Програмна система для управління вентиляцією

Software Requirements Specification

1.0

20.04.2025

Хижняк Дмитро Станіславович

**ІСТОРІЯ ЗМІН**

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Опис** | **Автор** | **Коментарі** |
| 01.04.2025 | Створено пункти 1.1 - 1.3 | Хижняк Дмитро Станіславович |  |
| 08.04.2025 | Створено пункти 1.4 - 1.5, 2.1 - 2.3 | Хижняк Дмитро Станіславович |  |
| 15.04.2025 | Створено пункти 2.4, 2.5, 3.1 - 3.2 | Хижняк Дмитро Станіславович |  |
| 20.04.2025 | Створено пункти 3.3 - 3.5 | Хижняк Дмитро Станіславович |  |

**ЗАТВЕРДЖЕННЯ ДОКУМЕНТУ**

Наступну специфікацію вимог до програмного забезпечення було прийнято та схвалено:

|  |  |  |  |
| --- | --- | --- | --- |
| **Підпис** | **Друковане ім’я** | **Назва** | **Дата** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**ЗМІСТ**

ІСТОРІЯ ЗМІН

ЗАТВЕРДЖЕННЯ ДОКУМЕНТУ

1. ВСТУП

1.1 Огляд продукту

1.2 Мета

1.3 Межі

1.4 Посилання

1.5 Означення та абревіатури

2. ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

2.2 Функції продукту

2.3 Характеристики користувачів

2.4 Загальні обмеження

2.5 Припущення й залежності

3. КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

3.1.2 Апаратний інтерфейс

3.1.3 Програмний інтерфейс

3.1.4 Комунікаційний протокол

3.1.5 Обмеження пам’яті

3.1.6 Операції

3.1.7 Функції продукту

3.1.8 Припущення й залежності

3.2 Властивості програмного продукту

3.3 Атрибути програмного продукту

3.3.1 Надійність

3.3.2 Доступність

3.3.3 Безпека

3.3.4 Супроводжуваність

3.3.5 Переносимість

3.3.6 Продуктивність

3.4 Вимоги бази даних

3.5 Інші вимоги

1. ВСТУП

1.1 Огляд продукту

Система автоматизованого керування вентиляцією призначена для моніторингу мікрокліматичних показників у різних приміщеннях та автоматичного налаштування обладнання відповідно до встановлених порогів. Вона складається з веб-інтерфейсу для адміністраторів та клієнтів, мобільного застосунку для оперативного доступу з пристроїв на Android, а також бекенд-сервера з API для обробки запитів та взаємодії з IoT-пристроями через MQTT. Продукт інтегрується зі сховищем даних, кешем та системою push-сповіщень, забезпечуючи цілісний підхід до збору, обробки та відображення телеметрії.

1.2 Мета

Основною метою проєкту є підвищення енергоефективності та комфорту в будівлях за рахунок адаптивного керування вентиляцією на основі реальних даних. Система повинна автоматично регулювати швидкість вентиляторів залежно від поточних параметрів (температура, вологість, CO₂), мінімізувати витрати ресурсів і зменшувати втручання операторів

1.3 Межі

Система охоплює: взаємодію користувачів з веб- і мобільними фронтендами; обробку REST- та MQTT-запитів; зберігання історичних даних у PostgreSQL і Redis; автоматичне керування вентиляційним обладнанням; push-оповіщення через Firebase FCM. Поза межами рішення залишено обробку великих аналітичних масивів із застосуванням машинного навчання, управління іншими типами кліматичного обладнання (наприклад, опаленням) та віддалене оновлення прошивки IoT-пристроїв

1.4 Посилання

Для реалізації та документації системи використано офіційну документацію та ресурси:

* ASP.NET Core (офіційний сайт Microsoft) – <https://docs.microsoft.com/aspnet/core>
* Vue.js (офіційна документація) – <https://vuejs.org/>
* Kotlin Compose for Android – <https://developer.android.com/jetpack/compose>
* Tailwind CSS – <https://tailwindcss.com/docs>
* Docker Compose – <https://docs.docker.com/compose/>
* Nginx – <https://nginx.org/en/docs/>
* PostgreSQL – <https://www.postgresql.org/docs/>
* Redis – <https://redis.io/documentation>
* EMQX (MQTT-брокер) – <https://docs.emqx.com/en/>
* Firebase Authentication та Cloud Messaging – <https://firebase.google.com/docs>
* Dapper (micro-ORM) – <https://github.com/DapperLib/Dapper>
* GitHub Actions – <https://docs.github.com/actions>
* Kubernetes – <https://kubernetes.io/docs/>

1.5 Означення та абревіатури

У цьому документі використані такі умовні позначення:

* API — Application Programming Interface
* FCM — Firebase Cloud Messaging
* IoT — Internet of Things
* MQTT — Message Queuing Telemetry Transport
* Redis — Remote Dictionary Server
* SRS — Software Requirements Specification
* UX/UI — User Experience / User Interface
* SQL — Structured Query Language

2. ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Система автоматизованого керування вентиляцією може розширюватися в декількох напрямах для підвищення ефективності та функціональної повноти. Ідеться про інтеграцію модулів прогнозного аналізу на основі машинного навчання для виявлення та запобігання аномаліям мікроклімату, що дозволить реагувати заздалегідь на потенційні порушення комфортних умов. Крім того, планується підтримка додаткових типів кліматичного обладнання (опалення, охолодження, зволоження), що розширить сферу застосування рішення від житлових будинків до промислових об’єктів. Розгортання у хмарному середовищі з використанням Kubernetes забезпечить горизонтальне масштабування відповідно до навантаження та безперервність обслуговування при високій доступності

2.2 Функції продукту

Основний функціонал системи спрямований на централізоване керування всіма елементами мікрокліматичної інфраструктури. Користувачі можуть створювати та налаштовувати середовища і кімнати з вказівкою місць розташування датчиків та пристроїв. Інструмент побудови кривих регулювання дозволяє задавати залежність вентиляційної швидкості від параметрів телеметрії за допомогою інтуїтивного графічного редактора. Модуль моніторингу в реальному часі відображає поточні значення температури, вологості та CO₂, а алгоритмічний блок забезпечує генерацію команд до пристроїв через MQTT при досягненні встановлених порогів. Система автоматизованих повідомлень інформує користувачів про критичні відхилення через push-сповіщення, а аналітичний модуль формує звіти про історичні дані та динаміку змін.

2.3 Характеристики користувачів

Цільовою аудиторією системи є:

* Власники та менеджери об’єктів, які потребують детального контролю та налаштування мікроклімату для різних зон приміщень; вони мають повний набір прав для адміністрування системи та аналізу даних.
* Адміністратори середовищ, відповідальні за налаштування користувацьких ролей, управління доступом та оптимізацію роботи пристроїв; вони використовують веб-інтерфейс для конфігурації та моніторингу.
* Інженери технічної підтримки, які забезпечують коректну роботу IoT-інфраструктури та реагують на технічні неполадки; для них важливі діагностичні інструменти та логи.
* Кінцеві користувачі (оператори, співробітники), які потребують лише перегляду стану середовища та отримання сповіщень про відхилення; їм доступні обмежені права для перегляду інформації через мобільний додаток.

2.4 Загальні обмеження

Система покладається на постійне з’єднання з мережею інтернет для взаємодії з API-сервером та отримання push-сповіщень, тому в умовах нестабільного зв’язку можливі затримки або втрата даних. Для обміну з IoT-пристроями використовується MQTT, отже середовище повинно підтримувати цей протокол і мати доступ до брокера EMQX. Серверна частина розгортається у контейнерах Docker із використанням Docker Compose, що вимагає попередньо встановленого Docker Engine відповідної версії. Клієнтський веб-інтерфейс сумісний із сучасними браузерами (Chrome, Firefox, Edge) із підтримкою TLS 1.2+, а мобільний застосунок працює на Android-пристроях з API рівнем 30 та вище. Рішення не розраховане на автономний режим роботи без мережевого з’єднання та не передбачає офлайн‑зберігання даних на клієнті

2.5 Припущення й залежності

У процесі розробки та впровадження системи прийнято такі ключові припущення:

Кожний користувач має стабільне інтернет-з’єднання для взаємодії з сервером та отримання push-сповіщень. MQTT-брокер EMQX налаштований і доступний за заданою URL-адресою без потреби у локальній інсталяції кожним користувачем. Сервіси Firebase Authentication і FCM працюють безперебійно для аутентифікації та розсилки повідомлень.

Середовище розгортання представлено контейнерами Docker, при цьому припускається наявність Docker Engine і Docker Compose відповідних версій на хост-системі. Серверна частина сумісна з .NET 8.0, клієнтський Android-застосунок підтримує мінімальний API рівень 30, а веб-інтерфейс — сучасні браузери з підтримкою ES6 та CSS3.

Залежності включають: ембеддинг Dapper для доступу до бази даних PostgreSQL, Redis для кешування, Nginx як reverse proxy, а також зовнішні API: Firebase, EMQX MQTT та будь-які REST-сервіси для інтеграцій.

3. КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Інтерфейс веб- та мобільної частин має забезпечувати інтуїтивну навігацію між основними розділами системи: моніторинг, налаштування порогів, історичні дані та керування користувачами. Всі елементи керування (кнопки, поля введення, випадаючі списки) повинні бути уніфіковані за стилем і реагувати на дії користувача не пізніше ніж за 200 мс. Для веб‑інтерфейсу необхідна адаптивність під різні розміри екрану (від 320 px до 1920 px) із використанням CSS Grid та Flexbox. Мобільний UI повинен бути оптимізований під сенсорне керування та враховувати стандарти Material Design.

3.1.2 Апаратний інтерфейс

Система взаємодіє із зовнішніми датчиками та вентиляційними пристроями через MQTT-брокер EMQX, тому обладнання повинно підтримувати протокол MQTT 3.1.1 та мати мережевий модуль із можливістю підключення до TCP/IP. Для сенсорних вузлів необхідна сумісність з Raspberry Pi або іншими Linux-майстрами на базі ARM, можливість підключення через UART або USB. Вентиляційні пристрої повинні підтримувати дистанційне керування швидкістю обертів за цифровим сигналом (PWM) або відповідним MQTT-топіком.

3.1.3 Програмний інтерфейс

REST API має відповідати специфікації OpenAPI 3.0 із чітким описом шляхів, методів (GET, POST, PUT, DELETE), параметрів та кодів відповіді (200, 201, 400, 401, 403, 404, 500). Вхідні та вихідні дані повинні бути представлені у форматі JSON, а всі об’єкти описані через JSON Schema. Для інтеграції з IoT-обладнанням передбачені MQTT-теми: sensors/{roomId}/telemetry для відправки даних та devices/{deviceId}/control для команд.

3.1.4 Комунікаційний протокол

Всі канали обміну даними повинні бути захищені TLS 1.2 або вище. REST-запити користувачів та внутрішні виклики сервісів між компонентами виконуються через HTTPS із сертифікатами X.509. MQTT-з’єднання між клієнтами та EMQX повинні бути налаштовані з TLS-шифруванням та взаємною аутентифікацією за допомогою client certificates. Частота публікації телеметрії має бути не більше ніж 1 повідомлення за секунду для кожного сенсора, а гарантія доставки забезпечується Quality of Service рівня 1 (at least once).

3.1.5 Обмеження пам’яті

Серверна частина має працювати у контейнері зі щонайменше 512 МБ оперативної пам’яті; рекомендовано 2 ГБ для виробничого середовища. Кеш Redis використовує до 256 МБ з політикою eviction LRU. Клієнтські веб‑сторінки повинні завантажувати не більше 2 МБ ресурсів при першому візиті, а мобільний застосунок — утримувати споживання пам’яті нижче 150 МБ навіть під час інтенсивного стрімінгу даних.

3.1.6 Операції

Продукт підтримує сценарії розгортання «one‑click» через Docker Compose. Передбачено щоденне резервне копіювання бази даних у зовнішнє сховище та автоматичне відновлення при відмові. Моніторинг здійснюється за допомогою Prometheus і Grafana, сповіщення надсилаються до Slack. Оновлення сервісів відбувається без простою завдяки rolling‑update у Kubernetes.

3.1.7 Функції продукту

Система надає користувачам:

* створення середовищ і кімнат;
* реєстрацію пристроїв;
* налаштування порогів і кривих регулювання;
* візуалізацію телеметрії у реальному часі;
* автоматичні push‑сповіщення;
* аналітичні звіти й експорт даних.

3.1.8 Припущення й залежності

Очікується стабільний доступ до мережі Інтернет зі швидкістю не нижче 1 Мбіт/с для коректної роботи потокових оновлень. Сервер розгортається на Linux‑хості з підтримкою Docker 24+. Для функціонування push‑сповіщень необхідна безперебійна робота сервісів Firebase. Пристрої повинні підтримувати MQTT 3.1.1 і бути налаштовані на TLS‑з’єднання з брокером.

3.2 Комунікаційний протокол

* + 1. Управління користувачами
       1. Вступ

Модуль відповідає за життєвий цикл облікових записів: реєстрацію, автентифікацію, призначення ролей (Owner, Admin, User) та видалення профілів.

* + - 1. Вхідні дані
* Дані форми (email, пароль) або Google OAuth‑токен.
* Запит на зміну ролі, що містить ідентифікатор користувача та нову роль.
  + - 1. Обробка
* Сервер перевіряє токен, створює або оновлює профіль у базі, фіксує зміну ролей у журналі.
* У відповідь сформовується JWT із актуальними правами доступу.
  + - 1. Вихідні дані
* Клієнт одержує підтвердження успішної операції та оновлені дані профілю.
* У графічному інтерфейсі змінюється відображення імені й ролі.
  + - 1. Обробка помилок
* Якщо електронна пошта вже існує, система повідомляє про дублювання.
* У разі некоректного пароля користувач бачить підказку щодо вимог до складності.
  + 1. Налаштування порогів
       1. Вступ

Дозволяє власникам і адміністраторам задавати граничні значення кліматичних параметрів та формувати криву регулювання швидкості вентилятора.

* + - 1. Вхідні дані

Ідентифікатор кімнати, параметр, набір точок кривої та критичні значення.

* + - 1. Обробка
* Сервер перевіряє коректність та послідовність точок, зберігає налаштування та оновлює кеш.
* Зміни реєструються у журналі налаштувань.
  + - 1. Вихідні дані
* Користувач отримує підтвердження збереження і бачить оновлену криву у візуальному редакторі.
  + - 1. Обробка помилок
* Якщо точка виходить за межі можливого діапазону, система підсвічує помилкове поле та пояснює причину.
  + 1. Моніторинг телеметрії
       1. Вступ

Забезпечує відображення показників мікроклімату в реальному часі.

* + - 1. Вхідні дані

Телеметричні повідомлення від сенсорів, що передаються через брокер MQTT.

* + - 1. Обробка
* Прийняті дані проходять згладжування, кешуються та зберігаються до бази.
* При перевищенні порогів генерується подія тривоги.
  + - 1. Вихідні дані
* Інтерфейс оновлює графіки й числові індикатори.
* Дані стають доступними для формування звітів.
  + - 1. Обробка помилок
* У разі втрати зв’язку із сенсором відображається індикатор «Немає даних».
  + 1. Керування вентиляторами
       1. Вступ

Автоматично коригує швидкість вентилятора, тримаючи параметри у допустимих межах.

* + - 1. Вхідні дані

Актуальне значення параметра та крива регулювання.

* + - 1. Обробка
* Алгоритм визначає цільову швидкість та надсилає команду пристрою через брокер.
* Чекає підтвердження від вентилятора; у разі затримки повторює запит.
  + - 1. Вихідні дані
* Пристрій повідомляє про прийняту швидкість, що відображається в інтерфейсі.
  + - 1. Обробка помилок
* Якщо пристрій не відповідає, користувач бачить попередження та рекомендації щодо перевірки підключення.
  + 1. Push‑сповіщення
       1. Вступ

Надсилає користувачам повідомлення про критичні події.

* + - 1. Вхідні дані

Подія перевищення порога та токен пристрою користувача.

* + - 1. Обробка
* Система формує текст повідомлення та передає його в службу FCM.
* Результат доставки фіксується у журналі.
  + - 1. Вихідні дані
* Користувач отримує сповіщення на телефон.
* Статус доставки доступний адміністратору у вигляді звіту.
  + - 1. Обробка помилок
* При недійсному токені система пропонує оновити налаштування сповіщень.
  + 1. Звіти та історія даних
       1. Вступ

Формує графіки та файли з історією параметрів для подальшого аналізу.

* + - 1. Вхідні дані

Вибраний користувачем період та ідентифікатор середовища.

* + - 1. Обробка
* Дані агрегуються з бази, структуруються та зберігаються у тимчасовий кеш.
* Готується графік.
  + - 1. Вихідні дані
* Графік відображається у веб‑інтерфейсі.
  + - 1. Обробка помилок
* Якщо даних у заданий період немає, користувач отримує повідомлення з рекомендацією змінити діапазон.
  + 1. Реєстрація та вхід
       1. Вступ

Дозволяє користувачам створювати облікові записи та входити до системи.

* + - 1. Вхідні дані

Електронна пошта, пароль або Google‑акаунт.

* + - 1. Обробка
* Система перевіряє коректність даних, у разі Google‑входу взаємодіє з OAuth‑сервісом, створює профіль та генерує токен.
  + - 1. Вихідні дані
* Після успішного входу користувач потрапляє на дашборд, токен зберігається у браузері чи мобільному застосунку.
  + - 1. Обробка помилок
* При некоректних даних система показує підказку з причиною та пропонує відновлення пароля.
  + 1. Додавання пристроїв
       1. Вступ

Дозволяє реєструвати нові сенсори та вентилятори.

* + - 1. Вхідні дані

Серійний номер, тип пристрою, прив’язка до кімнати, опис.

* + - 1. Обробка
* Система перевіряє унікальність серійного номера, створює запис у базі, надсилає команду ініціалізації.
  + - 1. Вихідні дані
* Підтвердження додавання пристрою та оновлений список в інтерфейсі.
  + - 1. Обробка помилок
* Якщо пристрій не відповідає, виводиться попередження «Пристрій офлайн».
  + 1. Управління середовищами
       1. Вступ

Надає можливість створювати та налаштовувати середовища (будівлі).

* + - 1. Вхідні дані

Назва й опис середовища або запит на його видалення.

* + - 1. Обробка
* Система перевіряє права користувача, створює або оновлює запис, журналюючи зміни.
  + - 1. Вихідні дані
* Оновлений перелік середовищ.
  + - 1. Обробка помилок
* Якщо назва вже зайнята або відсутні права, показується відповідне повідомлення.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система повинна забезпечувати безперервне функціонування у промисловому середовищі з мінімальним числом відмов. Впроваджені health‑check‑механізми автоматично перезапускають контейнер сервера при критичній помилці, а брокер EMQX підтримує збереження сесій для уникнення втрати повідомлень. Критичні дії логуються, що дозволяє швидко локалізувати та усунути збої.

3.3.2 Доступність

Продукт орієнтований на коефіцієнт доступності 99,9 % завдяки реплікації бази даних, використанню зворотного проксі Nginx із функціями балансування навантаження та можливості горизонтального масштабування сервера. При відмові одного вузла трафік автоматично перемикається на резервний.

3.3.3 Безпека

Передбачено TLS‑шифрування всіх мережевих з’єднань, автентифікацію через Firebase з перевіркою JWT‑токенів і рольовим доступом. Дані зберігаються в зашифрованих томах Docker. Обчислення критичних команд здійснюються на сервері, що унеможливлює їх підробку на стороні клієнта.

3.3.4 Супроводжуваність

Архітектура мікросервісів та використання Dapper з мінімальними залежностями спрощують внесення змін у код. Документація API генерується автоматично (Swagger). Пайплайн CI/CD виконує статичний аналіз та запускає тести, що дозволяє виявляти помилки на ранніх етапах.

3.3.5 Переносимість

Система контейнеризована в Docker, що забезпечує швидке розгортання в будь‑якому середовищі, сумісному з Docker або Kubernetes. Клієнтський фронтенд побудований на стандартних веб‑технологіях і сумісний з основними браузерами та мобільними платформами.

3.3.6 Продуктивність

Сервер обробляє запити користувачів і MQTT‑події паралельно, застосовуючи кеш Redis для зменшення звернень до бази даних. Оптимізовані SQL‑запити та використання асинхронних викликів забезпечують швидкий відгук інтерфейсу навіть при великій кількості підключених пристроїв.

3.4 Вимоги бази даних

3.4.1 Структура схеми

База даних побудована у PostgreSQL і містить сутності users, environments, rooms, sensors, devices, sensor\_data, device\_data, settings, parameters та таблиці зв’язків. Для кожної сутності визначено первинний ключ типу serial (автоінкремент) або комбінований складений ключ — у випадку таблиць, де важлива комбінація ідентифікаторів (наприклад, settings).

3.4.2 Цілісність і зв’язки

Між таблицями встановлені зовнішні ключі з політиками ON DELETE CASCADE або ON DELETE SET NULL, що забезпечують каскадне видалення повʼязаних даних або збереження історії. Кардинальності реалізовано у вигляді «один‑до‑багатьох» для environments → rooms та «один‑до‑одного» для settings.

3.4.3 Індекси і продуктивність

Для прискорення вибірок створені індекси на полях sensor\_data.timestamp, device\_data.timestamp, а також комбіновані індекси (environment\_id, role) у environment\_members. Показники продуктивності оптимізовані шляхом використання секційного (partition) зберігання даних телеметрії за місяцями.

3.4.4 Безпека даних

Доступ до бази здійснюється через окремого користувача з обмеженими правами. Підключення між сервером застосунку та PostgreSQL шифрується TLS. Конфіденційні дані (паролі, токени) зберігаються тільки у хешованому або зашифрованому вигляді.

3.4.5 Резервне копіювання та відновлення

Щодня виконується повне резервне копіювання бази з використанням pg\_dump, копії зберігаються у захищеному об’єкті S3‑сумісного сховища з політикою зберігання 30 днів. Відновлення перевіряється автоматичними smoke‑тестами на тестовому стенді щонайменше раз на місяць.

3.5 Інші вимоги

3.5.1 Сумісність

Програмний продукт має бути сумісним із провідними операційними системами серверного рівня (Ubuntu LTS, Debian, CentOS) і працювати в середовищах Docker та Kubernetes. Клієнтська частина підтримує сучасні браузери (Chrome, Firefox, Edge, Safari) з актуальними версіями JavaScript‑движків.

3.5.2 Юридичні та нормативні вимоги

Система повинна відповідати вимогам GDPR щодо зберігання та обробки персональних даних користувачів. Для передачі даних сенсорів у країнах ЄС необхідно врахувати директиви щодо захисту особистої інформації в IoT.

3.5.3 Екологічні вимоги

У разі розміщення серверів у локальних дата‑центрах рекомендовано використовувати енергоефективне обладнання та схеми охолодження з мінімальним впливом на навколишнє середовище.

3.5.4 Етичні аспекти

Система не повинна збирати чи передавати дані, що дозволяють ідентифікувати особу без явної згоди користувача. Всі алгоритми повинні працювати прозоро, без прихованих методів маніпуляції значеннями.

3.5.5 Масштабованість

Архітектура має підтримувати горизонтальне масштабування шляхом додавання додаткових інстансів сервісів. Планове навантаження — до 100 тис. одночасних сенсорних пристроїв та 10 тис. активних користувачів без погіршення часу відгуку інтерфейсу.

ДОДАТОКГ

Опубліковані тези доповіді

A document with text on it

AI-generated content may be incorrect.

A document with text and words

AI-generated content may be incorrect.