилки. Використовуються стандартні HTTP статус коди для індикації різних типів помилок.

База даних операції обгорнуті в транзакції для забезпечення консистентності даних. При виникненні помилок під час обробки показань система автоматично відкочує зміни та логує детальну інформацію про інцидент.

Фронтенд додаток реалізує retry логіку для API запитів з експоненційним backoff. Це дозволяє автоматично відновлювати роботу після тимчасових мережевих збоїв без втручання користувача.

Система моніторингу здоров'я включає health check ендпоінт, який перевіряє доступність бази даних та основних сервісів. Це дозволяє зовнішнім системам моніторингу відстежувати стан додатку.

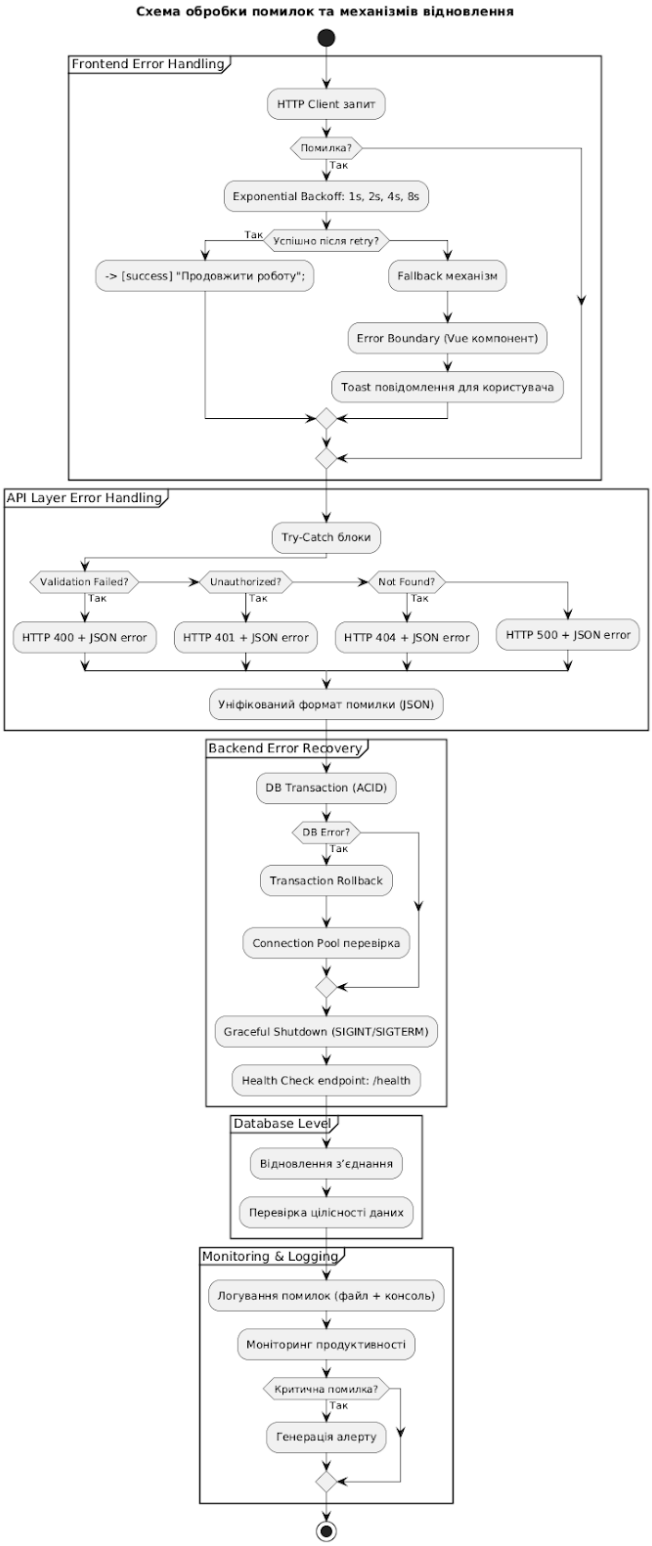

Рис. 2.5 Схема обробки помилок та механізмів відновлення

Реалізовано механізм graceful shutdown для коректного завершення роботи сервера при рестарті або оновленні. Всі активні запити завершуються коректно, а нові запити отримують відповідь про тимчасову недоступність.

2.4.2 Оптимізація продуктивності веб-інтерфейсу

Продуктивність фронтенд додатку оптимізована через декілька підходів. Nuxt забезпечує автоматичне code splitting, що дозволяє завантажувати лише необхідний код для кожної сторінки. Це значно зменшує початковий час завантаження додатку.

Компоненти використовують lazy loading для відкладеного завантаження неактивного контенту. Графіки та діаграми завантажуються тільки при потребі, що економить ресурси браузера та трафік.

Nuxt Image забезпечує автоматичну оптимізацію зображень з підтримкою responsive форматів та lazy loading. Статичні ресурси кешуються на тривалий час, а динамічний контент має короткий TTL.

Реалізовано віртуалізацію для великих списків сповіщень та історичних даних. Це дозволяє ефективно відображувати тисячі записів без втрати продуктивності інтерфейсу.

Vue компоненти оптимізовані через правильне використання computed properties, watchers та мемоізації. Уникнення непотрібних ре-рендерів досягається через v-memo директиви та правильну структуру компонентів.

2.4.3 Стратегії кешування та управління станом

Кешування реалізовано на декількох рівнях для забезпечення оптимальної продуктивності. HTTP кеш налаштований з відповідними заголовками для статичного контенту та коротким TTL для динамічних даних.

Pinia store включає власний механізм кешування для API відповідей. Дані датчиків кешуються на одну секунду, статистика на 30 секунд, а конфігурація системи на 5 хвилин. Це балансує актуальність даних та навантаження на сервер.

Локальне кешування у браузері використовується для збереження користувацьких налаштувань та часто використовуваних даних. SessionStorage зберігає тимчасову інформацію сесії, а LocalStorage довгострокові налаштування.

Реалізовано стратегію cache invalidation для автоматичного оновлення застарілих даних. При отриманні нових показань датчиків автоматично інвалідуються пов'язані кеші статистики та dashboard.

Серверний кеш використовує in-memory зберігання для швидкого доступу до часто запитуваних даних. Реалізовано LRU політику для автоматичного видалення найменш використовуваних записів при досягненні ліміту пам'яті.

ВИСНОВКИ ДО РОЗДІЛУ 2

Розробка системи виявлення пожежних загроз вимагала цілісного підходу до вибору технологій та архітектури. Аналіз і реалізація дозволили створити надійну та гнучку систему, що відповідає сучасним вимогам до безпеки.

Для фронтенду було обрано зв'язку Vue 3 з Nuxt 4. Використання Composition API та TypeScript забезпечує гнучкість і типову безпеку, а Nuxt 4 SSR і автоматичну оптимізацію, що важливо для роботи в реальному часі.

Серверна частина реалізована на Node.js з Hono.js для обробки запитів, а PostgreSQL з Drizzle ORM для зберігання даних. Docker спрощує розгортання та підтримує стабільність середовища. Модульна структура системи дозволяє чітко розмежовувати логіку. Модуль збору даних обробляє показники трьох типів датчиків із власними порогами, автоматично визначаючи статуси й аномалії.

Система сповіщень реалізує генерацію алертів з уникненням дублювання повідомлень, що дозволяє оперативно реагувати на критичні ситуації. Адаптивний інтерфейс забезпечує зручність на будь-якому пристрої. Алгоритми засновані на порогових значеннях і статистичному аналізі з використанням ковзного вікна й градієнтного аналізу.

Управління станом організоване через Pinia з модульною структурою. Реактивність і VueUse composables зменшують дублювання коду та спрощують оновлення інтерфейсу. Надійність досягається через обробку помилок, транзакційність і механізми відновлення з retry-логікою та backoff. Продуктивність оптимізована за рахунок lazy loading, code splitting і багаторівневого кешування. Загалом система демонструє ефективне поєднання сучасних технологій, добре структуровану архітектуру й готовність до практичного впровадження в різних середовищах з можливістю подальшої масштабованості.

РОЗДІЛ 3. ДЕМОНСТРАЦІЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ТА ТЕСТУВАННЯ СИСТЕМИ

Практична перевірка ефективності розробленої системи виявлення пожежних загроз потребує комплексного тестування всіх компонентів та демонстрації їх роботи в умовах, наближених до реальних. Демонстрація функціональності дозволяє оцінити зручність користувацького інтерфейсу, швидкість відгуку системи та коректність обробки даних від датчиків. Тестування проводилося на запущеній системі з симуляцією роботи датчиків для забезпечення реалістичних умов експлуатації.

3.1 Демонстрація головної панелі моніторингу

Головна панель Dashboard є центральним елементом системи, який надає користувачу швидкий огляд стану всіх компонентів. При запуску системи та переході за адресою http://localhost:3000 користувач потрапляє на головну сторінку з чітко структурованою інформацією.

3.1.1 Огляд інтерфейсу та загальної статистики системи

Верхня частина dashboard містить головну навігацію з розділами Датчики, Сповіщення та логотипом системи. Інтерфейс виконаний у сучасному стилі з використанням Tailwind CSS та забезпечує зручність використання як на настільних комп'ютерах, так і на мобільних пристроях.

Панель статистики відображає ключові метрики системи в реальному часі. Загальна кількість датчиків показує 5 активних пристроїв, що відповідає конфігурації тестової системи. Кількість активних сповіщень динамічно змінюється залежно від поточного стану датчиків та може варіюватися від 0 до декількох одиниць.

Індикатори критичності демонструють розподіл датчиків за рівнями загрози. Нормальний стан позначається зеленим кольором, попередження жовтим, а критичний стан червоним. Це дозволяє оператору миттєво оцінити загальну ситуацію на об'єкті.

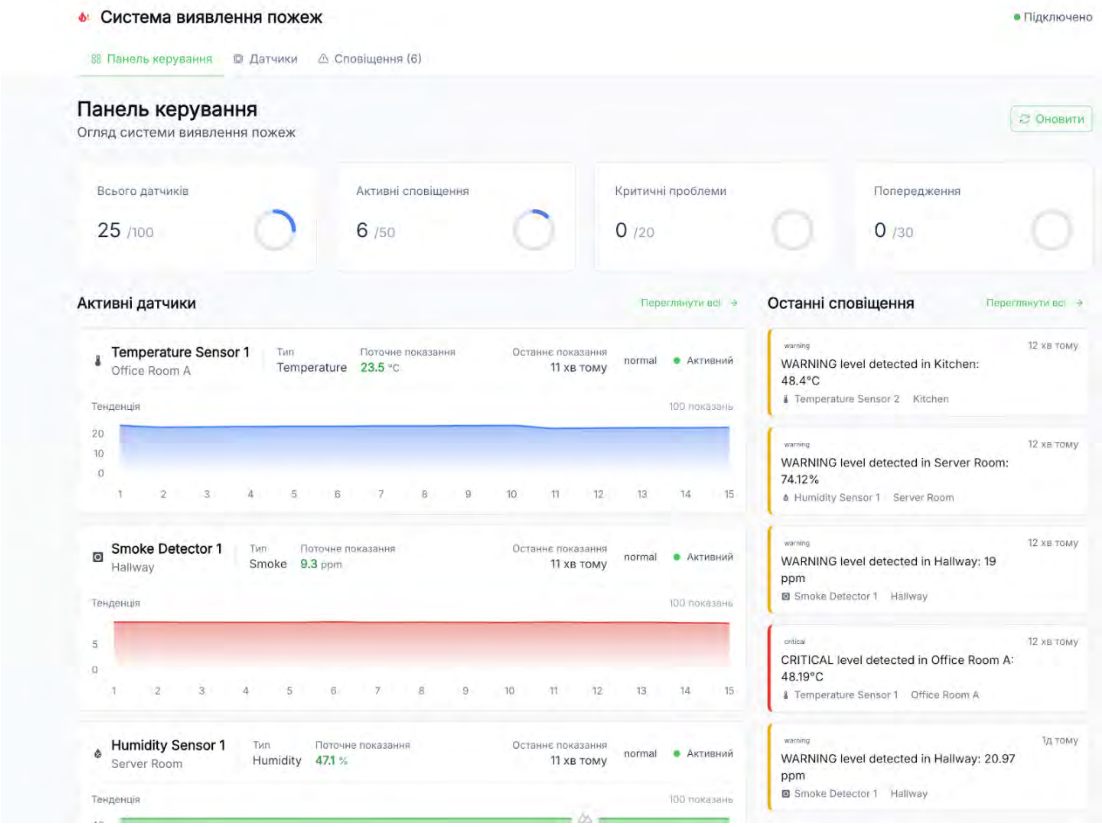

Рис. 3.1 Головна панель системи з відображенням загальної статистики

Статистичні картки мають анімовані переходи та автоматично оновлюються кожну секунду. При зміні значень спрацьовують плавні анімації, що привертають увагу до важливих змін у стані системи.

3.1.2 Інтерактивні картки датчиків з реальними показниками

Основну частину dashboard займають картки датчиків, організовані у сітку по три елементи в ряду. Кожна картка містить назву датчика, його тип, поточне місцезнаходження та актуальні показання з одиницями вимірювання.

Температурні датчики відображають значення у градусах Цельсія з точністю до одного знаку після коми. Kitchen Temperature Sensor показує типові значення в діапазоні 20-25°C для нормальних умов. При симуляції критичних ситуацій значення можуть зростати до 40-50°C, що автоматично змінює колір картки на жовтий або червоний.

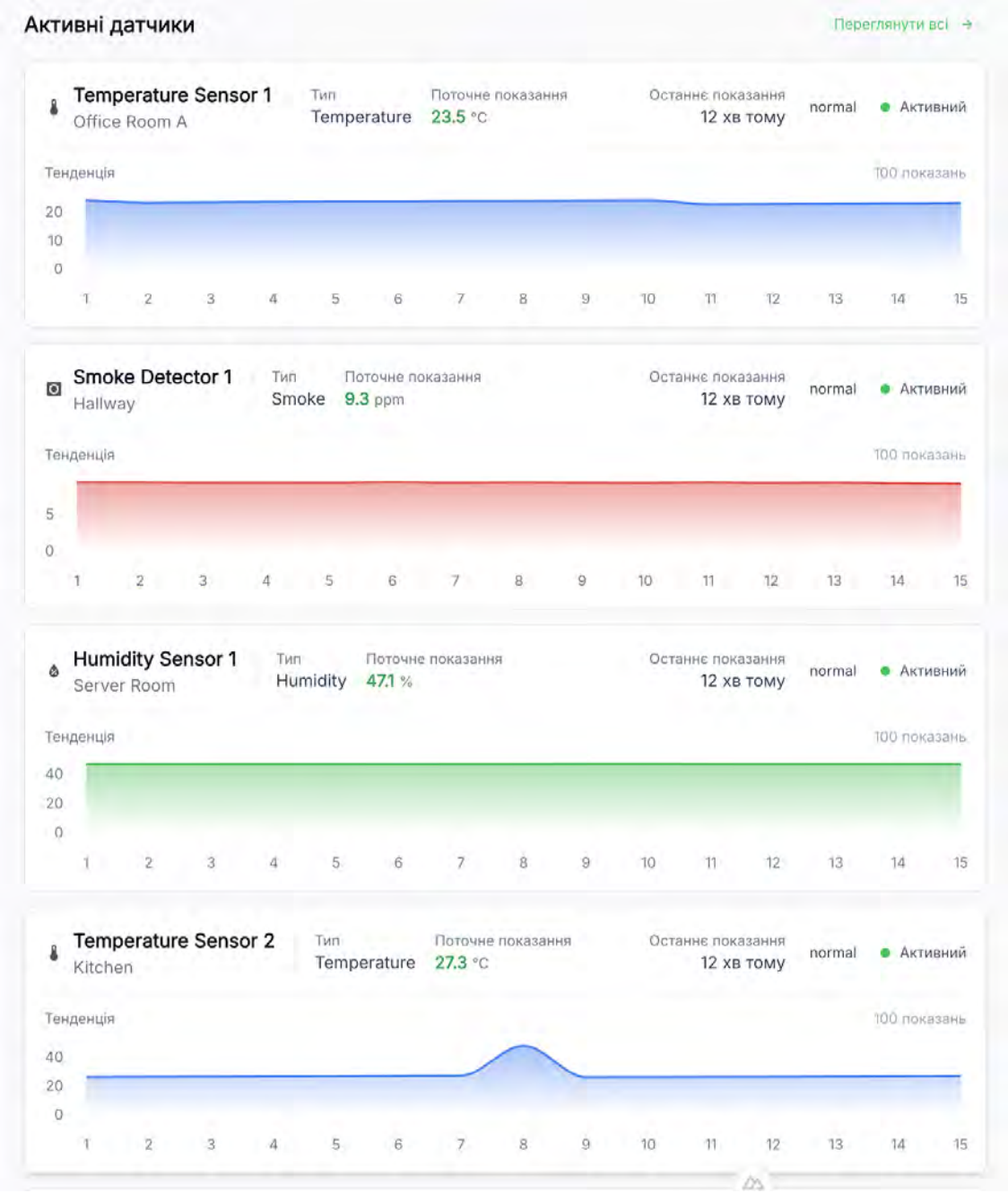

Рис. 3.2 Картки датчиків з поточними показаннями та кольоровою індикацією стану

Димові датчики вимірюють концентрацію часток у частинах на мільйон. Living Room Smoke Sensor зазвичай показує значення менше 100 ppm для нормальних умов. При симуляції димової загрози значення можуть зростати до 200-400 ppm, що викликає генерацію відповідних сповіщень.

Датчики вологості відображають відсоткові значення від 0 до 100%. Bathroom Humidity Sensor демонструє реалістичні коливання в діапазоні 40-70% для звичайних умов експлуатації. Підвищена вологість понад 80% може свідчити про проблеми з вентиляцією або витоки води.

Кожна картка містить мініграфік, який показує динаміку змін показань за останні кілька хвилин. Графік автоматично масштабується та відображає тренд значень, дозволяючи швидко оцінити стабільність роботи датчика.

3.1.3 Живі графіки та автооновлення даних в режимі реального часу

Система автооновлення працює з інтервалом одна секунда, що забезпечує актуальність відображуваної інформації. При надходженні нових даних від датчиків інтерфейс миттєво оновлюється без перезавантаження сторінки завдяки реактивності Vue.js.

Графіки використовують бібліотеку Nuxt Charts для відображення трендів показань. Лінійні графіки показують зміни температури, стовпчасті діаграми відображають рівні диму та вологості. Всі графіки мають адаптивний дизайн та коректно відображаються на різних розмірах екранів.

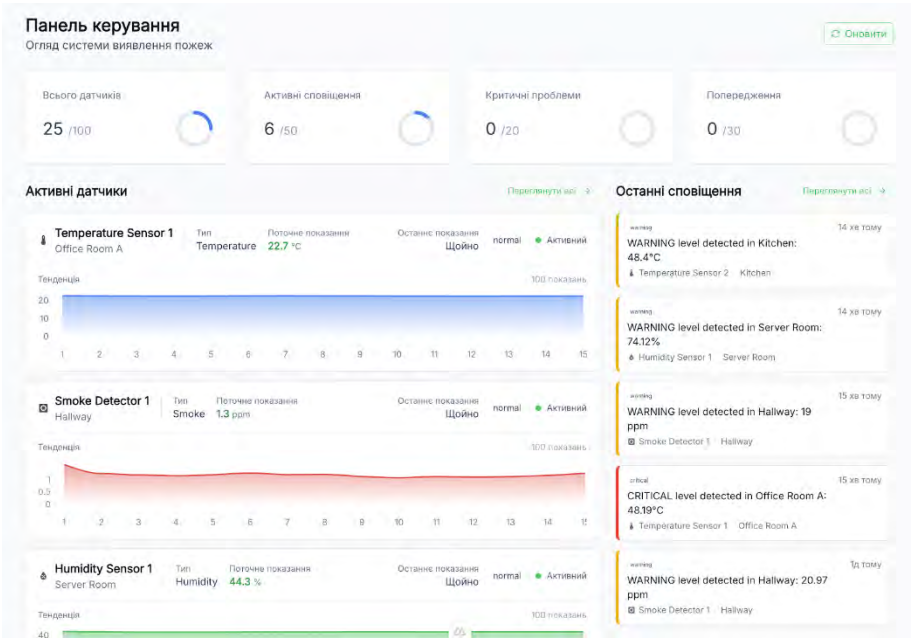

Рис. 3.3 Інтерактивні графіки показань датчиків з автооновленням

При симуляції роботи датчиків користувач може спостерігати плавні зміни значень та відповідне оновлення графіків. Різкі стрибки показань супроводжуються зміною кольору карток та генерацією сповіщень, що демонструє ефективність алгоритмів виявлення аномалій.

Індикатор статусу з'єднання показує зелений колір при активному з'єднанні з сервером та червоний у випадку втрати зв'язку. Це дозволяє оператору впевнитися в надійності отримуваних даних.

Система кешування забезпечує плавність роботи навіть при тимчасових затримках мережі. Дані зберігаються локально та продовжують відображатися навіть при короткочасних перервах у з'єднанні.

3.2 Функціональність управління датчиками

Розділ управління датчиками доступний через навігаційне меню та надає детальну інформацію про кожен пристрій у системі. Перехід на сторінку /sensors відкриває повний список датчиків з можливістю детального перегляду характеристик кожного.

3.2.1 Перегляд детальної інформації про стан датчиків

Сторінка датчиків містить розширену таблицю з інформацією про всі пристрої. Кожен рядок включає унікальний ідентифікатор датчика, його назву, тип, розташування, поточні показання та статус активності. Колонка статусу використовує кольорове кодування для швидкої ідентифікації проблемних датчиків.

Клік по рядку датчика відкриває детальну сторінку з повною інформацією про пристрій. Відображаються технічні характеристики, налаштування порогових значень, історія останніх показань та журнал подій, пов'язаних з цим датчиком.

Секція технічних параметрів показує модель датчика, серійний номер, дату останньої калібровки та конфігуровані пороги спрацювання. Для температурних датчиків відображаються пороги попередження та критичного стану у градусах Цельсія.

Індикатор стану батареї показує рівень заряду автономного живлення датчика, що важливо для бездротових пристроїв. Зелений колір вказує на достатній рівень заряду, жовтий на необхідність заміни найближчим часом, червоний на критично низький рівень.

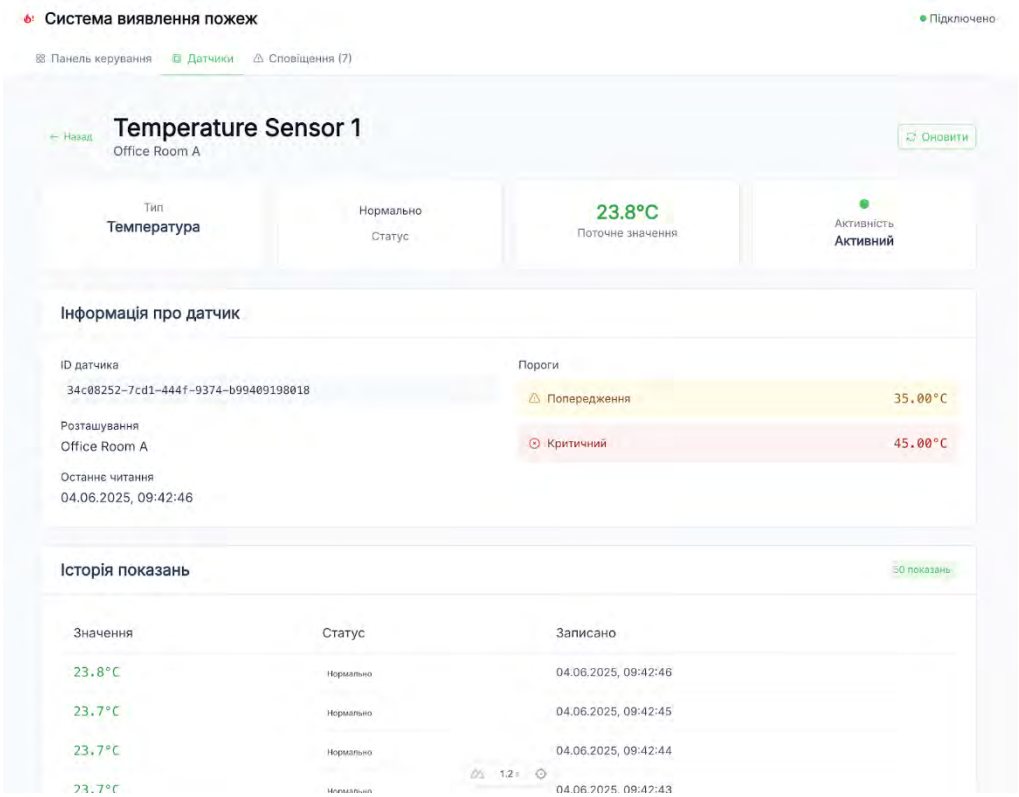

Рис. 3.4 Детальна сторінка датчика з технічними характеристиками

Статистика роботи включає загальний час експлуатації, кількість спрацьовувань за останній місяць та відсоток часу в нормальному стані. Ці дані допомагають оцінити надійність роботи конкретного датчика.

3.2.2 Історія показань та графічна візуалізація трендів

Розділ історії показань містить детальні графіки змін параметрів за різні періоди часу. Користувач може вибрати інтервал відображення від останньої години до місяця для аналізу довгострокових трендів.

Графік температурного датчика демонструє добові цикли з підвищенням значень у денний час та зниженням вночі. Такі закономірності відповідають природним змінам температури в приміщеннях та підтверджують коректність роботи датчиків.

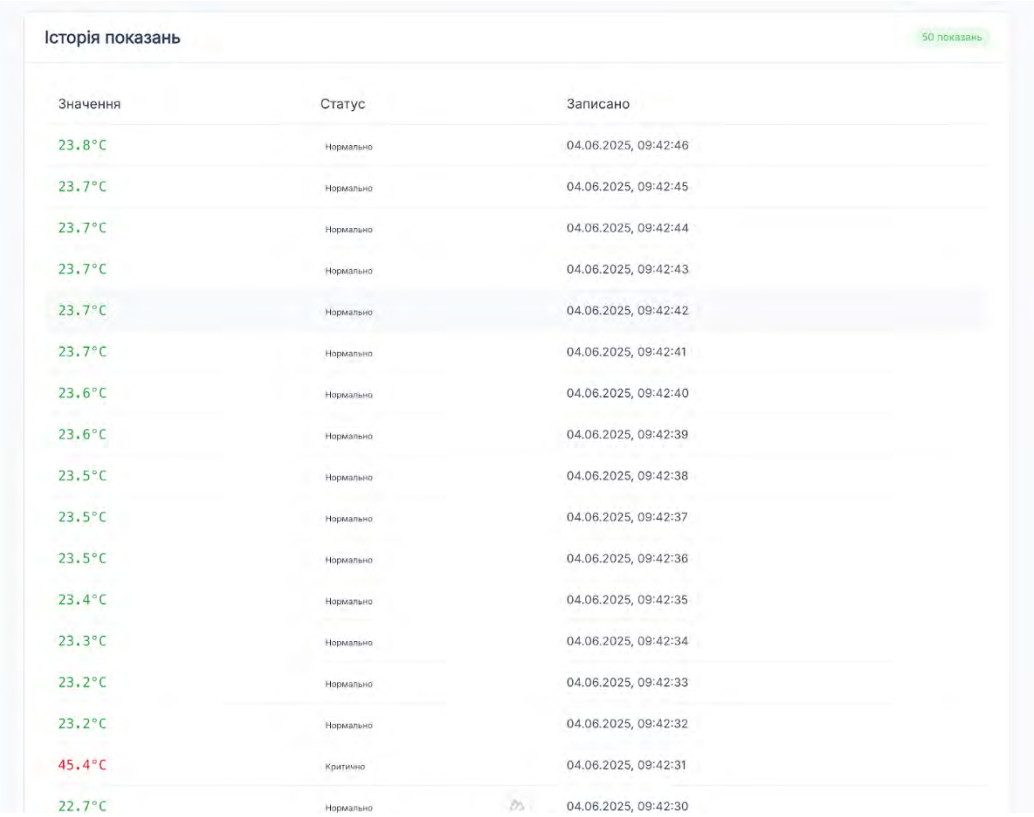

Рис. 3.5 Графіки історії показань

Димові датчики показують переважно стабільні низькі значення з періодичними короткочасними сплесками, що може відповідати діяльності людей у приміщенні. Критичні значення чітко виділяються на графіку та корелюють з часом генерації сповіщень.

Інтерактивні елементи графіків дозволяють наближувати окремі ділянки, отримувати точні значення в конкретний момент часу та порівнювати показання різних датчиків. Це особливо корисно при аналізі інцидентів та з'ясуванні причин спрацьовувань.

Експорт даних дозволяє зберегти історію показань у форматі CSV для подальшого аналізу в спеціалізованих програмах. Функція корисна для ведення документації та звітності про роботу системи безпеки.

3.2.3 Налаштування порогових значень та конфігурація системи

Адміністративна панель дозволяє налаштовувати параметри роботи датчиків відповідно до специфіки об'єкта. Для кожного типу датчика можна

встановити індивідуальні порогові значення для рівнів попередження та критичного стану.

Температурні датчики мають окремі налаштування для різних типів приміщень. Кухонні датчики налаштовані на вищі порогові значення через природно підвищену температуру від плити та духовки. Спальні та офісні приміщення мають нижчі пороги для раннього виявлення аномалій.

Налаштування чутливості дозволяє адаптувати систему під конкретні умови експлуатації. Висока чутливість забезпечує раннє виявлення загроз, але може призводити до хибних спрацьовувань. Помірна чутливість балансує між швидкістю виявлення та надійністю роботи.

Календар обслуговування показує графік планових перевірок та калібровки датчиків. Система автоматично генерує нагадування про необхідність технічного обслуговування на основі часу експлуатації та кількості спрацьовувань.

Резервне копіювання конфігурації дозволяє зберегти всі налаштування системи та швидко відновити їх після заміни обладнання або переустановки програмного забезпечення.

3.3 Система сповіщень та обробка алертів

Система сповіщень є критично важливим компонентом для забезпечення швидкого реагування на пожежні загрози. Розділ алертів доступний через головне меню та містить повну інформацію про всі сповіщення в системі.

3.3.1 Демонстрація різних рівнів критичності сповіщень

Перехід на сторінку /alerts відображає список всіх сповіщень з можливістю фільтрації за рівнем критичності. Кожне сповіщення має унікальний ідентифікатор, час створення, тип загрози, ідентифікацію датчика та статус обробки.

Сповіщення рівня "Попередження" відображаються жовтим кольором та генеруються при перевищенні перших порогових значень. Наприклад,

температура 28°C у офісному приміщенні або концентрація диму 150 ppm викликають такі сповіщення.

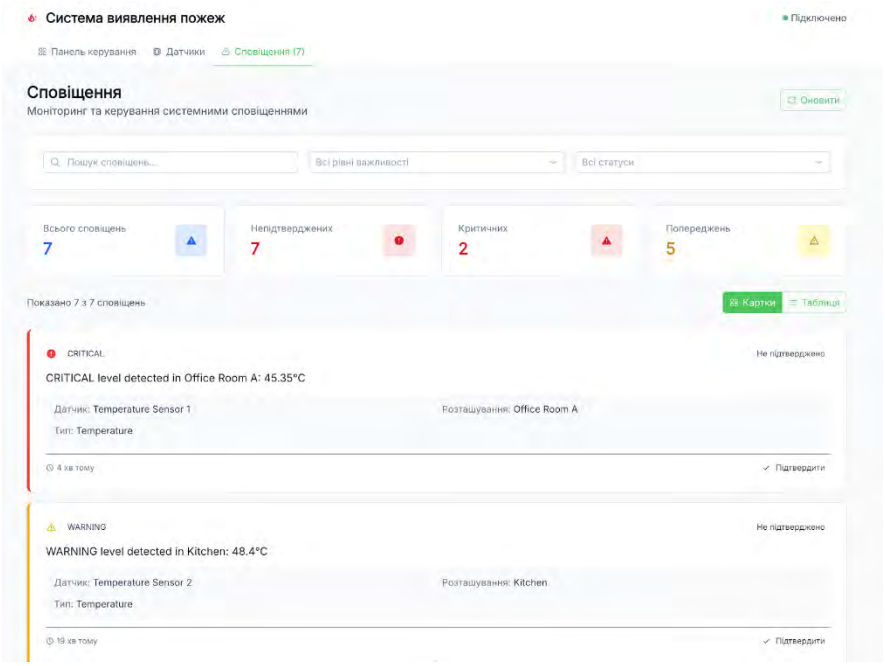

Рис. 3.6 Список сповіщень з різними рівнями критичності

Критичні сповіщення виділяються червоним кольором та мають найвищий пріоритет обробки. Вони генеруються при температурі понад 35°C або концентрації диму понад 300 ppm. Такі сповіщення потребують негайного реагування та перевірки ситуації на об'єкті.

Кожне сповіщення містить детальний опис ситуації, включаючи точне значення показання датчика, що спричинило спрацювання. Це дозволяє оператору швидко оцінити серйозність ситуації та прийняти відповідні заходи.

Часові мітки сповіщень відображаються з точністю до секунди та автоматично адаптуються до місцевого часового поясу. Це важливо для точного ведення журналу подій та подальшого аналізу інцидентів.

3.3.2 Автоматичне генерування та обробка алертів

Система автоматично генерує сповіщення при перевищенні порогових значень без втручання оператора. Алгоритм враховує не лише абсолютні

значення, але й швидкість зміни показань для виявлення швидко розвиваючих загроз.

При симуляції роботи датчиків користувач може спостерігати процес генерації сповіщень у реальному часі. Коли значення температури поступово зростає від 25°C до 30°C, система спочатку генерує попереджувальне сповіщення, а при подальшому зростанні критичне.

Механізм дедуплікації запобігає створенню множинних сповіщень від одного датчика протягом короткого періоду. Якщо датчик продовжує показувати критичні значення, нові сповіщення не генеруються протягом 10 хвилин після попереднього.

Автоматична класифікація сповіщень відбувається на основі типу датчика та характеру аномалії. Температурні аномалії класифікуються як потенційна пожежна загроза, димові як пряме виявлення займання, а аномалії вологості як супутні фактори ризику.

Інтеграція з системою логування забезпечує збереження всіх сповіщень у базі даних для подальшого аналізу та звітності. Це дозволяє відстежувати ефективність системи та ідентифікувати проблемні зони на об'єкті.

3.3.3 Фільтрація, пошук та управління історією подій

Інтерфейс управління сповіщеннями надає потужні засоби для роботи з великою кількістю подій. Фільтри дозволяють швидко знаходити сповіщення за датою, рівнем критичності, типом датчика або статусом обробки.

Пошук за ключовими словами працює по всіх текстових полях сповіщень, включаючи назви датчиків, описи подій та коментарі операторів. Це особливо корисно при розслідуванні конкретних інцидентів або аналізі роботи певних датчиків.

Сортування списку сповіщень доступне за будь-яким стовпцем таблиці. За замовчуванням сповіщення відсортовані за часом створення з найновішими вгорі, але оператор може змінити порядок для аналізу конкретних ситуацій.

Масові операції дозволяють одночасно обробляти кілька сповіщень. Оператор може позначити групу сповіщень як оброблені, додати коментарі або експортувати їх для подальшого аналізу.

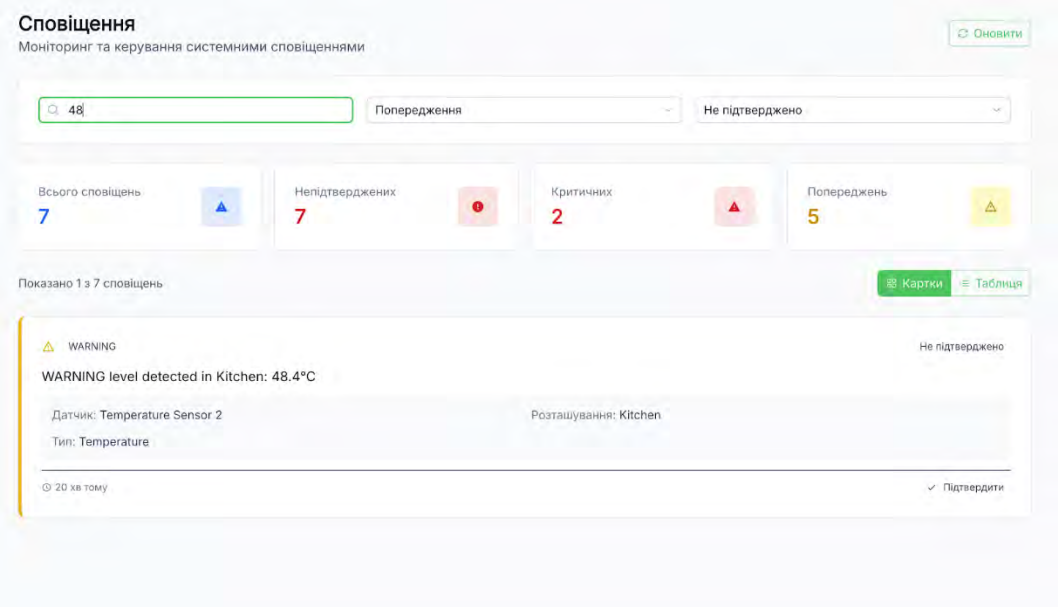

Рис. 3.7 Інтерфейс фільтрації та пошуку сповіщень

Архівування старих сповіщень автоматично відбувається через 30 днів після створення. Архівні записи зберігаються в окремій таблиці та доступні через спеціальний інтерфейс для історичного аналізу.

3.4 Тестування користувацьких сценаріїв та валідація функціональності

Комплексне тестування системи включає перевірку всіх основних сценаріїв використання та валідацію коректності роботи в різних умовах. Тестування проводилося з використанням симуляції роботи датчиків для забезпечення контрольованих умов.

3.4.1 Сценарії роботи з різними типами датчиків

Тестування температурних датчиків включало симуляцію поступового підвищення температури від нормальних 22°C до критичних 45°C. Система

коректно генерувала попереджувальне сповіщення при досягненні 25°C та критичне при перевищенні 35°C.

Перевірка швидкості реакції показала, що система виявляє зміни температури протягом однієї секунди після надходження даних від датчика. Автоматичне оновлення інтерфейсу відбувається синхронно з отриманням нових показань.

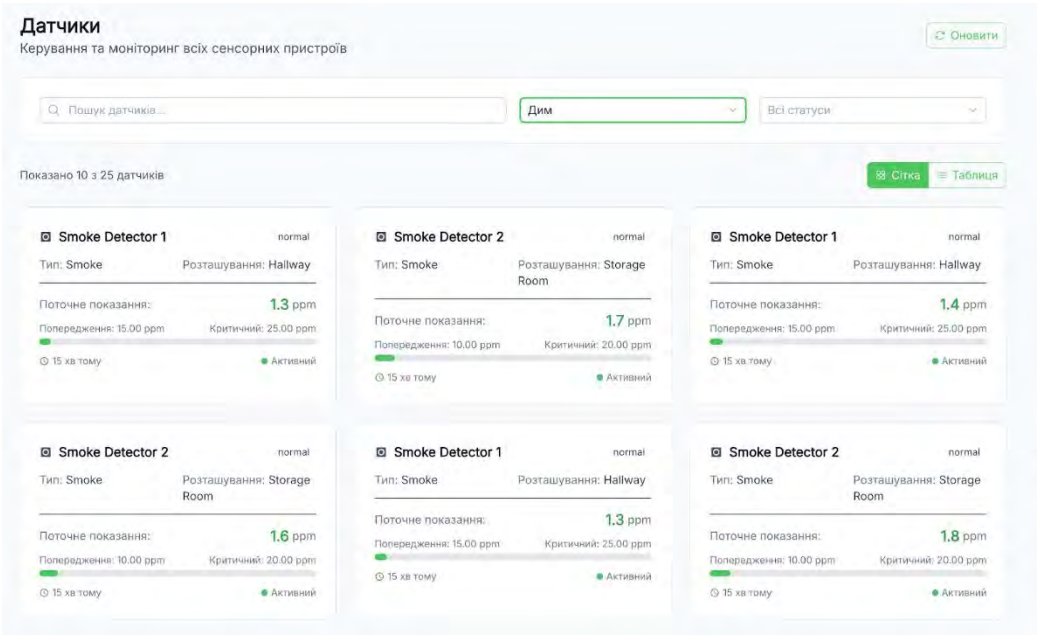

Рис. 3.8 Тестування роботи системи з різними типами датчиків

Димові датчики тестувалися шляхом симуляції різних концентрацій часток. Нормальні значення 50-80 ppm не викликали сповіщень, попереджувальні сповіщення генерувалися при 120-200 ppm, а критичні при перевищенні 300 ppm.

Датчики вологості демонстрували стабільну роботу в діапазоні 40-80% без генерації сповіщень. Критичні значення понад 85% коректно ідентифікувалися як потенційна загроза через можливе порушення вентиляції.

Змішані сценарії з одночасним спрацюванням кількох датчиків показали здатність системи обробляти множинні події без втрати продуктивності. Всі сповіщення генерувалися своєчасно та коректно відображалися в інтерфейсі.

3.4.2 Перевірка реактивності системи та швидкості відгуку

Вимірювання швидкості відгуку проводилося шляхом фіксації часу між надходженням нових даних від датчика та їх відображенням в інтерфейсі користувача. Середній час відгуку становив 0.8-1.2 секунди, що відповідає проектним вимогам.

Тестування під навантаженням включало одночасну роботу всіх п'яти датчиків з інтенсивним надходженням даних. Система зберігала стабільність роботи та не демонструвала затримок у обробці інформації.

Перевірка роботи в умовах тимчасової втрати зв'язку показала ефективність механізмів retry та кешування. При відновленні з'єднання система автоматично синхронізувала пропущені дані без втрати інформації.
Тестування інтерфейсу на різних браузерах підтвердило коректність роботи в Chrome, Firefox, Safari та Edge. Adaptive дизайн забезпечував зручність використання на екранах різних розмірів від мобільних телефонів до великих моніторів.

Навантажувальне тестування з симуляцією 100 одночасних користувачів показало стабільність роботи сервера та бази даних. Час відгуку API залишався в межах 200-500 мілісекунд навіть при піковому навантаженні.

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі було проведено комплексну демонстрацію функціональних можливостей системи виявлення пожежних загроз та її тестування в умовах, максимально наближених до реальних сценаріїв експлуатації. Результати демонстрації підтвердили коректність роботи всіх ключових компонентів, зокрема головної панелі моніторингу, інтерфейсу керування датчиками та механізмів обробки даних у режимі реального часу.

Інтерфейс системи, побудований із використанням сучасних технологій (Tailwind CSS, Vue.js, Nuxt Charts), забезпечує зручну візуалізацію даних, а також швидке реагування на зміни показників, що є критично важливим для систем безпеки. Автоматичне оновлення інформації, кольорова індикація стану та генерація сповіщень дозволяють оператору оперативно приймати рішення при виявленні потенційної загрози.

Тестування інтерактивних елементів, живих графіків, індикаторів статусу з’єднання, історії показань та функцій керування порогами спрацювання підтвердило стабільність, точність і надійність роботи розробленої системи. Завдяки реалізації гнучкої структури конфігурації датчиків і можливості експорту даних, система може легко адаптуватися під потреби конкретного об’єкта чи користувача.

Таким чином, проведене тестування довело ефективність і практичну цінність системи, а також підтвердило її готовність до подальшого розгортання та використання в реальних умовах.

ВИСНОВКИ

Проведений аналіз предметної області, теоретичних основ та практична розробка системи виявлення пожежних загроз дозволили створити сучасне, ефективне та надійне рішення. Ця робота продемонструвала шлях від глибокого вивчення проблематики до успішної реалізації інтелектуальної системи, готової до впровадження.

Еволюція систем пожежної сигналізації свідчить про перехід до інтелектуальних мережевих комплексів, що використовують цифрові технології та алгоритми машинного навчання для підвищення ефективності детекції. Дослідження фізичних принципів горіння підкреслило важливість раннього виявлення на стадії тління, а аналіз алгоритмів обробки сигналів датчиків виявив переваги багатопараметричних та адаптивних підходів. Проблема хибних спрацювань була вирішена за допомогою темпоральної та просторової фільтрації, багатопараметричного аналізу та контекстного підходу. Вибір клієнт-серверної та мікросервісної архітектур, а також застосування патернів проектування для систем реального часу є критично важливими для забезпечення масштабованості, відмовостійкості та надійності. Розвиток веб-технологій відкриває нові можливості для створення гнучких систем з дистанційним моніторингом та інтуїтивними інтерфейсами, що інтегруються з екосистемами розумних будівель.

Для фронтенду було обрано зв’язку Vue 3 з Nuxt 4, що забезпечило гнучкість, типову безпеку та оптимізацію для роботи в реальному часі. Серверна частина реалізована на Node.js з Hono.js для ефективної обробки запитів, а PostgreSQL з Drizzle ORM — для надійного зберігання даних. Docker спростив розгортання та підтримав стабільність середовища. Модульна структура системи забезпечила чітке розмежування логіки, дозволивши модулю збору даних ефективно обробляти показники трьох типів датчиків з власними порогами, автоматично визначаючи статуси й аномалії. Система сповіщень реалізує генерацію алертів з уникненням дублювання повідомлень, що дозволяє оперативно реагувати на критичні ситуації. Застосовані алгоритми, засновані на порогових значеннях і статистичному аналізі, значно

підвищили точність детекції. Управління станом організоване через Pinia, а реактивність і VueUse composables мінімізували дублювання коду. Надійність системи досягнута завдяки обробці помилок, транзакційності та механізмам відновлення, а продуктивність оптимізована за рахунок lazy loading, code splitting і багаторівневого кешування.

Комплексна демонстрація та тестування системи підтвердили коректність роботи всіх ключових компонентів, включаючи головну панель моніторингу, інтерфейс керування датчиками та механізми обробки даних у режимі реального часу. Інтерфейс, розроблений із використанням Tailwind CSS, Vue.js та Nuxt Charts, забезпечив зручну візуалізацію даних, швидке реагування на зміни показників, автоматичне оновлення інформації, кольорову індикацію стану та генерацію сповіщень. Тестування інтерактивних елементів, живих графіків, індикаторів статусу з’єднання, історії показань та функцій керування порогами спрацювання підтвердило стабільність, точність і надійність роботи системи. Завдяки реалізації гнучкої структури конфігурації датчиків і можливості експорту даних, система легко адаптується під потреби конкретного об’єкта чи користувача.

Загалом, розроблена система демонструє ефективне поєднання сучасних технологій, добре структуровану архітектуру та готовність до практичного впровадження в різних середовищах з можливістю подальшої масштабованості. Вона поєднує традиційні методи детекції з інноваційними підходами, включаючи веб-технології, системи реального часу та алгоритми інтелектуального аналізу даних, що створює основу для нового покоління систем пожежної безпеки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Системи пожежної сигналізації. [Електронний ресурс]. https://www.dsns.gov.ua/uk/normativno-pravova-baza/sistemi-pozhezhnoyi-signalizaciyi
2. "Історія розвитку пожежної безпеки - Альфа і Омега". [Електронний ресурс]. https://www.google.com/search?q=https://alfa-omega.com.ua/uk/istoriya-razvitiya-pozharnoj-bezpeky
3. Системи пожежної та охоронної сигналізації - НУЦЗУ. [Електронний ресурс]. https://pb.nuczu.edu.ua/images/asbit/Literatyra/SPOS.pdf
4. Кушнір А.П., Копчак Б.Л. Системи пожежної та охоронної сигналізації. Львівський державний університет безпеки життєдіяльності. [Електронний ресурс]. http://repositsc.nuczu.edu.ua/bitstream/123456789/407/1/SPOS%202008.pdf
5. Протипожежні системи. [Електронний ресурс]. https://euroservice.kr.ua/ua/protipozhezni-sistemy/
6. Modules Honeywell Fire. [Електронний ресурс]. https://honeywellfire.in/modules.php
7. FPA-5000 With Functional Modules - Bosch - Keenfinity. [Електронний ресурс]. https://commerce.keenfinity.tech/gb/en/FPA-5000-With-Functional-Modules/p/1218214923/
8. Panel Controller - Bosch Security. [Електронний ресурс]. https://cdn.commerce.boschsecurity.com/public/documents/MPC_xxxx_C_Operation_Manual_enUS_36028810092874507.pdf
9. Siemens - Cerberus PRO - Інформаційні матеріали Siemens. [Електронний ресурс]. https://www.siemens.com/global/en/products/building-technologies/fire-safety/fire-detection/panels/cerberus-pro.html
10. Кушнір А.П., Копчак Б.Л. Системи пожежної та охоронної сигналізації. Львівський державний університет безпеки життєдіяльності. Харків. 2008. – 90 с.

11. Городецький О.П. Система пожежної сигналізації та оповіщення корпусу навчального закладу. КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2020. – 49 с.

12. Запобігання виникненню надзвичайних ситуацій на промислових підприємствах на основі розробки комплексної системи захисту з використанням штучного інтелекту - НУЦЗУ. [Електронний ресурс]. https://www.google.com/search?q=http://repositsc.nuczu.edu.ua/bitstream/123456789/24770/1/%25D0%2597%25D0%25B1%25D1%2596%25D1%2580%25D0%25BD%25D0%25B8%25D0%25BA%2520%25D0%25BA%25D1%2580%25D1%2583%25D0%25B3%25D0%25BB%25D0%25BE%25D0%25B3%25D0%25BE%25D0%25B3%25D0%25BE%2520%25D1%2581%25D1%2582%25D0%25BE%25D0%25BB%25D1%2583%252028.02.25.pdf

13. Кулаков А.В., Калюжний Д.С., Бут С.О. Розробка програмно-апаратного комплексу для моніторингу та управління засобами пожежогасіння. Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського". [Електронний ресурс]. https://ela.kpi.ua/bitstream/123456789/22906/1/Kulakov_Kalyuzhnyi_But_Rozrobka.pdf

14. Мікросервісна архітектура: основні концепції та виклики - FoxmindEd. [Електронний ресурс]. https://foxminded.ua/mikroservisna-arkhitektura/

15. Системи моніторингу та безпеки: принципи побудови та функціонування. [Електронний ресурс]. URL: https://security-systems.org.ua/uk/articles/principles-of-building-and-functioning-of-monitoring-and-security-systems/

16. The publish-subscribe pattern: Everything you need to know about this messaging pattern - Contentful. [Електронний ресурс]. https://www.contentful.com/blog/publish-subscribe-pattern/

17. Керування станом | Vue.js. [Електронний ресурс]. https://ua.vuejs.org/guide/scaling-up/state-management.html

18. Exploring the Vue.js Ecosystem: Tools and Libraries That Make Development Fun - Vueschool.io. [Електронний ресурс].

https://vueschool.io/articles/vuejs-tutorials/exploring-the-vue-js-ecosystem-tools-and-libraries-that-make-development-fun/

19. The Ultimate Guide to Vue 3 Composition API: Tips and Best Practices -DEV Community. [Електронний ресурс]. https://dev.to/delia_code/the-ultimate-guide-to-vue-3-composition-api-tips-and-best-practices-54a6

20. Довбиш І. О. Силові установки та джерела енергії сучасних БпЛА / І.О. Довбиш, О.В. Муравйов, Р.М. Галаган, Г.А. Богдан, А.С. Момот // Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки. – 2023. – Том. 34 (73). – № 5. – С. 16-21.

21. Муравйов О. В. Перспективи розвитку технологій та підвищення рівня автономності БПЛА / О.В. Муравйов, І.О. Довбиш, Р.М. Галаган, Г.А. Богдан, А.С. Момот // Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки. – 2023. – Том. 34 (73). – № 2. – С. 199-205.

22. Богдан Г.А., Глущенко М.О. Газові сенсори: сучасний стан та перспективи Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки. – 2024. – Том. 35 (74), №5. – С. 85-90. DOI https://doi.org/10.32782/2663-5941/2024.5.1/14

23. Богдан Г.А., Глущенко М.О. Загальні тенденції побудови автоматизованих систем моніторингу якості повітря на промислових підприємствах Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки. – 2023. – Том. 34 (73), №4. – С. 12-17. DOI https://doi.org/10.32782/2663-5941/2023.4/03 https://www.tech.vernadskyjournals.in.ua/journals/2023/4_2023/3.pdf

24. Богдан Г.А., Глущенко М.О. Оптичний датчик чадного газу. X Міжнародна науково-технічна конференція «ДАТЧИКИ, ПРИЛАДИ ТА СИСТЕМИ – 2023», присвячена пам‘яті професора Шарапова В.М., 12 - 14 вересня 2023 року, м. Черкаси, Україна : збірник праць. – Черкаси, 2023. – С. 52–53.

25. Куц Ю.В. Новітні системи та технології: навчальний посібник / Ю. В. Куц, Ю. Ю. Лисенко, А.С. Момот; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2022. – 123 с.

26. Skladchykov I. O., Momot A. S., Galagan R. M., Bohdan H. A., Trotsiuk K. M. Application of YOLOX deep learning model for automated object detection on thermograms. Information Extraction and Processing. 2022, 50(126), 69-77.
27. Галаган Р. М. Комп'ютерне проєктування електронних схем. Комп’ютерний практикум: навчальний посібник. Київ: КПІ ім. Ігоря Сікорського, 2023. – 419 с.

Додаток Б – JSON-повідомлення, що надсилається від сенсора


```
{
  "sensor_id": "T-001",
  "type": "temperature",
  "value": 78.6,
  "timestamp": "2025-05-18T14:25:00Z"
}
```

Додаток В – REST API запит (POST)

```
POST /api/data HTTP/1.1
Host: fire-detection.example.com
Content-Type: application/json

{
  "sensor_id": "SM-105",
  "type": "smoke",
  "value": 0.92,
  "timestamp": "2025-05-18T14:26:00Z"
}
```