Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**«Автоматизація моніторингу умов зберігання зернових культур у мультимарній інфраструктурі»**

Виконав: студент 2 курсу, групи 1АКІТР-24м спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(шифр і назва спеціальності)

Володимир ФОУЧЕК
(ПІБ студента)

Керівник: к.т.н., професор кафедри АІІТ
Ілона БОГАЧ
(науковий ступінь, вчене звання / посада, ПІБ керівника)

«11» грудня 2025 р.

Опонент: д.т.н., проф. каф. КН
Ярослав ІВАНЧУК
(науковий ступінь, вчене звання / посада, ПІБ опонента)

«11» грудня 2025 р.

**Допущено до захисту**
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)

«12» грудня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти другий (магістерський)
Галузь знань 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітньо-професійна програма Інтелектуальні комп'ютерні системи

**ЗАТВЕРДЖУЮ**
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«26» вересня 2025 року

# ЗАВДАННЯ

## НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Фоучеку Володимиру Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Автоматизація моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі»
керівник роботи Богач Ілона Віталіївна, к.т.н, доцент, доцент кафедри АІІТ
Затверджені наказом ВНТУ від «24» вересня 2025 року № 313.
2. Строк подання роботи студентом: до «10» грудня 2025 року.
3. Вихідні дані до роботи: Розроблене програмне забезпечення повинне запускатись на будь-якій ОС, де є доступ до мережі інтернет. Модель програмного забезпечення — архітектурна. Мова програмування — JavaScript і Java. Фреймворк — React.
4. Зміст текстової частини: Вступ; Аналіз предметної галузі; Проектування архітектури та огляд технологій розробки; Реалізація автоматизованої системи моніторингу; Економічний розділ; Висновки; Список використаних джерел.
5. Перелік ілюстративного (або графічного) матеріалу: Діаграма сутностей та зв'язків; Діаграма клієнт-серверного системного збору, оброблення й візуалізації телеметрії; Діаграма прав доступу в системі; Ієрархія розмежування доступу: холдинг – підприємство – посади користувачів; Інтерфейс графічного налаштування термопідвісок у силосі; Супутникове зображення майданчика з силосами.
6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-3 | Ілона БОГАЧ, к.т.н., доц. каф. АІІТ | | |
| 4 | Володимир КОЗЛОВСЬКИЙ, проф. каф. ЕПВМ | | |

7. Дата видачі завдання: «25» вересня 2025 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів магістерської кваліфікаційної роботи | Строк виконання етапів роботи | Примітки |
|---|---|---|---|
| 1 | Дослідження існуючих систем моніторингу зерносховищ та мультихмарних підходів | 25.09–10.10.2025 | виконано |
| 2 | Проєктування математичного, алгоритмічного та інформаційного забезпечення | 10.10 – 29.10.2025 | виконано |
| 3 | Аналіз та обґрунтування вибору програмних інструментів | 29.10 – 15.11.2025 | виконано |
| 4 | Розробка програмного забезпечення | 15.11 – 20.11.2025 | виконано |
| 5 | Підготовка економічної частини | до 01.12.2025 | виконано |
| 6 | Оформлення пояснювальної записки, графічного матеріалу і презентації | 25.11 – 01.12.2025 | виконано |
| 7 | Попередній захист роботи | до 02.12.2025 | виконано |
| 8 | Захист роботи | до 19.12.2025 | виконано |

Студент _____ Володимир ФОУ
(підпис) (прізвище та ініц

Керівник роботи _____ Ілона БОГАЧ
(підпис) (прізвище та ініц

# АНОТАЦІЯ

УДК 004.43

Фоучек В. О. Автоматизація моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі. Магістерська кваліфікаційна робота зі спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-професійна програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2025. 134 с.

На укр. мові. Бібліогр.: 29 назв; рис.: 16; табл.: 1.

Магістерська кваліфікаційна робота присвячена розробленню автоматизованої системи моніторингу умов зберігання зернових культур у силосах у багатохмарній інфраструктурі. На основі аналізу сучасних рішень для термометрії елеваторів та вимог промислових зерносховищ визначено функціональні вимоги до системи контролю температури, вологості, рівня зерна та $CO_2$, а також до багаторівневої візуалізації даних на рівнях «підприємство – силос – термопідвіска – датчик». Практична цінність роботи полягає у підвищенні надійності та якості довготривалого зберігання зерна за рахунок своєчасного виявлення небезпечних температурних режимів, зменшенні ризиків псування та фінансових втрат, а також у можливості інтеграції розробленого модуля з існуючими системами автоматизованого керування елеватором.

Ключові слова: автоматизована система моніторингу, зерновий елеватор, умови зберігання зерна, температурне поле силоса, тривимірна візуалізація, Java, Spring Boot, React, PostgreSQL, Grafana, багатохмарна інфраструктура.

# ABSTRACT

Fouchek V. O. Automation of Authentication and Access Management Processes in Banking Systems. Master's Qualification Thesis in the specialty 174 - Automation, Computer-Integrated Technologies, and Robotics, educational-professional program - Intelligent Computer Systems. Vinnytsia: VNTU, 2025. 134 pages.

In Ukrainian. Bibliography: 29 titles; figures: 31; table: 1.

The thesis is devoted to the development of an automated system for monitoring the storage conditions of grain crops in silos within a multi-cloud infrastructure. Based on an analysis of modern solutions for elevator thermometry and the requirements of industrial grain storage facilities, the functional requirements have been defined for a system to control temperature, humidity, grain level and $CO_2$, as well as for multi-level data visualization at the "enterprise – silo – temperature cable – sensor" levels. The practical value of the work lies in improving the reliability and quality of long-term grain storage through timely detection of hazardous temperature regimes, reducing the risks of spoilage and financial losses, and enabling integration of the developed module with existing automated elevator control systems.

Keywords: automated monitoring system, grain elevator, grain storage conditions, silo temperature field, three-dimensional visualization, Java, Spring Boot, React, PostgreSQL, Grafana, multi-cloud infrastructure.

# ЗМІСТ

# ВСТУП

**Актуальність теми.** Сучасне сільське господарство стикається з дедалі більшими викликами зберігання й збереження якості зернових культур. В умовах глобалізації ринків, зміни клімату та зростання обсягів виробництва надзвичайно важливо впроваджувати автоматизовані системи моніторингу, які забезпечують своєчасне виявлення критичних відхилень температури, вологості, рівня $CO_2$ та інших параметрів у зерносховищах. Клієнтська частина такої системи відіграє ключову роль, оскільки саме вона відповідає за інтуїтивно зрозумілу візуалізацію даних, оперативне інформування користувача та інтерактивний контроль стану обладнання в реальному часі.

Зростання обсягів виробництва зернових культур викликає необхідність удосконалення методів зберігання, адже навіть незначні відхилення температури чи вологості здатні призвести до значних втрат якості та обсягів продукції. Своєчасний моніторинг параметрів середовища в силосах і елеваторах дозволяє запобігти зростанню плісняви, бактеріального розкладу та знизити ризики економічних збитків. [1-4]

Розвиток Інтернету речей (IoT) і мультихмарних технологій відкриває нові можливості для агропромисловості: дані з великої кількості сенсорів можуть збиратись і оброблятись розподілено, що підвищує відмовостійкість і масштабованість системи. Водночас без якісної клієнтської частини, що інтегрує показники з різних хмарних сервісів [5] та забезпечує користувачам зрозумілий інтерфейс, переваги таких технологій не можуть бути реалізовані повністю.

Для операторів зерносховищ важливо мати інструменти, які дозволяють в режимі реального часу візуалізувати показники в 2D/3D форматах, отримувати сповіщення про аварійні ситуації й оперативно змінювати нормативні значення [6-7]. Використання сучасних веб-технологій (React, Grafana, Three.js) забезпечує високу швидкодію інтерфейсу, адаптивність під різні пристрої та гнучке розширення функціоналу.

Клієнтська частина, яка орієнтується на потреби кінцевого користувача стає «мостом» між складними алгоритмами обробки даних у хмарі та практичними завданнями операторів. Це обумовлює високу актуальність теми розробки системи моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі.

**Мета роботи:** підвищення ефективності та продуктивності системи моніторингу умов зберігання зернових культур шляхом створення високофункціональної системи в мультихмарній інфраструктурі, що дозволить знизити втрати продукції, пришвидшити прийняття рішень операторами та оптимізувати експлуатацію зерносховищ.

Для досягнення поставленої мети необхідно вирішити **наступні задачі**:

1. Проаналізувати існуючі рішення модулів систем моніторингу сільськогосподарських об’єктів.
2. Спроектувати архітектуру модуля платформи.
3. Розробити та реалізувати ключові компоненти інтерфейсу користувача: інтерактивну візуалізацію параметрів, таблиці звітів, модулі сповіщень і панель управління налаштуваннями..
4. Інтегрувати Three.js для 3D-візуалізації параметрів середовища [8-9].
5. Провести тестування розробленого програмного забезпечення для виявлення та усунення помилок.

**Об’єктом дослідження** є процес розробки системи моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі.

**Предметом дослідження** є методи та інструменти розробки системи моніторингу умов зберігання зернових культур.

**Методи дослідження**. Для досягнення поставленої мети був використаний аналіз літературних джерел, розробка, тестування та налагодження програмного забезпечення.

**Науково-технічний результат** роботи полягає у створенні високофункціонального модуля системи моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі, який забезпечує збирання даних із розподілених IoT-сенсорів, інтегровану 2D та 3D візуалізацію середовища,

динамічне фільтрування табличних звітів і гнучке налаштування порогових значень датчиків в єдиному інтерфейсі з масштабованою мультихмарною архітектурою.

**Практичне значення** отриманих результатів полягає у можливості впровадження розробленого модуля в реальних зерносховищах для підвищення точності та оперативності виявлення аварійних відхилень, зниження втрат продукції, оптимізації роботи операторів, пришвидшення прийняття управлінських рішень та зменшення експлуатаційних витрат за рахунок мінімізації людської похибки.

**Апробація результатів роботи**. Результати роботи було представлено на Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації (2025) [10], подано до Міжнародної науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)» та підготовлено свідоцтво на реєстрацію авторського права на комп’ютерну програму.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Актуальність та фактори збереженості зернових культур

Зберігання зернових культур є задачою із виразною міждисциплінарною природою, де перетинаються фізика пористих середовищ, мікробіологія, ентомологія та інженерія керованих середовищ. Комерційна цінність партії зерна упродовж усього періоду зберігання визначається динамічною рівновагою між інтенсивністю біологічних процесів у насипі (дихання зерна, розвиток грибів і комах) та здатністю оператора підтримувати мікроклімат у безпечних межах. Класичні настанови для складів і силосів формулюють дві базові вимоги: тримати зерно сухим та утримувати рівномірне, якнайнижче з технологічно можливих, температурне поле; саме ці умови мінімізують швидкість псувальних реакцій і знижують ризики мікотоксинів та інфестацій.

Ключова небезпека зумовлена активністю так званих «складських» грибів, діапазон оптимуму температур для яких припадає приблизно на 23–40 °C; за наявності підвищеної вологовмістності такі гриби швидко колонізують міжзерновий простір і ініціюють самонагрівання. Саме тому контроль температури й вологи в об’ємі насипу є обов’язковим елементом технології зберігання, незалежно від типу сховища. [11]

Термін «вологість зерна» є спрощенням для більш фундаментальної характеристики - водоактивності (*aw*), що відображає термодинамічно доступну воду для мікробіологічних процесів і прямо пов’язана з рівноважною відносною вологістю повітря (*ERH*) у контакті з зерном: aw≈ERH/100aw≈ERH/100. Для більшості бактерій критичним є поріг $aw > 0{,}91$; дріжджі переважно потребують $aw > 0{,}88$; цвілеві гриби можуть виживати й рости при ще нижчих *aw*, з нижньою межею близько 0,65, що пояснює випадки пліснявіння навіть у на вид сухих партіях. Відповідно, безпечність зберігання слід оцінювати не лише за масовою часткою вологи, а й через рівноважні стани «зерно–повітря», які залежать від температури.

Кількісну основу для таких оцінок задають сорбційні ізотерми (моделі ГАВ/GAB, Гендерсона тощо), що описують зв'язок між вологістю зерна, температурою та відносною вологістю навколишнього повітря. На їх базі будують таблиці рівноважної вологості (EMC) для операційних рішень: якщо повітря з певними T/RH подається на аерацію досить довго, вологість зерна прагне до відповідного EMC. Це дозволяє керувати процесами «досушування» та «охолодження» без перевитрат енергії й уникати конденсації в зоні «холодної стелі» [12].

Фізіологічне дихання зерна та мікробіологічні процеси мають температурну залежність типу Q10/Арреніуса: невелике підвищення температури у вологій масі призводить до непропорційного зростання швидкості реакцій, що, у свою чергу, прискорює виділення тепла та «розкручує» позитивний зворотний зв'язок самонагрівання. До теплотворного балансу долучається й «сухе» нагрівання від активних комах-шкідників [13]: навіть у партіях із формально «безпечними» показниками вологості локальні популяції здатні сформувати гарячі осередки. Це підкреслює необхідність просторово розрізненого контролю (по висоті та радіусу силоса), а не лише «середніх» значень.

У практиці прийнято оперувати категоріями «коротко-/середньо-/довгострокового» зберігання з поправкою на наявність аерації та рівень механічних домішок/битого зерна. Зокрема, без активної аерації тривалість безпечного зберігання істотно обмежена навіть для партій із нормативною вологістю — через повільне, але неминуче накопичення біотепла; системи аерації знімають це обмеження, але потребують надійних датчиків і алгоритмів запуску в «правильні вікна» температурно-вологісних умов [14].

На рівні раннього виявлення ризиків традиційні канали (температура, відносна вологість) дедалі частіше доповнюють моніторингом вуглекислого газу ($CO_2$) у надзерновому просторі та/або в повітропроводах. Концептуально $CO_2$ — це інтегральний побічний продукт дихання зерна, грибів і комах; його концентрація в «головному просторі» часто зростає раніше, ніж встигає змінитися поле температури в масиві (тепло поширюється повільніше, ніж конвективні потоки газу

несуть $CO_2$ до сенсорів). Низка експериментальних і прикладних робіт (у т. ч. 2025 р.) показала, що безперервний $CO_2$-моніторинг забезпечує швидше та чутливіше сповіщення про початок псування та зростання ризику мікотоксинів, ніж ізольовані вимірювання T/RH; отже, поєднання «T/RH усередині насипу + $CO_2$ у headspace/plenum + зовнішні T/RH» є науково й практично обґрунтованим базисом для сучасних систем.

З економічного погляду наслідки ігнорування цих факторів вимірюються не лише втратою якості (класифікація, зниження ціни продажу) та кількості (усушка, висипання), а й прямими масовими втратами внаслідок інтенсивного дихання за небезпечних T/RH. За оцінками галузевих оглядів, сумарні втрати маси можуть досягати 0,5–3 % залежно від температури, вологості та тривалості зберігання; для великих парків силосів це конвертується в суттєві фінансові втрати навіть без катастрофічних інцидентів. Саме тому інвестиції у моніторинг і керовану аерацію мають доведену окупність.

### 1.2 Методи і засоби моніторингу системи

Методи моніторингу умов зберігання зернових культур охоплюють як фізичні засоби вимірювання у насипі та надзерновому просторі, так і інформаційно-комунікаційні підсистеми збору, валідації та передавання телеметрії до аналітичних сервісів. У практиці домінує кабельна термометрія з багатоточковими цифровими датчиками, що встановлюються у підвісках по висоті та, за потреби, з різними відступами від стінки силоса для формування просторового «січення» температурного поля. Розміщення й кількість кабелів впливають на репрезентативність даних і енергоефективність алгоритмів аерації: експериментальні та інженерні огляди показують, що різні схеми розподілу сенсорів здатні давати відхилення усереднених температур до кількох градусів та впливати на коректність керування вентиляторами [15]; індустріальні рекомендації для розстановки кабелів наголошують на необхідності аналітичного проєктування

сітки вимірювань з урахуванням діаметра силоса, наявності «корінгу» та теплоізоляційних властивостей зернової маси.

Типовою сенсорною базою для температури в підвісках є цифрові термометри сімейства 1-Wire, зокрема DS18B20 [16], що забезпечують вимірювання в діапазоні −55…+125 °C із типовою точністю ±0,5 °C у робочому діапазоні −10…+85 °C і програмованою роздільністю 9–12 біт; багатокрапкове адресування на одній шині спрощує монтаж та обслуговування великої кількості точок у кожній підвісці. У поєднанні з ними широко застосовують датчики відносної вологості/температури повітря (наприклад, сімейство SHT3x), для яких характерні типові точності близько ±2 %RH і ±0,3 °C та фабричне калібрування з температурною компенсацією, що є критичним для коректної оцінки рівноважних станів «зерно–повітря». На етапі експлуатації важливі процедури періодичної перевірки та рекалібрування з урахуванням довгострокового дрейфу.

Система моніторингу не обмежується вимірюванням у масиві зерна. Для раннього виявлення деградаційних процесів впроваджують контроль вуглекислого газу у надзерновому просторі (headspace) та в пленумі/повітропроводах. NDIR-сенсори $CO_2$ фіксують інтегральний ефект дихання зерна, грибів і комах; численні дослідження показують, що безперервний $CO_2$-моніторинг забезпечує більш швидку й чутливу індикацію початку псування та підвищення ризику мікотоксинів порівняно з ізольованими каналами температури та відносної вологості. Окремі роботи відзначають діагностичну цінність відхилень $CO_2$ від фонового рівня 400–500 ppm як практичного порогу для настороження, тоді як інженерні керівництва рекомендують поєднувати $CO_2$ з температурою/вологістю для підвищення надійності рішень. Експлуатація NDIR-сенсорів передбачає контроль температурного дрейфу, часової стабільності й калібрувальних механізмів (включно з двоканальними схемами та самокалібруванням), що має бути задокументовано на рівні регламентів.

Оскільки керування аерацією є ключовою операцією підтримання якості, у системі моніторингу доцільно вимірювати також параметри зовнішнього повітря й стан у пленумі. Класичні довідники та керівництва FAO описують закономірності

«просування фронту охолодження/осушення» та залежність рівноважної вологості зерна (EMC) від температури й відносної вологості повітря; практика безпечного зберігання спирається на ці залежності для вибору «вікон» аерації та попередження конденсації під «холодною стелею». У підсумку підсистема датчиків зовнішніх умов стає невід'ємною частиною інформаційного контуру, що замикає вимірювання «всередині насипу» з керуванням вентиляторами/нагрівачами.

Окрема група засобів — ручні зонди/термоштанги для оперативного контролю й верифікації даних стаціонарної системи, особливо в плоских складах та на об'єктах із неповною кабельною інфраструктурою. Профільні рекомендації університетських та галузевих видань наголошують на необхідності регламентних оглядів і частішого інспектування за підвищених температур зовнішнього повітря; водночас для великих діаметрів силосів рекомендовано інсталяцію вбудованих систем термометрії як базового інструмента скорочення трудовитрат і попередження деградації [17].

Щодо архітектури збору даних, сучасні системи покладаються на edge-контролери, що опитують мережі підвісок і локальних датчиків, виконують первинну фільтрацію (перевірка діапазонів, усунення «стрибків», усереднення ковзаючим вікном), буферизують часові ряди та публікують їх у шину повідомлень. Для інтероперабельності у виробничому контурі використовують OPC UA (єдина модель даних, сервісно-орієнтована архітектура, надійна і захищена взаємодія клієнт–сервер і PubSub), тоді як для легковагової телеметрії до хмарних брокерів типовим вибором є протокол MQTT (OASIS-стандарт публікації/підписки, оптимізований для нестабільних каналів і обмежених пристроїв). Такий «OPC UA всередині об'єкта + MQTT назовні» забезпечує сумісність із наявними АСУ ТП/SCADA та масштабовану публікацію в мультихмарні аналітичні сервіси, не порушуючи семантики даних.

Вимоги до кібербезпеки при підключенні сенсорної мережі до корпоративних і хмарних середовищ формулюються в термінах серії ISA/IEC 62443: сегментація на зони й канали, моделювання загроз, контроль доступу, журналювання та відстежуваність, керування оновленнями, випробування безпеки

й забезпечення життєвого циклу. Стандартизована зональна модель дозволяє мінімізувати поверхню атаки, розмежувати OT/IT та формалізувати політики взаємодії брокерів повідомлень, OPC UA-серверів і зовнішніх API.

З точки зору обчислювальної обробки, якісне відтворення фізичних процесів у насипі потребує не лише «точних» сенсорів, а і методично обґрунтованих частот опитування, просторового покриття та процедур контролю якості даних. На етапі проєктування схеми розміщення датчиків враховують теплоізоляційні властивості зернової маси, можливі «тіні» від центральних колон/кореневих каналів та особливості завантаження/розвантаження; імітаційні та польові дослідження показують, що кабелі біля центру чи біля стінки можуть давати нерепрезентативні результати, а отже, для задач керування аерацією потрібне збалансоване покриття по радіусу. Це безпосередньо впливає на якість клієнтських візуалізацій: інтерфейс має коректно інтерполювати дані між підвісками, підсвічувати невизначеність і не створювати ілюзій точності там, де просторової роздільності недостатньо.

Узагальнюючи, «засоби» моніторингу — це не лише сенсори в насипі, headspace та зовнішні метеоканали, а й сформована методика їх інсталяції/калібрування, телеметричний стек (OPC UA/MQTT), політики безпеки (ISA/IEC 62443) і клієнтські механізми обробки/візуалізації, які відображають специфіку фізичних процесів у зерні. Саме інтегральність цього підходу забезпечує своєчасне виявлення «гарячих осередків», енергоефективну аерацію та зниження ризиків псування в реальних умовах експлуатації — відповідно до усталених настанов із безпечного зберігання та сучасних науково-практичних результатів.

### 1.3 Підходи до хмарної і мультихмарної обробки телеметрії

Хмарна обробка телеметрії у задачі моніторингу умов зберігання зернових культур спирається на чітко визначену модель сервісів і розгортань, що забезпечує еластичність, масштабованість і оперативний доступ до даних. Визначення Національного інституту стандартів і технологій США окреслює хмарні

обчислення як модель «зручного, всюдисущого, на вимогу» доступу до спільного пулу конфігурованих ресурсів із п’ятьма сутнісними ознаками, трьома сервісними моделями та чотирма моделями розгортання; ця таксономія є загальноприйнятою відправною точкою для порівняння варіантів побудови телеметричних сервісів (гарячі панелі моніторингу, історичні сховища часових рядів, аналітика) та їх подальшої еволюції у багатохмарні сценарії.

У виробничому контурі з підключеним обладнанням критичне значення має інтероперабельність між рівнями «датчик/контролер — edge — хмара». Практично виправданою є комбінація двох стандартів: OPC UA як «семантичного ядра» обміну даними усередині об’єкта (моделі об’єктів, узгоджені сервіси, автентифікація клієнтів/серверів, цілісність і конфіденційність трафіку) та MQTT як легковагового транспорту подій у напрямі хмари з підтримкою публікації/підписки і різних класів гарантій доставки. Такий розподіл дозволяє зберегти «зміст» (інформаційні моделі та атрибути) у межах технологічної мережі й водночас ефективно винести масові телеметричні потоки до брокерів і хмарних сервісів аналітики.

Сучасна специфікація MQTT 5.0 [18] є релевантною для високонавантажених IoT-сценаріїв: окрім класичних рівнів QoS для керування компромісом «затрати ↔ гарантії доставки», вона містить розширення, що спрощують масштабування і надійність (поліпшена діагностика помилок, властивості користувача для розширення протоколу, сесійні налаштування тощо). Для каналу «edge → хмара» це означає кращий контроль життєвого циклу повідомлень, прозоріші відмови та можливості адаптації до обмежених або нестабільних каналів зв’язку — типового випадку для віддалених зерносховищ.

Архітектурно багатохмарні (multi-cloud) підходи до телеметрії вирішують одразу кілька завдань: усувають залежність від одного провайдера, покращують відмовостійкість, дозволяють розміщувати «гарячі» сервіси ближче до об’єкта (зменшення затримок) і оптимізують витрати завдяки вибору найкращих сервісів у кожній хмарі [27]. Оглядові роботи 2024–2025 років унаочнюють, що техніки федерації й гібридного розподілу (контейнери/оркестрація, абстракція даних і

політик, уніфіковане спостереження) є ключовими провайдерами зрілості багатохмарних екосистем, а також підкреслюють виклики — інтероперабельність, узгоджене управління політиками доступу і наскрізна спостережуваність потоку даних. Для систем моніторингу з великою кількістю віддалених вузлів це безпосередньо транслюється у підвищення надійності оповіщень і сталу доступність історії вимірювань.

Безпека промислових систем у контексті підключення до хмар вимагає проектування за принципами серії ISA/IEC 62443. Модель «зон і каналів» (zones & conduits) дозволяє сегментувати виробниче середовище (OT) і визначити строго контрольовані канали взаємодії з інформаційними сервісами (IT) та хмарними брокерами; це охоплює ідентифікацію/автентифікацію, шифрування, аудит та політики доступу на межах зон. Розміщення брокерів повідомлень і OPC UA-серверів у демілітаризованих сегментах, погоджене з цими принципами, суттєво зменшує поверхню атаки та формалізує вимоги до наскрізної захищеності «edge → хмара» [19].

Узгодження часової семантики та якості даних у багатохмарному конвеєрі має неабияке значення для коректної аналітики: на рівні edge доцільно виконувати штампування вимірювань «часом події», локальне буферування і «store-and-forward», дедублікацію/ідемпотентні оновлення при повторній доставці (характерній для режимів QoS≥1), а також контроль дрейфу годинника (NTP/PTP) для збереження порівнюваності часових рядів у різних доменах. На «хмарному» рівні багатошарові сховища (оперативний TSDB для «гарячих» вікон і об’єктне «холодне» сховище для довгострокового зберігання) компенсуються наскрізними SLO за затримкою доставки й доступністю, тоді як обчислювальні служби (потокова обробка і пакетна аналітика) отримують дані через брокери публікації/підписки з урахуванням гарантій MQTT 5.0. Така рамка, поєднана з OPC UA як «семантичним каркасом» у технологічній мережі, створює стійкий до збоїв і переносний у межах кількох хмар конвеєр телеметрії, адекватний вимогам аграрної галузі до безперервності моніторингу та простежуваності рішень оператора.

## 1.4 Огляд існуючих рішень моніторингу умов зберігання зернових культур

У промисловій практиці зберігання зерна сформувався помітний спектр рішень від класичної кабельної термометрії до хмарних платформ із раннім виявленням ризиків $CO_2$. Ці системи різняться сенсорною базою (багатоточкові підвіски в насипі, headspace/plenum-датчики, зовнішні метеоканали), рівнем аналітики (тренди, індекси ризику, 2D/3D-візуалізація), ступенем автоматизації (від сповіщень до керування вентиляторами/нагрівачами) та здатністю до інтеграції в мультихмарні архітектури (OPC UA/MQTT, API). Розглянуто рішення для українського ринку, зокрема OPIsystems Inc, Temix LLC та SmartTerm LLC з акцентом на їхні функціональні можливості, архітектуру збору телеметрії, сценарії експлуатації, сильні сторони й обмеження.

### 1.4.1 OPIsystems Inc

OPIsystems — один із найвідоміших міжнародних виробників рішень для моніторингу й керування зберіганням зерна. Актуальна продуктова лінійка охоплює дві «вісі»: (а) редизайн хмарного моніторингу EPIQ Remote Monitoring як еволюцію OPI Blue, зорієнтований на спрощення інсталяції та зниження вартості входу, і (б) екосистему OPI BLUE Full Automation із підтримкою автоматизованого керування вентиляторами/нагрівачами на базі показників у зерні та стану зовнішнього/пленумного повітря. За офіційним описом, EPIQ «усуває потребу в окремому шлюзі (gateway)», що «зменшує стартові витрати на тисячі доларів», має «посилені магнітні з’єднання» й «збільшений ресурс батареї»; рішення позиціонується як «сучасний дизайн, надійна робота, доступна вартість володіння»

з постійним 24/7 моніторингом і сповіщеннями у мобільних і desktop-клієнтах. OPIsystems – Advancing Grain Management У свою чергу, OPI BLUE — це «бездротова система керування зберіганням», що «автоматично доставляє дані про стан зерна» на смартфони/ПК із «24/7 моніторингом і тривогами», а версія Full Automation інтегрує Fan & Heater Control (сторінки продукту/мануали демонструють окрему сторінку керування вентиляторами і журнали рішень ON/OFF).

З інженерної точки зору OPIsystems традиційно спирається на кабельну термометрію (багатоточкові підвіски із цифровими датчиками температури, у конфігураціях — вологи), що забезпечує щільну «внутрішню» карту температурного поля; дані агрегуються в хмарі, відображаються на карті об'єкта/силосів, у трендах і таблицях; конфігуруються пороги та сценарії сповіщень. В описах/реселерських матеріалах наголошено на автоматизації аерації: інтеграція вентиляторів/нагрівачів дає змогу оптимізувати кондиціонування зерна (якість/вологість) та зменшувати витрати (електроенергія, фумігація, праця).

- Переваги: Зріла екосистема, масштабованість від фермерських бункерів до елеваторних парків, глибока підтримка алгоритмічної аерації (вентилятори/нагрівачі), зручні мобільні клієнти; редизайн EPIQ знижує бар'єр входу (без окремого шлюзу) й прискорює інсталяцію.
- Потенційні обмеження: Як і для будь-яких кабельних систем — вартість/регламент обслуговування підвісок та потреба ретельно проектувати їх розміщення; $CO_2$-канал частіше виступає як опція/інтеграція, тож у проєктах із наголосом на ранніх індикаторах псування доведеться передбачити відповідні модулі. (Висновок базується на доступних матеріалах постачальника/реселерів і мануалах із Fan Control/Cloud.) [20].

### 1.4.2 Temix LLC

Temix — інженерно-виробнича компанія з понад 20-річним фокусом на системах контролю температури зерна: стаціонарні термопідвіски (багатоточкове вимірювання по висоті насипу) і термоштанги(ручні/бездротові зонди) для силосів, бункерів і плоских складів; компанія також постачає обладнання вимірювання рівня та виконує монтаж і пусконалагодження. Офіційні сторінки формулюють призначення як «комплекс технічних засобів для безперервного або циклічного контролю температури зернової насипі», що «усуває вплив людського фактору і попереджає недопустиме самосогрівання».

Типова архітектура Temix — мережа цифрових підвісок + блоки опитування/ПЗ із картою об’єкта, журналами, тривогами. Перевагою є локальна сервісна підтримка, ремонтопридатність та можливість модернізації існуючих систем за рахунок сумісності з поширеними типами підвісок; профільні огляди ринку також згадують систему ІТУ-3 як «універсальний засіб контролю температури», що «легко нарощується та інтегрується у системи верхнього рівня». У практиці українських елеваторів це зручно для поступового дооснащення без зупинки об’єкта.

Переваги: Повний «кабельний» стек для термометрії (підвіски, штанги, ПЗ), локальний сервіс/монтаж, досвід роботи з різними типами сховищ.

Потенційні обмеження: Публічні матеріали Temix фокусуються переважно на температурі; канали $CO_2$/headspace/plenum і автоматизована аерація зазвичай реалізуються як інтеграційні проєкти або в партнерстві

### 1.4.3 ДНВП «Ельдорадо»

ДНВП «Ельдорадо» пропонує системи термометрії для силосів, плоских складів і насіннєвих елеваторів. Офіційні сторінки подають призначення: вимірювання температури зерна по висоті, у низці комплектацій — також вимірювання рівня; для насіннєвих елеваторів наведено спеціалізовані

комплектації. Технічні описи вказують на використання цифрових датчиків температури Dallas (DS18B20) у складі підвісок, опитуваних блоками віддаленого контролю з передаванням у ПЗ/панелі оператора [21].

Архітектура відповідає усталеній практиці: стаціонарні підвіски у силосах/складах, блоки опитування, візуалізація, тривоги. Headspace/$CO_2$-контроль та автоматизована аерація зазвичай додаються на проєктній основі (через зовнішні датчики/шлюзи).

Переваги: стандартизована конструкція підвісок, локальна документація/сервіс і наявність рішень для насіннєвих об’єктів.

Обмеження: для впровадження раннього виявлення ($CO_2$) та алгоритмічної аерації потрібна інтеграція із зовнішніми модулями/сервісами.

## 1.5 Висновки до розділу 1

У першому розділі було здійснено огляд предметної області зберігання зернових культур, визначено ключові фактори збереженості та окреслено вимоги до первинних вимірювань і якості даних. Показано, що термічні та газові процеси у товщі зернової маси мають виражену просторово-часову неоднорідність, а їхній перебіг суттєво залежить від зовнішнього мікроклімату, тривалості зберігання та історії технологічних операцій. Це зумовлює необхідність багатоканального моніторингу з ієрархічною прив’язкою «елеватор, силос, підвіска, датчик», синхронізацією часових міток і зберіганням повних метаданих (геометрія, культура, партія, профіль порогів).

Було проведено аналіз ринку рішень, які забезпечують класичний моніторинг (збір, двовимірна візуалізація, оповіщення і частково керування вентиляцією). Водночас виявлено спільні обмеження: закритість апаратних екосистем, ризик залежності від постачальника, брак відкритих промислових протоколів, стандартні інтерфейси для інтеграції з виробничими та корпоративними системами, відсутність тривимірних перерізів внутрішньосилосних структур; фокус на

статичних порогах без формалізованих динамічних індикаторів (добова зміна температури, коефіцієнт варіації, детекція змінних режимів). Крім того, у відкритих матеріалах не декларується мультихмарна стратегія з цілями відновлення та сценаріями відновлення після аварій, що обмежує масштабоване промислове використання в умовах високих вимог до доступності.

Зіставлення вимог предметної області з ринковими спроможностями дозволило сформувати набір принципів для подальшого проєктування автоматизованої системи моніторингу: інтероперабельність через відкриті промислові протоколи (Modbus, OPC UA) і контрактні інтерфейси REST, ієрархічна двовимірна та тривимірна візуалізація з можливістю одночасного аналізу підвісок і датчиків, динамічні порогові моделі для аналітики (добова зміна температури), індикатори варіації та детекція точок змін, а також мультихмарність та відмовостійкість.

# 2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ОГЛЯД ТЕХНОЛОГІЙ РОЗРОБКИ

## 2.1 Концептуальна модель системи і критерії якості функціювання

Концептуальна модель ґрунтується на ієрархії «підприємство → силос → підвіска → датчик → вимір» та охоплює три узгоджені рівні:

- Рівень представлення. Веб-інтерфейс оператора з оглядовими панелями, тривимірною сценою силосів, таблицями і графіками часових рядів; підтримує контекстні підказки, порівняння періодів, фільтри та керування порогами.
- Рівень логіки. Сервіси нормалізації телеметрії, обчислення індикаторів стану (зміна температури за добу, індекси варіації), виявлення аномалій, формування інцидентів і рекомендацій. Тут зосереджено правила ескалацій і ведення журналу рішень.
- Рівень даних. Реляційне ядро для довідників і конфігурацій; таблиці часових рядів для телеметрії; кеш «гарячих» агрегатів; архівація і відновлення.

Критерії якості функціювання формулюються операційно й підлягають верифікації під час експлуатації:

- Свіжість даних: різниця між моментом вимірювання та відображення — не більше 1 секунди.
- Затримка інтерфейсу: час від дії користувача до повного відмальовування не більше 0,5 секунди.
- Час доставки сповіщення: від події до повідомлення — не більше 10 секунд.
- Доступність критичних функцій: не менше 99,5% на місяць.
- Стійкість до деградацій: відставання оброблення черг стабілізується в межах, визначених політикою автомасштабування.

## 2.2 Архітектура системи і моделі взаємодії

Архітектура побудована за принципом клієнт–серверної взаємодії з подієво орієнтованим транспортом (рис. 2.1). Такий підхід чітко розділяє відповідальності, спрощує незалежне масштабування підсистем і локалізує збої.

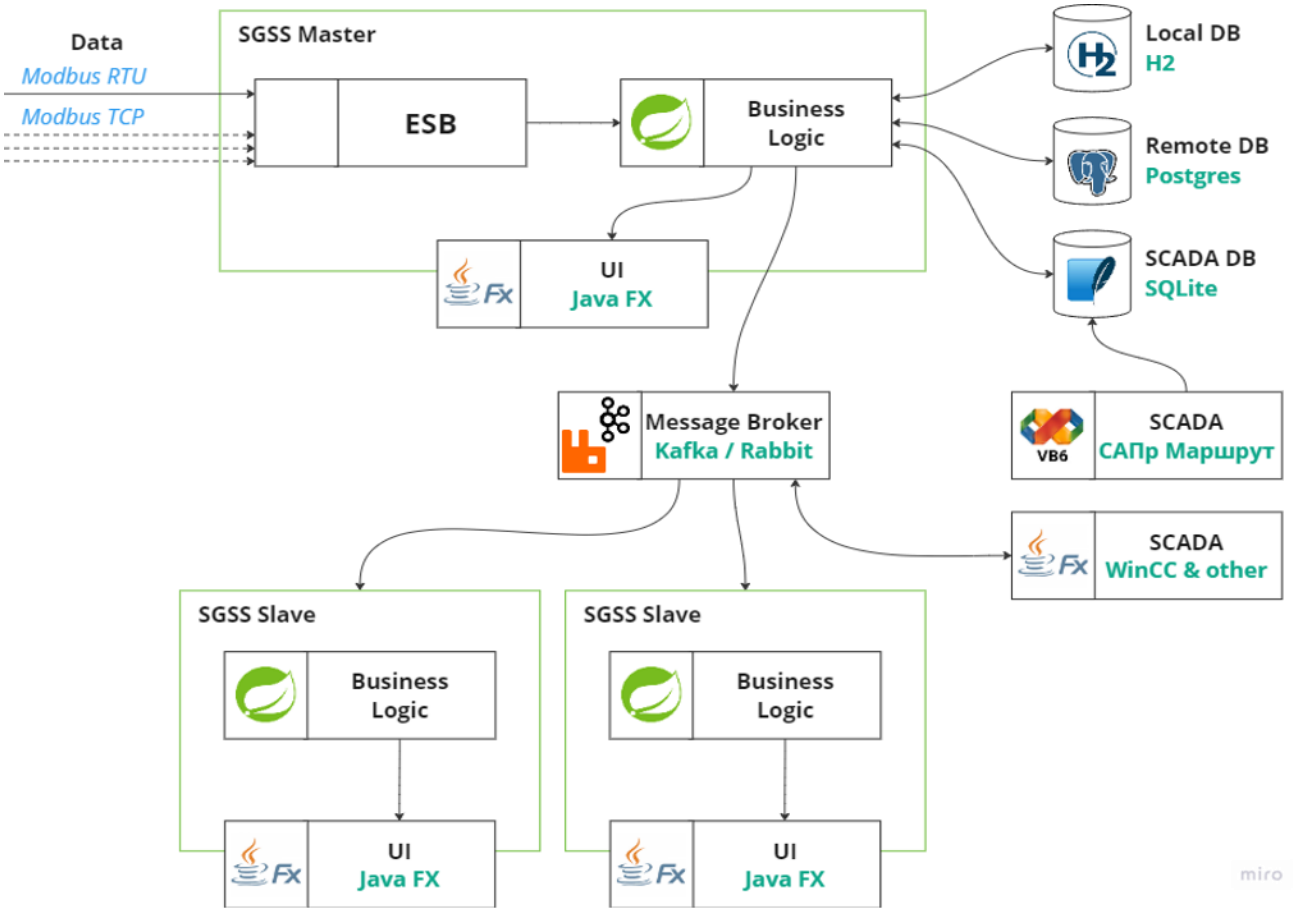


Рисунок 2.1 – Архітектура клієнт-серверної системи

Основні компоненти і потоки даних:

1. Збір телеметрії. Драйвери промислових протоколів опитують контролери термопідвісок та допоміжних сенсорів, встановлюють часові мітки, виконують первинні перевірки діапазонів і збагачують дані метаданими (ідентифікатор, геометрія, культура, партія)

2. Транспортний шар. Брокер подій забезпечує партиціювання повідомлень за ознаками підприємства, силоса та датчика, повторні спроби з контрольованими затримками і спеціальну чергу для відбракування некоректних записів.

3. Сервіси нормалізації та аналітики. Формують уніфіковані часові ряди, обчислюють індикатори зміни стану, виявляють аномалії і створюють інциденти з повним контекстом.

4. Сховище та доступ. Конфігурації та довідники — у реляційній базі; телеметрія у таблицях часових рядів із партиціюванням; агрегати «для екрана» кешуються. Програмний інтерфейс застосунків (API) надає методи читання, фільтрації та агрегації.

5. Візуалізація і сповіщення. Двовимірні огляди, тривимірні зрізи та табличні подання синхронізовані за часовими вікнами; єдиний інбокс подій забезпечує ескалації, кореляцію і глибокі посилання у контекст.

Моделі взаємодій і спостережність: для ланцюжка «від сенсора до екрана» визначено контрольні точки вимірювання затримок (опитування, доставка, нормалізація, кеш, відмальовування). Контракти повідомлень мають версіонування; ідемпотентність гарантується природними ключами (ідентифікатори та часові мітки).

Надійність і безпека: підтримуються сценарії «активний — резервний» і «активний — активний» з географічним рознесенням; визначено цілі точки та часу відновлення; застосовано керування ідентичностями і доступами, шифрування каналів і даних на зберіганні, аудит і політику мінімальних привілеїв.

## 2.3 Огляд технологій для розробки

Перш за все, критичною є підтримка промислових протоколів та робота з послідовними інтерфейсами через RS-485/USB-перехідники, стійкість до завад, кросплатформність (Windows 7–11 і Linux), наявність імітаторів для тестування та прозора діагностика каналів. Прагматичним вибором у зерновій термометрії виступає Modbus RTU/Modbus TCP як де-факто стандарт для контролерів термопідвісок із готовими бібліотеками для роботи з картами регістрів, таймаутами, повторними спробами та лінійною діагностикою; для інтеграції з

наявними системами диспетчеризації доцільно застосовувати OPC UA як уніфікований інтерфейс читання даних. Доступ до COM/tty забезпечує роботу з віртуальними послідовними портами з «щадними» таймаутами та контрольованою швидкістю обміну.

Основні ризики — електромагнітні завади й «німі» пристрої — нівелюються екранованими кабелями, гальванічною розв’язкою, експоненційними повторними спробами, а також чорними списками з поступовим поверненням. Додатково, при роботі з зовнішніми джерелами телеметрії доцільний режим доступу «лише читання», що зменшує поверхню атаки й операційні ризики.

Шар транспорту та подієвої шини покликаний гарантувати надійну доставку, ізоляцію потоків і спостережність. Вимоги включають горизонтальне масштабування, партиціювання за ідентифікаторами підприємства, силоса, датчика, наявність повторних спроб, підтверджень, збереження та «черг відбракування».

Доцільним є застосування брокера повідомлень промислового класу з семантикою тем і черг; для опису корисного навантаження виправдані JSON як прозорий формат і Apache Avro зі «реєстром схем» для компактності та еволюційної сумісності під високим навантаженням. Типова помилка — надто груба чи надто дрібна гранулярність тем, що призводить до «гарячих» партицій; її уникають добором ключа партиціювання з високою ентропією, наприклад композицією «силос × датчик × квант часу». Еволюція форматів даних потребує суворого сумісного версіонування та відокремлення обов’язкових і необов’язкових полів, що мінімізує ламкі зміни.

У шарі оброблення потоку даних і бізнес-логіки визначальними є здатність працювати в реальному часі з цільовою затримкою до однієї секунди, ідемпотентність операцій та експлуатаційна простота. Практика показує, що за сценаріїв зі «середньою» обчислювальною складністю (добові різниці, коефіцієнти варіації, детекція змін режимів) раціонально застосовувати легкі сервісні обробники, що підписані на теми брокера, замість важковагових потокових рушіїв; це знижує операційну складність і спрощує життєвий цикл розгортань. Для

періодичних завдань — агрегацій, архівацій, перевірок цілісності — доцільно використовувати вбудовані планувальники або Quartz.

Головні ризики — неузгоджені часові пояси й дублікати подій — нівелюються нормалізацією часових міток до UTC на вході та впровадженням ідемпотентної обробки на базі природних ключів.

Шар зберігання та кешування має одночасно підтримати транзакційну цілісність довідникових сутностей і високу пропускну здатність часових рядів. Реляційна СУБД на кшталт PostgreSQL або MySQL залишається виправданою для реєстрів, конфігурацій, журналів і посилальних зв’язків, тоді як часові ряди ефективно розміщуються в ній же за рахунок партиціювання за часом і силосом та цільових індексів за міткою часу, силосом, підвіскою й датчиком. Гарячі агрегати інтерфейсу доцільно кешувати в оперативній пам’яті (Redis або Hazelcast) з чіткими політиками інвалідизації при надходженні нових вимірів. Для обміну та архівації обґрунтовано використовувати JSON як універсальний формат, CSV для імпорту зі електронних таблиць, а Parquet — для історичних архівів із колонковим стисненням. Ріст об’єму телеметрії пом’якшується політикою життєвого циклу з поділом на гарячий, теплий і холодний шари та ізоляцією аналітичних запитів на репліках «тільки для читання», що зменшує конкуренцію за ресурси.

Клієнтська частина, включно з двовимірним і тривимірним інтерфейсом, має відповідати вимогам продуктивності, доступності й довгострокової підтримуваності. Раціонально обрати фреймворк інтерфейсу на сучасному типізованому TypeScript із компонентним підходом, зрілою екосистемою, підтримкою інтернаціоналізації та уніфікованим керуванням станом і маршрутизацією між «Мапою силосів», «Карткою силоса», «Звітами» та «Сповіщеннями». Для тривимірної графіки доречно спиратися на WebGL із використанням бібліотеки сценографії на кшталт Three.js, що забезпечує сцени, камери, матеріали, освітлення, перетинання променем і інстансинг великої кількості однотипних об’єктів датчиків. Ризики стосуються перевантаження клієнта великою кількістю примітивів і зниження FPS; їх пом’якшують батчинг,

рівні деталізації, відкладене завантаження та використання апаратного інстансингу, а також уважна оптимізація обчислювально містких шейдерів.

Шар спостережності та безпеки забезпечує керованість і відповідність вимогам експлуатації. З боку спостережності доречні централізоване журналювання подій, метрик і трасувань, визначення SLI/SLO для затримок оброблення та успішності доставки, а також дашборди та оповіщення за порогами. З боку безпеки виправдані наскрізне шифрування трафіку, рольова авторизація з мінімально необхідними правами, аудит дій користувачів і сервісів, захист секретів у сховищах на кшталт «vault», а також твердіння кінцевих точок і перевірка залежностей. Типові ризики — ескалація привілеїв, витік секретів, ін’єкції та помилки конфігурацій; їх зменшують регулярним пентестом, політиками обертання ключів, WAF-захистом для публічних інтерфейсів і відокремленням середовищ.

Нарешті, шар керування конфігураціями та розгортанням визначає відтворюваність і масштабованість. Доцільно застосувати декларативне керування інфраструктурою та конфігураціями, контейнеризацію сервісів і оркестрацію з можливістю горизонтального масштабування, а також безперервну інтеграцію й доставку з автоматизованими тестами контрактів та перевірками безпеки. Розумними стратегіями розгортання є blue-green або canary з автоматичним відкотом у разі деградації показників. Ризики «дрейфу» конфігурацій і людських помилок мінімізуються Git-керованими джерелами істини, шаблонізацією параметрів середовищ, політиками перевірки змін і поетапним промоушном артефактів від середовища розробки до продуктивного.

Узгоджена реалізація перелічених шарів — від надійного знімання телеметрії з термопідвісок до тривимірної візуалізації стану силосів і керованого життєвого циклу розгортань — формує цілісну архітектуру, здатну масштабовано, безпечно й відтворювано обробляти потоки даних, забезпечувати коротку затримку оновлення інтерфейсу та надавати операторам прозорий інструментарій для своєчасного виявлення відхилень і ухвалення рішень.

## 2.4 Вибір інструментів розробки для рендерингу тривимірної графіки

Візуалізація внутрішніх процесів у силосі потребує тривимірного подання з геометричною прив’язкою датчиків і можливістю оперативного аналізу просторово-часових патернів. Для клієнтської частини обрано бібліотеку Three.js як високорівневу надбудову над Web Graphics Library (WebGL), що надає сценограф (сцена, камери, джерела світла), матеріали, механізми інстансингу однотипних об’єктів та засоби взаємодії (перетинання променем).

Геометрична модель силоса (рис. 2.2) задається як поверхня обертання з параметрами радіуса і висоти, а також аналітичними спрощеннями для даху та днища; усередині формується логічна сітка за кільцями і висотами для маркування підвісок, кожна підвіска моделюється як вертикальний стрижень з вузлами на заданих відмітках, а координати вузла у циліндричній системі зберігаються разом із метаданими датчика. Значення вимірювань проектуються у колірний простір за допомогою перцептивно рівномірних палітр, шкали та легенди синхронізуються з пороговими профілями інцидентів, а отже оператор інтуїтивно співвідносить колір на сцені з кількісним діапазоном параметра і класом стану.

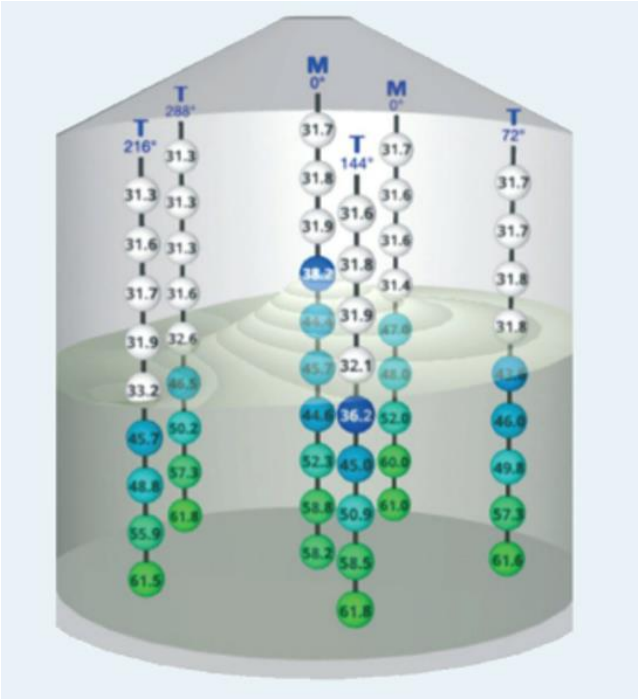


Рисунок 2.2 – Візуалізація силуса з розміщення термопідвісок

Конвеєр рендерингу вибудовується як стабільний ланцюжок від даних до сцени: клієнт отримує з програмного інтерфейсу застосунків батчі телеметрії для заданого інтервалу, нормалізує одиниці, фіксує часові мітки і статуси якості, готує буфери геометрії та атрибутів, обчислює матриці трансформацій для інстансів, а далі оновлює лише ті частини сцени, що змінилися, не перевизначаючи геометрію повністю. Обчислювально витратні перетворення виносяться у робітники браузера, що знижує вплив на основний потік взаємодії, а синхронізація часових вікон між графіками і тривимірною сценою забезпечує когерентне відображення одних і тих самих даних у різних поданнях без візуальних розбіжностей.

Просторова аналітика реалізується через горизонтальні та вертикальні зрізи, які дозволяють ізолювати потрібні шари і меридіани та швидко локалізувати проблемні зони у товщі зернової маси; у місцях рідкої сенсорної сітки застосовується локальна інтерполяція з обмеженням дальності впливу, щоб уникати штучних осциляцій, причому інтерполяційні обчислення, за потреби, переносяться у вершинні або фрагментні шейдери, що зберігає реактивність інтерфейсу. Зони з низькою довірою до відновленого поля позначаються графічними індикаторами, а користувач завжди може вимкнути інтерполяцію, залишивши лише фактичні вузли вимірювань.

Взаємодія з тривимірною сценою підтримує природну навігацію з обертанням, панорамуванням і масштабуванням, збереження корисних ракурсів у профілі користувача і швидке повернення до них, виділення об’єктів за допомогою перетинання променем та показ контекстних підказок зі значеннями, добовими змінами і класом стану, а також переходи до детальної картки датчика і відповідного фрагмента часового графіка. Для порівняння періодів передбачено режим синхронних камер, коли два види сцени відображають поточний і базовий інтервали з ідентичними параметрами огляду, що дозволяє об’єктивно оцінювати динаміку без паралаксних спотворень.

Продуктивність і стабільність кадру забезпечуються сукупністю прийомів: інстансингом маркерів датчиків, відсіканням невидимих об’єктів за межами обрізного об’єму камери та маскуванням груп, рівнями деталізації геометрії корпусу силоса і підвісок, лінивим завантаженням текстур і сіток, а також адаптивною частотою оновлень атрибутів у «гарячих» станах, коли пріоритет віддається оновленню індикаторів стану, тоді як декоративні ефекти можуть відмальовуватися зниженою частотою. Цільові орієнтири включають час до першого змістовного кадру для картки силоса в межах узгодженого стандартом експерименту значення на референсному комп’ютері, стабільну частоту кадрів без провалів під час взаємодії і час відгуку на зміну шару або фільтра, що вкладається у половину секунди для дев’яносто п’ятого перцентиля. У разі виявлення апаратних обмежень система автоматично переходить у двовимірні або табличні подання з ідентичними фільтрами, легендами і логікою вибору.

Вимоги доступності реалізуються через перцептивно рівномірні палітри, узгоджені з двовимірними екранами, висококонтрастні теми, масштабовані шрифти, текстові описи ключових елементів сцени для екранних дикторів і повну клавіатурну навігацію за допомогою фокусованих гарячих областей. Таке поєднання гарантує, що користувачі з різними потребами і на різному обладнанні отримують однаково інформативний і керований інтерфейс.

Інтеграція модуля сцени з клієнтською архітектурою виконується за принципом суворої інкапсуляції: графічний модуль відповідає за створення сцени,

камер, світла, матеріалів і життєвого циклу кадру, натомість шар стану надає чисті дані і конфігурацію через уніфікований контракт, у якому для кожного вузла передаються ідентифікатори об’єктів, координати у циліндричній системі, поточне значення, одиниці виміру, часова мітка і статус контролю якості. Маршрути між «мапою силосів», «карткою силоса», «звітами» і «сповіщеннями» зберігають контекст вибору, а отже перехід між екранами не руйнує користувацький сценарій аналізу.

Контроль якості будується на поєднанні функціональних і навантажувальних випробувань, синтетичних транзакцій для вимірювання часу до першого змістовного відображення та часу відгуку, візуальної регресії зі порівнянням еталонних знімків сцени, а також тестів на надійність з відновленням після втрати контексту WebGL, змін розміру вікна і очищенням графічних ресурсів під час навігації. Така дисципліна випробувань дає відтворювані оцінки й дозволяє запобігати непомітним деградаціям.

## 2.5 Висновки до розділу 2

У другому розділі сформовано цілісну архітектурно-технологічну концепцію автоматизованої системи моніторингу умов зберігання зернових культур, яка поєднує чітку інформаційну модель з подієво орієнтованими механізмами оброблення даних та вимірюваними критеріями якості функціювання. Концептуальна модель, що ґрунтується на ієрархії «підприємство → силос → підвіска → датчик → вимір», забезпечує трасованість кожного значення від первинного джерела до керувального рішення, узгоджує відповідальності між рівнями представлення, логіки та даних і задає єдині контракти взаємодії. Така декомпозиція створює необхідні передумови для керованої еволюції системи, повторного використання компонентів і прозорості експлуатації.

Показано, що критерії якості — свіжість даних, затримка інтерфейсу, час доставки сповіщень, доступність критичних функцій і стійкість до деградацій —

сформульовано в операційних термінах та прив’язано до способів вимірювання в реальному середовищі. Для кожного критерію визначено точки інструментування у ланцюжку «від сенсора до екрана», що уможливлює не лише контроль дотримання цілей, а й швидке локалізування джерел деградації. Запропонований підхід інтегрує вимоги предметної області з інженерними обмеженнями, перетворюючи «якість» із декларації на керований процес.

Архітектура системи обґрунтована як клієнт–серверна з подієвою шиною у серці оброблення. Рівень збору забезпечує надійне опитування промислових контролерів із часовою синхронізацією та первинною валідацією; транспортний шар надає збереження, партиціювання та ідемпотентні повтори із чергою відбракування; сервіси нормалізації та аналітики виконують перетворення «сирих» вимірів на уніфіковані ряди, розрахунок добової зміни температури, індексів варіації та маркування інцидентів з повним контекстом. Сховище поєднує реляційне ядро для довідників і конфігурацій із таблицями часових рядів, доповненими кешем «гарячих» агрегатів для швидкого відображення на екрані, а також політиками архівації та відновлення. Такий поділ ролей зменшує зв’язаність компонентів, полегшує ізоляцію збоїв і спрощує незалежне масштабування найнавантаженіших ділянок.

З позиції надійності та безпеки описано стратегії «активний — резервний» і «активний — активний» з географічним рознесенням, цілі точки та часу відновлення, вимоги до шифрування на каналах і на зберіганні, керування ідентичностями та доступами, аудит і політику мінімальних привілеїв. Передбачено керованість змін завдяки версіонуванню контрактів, міграціям схем даних і поетапним оновленням із можливістю швидкого відкату. Обрані технологічні засоби — відкриті промислові протоколи для взаємодії з обладнанням, продуктивний брокер подій, реляційне сховище з партиціюванням часових рядів, стандартизовані програмні інтерфейси із машинночитною специфікацією, контейнеризація та інфраструктура як код — забезпечують портативність розгортань та мінімізують залежність від конкретного постачальника.

Окремо обґрунтовано вибір засобів клієнтської візуалізації у трьох вимірах. Використання Three.js як високорівневої бібліотеки над Web Graphics Library дає змогу реалізувати геометрично коректну модель силоса, інстансинг великої кількості вузлів-датчиків, інтерактивні просторові зрізи та, за потреби, обчислювально ощадливу інтерполяцію полів у шейдерах, зберігаючи цільові орієнтири за стабільністю кадру і часом відгуку інтерфейсу. Узгодження тривимірних подань із двовимірними екранами та табличними оглядами, а також вимоги доступності до палітр, шрифтів і навігації забезпечують єдиний користувацький досвід на різних класах пристроїв і в різних експлуатаційних сценаріях.

Ризики експлуатації і розвитку проаналізовано та адресовано інженерними засобами: електромагнітні перешкоди на лініях усуваються правильним монтажем і повторними спробами на рівні протоколу, «гарячі» партиції у транспорті нівелюються коректним вибором ключів партиціювання, артефакти інтерполяції обмежуються просторовими фільтрами та індикаторами достовірності, продуктивні обмеження застарілих пристроїв компенсуються деградацією з гідністю у двовимірні або табличні представлення без втрати семантики. Пропонований набір практик спостережності — показники, журнали, розподілене трасування з кореляційними ідентифікаторами — закриває потребу в прозорості «від сенсора до екрана» та підтримує прийняття управлінських рішень на підставі даних.

Загалом розділ доводить технічну здійсненність і внутрішню узгодженість обраної архітектури, підтверджує здатність системи досягати визначених цілей якості функціювання та окреслює практичний шлях до промислового впровадження. Сформований фундамент дає змогу перейти до реалізації прототипу, підготовки стендів і сценаріїв випробувань та виконання експериментальної перевірки показників — свіжості даних, затримок інтерфейсу, часу доставки сповіщень, стійкості до деградацій і якості виявлення відхилень за динамічними індикаторами. Отримані в наступному розділі емпіричні результати стануть підставою для остаточного налаштування порогів, політик ескалацій та

параметрів масштабування, а також для уточнення рекомендацій щодо промислової експлуатації.

# 3 РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ МОНІТОРИНГУ

## 3.1 Алгоритмічне забезпечення аналізу температурних режимів

Алгоритмічне забезпечення є ключовим елементом автоматизованої системи моніторингу параметрів зберігання зерна, оскільки саме воно перетворює масиви вимірювань із датчиків на інформацію, придатну для прийняття технологічних рішень. У попередніх розділах було розглянуто архітектуру системи, структуру обладнання елеватора та організацію збору даних з термопідвісок. У цьому підрозділі зосередимося на методах аналізу температурних режимів, що реалізовані в програмному забезпеченні.

У типовій конфігурації елеваторного комплексу температура вимірюється в дискретні моменти часу у множині просторово рознесених точок: на окремих датчиках, змонтованих на термопідвісках, які розташовані у внутрішньому об’ємі силоса за певною схемою. Таким чином, на вхід алгоритмів надходить часовий ряд $T_{i,j}(t)$, де i – номер силоса, j – номер датчика в цьому силосі, t – час вимірювання. Для кожного датчика, групи датчиків (термопідвіски) та силоса в цілому необхідно оцінити:

- стабільність температурного режиму у часі;
- відхилення температури від допустимих значень для конкретної культури;
- просторовий розподіл температури та його зміни;
- орієнтовний рівень заповнення силоса зерном [22-23].

Особливістю процесу зберігання зерна є його тривалість, повільна інерційна динаміка температури в масі зерна та водночас потенційна небезпека локальних зон самозігрівання. Через це недостатньо аналізувати лише миттєві значення температури; необхідно розглядати їх зміну в часі, порівнювати з попередніми періодами, а також враховувати просторовий контекст (сусідні датчики, різні рівні по висоті силоса). Алгоритмічне забезпечення системи має забезпечити виявлення як повільних трендів (поступове зростання температури), так і відносно швидких аномальних змін, що можуть свідчити про початок псування зерна.

Для вирішення цих завдань у системі реалізовано багаторівневий підхід до аналізу даних:

1. на рівні окремого датчика обчислюються показники стабільності температури, визначаються локальні тенденції зростання або зниження, формуються індекси рівня температури;

2. на рівні термопідвіски здійснюється агрегування показників окремих датчиків, аналіз вертикального профілю температури та його зміни в часі;

3. на рівні силоса визначається загальний стан зернової маси, просторовий розподіл температури по об’єму та приблизний рівень заповнення.

Однією з базових задач є кількісна оцінка коливань температури за певний період часу. Для цього використовується аналіз часових інтервалів (наприклад, добових або тижневих вікон), у межах яких для кожного датчика обчислюються статистичні характеристики: середнє значення температури, діапазон відхилень, стандартне відхилення та коефіцієнт варіації. Останній дає змогу оцінити відносну мінливість температури незалежно від її абсолютного рівня. На основі цих характеристик формуються критерії, що дозволяють розрізняти стабільний, помірно нестабільний та явно нестабільний температурні режими.

Другою важливою задачею є класифікація стану температурного режиму за допомогою індексів рівня температури. Для практичної експлуатації системи технологам необхідно отримувати не просто числові значення в градусах Цельсія, а зрозумілі категорії, наприклад: «норма», «підвищений контроль», «небезпечний рівень». У системі використовується підхід, за якого на основі фактичної температури, культури зерна, її сорту та, за потреби, вологості формується інтегральний індекс, що відображає ступінь наближення до небезпечних режимів. Індекс може бути визначений як для окремого датчика, так і для термопідвіски або силоса в цілому, що полегшує візуалізацію стану об’єкта.

Окремий клас алгоритмів пов’язаний із визначенням рівня заповнення силоса. Ця задача важлива як з точки зору технологічного управління (планування завантаження та розвантаження), так і для коректної інтерпретації температурних полів. У найпростішому випадку рівень заповнення може визначатися за даними

спеціалізованих вимірювачів рівня; однак у пропонованій системі передбачено використання також непрямих ознак, зокрема характеру вертикального розподілу температури по термопідвісках. Зміна температурного профілю після завантаження чи часткового розвантаження силоса дозволяє оцінити положення межі «зерно–повітря» та, відповідно, висоту шару зерна.

Таким чином, алгоритмічне забезпечення аналізу температурних режимів у запропонованій системі базується на поєднанні:

1. статистичної обробки часових рядів (оцінка коливань, трендів, сезонних варіацій);
2. нормування температури відносно допустимих режимів зберігання для конкретних культур;
3. просторового аналізу розподілу температури по висоті та площі силоса;
4. інтегрального опису стану об’єкта у вигляді індексів і рівнів.

### 3.1.1 Алгоритм визначення коливань температури

Метою алгоритму визначення коливань температури є кількісна оцінка стабільності температурного режиму в зерновій масі протягом заданого інтервалу часу. Результати цього аналізу використовуються для подальшого формування індексів рівня температури, виявлення небезпечних тенденцій самозігрівання, а також для візуалізації стану окремих датчиків, термопідвісок і силоса в цілому.

Автоматизована система моніторингу реєструє значення температури $T_j(t_k))$ для кожного датчика j у дискретні моменти часу $t_k$ (наприклад, з періодом 10–15 хвилин). Для аналізу коливань використовується вибірка значень температури за певний період $\Delta T$, який у практичних умовах доцільно обирати рівним одній добі або декільком добам. Для датчика j набір вимірювань за обраний період можна подати у вигляді:

$$T_j = \{T_j(t_1), T_j(t_2), \dots, T_j(t_n)\}, \tag{3.1}$$

де n – кількість вимірювань у вікні аналізу.

Інтервал аналізу може бути: фіксованим (наприклад, календарна доба з 00:00 до 23:59) або реалізованим як ковзне вікно тривалістю $\Delta T$, що пересувається по часовому ряду з певним кроком (наприклад, одна година).

У реалізованому програмному забезпеченні використовується ковзне вікно, що дозволяє оперативно оцінювати зміну температурної стабільності без очікування завершення повної доби.

Розрахуємо статистичні показники коливань. Для кожного датчика в межах вибраного вікна аналізу обчислюються основні статистичні характеристики:

1. Середнє значення температури:

$$\overline{T_j} = \frac{1}{n}\sum_{k=1}^{n} T_j(t_k), \tag{3.2}$$

2. Мінімальне та максимальне значення:

$$T_j^{\min} = \min_{1\le k\le n} T_j(t_k),\ \mathrm{T}_j^{\max} = \max_{1\le k\le n} T_j(t_k), \tag{3.3}$$

3. Амплітуда коливань температури:

$$A_j = T_j^{\max} - T_j^{\min}, \tag{3.4}$$

4. Дисперсія та стандартне відхилення:

$$\sigma_j^2 = \frac{1}{n}\sum_{k=1}^{n}\left(T_j(t_k) - \overline{T_j}\right)^2, ][\sigma_j = \sqrt{\sigma_j^2}, \tag{3.5}$$

Амплітуда Aj відображає максимальний розкид температури в абсолютних значеннях (°C), тоді як коефіцієнт варіації Vj дозволяє оцінити відносну

нестабільність режиму незалежно від середнього рівня температури. Це особливо важливо при порівнянні датчиків, що знаходяться на різній висоті або в силосах з різними культурами, де допустимі температурні діапазони можуть істотно відрізнятися.

На основі розрахованих статистичних показників вводяться градації стабільності температурного режиму для кожного датчика. У найпростішому варіанті класифікація може базуватися на двох показниках – амплітуді Aj та коефіцієнті варіації Vj. У системі передбачено налаштовувані порогові значення, які можуть бути встановлені технологом або адміністратором на підставі практичного досвіду експлуатації елеватора.

Приклад класифікації:

- Стабільний режим:
  - $\mathrm{A_j} \leq \mathrm{A_{стаб}}$
  - $\mathrm{V_j} \leq \mathrm{V_{стаб}}$
- Помірні коливання:
  - $\mathrm{A_{стаб}} < \mathrm{A_j} \leq \mathrm{A_{кр}}$ або $\mathrm{V_{стаб}} < \mathrm{V_j} \leq \mathrm{V_{кр}}$
- Підвищені коливання (нестабільний режим):
  - $A_j > A_{\text{кр}}$ або $V_j > V_{\text{кр}}$ ,

де $A_{\text{стаб}}, V_{\text{стаб}}, A_{\text{кр}}, V_{\text{кр}}$ – порогові значення амплітуди та коефіцієнта варіації, що задаються в налаштуваннях системи. Для різних культур (пшениця, кукурудза, ячмінь тощо) та умов зберігання ці пороги можуть мати різні значення.

За результатами класифікації для кожного датчика формується категорія стану коливань, яка може відображатися у вигляді:

- числового коду (0 – стабільно, 1 – помірні коливання, 2 – підвищені коливання),
- кольорової індикації (наприклад, зелений – стабільно, жовтий – помірно, червоний – нестабільно),
- текстового опису в таблиці або звіті.

Для більш повної оцінки коливань доцільно враховувати не лише розкид значень відносно середнього, а й наявність монотонної тенденції зростання або

зниження температури. У реалізованому алгоритмі це досягається шляхом порівняння середнього значення за поточне вікно $\overline{T}_j^{(\text{пот})}$ із середнім значенням за попереднє вікно такої ж тривалості $\overline{T}_j^{(\text{поп})}$: $\Delta\overline{T}_j = \overline{T}_j^{(\text{пот})} - \overline{T}_j^{(\text{поп})}$

За величиною $\Delta\overline{T}_j$ можна додатково класифікувати характер динаміки:

- $\Delta\overline{T}_j \approx 0$ – істотного тренду не виявлено;

- $\Delta\overline{T}_j > \Delta T_{\text{пор}}$ – спостерігається стійка тенденція до підвищення температури;

- $\Delta\overline{T}_j < -\Delta T_{\text{пор}}$ – спостерігається стійка тенденція до зниження температури,

де $\Delta T_{\text{пор}}$ – порогове значення зміни середньої температури (наприклад, 0,5–1,0 °C за добу), яке задається в налаштуваннях.

Отримана інформація про тренд може використовуватися спільно з показниками коливань: наприклад, навіть при відносно невеликих миттєвих коливаннях, але при стійкому зростанні $\Delta\overline{T}_j$ система може перевести датчик у більш високий рівень контролю.

Використання результатів на вищих рівнях ієрархії

Результати роботи алгоритму на рівні окремих датчиків є основою для подальшої агрегації:

- на рівні термопідвіски – за рахунок узагальнення коливань по всіх датчиках, закріплених на даній підвісці;

- на рівні силоса – через інтеграцію інформації від усіх термопідвісок та оцінку загальної стабільності температурного поля.

### 3.1.2 Визначення індексів рівня температури

Для практичного використання автоматизованої системи моніторингу оператору й технологу недостатньо бачити лише числові значення температури в точках вимірювання. Необхідно мати компактну, інтерпретовану оцінку стану температурного режиму, яка б дозволяла швидко ідентифікувати проблемні зони, порівнювати між собою різні силоси та оцінювати динаміку змін у часі. З цією метою в системі введено поняття індексів рівня температури.

Індекс рівня температури розглядається як інтегральний показник, що відображає ступінь наближення фактичної температури у заданій точці до нормативних або критичних значень для конкретної культури зерна [23]. На відміну від «сирих» вимірювань, індекс має обмежену кількість станів і безпосередньо використовується для:

- кольорової індикації датчиків, термопідвісок і силосів на 2D- та 3D-схемах;
- формування текстових повідомлень і сповіщень;
- агрегованої оцінки стану термопідвісок і силоса в цілому.

Як вихідні дані для розрахунку індексів використовуються:

- усереднена за вікно аналізу температура в точці вимірювання
- довідкова інформація про допустимі температурні режими для культури, що зберігається в силосі (нормативний діапазон та допустимі відхилення);
- за потреби – оцінка тренду зміни температури за останні інтервали спостережень.

У базовому варіанті для кожної культури задається нормативний діапазон температур зберігання. Для пшениці, кукурудзи, ячменю тощо ці діапазони відрізняються, а також можуть коригуватися залежно від вологості зерна та тривалості зберігання. Крім того, визначаються допустимі відхилення від нормативного діапазону, які поділяються на «попереджувальний» та «критичний» рівні. Це дозволяє розділити весь діапазон можливих температур на кілька зон:

1. зона безпечної експлуатації (норма);
2. зона підвищеного контролю (помірне відхилення від нормативу);
3. зона небезпечних режимів (суттєве відхилення, що вимагає оперативного втручання).

Для кожного датчика обчислюється інтегральний індекс рівня температури як дискретна величина з обмеженою кількістю станів. У роботі доцільно використовувати чотирирівневу шкалу, наприклад:

1. рівень 0 – занижена температура (нижній рівень, умови ближче до переохолодження);
2. рівень 1 – нормальний режим зберігання;
3. рівень 2 – підвищений контроль (попереджувальний рівень);
4. рівень 3 – аварійний рівень (небезпечна температура).

Таке кодування дозволяє одночасно врахувати як відхилення в бік зниження температури (що може бути небажаним, але зазвичай менш критичним), так і відхилення в бік підвищення (які пов’язані з ризиком самозігрівання зерна).

На першому етапі для кожного датчика визначається відхилення фактичної температури від опорного значення. Як опорне значення може використовуватися центр нормативного діапазону температур для даної культури. Далі обчислюється модуль цього відхилення, який показує, наскільки далеко фактична температура відійшла від рекомендованого режиму. Порівнюючи модуль відхилення з двома порогами (для попереджувального та критичного рівнів), система відносить точку вимірювання до відповідного класу.

У разі, коли потрібна більш гнучка візуалізація (наприклад, плавні колірні градієнти на 3D-моделі силоса), додатково може обчислюватися нормований індекс ризику температури у діапазоні від 0 до 1, який відображає, яку частку від максимально допустимого відхилення становить поточна температура [24]. Значення, близькі до нуля, відповідають умовно ідеальному режиму, а значення, близькі до одиниці, – гранично допустимим або аварійним станам.

Якщо для датчика спостерігається стійкий тренд зростання середньої температури за останні інтервали спостережень, індекс рівня температури може бути підвищений на один щабель навіть у випадку, коли абсолютне значення температури ще не досягло попереджувальної або критичної межі. Це дозволяє системі реагувати не лише на вже сформовані аварійні ситуації, а й на потенційно небезпечні тенденції.

Після визначення індексів на рівні окремих датчиків виконується агрегація індексів для термопідвісок і силосів у цілому. Для термопідвіски інтегральний індекс може визначатися, наприклад, як максимальне значення індексів серед усіх датчиків, закріплених на цій підвісці. Такий підхід відповідає практичній логіці: наявність хоча б однієї «гарячої» точки робить підвісці необхідною підвищену увагу. Додатково для аналітичних задач може використовуватися середнє значення індексів по термопідвісці.

Аналогічний підхід застосовується до рівня силоса. Для нього інтегральний індекс може визначатися як максимум індексів усіх датчиків силоса, що дозволяє швидко виявити силоси з потенційно небезпечними зонами. Разом з тим для якісної оцінки стану силоса може використовуватися частка датчиків, що перебувають у кожній з категорій (норма, попереджувальний, аварійний рівні). Це дає змогу відрізнити ситуації локального перегріву від випадків, коли підвищена температура спостерігається в значній частині об’єму.

Розраховані індекси рівня температури використовуються в автоматизованій системі для:

- визначення кольорів відображення датчиків, термопідвісок і силосів на графічних схемах;
- фільтрації та сортування записів у табличних представленнях;
- генерації попереджень і аварійних сповіщень;
- формування звітів про стан температурного режиму за обрані періоди.

Таким чином, індекси рівня температури є ключовою ланкою між «сирими» даними датчиків та інформаційними представленнями, що використовуються персоналом для оперативного контролю й прийняття рішень.

### 3.1.3 Алгоритми визначення рівня заповнення силоса

У межах автоматизованої системи моніторингу рівень заповнення силоса використовується не лише для візуалізації, а й для орієнтовного розрахунку маси зерна, аналізу ефективності завантаження та формування рекомендацій щодо режимів аерації та сушіння. У відповідності до функціональних вимог система має визначати рівень зерна за даними температурних датчиків та, за наявності, аналогових датчиків рівня, а також виконувати приблизний розрахунок ваги культури в силосі.

Нехай силос має висоту H та відому геометрію (циліндричну, конічну тощо), яка задається функцією корисного об’єму від висоти заповнення $V_{силос}$. (h) Ця функція одержується з конструкторської документації та зберігається в базі даних у вигляді аналітичної залежності або табличної апроксимації. У кожному силосі встановлено M термопідвісок (кабельних підвісок), на j-тій підвісці розміщено Kj датчиків температури. Висота розташування k-го датчика на j-тій підвісці позначається як $z_{j,k}$ , де $0 \leq z_{j,k} \leq H$. Для кожного датчика у момент часу t система отримує значення температури $T_{j,k}(t)$. За наявності аналогового датчика рівня (радар, ультразвуковий або ємнісний перетворювач) додатково формується вимірюване значення геометричного рівня зерна $h_{ан(t)}$ .

Алгоритм визначення рівня спирається на те, що датчики, занурені у зернову масу, мають інші значення температури та іншу динаміку їх зміни, ніж датчики, розташовані в газовому просторі над зерном. Основна ідея алгоритму полягає в тому, що датчики, занурені у зернову масу, характеризуються іншою температурою та динамікою її зміни, ніж датчики, які знаходяться в газовому просторі над зерном. Цю відмінність можна використати для класифікації датчиків на «занурені» та «розташовані в повітрі», а далі – для оцінки висоти фронту зерна вздовж кожної підвіски.

Для кожної підвіски попередньо виділяється набір нижніх датчиків, які в будь-якому робочому режимі гарантовано знаходяться в зерні (наприклад, 2–3 нижні датчики).

Для них обчислюється усереднена температура:

$$T_j^{\text{gr}}(t) = \frac{1}{n_{\text{gr}}} \sum_{k=1}^{n_{\text{gr}}} T_{j,k}(t), \tag{3.6}$$

де $n_{gr}$ – кількість «опорних» нижніх датчиків на j-тій підвісці.

Усереднена температура повітря в газовому просторі над зерном може визначатися або за даними зовнішніх/верхніх датчиків, або за спеціальним датчиком довкілля:

$$T^{\text{air}}(t) = \frac{1}{M} \sum_{j=1}^{M} T_{j,K_j}(t), \tag{3.7}$$

Для кожного датчика обчислюється відхилення від температури повітря: $\Delta T_{j,k}(t) = T_{j,k}(t) - T^{\text{air}}(t)$.

Датчик вважається зануреним у зерно, якщо виконується умова

$$\left| \Delta T_{j,k}(t) - \Delta T_j^{\text{gr}}(t) \right| \leq \varepsilon_T, \tag{3.8}$$

де $\varepsilon_T$ – допустиме порогове відхилення, яке налаштовується в модулі. Альтернативно може застосовуватись градієнтний критерій:

$$G_{j,k}(t) = \frac{T_{j,k+1}(t) - T_{j,k}(t)}{z_{j,k+1} - z_{j,k}}, \tag{3.9}$$

і точка переходу «зерно–повітря» визначається як шар, де величина $\left| G_{j,k}(t) \right|$ різко змінюється.

Для кожної підвіски визначається індекс останнього датчика, що належить зерновій масі: $k_j^*(t) = \max\{k \mid T_{j,k}(t)$ класифіковано як «зерно»$\}$.

Відповідна локальна оцінка рівня заповнення вздовж j-тої підвіски: $h_j(t) = z_{j,k_j^*(t)}$. Щоб уникнути «стрибків» рівня при випадкових шумових коливаннях температури, для $h_j(t)$ застосовується часовий фільтр ковзного середнього або експоненційне згладжування:

$$\bar{h_j}(t) = \alpha\, h_j(t) + (1-\alpha)\, \bar{h_j}(t-\Delta t), \tag{3.10}$$

де $0 < \alpha \leq 1$– параметр згладжування.

Якщо в силосі додатково встановлено аналогові датчики рівня, їх покази використовуються як незалежне джерело інформації про геометричну висоту зерна. У найпростішому випадку, за наявності коректного сигналу, рівень визначається за аналоговим датчиком:

$$h_{\text{ан}}(t) = f_{\text{кал}}\big(U(t)\big), \tag{3.11}$$

де U(t) – виміряна напруга/струм, $f_{\text{кал}}$ – калібрувальна характеристика перетворювача.

У разі паралельного використання двох джерел (температурні підвіски та аналоговий датчик) доцільно застосувати зважене об’єднання: μ, де $h_{\text{темп}}(t)$ – оцінений за температурою рівень, а $\beta \in [0,1]$ – коефіцієнт довіри до аналогового каналу, який задається під час налаштування системи.

Відповідно до вимог до інтерфейсу система має не лише відображати рівень заповнення, а й орієнтовну вагу культури в силосі в режимах 2D та 3D візуалізації.

На основі геометричної моделі силоса визначається функція корисного об’єму від висоти:

$$V_{\text{гр}}(t) = V_{\text{силос}}\big(h_{\text{fill}}(t)\big). \tag{3.12}$$

Для металевого циліндричного силоса без урахування конічного днища в простому випадку:

$$V_{\text{силос}}(h) = \pi R^2 h, \tag{3.13}$$

де R – радіус силоса. Для силосів із конічним дном або складнішою геометрією функція $V_{\text{силос}}(h)$ задається як таблична залежність та інтерполюється.

1. Оцінка маси зерна.

Маса зерна в силосі визначається через об’єм та насипну густину:

$$m(t) = \rho_{\text{нас}} \cdot V_{\text{гр}}(t), \tag{3.14}$$

де $\rho_{\text{нас}}$ – насипна густина конкретної зернової культури та її класу, що зберігається в силосі.

Опишемо алгоритми візуалізації рівня заповнення в 2D та 3D. Отримане значення $h_{\text{fill}}(t)$ використовується на рівні візуалізації даних для побудови 2D та 3D відображення силоса. У 2D-режимі силос подається у вигляді прямокутника або кола з частковим зафарбуванням згідно з відносною висотою заповнення:

$$k_{\text{fill}}(t) = \frac{h_{\text{fill}}(t)}{H}. \tag{3.15}$$

Стільки ж відсотків геометричної фігури зафарбовується кольором зерна, при цьому поверх накладається кольорова індикація температурних полів.

У 3D-режимі рівень заповнення відображається як тривимірне «тіло» зерна всередині геометрії силоса. Для побудови поверхні насипу проводиться інтерполяція значень $\overline{h_J}(t)$ між підвісками, після чого візуалізатор Three.js формує полігональну модель поверхні зерна з урахуванням температурного поля по висоті та радіусу силоса.

## 3.2 Модуль зберігання та попередньої обробки даних

В цьому розділі розглянемо, як саме виміряні значення потрапляють до бази даних і стають доступними для подальшої обробки та відображення в інтерфейсі користувача. Логічний ланцюжок починається з брокера повідомлень RabbitMQ,

який виступає проміжною ланкою між контролерами термометрії та серверною частиною автоматизованої системи моніторингу. Контролери або локальний агент формують повідомлення з виміряними значеннями датчиків і публікують їх у відповідні черги RabbitMQ за протоколом AMQP (Advanced Message Queuing Protocol). Повідомлення містить ідентифікатор датчика, час вимірювання, значення температури й допоміжні службові поля, необхідні для трасування та ідемпотентної обробки.

Далі ці повідомлення асинхронно споживаються серверною частиною системи. На стороні бекенду працює один або кілька сервісів-консьюмерів, які підписані на черги RabbitMQ та отримують дані по AMQP у міру їх надходження. На цьому етапі виконується перший рівень валідації: перевіряється цілісність структури повідомлення, коректність форматів полів, наявність відповідних записів про датчик і силос у базі даних, а також допустимість самого виміряного значення відносно діапазону роботи датчика. У разі виявлення помилок спрацьовує механізм реєстрації й повторної обробки, а для критичних відхилень формується службове сповіщення.

Після успішної валідації дані перетворюються до внутрішньої моделі домену й зберігаються в реляційній базі даних PostgreSQL. Для цього бекенд виконує транзакцію, яка додає нові записи до таблиці часових рядів вимірювань (SensorData) та, за потреби, оновлює агреговані показники або пов’язані сутності (наприклад, поточний максимальний рівень температури в силосі). Завдяки транзакційному механізму забезпечується атомарність операцій, а використання індексів за ідентифікатором датчика та часовою міткою дозволяє надалі ефективно виконувати аналітичні запити.

Останньою ланкою цього ланцюга є кешування даних під час звернень клієнтських застосунків. Коли веб-клієнт або мобільний додаток запитує, наприклад, актуальні температурні профілі силоса чи історичний графік за останні 24 години, серверний модуль спочатку звертається до кешу (локального в пам’яті або розподіленого, реалізованого засобами на кшталт Redis). Якщо необхідний набір даних уже присутній у кеші й не втратив актуальності, він негайно

повертається клієнту без виконання запиту до PostgreSQL. У протилежному випадку сервіс формує запит до бази даних, виконує необхідну попередню обробку (агрегування, відсікання шуму, обчислення похідних показників), зберігає отриманий результат у кеш із заданим часом життя і лише після цього надсилає відповідь клієнту. Така схема «RabbitMQ → бекенд з валідацією → PostgreSQL → кеш на запиті клієнта» забезпечує поєднання надійного збереження даних з високою швидкодією інтерфейсів і дає змогу масштабувати систему при збільшенні кількості датчиків та обсягу історичних вимірювань.

### 3.2.1 Структура бази даних

Структура бази даних модуля зберігання та попередньої обробки даних відображає логіку роботи автоматизованої системи моніторингу та побудована як ієрархія «користувач – підприємство – силос – підвіска – датчик – часовий ряд вимірювань» з додатковими сутностями для керування обладнанням, налаштуваннями та сповіщеннями і відповідає концептуальній схемі сутностей, зображеній на рисунку 3.1 . На верхньому рівні зберігається інформація про профілі користувачів. Таблиця Profile містить унікальний ідентифікатор, ім’я користувача, електронну адресу та номер телефону. Вона використовується для аутентифікації й комунікації, а також дозволяє відокремити облікові дані від прикладної доменної моделі. З профілем пов’язується сутність User, яка представляє конкретного користувача системи та містить посилання на профіль і на підприємство, де цей користувач працює. Таким чином реалізується прив’язка облікового запису до певного елеватора та забезпечується можливість обмеження доступу до даних у межах одного підприємства.

| id | name | display_name | bin_type | enabled_calculation | enabled_ui | x_position | y_position | radius | container_width | container_height | annotations | created_date | created_by | last_modified_date | last_modified_by | deleted | grain_type_id |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Силос №1 | Силос №1 | METAL | true | true | 100 | 100 | 1 | 1920 | 1080 | | | | | | false | 7 |
| 2 | Силос №2 | Силос №2 | METAL | true | true | 500 | 100 | 1 | 1920 | 1080 | | | | | | false | 1 |
| 3 | Силос №3 | Силос №3 | METAL | true | true | 900 | 100 | 1 | 1920 | 1080 | | | | | | false | 2 |
| 4 | Силос №4 | Силос №4 | METAL | true | true | 1300 | 100 | 1 | 1920 | 1080 | | | | | | false | 1 |
| 5 | Силос №5 | Силос №5 | METAL | true | true | 100 | 500 | 1 | 1920 | 1080 | | | | | | false | 3 |
| 6 | Силос №6 | Силос №6 | METAL | true | true | 500 | 500 | 1 | 1920 | 1080 | | | | | | false | 5 |
| 7 | Силос №7 | Силос №7 | CONCRETE | true | true | 1000 | 500 | 1 | 1920 | 1080 | | | | | | false | 4 |
| 8 | Силос №8 | Силос №8 | METAL | true | true | 1500 | 500 | 1 | 1920 | 1080 | | | | | | false | 7 |
| 9 | Силос №9 | Силос №9 | METAL | true | true | 100 | 900 | 1 | 1920 | 1080 | | | | | | false | 8 |
| 10 | Силос №10 | Силос №10 | CONCRETE | true | true | 700 | 900 | 1 | 1920 | 1080 | | | | | | false | 6 |
| 11 | Силос №11 | Силос №11 | CONCRETE | true | true | 1300 | 900 | 1 | 1920 | 1080 | | | | | | false | 9 |

Рисунок 3.1 – Фрагмент таблиці конфігурації силосів у базі даних

Поруч із цим зберігаються індивідуальні налаштування користувачів, що представлені таблицею Config. Для кожного запису вказується ідентифікатор конфігурації, набір параметрів (наприклад, у форматі JSON) та посилання на користувача, якому належить ця конфігурація. Це дозволяє зберігати особисті вподобання щодо відображення даних, порогових значень тривог, мови інтерфейсу тощо, не змінюючи основну логіку предметної області. Саме користувачі, до яких прив’язані профілі й конфігурації, працюють у контексті певних підприємств, що моделюються таблицею Enterprise і показано на рисунку 3.2 з полями ідентифікатора, назви та місця розташування. Одне підприємство може містити множину силосів, тому між Enterprise та Silage реалізується зв’язок «один-до-багатьох».

Загалом запропонована структура бази даних є логічно узгодженою та масштабованою, що дозволяє забезпечити зручне зберігання, обробку й аналіз даних моніторингу. Ієрархічна організація сутностей спрощує навігацію між об’єктами системи та підтримує розмежування доступу відповідно до ролей користувачів і підприємств. Обрана модель також створює передумови для подальшого розширення функціональності системи, зокрема шляхом додавання нових типів обладнання, датчиків або механізмів аналітики, без необхідності суттєвої перебудови існуючої структури.

```xml
<?xml version="1.0" encoding="utf-8"?>
<databaseChangeLog
    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
    xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-latest.xsd
                        http://www.liquibase.org/xml/ns/dbchangelog-ext http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-ext.xsd">

    <!--
        Added the entity Enterprise.
    -->
    <changeSet id="20241119150023-1" author="IVP">
        <createTable tableName="enterprise">
            <column name="id" type="bigint">
                <constraints primaryKey="true" nullable="false"/>
            </column>
            <column name="cloud_id" type="${uuidType}">
                <constraints nullable="true" />
            </column>
            <column name="name" type="varchar(255)">
                <constraints nullable="true" />
            </column>
            <column name="short_name" type="varchar(255)">
                <constraints nullable="true" />
            </column>
            <column name="address" type="varchar(255)">
                <constraints nullable="true" />
            </column>
        </createTable>
        <dropDefaultValue tableName="enterprise" columnName="created_date" columnDataType="${datetimeType}"/>
        <dropDefaultValue tableName="enterprise" columnName="last_modified_date" columnDataType="${datetimeType}"/>
    </changeSet>

    <changeSet id="20241119150023-1-data" author="IVP" context="faker">
        <loadData
                  file="config/liquibase/fake-data/enterprise.csv"
                  separator=";"
                  tableName="enterprise"
                  usePreparedStatements="true">
            <column name="id" type="numeric"/>
            <column name="cloud_id" type="${uuidType}"/>
            <column name="name" type="string"/>
            <column name="short_name" type="string"/>
            <column name="address" type="string"/>
            <column name="contact" type="string"/>
            <column name="annotations" type="string"/>
            <column name="registered" type="boolean"/>
            <column name="created_date" type="date"/>
            <column name="created_by" type="string"/>
            <column name="last_modified_date" type="date"/>
            <column name="last_modified_by" type="string"/>
            <column name="deleted" type="boolean"/>
        </loadData>
    </changeSet>
</databaseChangeLog>
```

Рисунок 3.2 – Фрагмент XML-конфігурації Liquibase для створення таблиці

Сутність Silage описує окремий силос (зерновий бункер) на підприємстві й містить ідентифікатор, посилання на підприємство, а також назву й локалізаційні атрибути. Саме з цією таблицею пов’язується технічне обладнання: контролери та система сповіщень. Таблиця Controller зберігає інформацію про контролери термометрії, які обслуговують силосу: ідентифікатор пристрою, посилання на силос, тип контролера та його поточний стан. Для фіксації подій, пов’язаних із перевищенням порогів температури або зі збоями в роботі обладнання, використовується таблиця Alert. У ній зберігаються ідентифікатор сповіщення, посилання на силос, текстове повідомлення та час виникнення події. Такий підхід

дозволяє будувати журнал подій і оперативно переходити від тривоги до відповідного силоса та його датчиків.

Нижче по ієрархії розташований рівень геометричної структури силоса, що описує підвіски й датчики. Таблиця StripCircle моделює кільце підвісок всередині силоса, яке містить посилання на Silage і число підвісок у цьому кільці. Далі йде таблиця Strip, що відповідає окремій термопідвісці; вона пов’язана з конкретним кільцем і зберігає кількість датчиків, змонтованих на підвісці. Така двоступенева модель дозволяє відтворювати реальне просторове розташування підвісок у силосі і використовувати його під час 2D/3D-візуалізації.

Кожен фізичний датчик описується сутністю Sensor. У ній задається унікальний ідентифікатор, посилання на підвіску, тип датчика (наприклад, температурний або комбінований) і робочий діапазон у вигляді мінімального та максимального значень. Окремо зберігаються конфігураційні та сервісні дані для датчиків у таблиці SensorConfig. Вона містить посилання на датчик, структуровані конфігураційні дані (параметри калібрування, індивідуальні пороги тривог, коефіцієнти фільтрації) і, за потреби, посилання на конкретне вимірювання, з яким пов’язана конфігурація. Такий поділ між Sensor і SensorConfig робить модель гнучкою: фізичні властивості датчика й алгоритмічні налаштування можуть змінюватися незалежно.

Нарешті, найдинамічніший шар структури становлять часові ряди вимірювань, які зберігаються в таблиці SensorData. Кожний запис містить ідентифікатор, посилання на датчик, часову мітку й виміряне значення температури. У перспективі структура може бути розширена додатковими полями для інших параметрів (вологості, вмісту $CO_2$ тощо), однак уже поточна модель дає змогу формувати детальні часові ряди для кожного датчика. Зв’язок SensorData з Sensor, а через нього – з Strip, StripCircle, Silage та Enterprise дозволяє будувати складні запити: від вибірки всіх вимірювань для одного датчика до отримання агрегованих показників по всіх силосах підприємства.

Таким чином, структура бази даних поєднує в собі рівень користувачів і їхніх налаштувань, рівень підприємств і силосів, рівень контролерів і сповіщень, а також

детальну модель підвісок, датчиків і часових рядів вимірювань. Логічні зв’язки між сутностями реалізуються через зовнішні ключі, а ієрархічна організація дозволяє одночасно підтримувати зручне адміністрування, повноцінну візуалізацію стану зерна та ефективне виконання аналітичних запитів у модулі попередньої обробки даних.

### 3.2.2 Кешування та оптимізація доступу

На відміну від попередніх розділів, де розглядалися алгоритми аналізу, у цьому підрозділі увага зосереджується на тому, як саме виміряні значення потрапляють до бази даних і стають доступними для подальшої обробки та відображення в інтерфейсі користувача. Логічний ланцюжок починається з брокера повідомлень RabbitMQ, який виступає проміжною ланкою між контролерами термометрії та серверною частиною автоматизованої системи моніторингу. Контролери або локальний агент формують повідомлення з виміряними значеннями датчиків і публікують їх у відповідні черги RabbitMQ за протоколом AMQP (Advanced Message Queuing Protocol). Повідомлення містить ідентифікатор датчика, час вимірювання, значення температури й допоміжні службові поля, необхідні для трасування та ідемпотентної обробки.

Далі ці повідомлення асинхронно споживаються серверною частиною системи. На стороні бекенду працює один або кілька сервісів-консьюмерів, які підписані на черги RabbitMQ та отримують дані по AMQP у міру їх надходження [25]. На цьому етапі виконується перший рівень валідації: перевіряється цілісність структури повідомлення, коректність форматів полів, наявність відповідних записів про датчик і силос у базі даних, а також допустимість самого виміряного значення відносно діапазону роботи датчика. У разі виявлення помилок спрацьовує механізм реєстрації й повторної обробки, а для критичних відхилень формується службове сповіщення.

Після успішної валідації дані перетворюються до внутрішньої моделі домену й зберігаються в реляційній базі даних PostgreSQL. Для цього бекенд виконує транзакцію, яка додає нові записи до таблиці часових рядів вимірювань (SensorData) та, за потреби, оновлює агреговані показники або пов’язані сутності (наприклад, поточний максимальний рівень температури в силосі). Завдяки транзакційному механізму забезпечується атомарність операцій, а використання індексів за ідентифікатором датчика та часовою міткою дозволяє надалі ефективно виконувати аналітичні запити.

Останньою ланкою цього ланцюга є кешування даних під час звернень клієнтських застосунків. Коли веб-клієнт або мобільний додаток запитує, наприклад, актуальні температурні профілі силоса чи історичний графік за останні 24 години, серверний модуль спочатку звертається до кешу (локального в пам’яті або розподіленого, реалізованого засобами на кшталт Redis). Якщо необхідний набір даних уже присутній у кеші й не втратив актуальності, він негайно повертається клієнту без виконання запиту до PostgreSQL. У протилежному випадку сервіс формує запит до бази даних, виконує необхідну попередню обробку (агрегування, відсікання шуму, обчислення похідних показників), зберігає отриманий результат у кеш із заданим часом життя і лише після цього надсилає відповідь клієнту. Така схема «RabbitMQ → бекенд з валідацією → PostgreSQL → кеш на запиті клієнта» забезпечує поєднання надійного збереження даних з високою швидкодією інтерфейсів і дає змогу масштабувати систему при збільшенні кількості датчиків та обсягу історичних вимірювань.

### 3.3 Розробка інтерфейсу користувача

Розпочнемо з побудови інтерфейсу користувача, який забезпечує наочне подання результатів цих обчислень, зручну взаємодію з даними та підтримку типових сценаріїв роботи операторів, технологів і адміністраторів. Саме модуль інтерфейсу пов’язує між собою серверні сервіси, модуль обробки даних та

кінцевого користувача, перетворюючи числові часові ряди й конфігураційні параметри на інтерактивні дашборди, графіки та форми налаштувань.

З технічної точки зору клієнтська частина реалізована як веб-застосунок на основі сучасного JavaScript-фреймворку з використанням компонентного підходу. Це дає змогу розділити інтерфейс на набір незалежних компонентів (віджети силоса, графіки температури, таблиці показників, панелі фільтрів), які перевикористовуються в різних частинах системи та спрощують супровід. Обмін даними з сервером виконується через REST/GraphQL-API та WebSocket-канали для отримання актуальних значень температури й тривог у режимі, наближеному до реального часу. Для візуалізації часових рядів і індексів використано спеціалізовані бібліотеки діаграм, а тривимірне представлення силоса реалізується із залученням засобів WebGL, а саме Three.js, що забезпечує узгодженість з алгоритмами 3D-оцінки рівня заповнення.

Архітектура інтерфейсу відображає логіку предметної області. На верхньому рівні передбачено окремі представлення для «елеватора» та для окремого силоса, де користувач може переходити від агрегованого огляду підприємства до детальної інформації по обраній ємності. Інформаційні панелі формуються динамічно на основі конфігурацій, що зберігаються в базі даних: користувач може змінювати склад і розташування віджетів, налаштовувати порогові значення, обирати періоди для аналізу та зберігати власні варіанти дашбордів. Окремі компоненти інтерфейсу відповідають за редагування індивідуальних температурних налаштувань для силоса, підвіски чи датчика, дозволяючи тонко адаптувати алгоритми індексів і сповіщень під конкретну культуру та технологічний режим.

Важливою особливістю є підтримка рольової моделі доступу. Шари інтерфейсу для адміністратора, менеджера даних та звичайного користувача відрізняються набором доступних функцій та елементів управління: адміністратор працює з системними конфігураціями й правами доступу, менеджер даних – з підприємствами, силосами та шаблонами дашбордів, оператор – з прикладними сценаріями моніторингу й реакції на тривоги. Усі ці режими реалізовано в межах єдиного веб-застосунку на основі спільної бібліотеки компонентів, при цьому

доступ до окремих елементів визначається конфігурацією прав, що надходить з серверної частини. У наступних підрозділах детально розглянуто інтерфейс рівня «Елеватор», представлення окремого силоса, інструмент інтерактивного редагування дашборду та модуль індивідуальних налаштувань температур, а також показано, як вони взаємодіють із сервісами бекенду та модулем зберігання даних.

### 3.3.1 Інтерфейс рівня «Елеватор»

Інтерфейс рівня «Елеватор» програмно реалізований як веб-клієнт на основі фреймворку React з використанням TypeScript та каскадних таблиць стилів SCSS. Візуалізація карти силосів винесена в окремий функціональний компонент GrainBinMap, що імпортується до модуля термометрії й відповідає за побудову оглядової панелі, зображеної на рисунку 3.3.


Рисунок 3.3 – Інтерфейс рівня “Елеватор” (мапа силосів)

Компонент GrainBinMap (рис. 3.4) отримує з батьківського контейнера параметри «еталонного» полотна (originalContainerWidth, originalContainerHeight)

та структуру даних sections: IGrainBinSections. Ця структура містить три основні підрозділи:

- grainBins – масив силосів з геометричними (тип, радіус, координати, ширина, висота) та технологічними атрибутами;
- elementGrainBins – масив допоміжних елементів (умовні позначення дверей, драбин, входів тощо), типи яких описані переліком ElementGrainBinType;
- thermometryLegends – опис легенди температурних статусів і умовних позначень, яка відображається під картою силосів.

```
const GrainBinMap = ({ originalContainerWidth, originalContainerHeight, sections }: GrainBinMapProps) => {
  const { theme } = useContext(ThemeContext);
  const containerRef = useRef<HTMLDivElement>(null);
  const { containerWidth, containerHeight } = useResponsiveContainer({ containerRef });

  const [isLoading, setIsLoading] = useState<boolean>(true);

  const renderGrainBins = useMemo(() => {
    if (!sections?.grainBins?.length) return null;

    return sections.grainBins.map(grainBin => {
      const { id, grainBinType, xPosition, yPosition, width, height, radius } = grainBin;
      const scaledX = scaleValue(xPosition, originalContainerWidth, containerWidth);
      const scaledY = scaleValue(yPosition, originalContainerHeight, containerHeight);

      if (grainBinType === GrainBinType.CIRCLE) {
        const scaledRadius = scaleValue(radius || 0, originalContainerWidth, containerWidth);

        return <CircleBinType grainBin={grainBin} scaledRadius={scaledRadius} scaledX={scaledX} scaledY={scaledY} />;
      }

      if (grainBinType === GrainBinType.SQUARE || grainBinType === GrainBinType.RECTANGLE) {
        const scaledWidth = scaleValue(width || 0, originalContainerWidth, containerWidth);
        const scaledHeight =
          grainBinType === GrainBinType.SQUARE ? scaledWidth : scaleValue(height || 0, originalContainerHeight, containerHeight);

        return (
          <SquareBinType
            grainBin={grainBin}
            scaledWidth={scaledWidth}
            scaledHeight={scaledHeight}
            scaledX={scaledX}
            scaledY={scaledY}
          />
        );
      }

      return null;
    });
  }, [sections.grainBins, containerWidth, containerHeight]);
```

Рисунок 3.4 – Компонент GrainBinMap

Для відтворення різних типів ємностей використовуються спеціалізовані підкомпоненти: CircleBinType (круглі металеві силоси) та SquareBinType

(прямокутні/квадратні силокорпуси, склади підлогового зберігання). Тип силоса задається переліком GrainBinType, що дозволяє додавати нові геометричні представлення без зміни базового алгоритму побудови карти.

Адаптивність інтерфейсу забезпечується за рахунок хука useResponsiveContainer. Компонент створює посилання containerRef на DOM-вузол, у якому відображається карта силосів, і передає його в useResponsiveContainer. Хук повертає фактичні розміри області візуалізації (containerWidth, containerHeight), які можуть змінюватися при ресайзі вікна браузера або переході між різними роздільними здатностями дисплея.

Масштабування координат та розмірів виконується за допомогою допоміжної функції scaleValue. Для кожного силоса з масиву sections.grainBins використовується такий підхід:

```
const scaledX = scaleValue(xPosition, originalContainerWidth, containerWidth);
const scaledY = scaleValue(yPosition, originalContainerHeight, containerHeight);
```

Аналогічно масштабуються радіус, ширина та висота. Таким чином, усі координати спочатку задаються в системі відліку «еталонного» полотна макета з Figma, а під час рендерингу лінійно перераховуються в систему координат поточного контейнера. Це дозволяє один раз описати схему розміщення силосів для конкретного елеватора і коректно відображати її на будь-якому екрані.

Для оптимізації продуктивності побудова DOM-дерева винесена в мемоізовані обчислення. Масив силосів обробляється всередині виразу useMemo (renderGrainBins), де формується набір компонентів CircleBinType / SquareBinType з уже масштабованими координатами. Аналогічно, у другому виразі useMemo (renderElementGrainBins) готуються компоненти ElementGrainBin, що відповідають за відображення піктограм (двері, драбина тощо) на карті. Перерахунок відбувається лише тоді, коли змінюються вхідні дані (sections) або фактичні розміри контейнера, що зменшує кількість зайвих перерендерів.

Стан завантаження даних керується локальним стейтом isLoading, оголошеним через хук useState<boolean>. Поки дані про силоси не готові до відображення, у центральній області показується компонент Loader, що візуально

інформує оператора про триваючу ініціалізацію інтерфейсу. Після завершення завантаження прапорець isLoading встановлюється в false, і замість індикатора рендериться карта силосів.

Для підтримки узгодженого оформлення в інтерфейсі використовується контекст теми ThemeContext. Поточна тема (theme) передається, зокрема, в компонент ElementGrainBin, який підбирає кольорові схеми та стилі піктограм відповідно до активного варіанту оформлення (світла/темна тема). Глобальні стилі компонента карти силосів зосереджені в файлі style.scss, де задаються правила позиціонування елементів, оформлення фонів, шрифтів та адаптивні медіа-запити.

У нижній частині компонента умовно відображається Legend, який отримує дані з sections.thermometryLegends і візуально відтворює легенду для температурних статусів (аварійна межа, попереджувальна межа, норма, нижче норми, відсутні дані) та службових позначок. Це дозволяє безпосередньо пов’язати програмну логіку класифікації станів з їх графічним представленням на карті.
Завдяки такій архітектурі інтерфейс рівня «Елеватор» є модульним, легко розширюваним (за рахунок нових типів силосів та елементів), стійким до змін розмірів екрана та готовим до подальшої інтеграції з деталізованими представленнями рівня «Силос» і модулями аналітики.

### 3.3.2 Інтерфейс рівня «Силос»

Інтерфейс рівня «Силос» реалізовано як набір спеціалізованих React-компонентів, які відображають детальну структуру однієї ємності, температурний профіль по висоті та в поперечному перерізі, а також дозволяють налаштовувати порогові значення для окремих датчиків і підвісок. Відповідні варіанти екранів показано на рисунках 3.5 та 3.7: у першому випадку використовується псевдо-3D-зображення силоса з вертикальними підвісками, у другому – «вид зверху» з радіальним розташуванням датчиків всередині силоса.

Загальний контейнер «картки силоса» поділено на три логічні області. Ліва частина містить графічне подання силоса (3D або 2D), центральна/нижня – графік зміни показників датчиків у часі, права – панель параметрів і таблицю станів окремих датчиків. Над правою панеллю розташований перемикач між режимами «Мапа силосів» та «Картка силоса», який повертає користувача на рівень «Елеватор».

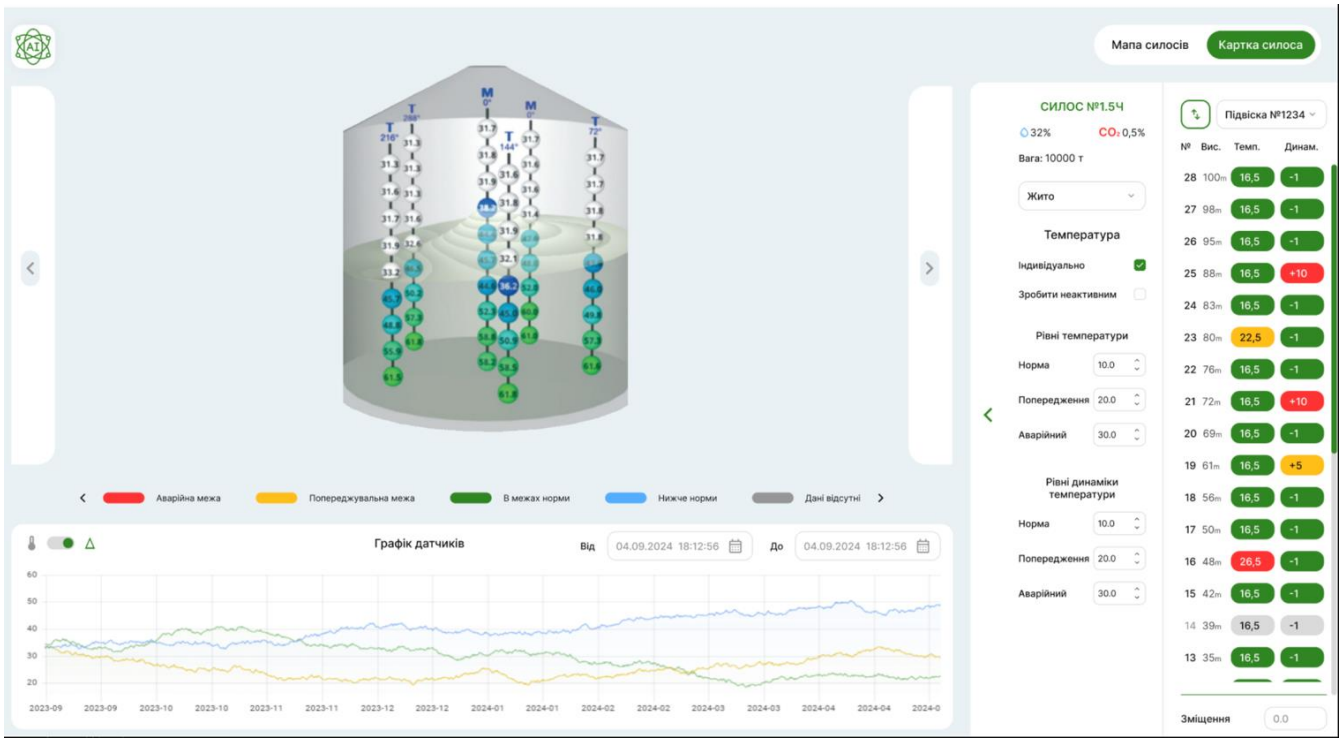


Рисунок 3.5 – Інтерфейс з тривимірною візуалізацією датчиків та детальною панеллю параметрів

Графічний блок працює в двох режимах відображення:

- Вертикальний профіль – зображення силоса з натягнутими підвісками (рисунок 3.5). На кожній підвісці виводяться значення температури на різних висотах, а колір маркерів відповідає температурному статусу (норма, попередження, аварія, нижче норми, відсутні дані). На рисунку 3.6 наведено тривимірне зображення окремої термопідвіски, де колір та інтенсивність світіння відповідають переходу датчика в аварійний температурний режим згідно з прийнятою кольоровою шкалою. У технічній реалізації цей режим побудовано як

композицію фонового зображення силоса та набору позиційованих елементів-маркерів. Кожен маркер має відносні координати (відсоток від ширини/висоти силоса), які при рендерингу трансформуються в конкретні піксельні значення на основі фактичного розміру контейнера. Такий підхід дозволяє без змін логіки масштабу виділяти силос на різних екранах і роздільних здатностях.

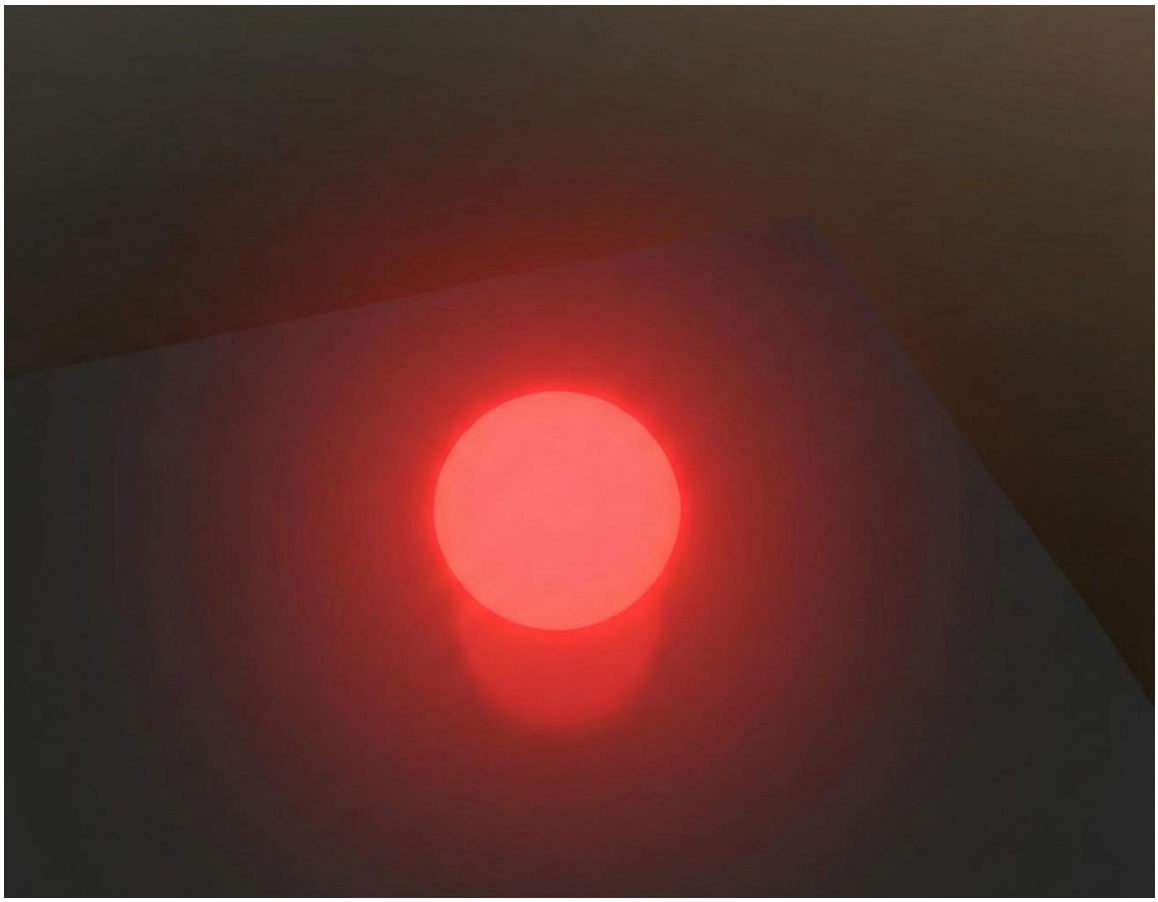

Рисунок 3.6 – Тривимірне зображення окремої термопідвіски

- Вид зверху (радіальна схема) – кругова діаграма внутрішнього простору силоса (рисунок 3.7). Датчики, розташовані на підвісках, проектуються на горизонтальну площину та відображаються як маркери всередині кількох концентричних кіл. Для побудови цієї схеми використовується спеціалізований компонент SliderItem, що рендерить SVG-графіку.

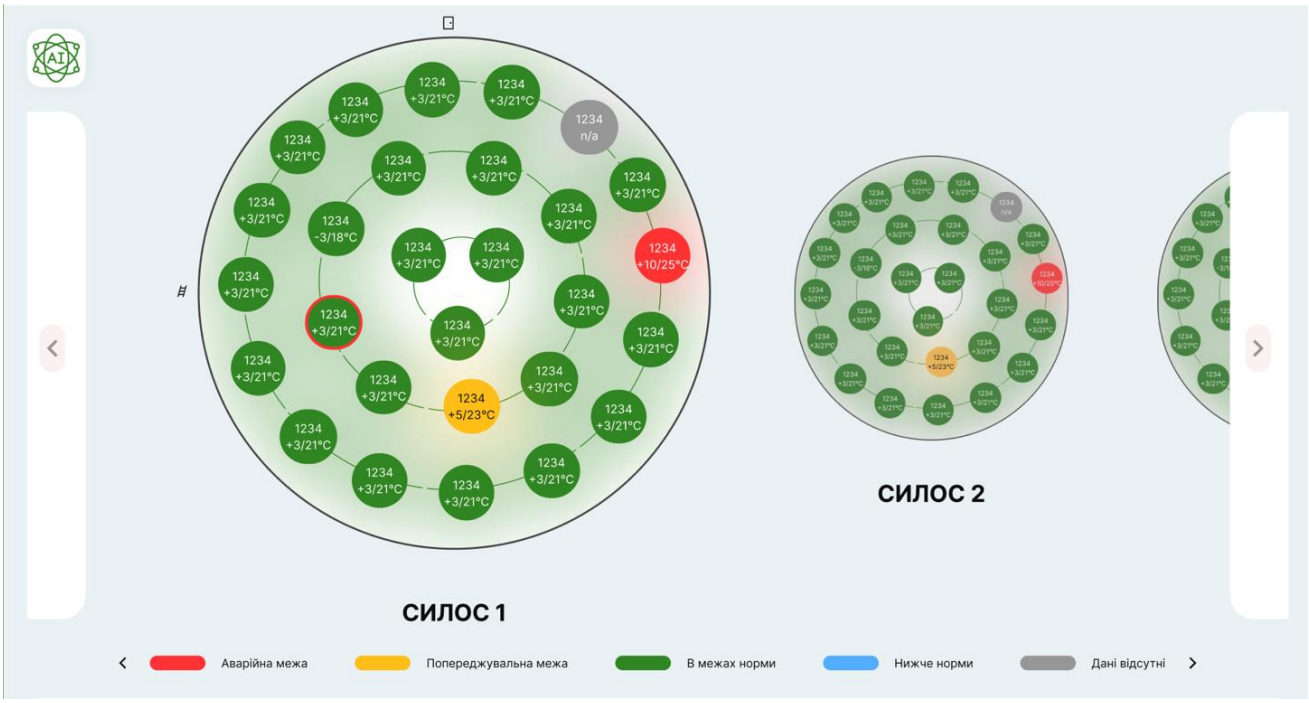


Рисунок 3.7 – Інтерфейс у 2D-режимі з відображенням температурних датчиків по перерізу

Компонент SliderItem отримує параметри розміру контейнера (containerWidth, containerHeight) та масив об’єктів thermalSuspension, кожен з яких описує одну термопідвіску чи групу датчиків. Для кожного об’єкта зберігаються, зокрема, такі поля: ідентифікатор id, логічний прапорець enabledUI, положення по радіусу (position), полярний кут theta, колір color, текстова назва displayName, поточна температура currentTemperature та добова зміна deltaTemperature.

Усередині компонента формується система концентричних кіл із заздалегідь визначеними радіусами:

```
const baseRadius = Math.min(centerX, centerY) / 4;
const radii = {
  0: baseRadius,
  1: baseRadius * 2,
  2: baseRadius * 3,
  external: baseRadius * 4,
};
```

Зовнішнє коло (external) використовується як рамка силоса: воно заповнюється напівпрозорим фоном і отримує товсту обводку, яка підсвічується, якщо силос є активним на слайдері. Внутрішні кола позначають різні рівні розташування датчиків ближче/далі від центру.

Для обчислення координат маркерів використовується допоміжна функція calculateCoordinates, яка переводить полярні координати (радіус r та кут theta у градусах) у декартову систему:

```
const thetaRadians = (theta * Math.PI) / 180;
const x = x0 + r * Math.cos(thetaRadians);
const y = y0 + r * Math.sin(thetaRadians);
```

Ця функція викликається для кожного об’єкта thermalSuspension, причому радіус r обирається відповідно до рівня position (0, 1 або 2) чи, за замовчуванням, зовнішнього кола. Таким чином датчики на різних підвісках автоматично розкладаються по концентричних «кільцях» та різних азимутальних кутах, що відповідає реальному розміщенню підвісок у силосі.

Інтерфейс рівня «Силос» активно використовує події миші для виділення елементів та синхронізації з правою панеллю параметрів. У компоненті SliderItem реалізовано два сценарії взаємодії:

- *Вибір конкретної підвіски* – при натисканні на маркер (<circle>) генерується подія onClick, яка перехоплюється у групі <g>. Обробник викликає функцію setActiveSuspension(suspension.id), що змінює поточний активний стан у батьківському контейнері. Для активної підвіски додається контрастна обводка (чорний контур) навколо маркера, а в правій панелі автоматично прокручується таблиця до відповідного рядка та відкриваються детальні параметри вибраної підвіски.

- *Скидання вибору* – клік по вільній області SVG обробляється функцією handleSvgClick. Якщо подія не стосується кола типу suspension (перевірка через e.target.dataset.type), активний стан скидається (setActiveSuspension(null)), і система повертається до «загального» режиму, коли на правій панелі показуються агреговані пороги для силоса загалом.

Окрім реакції на клацання, SVG-графіка використовує фільтри feDropShadow для кожного маркера, формуючи візуально виразні «підсвічені» точки з глибиною. Ідентифікатор фільтра параметризується ідентифікатором підвіски (shadow-${suspension.id}), а колір тіні відповідає температурному статусу, що дозволяє ще інтенсивніше виділити датчики з аномальними значеннями.

Активний стан силоса (isActive) додатково використовується для підсвічування зовнішнього кола на радіальній схемі, завдяки чому користувач може швидко зорієнтуватися, який саме силос у слайдері зараз прив’язаний до правої панелі налаштувань.

Права частина інтерфейсу рівня «Силос» представлена у вигляді гнучкого компонента форми з таблицею, що відображає список усіх датчиків конкретної підвіски на рисунку 3.8. Для кожного датчика виводяться: номер, висота над дном силоса, поточна температура, добова динаміка, а також кольоровий індикатор стану. Нижче розташовані поля для налаштування порогів:

- Рівні температури» – значення для станів *норма*, *попередження*, *аварійний*;
- «Рівні динаміки температури» – аналогічні пороги для добової зміни температури;
- Опція «Індивідуально», яка переводить силос у режим використання індивідуальних порогів (відмінних від загальноелеваторних значень).

Усі елементи форми реалізовані як контрольовані React-компоненти: при зміні значень події onChange оновлюють локальний або глобальний стан, а також ініціюють відправку змін до серверної частини (API модуля термометрії). Після збереження налаштувань, обчислені статуси датчиків оновлюються як у правій таблиці, так і в графічних представленнях (3D-вид і радіальна схема), що забезпечує узгодженість усіх представлень даних.

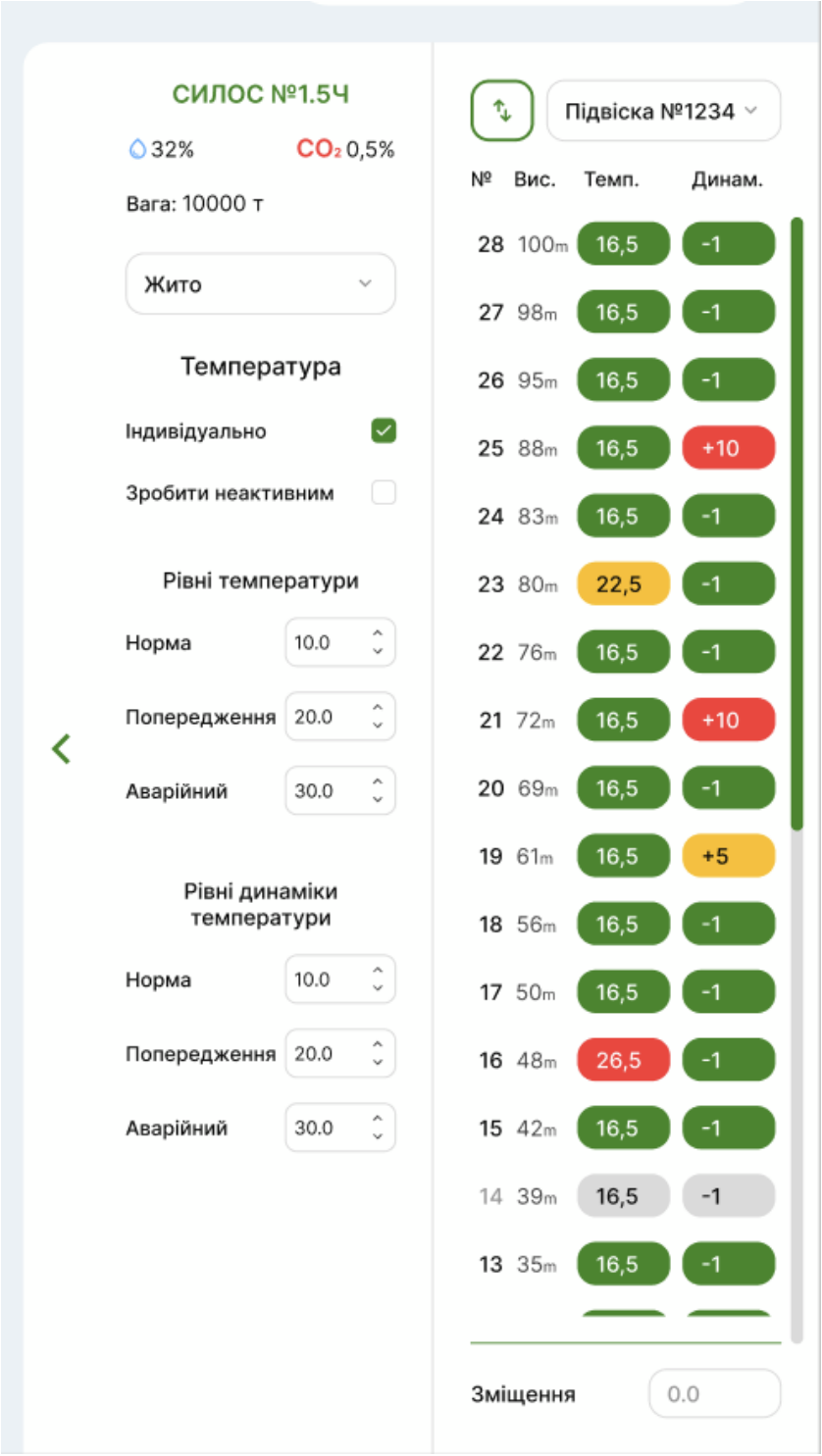


Рисунок 3.8 – Панель індивідуальних температурних налаштувань для вибраного силоса та термопідвіски

Часові графіки температури в інтерфейсі рівня «Силос» реалізовано як окремий підмодуль візуалізації часових рядів, що працює поверх REST-API модулю термометрії. Для кожної вибраної термопідвіски формується набір часових рядів: окремий ряд відповідає одному датчику, вісь *x* інтерпретується як час

вимірювання, вісь *y* – як температура (за потреби додаються похідні параметри, наприклад, динаміка температури). За замовчуванням при відкритті вкладки «Картка силоса» на графіку відображаються дані першої за номером підвіски, причому одночасно візуалізуються всі датчики цієї підвіски, що узгоджується з вимогами бізнес-логіки модулю термометрії.


Рисунок 3.9 – Графік зміни температури датчиків у часі для вибраного силоса

Графік є невід'ємною частиною інтерфейсу й не може бути прихований або деактивований для окремих підвісок (рис. 3.9): це забезпечує постійний доступ оператора до історії температури при будь-яких операціях із налаштуванням порогів чи переглядом 2D/3D представлень. У разі, якщо користувач не звужує вибір до конкретного датчика, відображаються всі серії одночасно, що дає можливість оцінити просторово-часову структуру температурного поля по висоті силоса та виділити аномальні криві. Діапазон часу задається за допомогою пари контролів «Від» / «До»; на рівні серверної частини реалізується агрегування даних (усереднення, мінімум, максимум за фіксований інтервал), що дозволяє працювати з великими обсягами архіву без помітних затримок в інтерфейсі.

Реалізоване реактивне оновлення графіка: зміна активної підвіски, перемикання датчиків у правій таблиці або модифікація часових меж генерує новий запит до API, після чого компоненти візуалізації отримують вже підготовлений набір точок. При цьому застосовується уніфікована кольорова схема, синхронізована з легендою температурних статусів (норма, попередження, аварія, нижче норми, відсутні дані), що забезпечує семантичну узгодженість між табличним поданням, 2D/3D-візуалізацією силоса та часовими графіками.

Для розширеного аналізу часових рядів передбачається інтеграція з платформою *Grafana*, яка використовується в системі як стандартний інструмент моніторингу та візуалізації метрик додатку й технологічних параметрів. Grafana є відкритою аналітичною платформою для роботи з часовими рядами, підтримує підключення до різних джерел даних (PostgreSQL, промислові time-series-БД тощо), гнучку фільтрацію, побудову складних панелей, накладання кількох параметрів на один графік, а також збереження користувацьких дашбордів. У контексті автоматизованої системи моніторингу передбачається використання Grafana-дашбордів, попередньо налаштованих для задач термометрії силосів: на них реалізовано перемикачі «елеватор – силос – підвіска – датчик», вибір часових інтервалів, накладання температури, динаміки, рівня заповнення та супутніх параметрів [26].

З технічної точки зору взаємодія реалізується через вбудовування Grafana-панелей (iframe або спеціалізований компонент) у вкладку з графіками; автентифікація й авторизація здійснюються централізовано відповідно до ролей користувача в системі. Такий підхід дозволяє, з одного боку, зберегти легковаговий, орієнтований на оперативну роботу графік у веб-інтерфейсі модуля термометрії, а з іншого – надати інженерам і аналітикам повнофункціональне середовище дослідження часових рядів без необхідності розробки власного «велосипеда» для складних сценаріїв аналізу даних.

### 3.3.3 Інструмент інтерактивного редагування дашборду

Для того щоб інтерфейс користувача був не лише інформативним, а й гнучко пристосовувався до потреб конкретного підприємства та ролі користувача, в автоматизованій системі моніторингу реалізовано інструмент інтерактивного редагування дашборду. Цей інструмент дозволяє менеджеру даних або адміністратору самостійно формувати склад інформаційних віджетів на екрані, змінювати їхній розмір і розташування, додавати нові графіки та табличні панелі, а

також зберігати декілька варіантів конфігурацій для різних сценаріїв роботи. З точки зору прав доступу редагування доступне лише користувачам із відповідними дозволами, тоді як оператори бачать той самий дашборд у режимі лише перегляду. Технічно редактор дашборду реалізований як окремий компонент клієнтського веб-застосунку, побудованого на основі React з використанням типового для SPA підходу «стан – подання». Віджети (графіки температури, карти силосів, списки тривог, агреговані показники) описуються єдиною конфігураційною моделлю у вигляді JSON-структури, яка містить тип віджета, прив’язку до джерела даних (наприклад, до конкретного силоса чи датчика), а також параметри компонування: координати у сітці, ширину, висоту, мінімальний розмір. Під час завантаження сторінки клієнт отримує цю конфігурацію з серверної частини через REST-запит; на її основі конструктор динамічно створює дерево React-компонентів і розташовує їх у контейнері сіткового макета.

Режим редагування активується окремою кнопкою або пунктом меню, після чого поверх стандартного подання вмикаються механізми drag-and-drop. Для реалізації перетягування та зміни розмірів використовується бібліотека керування сіткою React-Grid-Layout, яка відслідковує переміщення курсора, накладає обмеження по сітці та генерує події зміни макета. Кожна така подія призводить до оновлення локального стану: для відповідного віджета змінюються координати та розміри, а оновлена конфігурація одразу відображається на екрані. Усі обчислення виконуються на клієнті, тому користувач отримує миттєвий візуальний зворотний зв’язок без додаткових запитів до сервера.

Коли користувач завершує редагування, оновлена конфігурація відправляється на сервер у вигляді JSON-об’єкта. На рівні бекенду ця структура зберігається в таблиці налаштувань: для кожного підприємства, ролі або конкретного користувача фіксується набір віджетів, їхні параметри й версія дашборду. За потреби підтримується історія змін, що дозволяє відкотитися до попередньої конфігурації. Для забезпечення узгодженості даних перед збереженням виконується валідація: перевіряється, чи всі віджети відповідають

відомим типам компонентів, чи коректно вказані ідентифікатори силосів і датчиків, чи не порушено обмеження на мінімальні розміри елементів [28].

Окрему увагу приділено тому, щоб дашборди, налаштовані в інструменті редагування, коректно працювали у звичайному режимі моніторингу. Для цього кожний віджет описує не тільки свою геометрію, а й параметри підключення до джерел даних: ім’я API-методу, фільтри (ідентифікатор підприємства, силоса, датчиків), налаштування періоду агрегації, порогові значення для підсвічування тощо. Під час відтворення дашборду в робочому режимі клієнтський застосунок, базуючись на конфігурації, ініціалізує для кожного віджета відповідні запити до бекенду або підписки на WebSocket-канал. Таким чином, зміна структури дашборду не потребує модифікації коду прикладної логіки: достатньо оновити конфігурацію, і інтерфейс автоматично підлаштовується під нове компонування.

Завдяки такому підходу інструмент інтерактивного редагування дашборду перетворює статичний інтерфейс на гнучку конфігуровану панель керування, в якій зміни можуть виконуватися без участі розробників. Менеджер даних має змогу швидко адаптувати подання інформації під нову культуру, сезон або режим роботи елеватора: винести на перший план силоси з найбільшим ризиком, додати детальні графіки для окремих зон, згрупувати віджети за технологічними лініями. Це підвищує ефективність використання автоматизованої системи моніторингу та зменшує час реакції персоналу на зміни стану зерна.

### 3.3.4 Інтерфейс індивідуальних налаштувань температур

Індивідуальні налаштування температури є зв’язувальною ланкою між алгоритмами аналізу та щоденною роботою операторів. Саме через цей інтерфейс технолог задає порогові значення «норма – попереджувальний – аварійний» для різних культур і партій зерна, а також визначає, які силоси, термопідвіски та датчики беруть участь в обчисленні індексів температури й динаміки. На основі

цих параметрів модуль обробки даних формує індекси, спрацьовують тривоги і будується візуалізація.

Інтерфейс реалізовано у вигляді окремого вікна «Налаштування», що відкривається з адміністративного розділу. У верхній частині розміщено вкладки «Культури», «Договори» та «Термометрія». Підрозділ «Термометрія», представлений на рисунку 3.10, містить табличну форму, у якій по рядках перелічені культури і їхні класи («За замовчуванням», «Пшениця 2 кл.», «Ячмінь продовольчий» тощо), а по стовпцях – три рівні температури та три рівні динаміки температури. Для кожної комірки використовується контроль типу numeric input зі стрілками, що дозволяє задавати значення з кроком, погодженим із дискретністю вимірювань датчиків. Значення в стовпцях «Норма», «Попереджувальний», «Аварійний» по температурі відповідають порогам, які використовуються в алгоритмах класифікації стану зернової маси; аналогічні стовпці для «Рівнів динаміки температури» визначають допустимі швидкості зміни температури за одиницю часу, що застосовуються при розрахунку індексу динаміки.

Перший рядок таблиці – «За замовчуванням» – задає глобальні значення, які використовуються у випадку, коли для конкретної культури не налаштовано спеціальні пороги. Це дає змогу швидко розгорнути систему з типовими параметрами, а потім поступово уточнювати налаштування для окремих культур і класів. Кнопка «Додати культуру» над таблицею відкриває діалог створення нового рядка: користувач вводить назву культури, після чого отримує можливість задати для неї власні порогові значення. Усі зміни, внесені в таблицю, відстежуються у локальному стані форми; кнопки «Відмінити» та «Зберегти» в нижній частині вікна відповідно скидають поточні правки або надсилають оновлену конфігурацію на сервер.

Вкладка «Термометрія» розроблена як набір React-компонентів, що реалізують «табличний» редактор конфігурацій. Структура даних для однієї культури має вигляд об’єкта з полями cropName, tempLevels та tempDynamics, де кожен блок містить значення для «норма / попереджувальний / аварійний». Компонент верхнього рівня отримує масив таких об’єктів з бекенду через REST -

API, відображає його у вигляді таблиці та на кожну зміну в комірці оновлює відповідний елемент масиву. Валідація виконується безпосередньо і з клієнтської частини, так і з серверної: не допускається введення некоректних числових значень, попереджувальний рівень має бути більшим за норму, а аварійний – не меншим за попереджувальний; у разі порушення цієї логіки комірки підсвічуються, а кнопка «Зберегти» блокується.

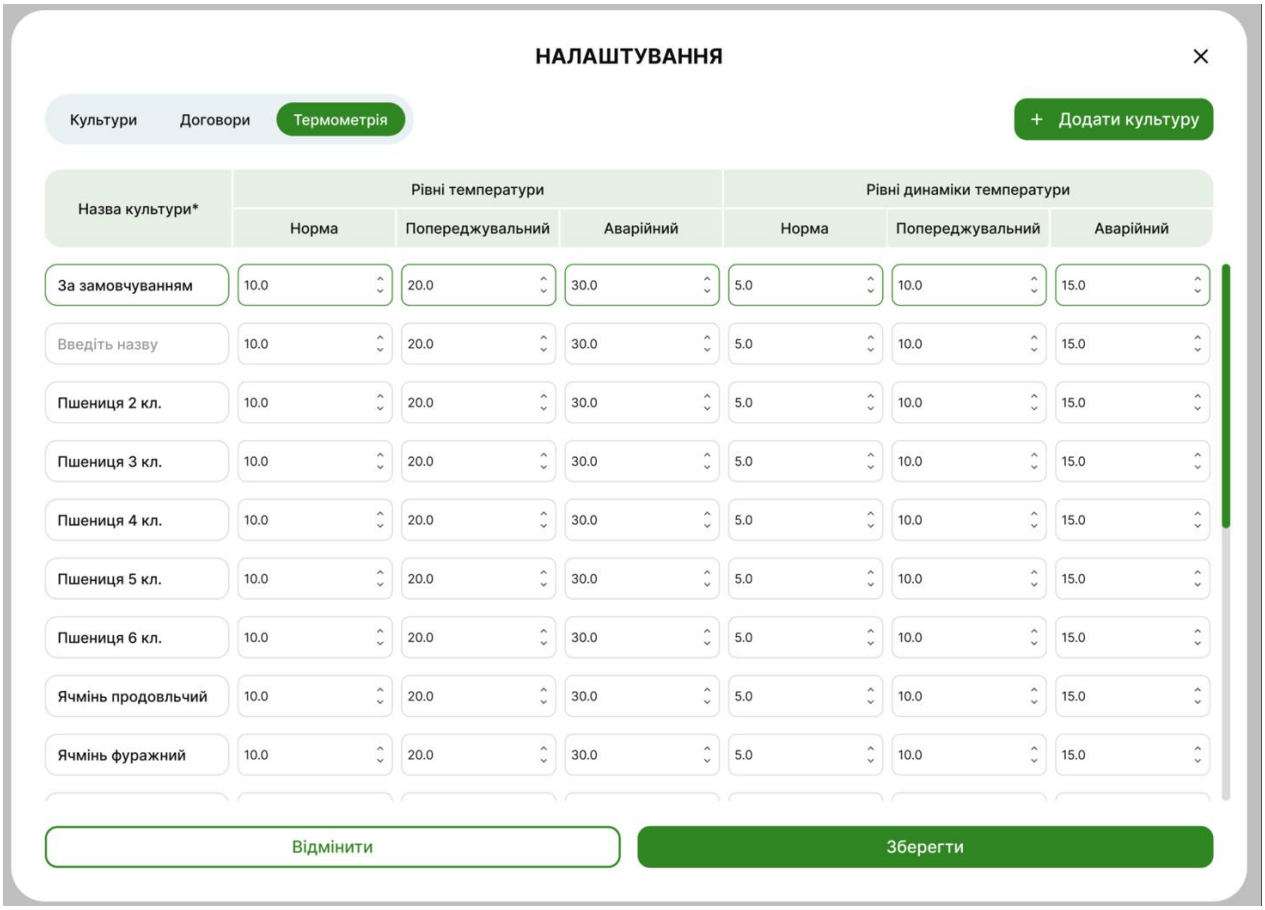


Рисунок 3.10 – Інтерфейс налаштування порогових рівнів температури та її динаміки для різних культур у модулі термометрії

Після натискання «Зберегти» вся таблиця серіалізується в JSON-структуру й надсилається на сервер одним запитом. На стороні бекенду ця структура розкладається по сутностях бази даних (наприклад, таблиця CropTemperatureSettings), де для кожної культури зберігаються її унікальний ідентифікатор, тип культури та відповідні порогові значення. Збереження виконується в транзакції, що дозволяє уникнути ситуацій із частковим

оновленням. Після успішного запису сервер повертає актуальну конфігурацію, яка оновлює стан клієнтського застосунку, а також кеш температурних налаштувань у модулі обробки даних.

Налаштування на рівні культур доповнюють конфігурацію на рівні силосів, термопідвісок і датчиків, описану вище. Коли оператор створює або редагує силос у інтерфейсі керування силосами, він обирає тип культури зі списку; системний сервіс конфігурацій підтягує для цього силоса відповідні пороги з вкладки «Термометрія». Якщо згодом технологу потрібно локально «піджати» межі для окремого силоса (наприклад, для партії з підвищеною вологістю), це робиться через додаткові параметри в конфігурації силоса: для нього можуть задаватися поправки або повне перевизначення порогів, які зберігаються в таблицях SilageConfig / SensorConfig. У такому випадку алгоритми аналізу спочатку шукають локальні налаштування, а за їхньої відсутності – звертаються до глобальних значень з модулю «Термометрія».

Інтерфейс індивідуальних налаштувань температур тісно інтегрований із підсистемою тривог. Сервіс, що розраховує індекси температури та динаміки, при кожному новому вимірюванні зчитує з кешу відповідний набір порогових значень: спочатку визначається культура, потім – локальні налаштування силоса чи датчика. Порівнюючи поточні значення індексів з порогами «норма – попереджувальний – аварійний», система встановлює стан контрольної точки й, у разі перевищення, створює запис у таблиці Alert. Завдяки цьому будь-яка зміна у вікні «Налаштування» прямо впливає на логіку спрацьовування тривог без необхідності перезбирання або перезапуску застосунку.

Доступ до інтерфейсу індивідуальних налаштувань температур обмежується ролями. Змінювати глобальні пороги на вкладці «Термометрія» мають право лише користувачі з ролями адміністратора або менеджера даних; оператори можуть переглядати поточні значення, але не можуть їх редагувати. Це реалізовано як на рівні бекенду (перевірка прав перед застосуванням змін), так і на рівні інтерфейсу – кнопки «Додати культуру» та «Зберегти» приховуються або блокуються, якщо користувач не має відповідних Permission. Таким чином інтерфейс забезпечує

водночас гнучкість конфігурування температурних режимів для різних культур та контрольований, захищений доступ до критично важливих параметрів, від яких залежить безпечність і якість зберігання зерна.

### 3.4 Модуль формування звітів

Модуль формування звітів забезпечує користувачам можливість отримувати узагальнену інформацію про стан зернових мас у силосах у вигляді зафіксованих «зрізів» даних. На відміну від дашбордів, що працюють у режимі близькому до реального часу, звіти орієнтовані на документування ситуації на певний момент або за окремий період, подальший аналіз технологами, передавання результатів керівництву й контролюючим органам. На рівні серверної частини модуль використовує ті самі механізми зберігання й кешування даних, однак додає до них шар бізнес-логіки для формування структурованих представлень і збереження параметрів звіту.

Інтерфейс модуля складається з двох основних представлень. Перше – це список сформованих звітів (рисунок 3.11). У верхній частині екрана розташована кнопка «Створити звіт», поле пошуку та заголовок «Звіти». Нижче подано табличний перелік наявних звітів із такими ключовими колонками: «ID звіту», «Ким створений звіт», «Силоси» та «Дата створення звіту». Колонка «Силоси» містить перелік силосів, що були включені до відповідного звіту; таким чином користувач може швидко зорієнтуватися, для яких об’єктів сформовано певний документ. Ліворуч від таблиці розташовано контекстне меню дій по рядку, що дозволяє переглянути обраний звіт або видалити його з журналу. У нижній частині екрана реалізовано пагінацію та вибір кількості записів на сторінці, що забезпечує зручну роботу навіть за великої кількості архівних звітів.

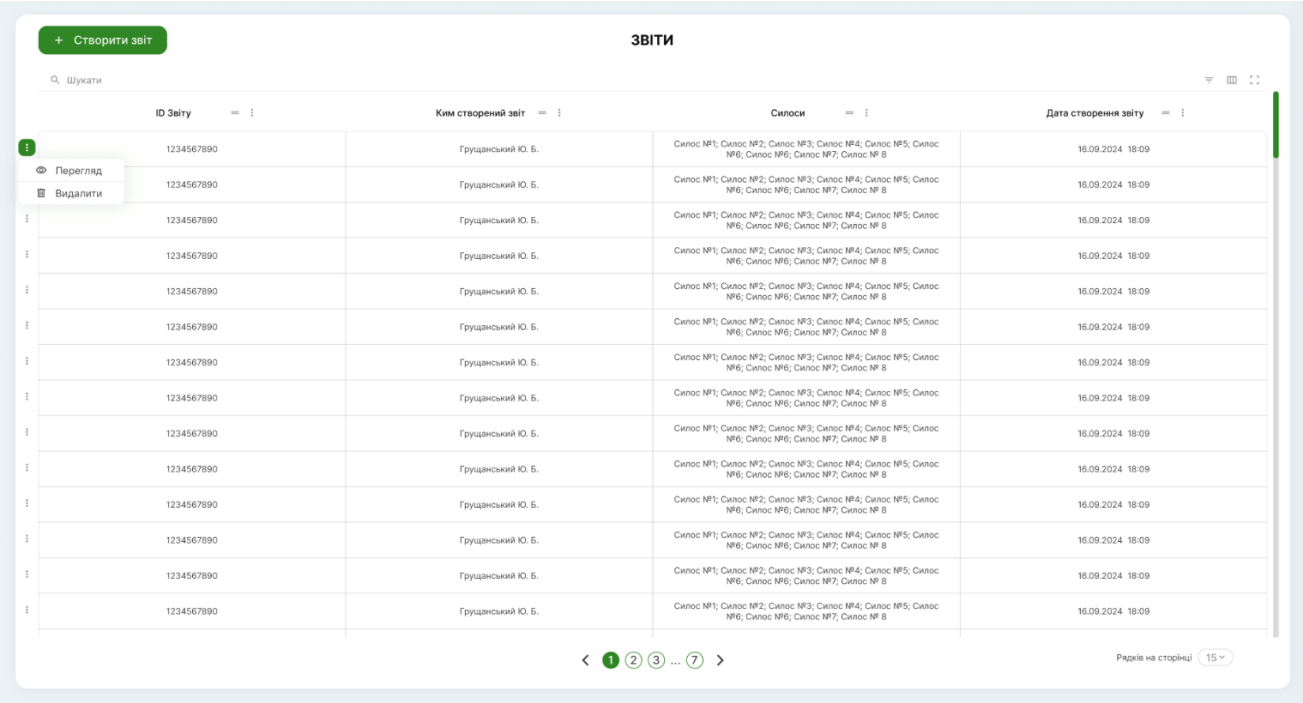


Рисунок 3.11 – Інтерфейс журналу сформованих звітів в автоматизованій системі моніторингу

Друге представлення – це форма створення й перегляду звіту по температурі (рисунок 3.12). У центральній частині екрана розташована матриця значень, де по горизонталі відкладаються термопідвіски (позначені як «П №1», «П №2» тощо), а по вертикалі – номери датчиків. Кожна комірка містить зафіксоване значення температури для відповідної пари «підвіска–датчик» на вибраний момент часу. Для наочності використано кольорове кодування: значення, що відповідають нормальному режиму, відображаються нейтральним кольором, підвищені – жовтим або помаранчевим, а аварійні – червоним. Межі між цими станами визначаються порогами, налаштованими у модулі індивідуальних налаштувань температур, тому візуальне представлення безпосередньо відображає результат алгоритмів класифікації.

Таке представлення даних забезпечує швидку візуальну оцінку температурного стану силосу та спрощує виявлення критичних відхилень під час моніторингу.

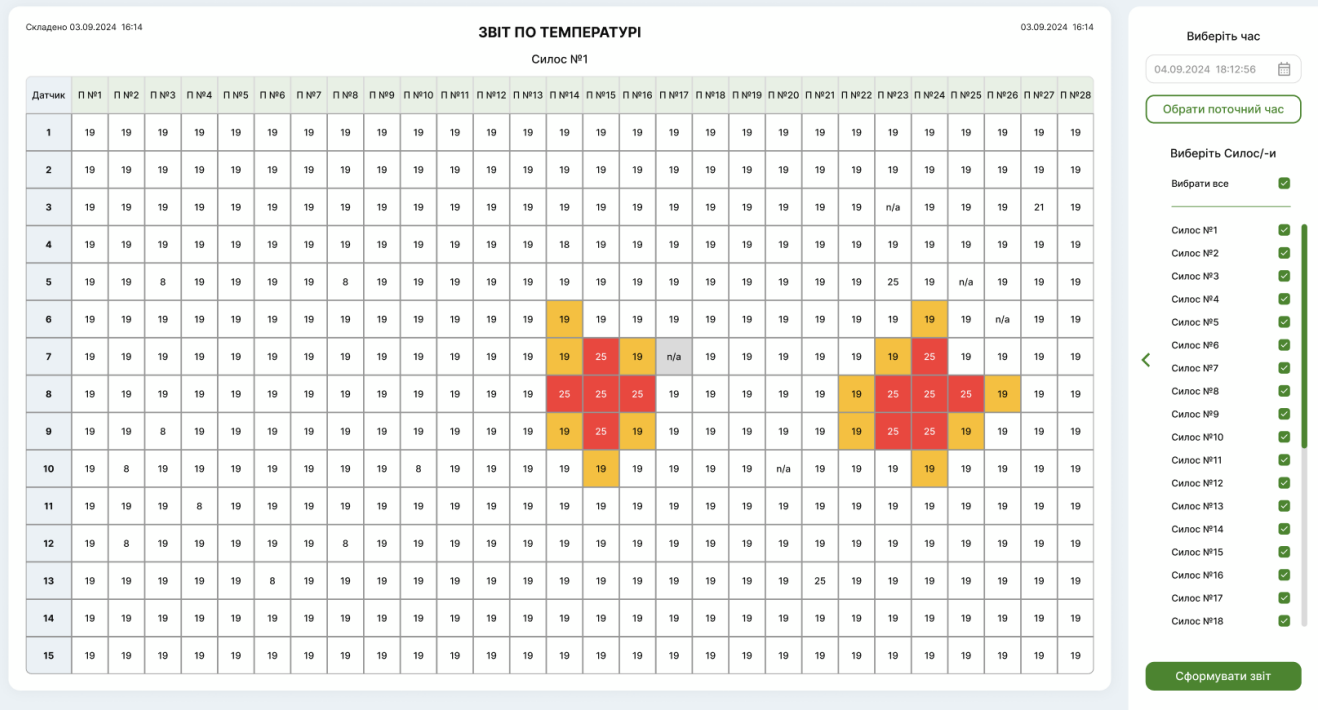


Рисунок 3.12 – Інтерфейс формування та перегляду звіту по температурі для вибраних силосів

Праворуч розміщується панель параметрів формування звіту. Блок «Виберіть час» дозволяє задати момент, на який формується зріз: або вибрати його з календаря та таймпікера, або скористатися кнопкою «Обрати поточний час» для фіксації актуальної ситуації. Нижче розташований список силосів з чекбоксами – користувач може обрати один або кілька силосів, для яких буде побудовано звіт; доступна також опція «Вибрати все». Кнопка «Сформувати звіт» ініціалізує процес звернення до серверної частини: фронтенд надсилає параметри (час, перелік силосів, тип звіту), бекенд отримує для кожного вибраного силоса останні на той момент вимірювання від відповідних датчиків, застосовує налаштовані порогові значення, формує структуру матриці й повертає її у вигляді JSON-відповіді. Клієнтський застосунок на основі цієї відповіді заповнює матрицю значень та присвоює кожній комірці відповідний колір.

З технічної точки зору для кожного сформованого звіту на сервері може створюватися запис у таблиці Report, де зберігаються його ідентифікатор, автор, час формування, перелік силосів та параметри побудови (тип звіту, джерело даних, версія шаблону). Це дозволяє згодом повторно відкривати звіт у тому вигляді, в

якому він був сформований, навіть якщо поточні дані сенсорів або налаштування порогів змінилися. Для прискорення роботи, особливо коли звіти охоплюють багато силосів, результати обчислень можуть кешуватися в модулі зберігання даних: при повторному відкритті вже сформованого звіту система спочатку звертається до кешу, а не виконує повторну вибірку й обробку великої кількості вимірювань.

Модуль формування звітів інтегрований із рольовою моделлю доступу. Створювати й видаляти звіти можуть користувачі з розширеними правами (наприклад, технологи або менеджери даних), тоді як оператори мають доступ лише до перегляду. Відповідні обмеження перевіряються як на рівні інтерфейсу (приховування або блокування кнопок), так і на рівні бекенду. Завдяки цьому забезпечується одночасно прозоре документування стану силосів і контрольований доступ до керування архівом звітів.

У підсумку модуль формування звітів доповнює дашборди й аналітичні компоненти системи, надаючи користувачам зручний інструмент для фіксації та аналізу стану зернових запасів у задані моменти часу. Він базується на єдиній моделі даних та алгоритмах оцінки температурних режимів, описаних у попередніх підрозділах, і забезпечує відтворюваність результатів, що є важливим для підтвердження якості зберігання зерна та обґрунтування прийнятих технологічних рішень.

### 3.5 Модуль оповіщень та керування подіями

Модуль оповіщень є зв’язувальною ланкою між алгоритмами аналізу даних та користувачем, який приймає рішення. Саме він перетворює сирі вимірювання й розраховані індекси на зрозумілі повідомлення про події: перехід датчика в попереджувальний або аварійний стан, вихід температури за нижню межу, критичну динаміку, відновлення норми тощо. На основі цих подій оператори

контролюють ситуацію в силосах, а технологи та керівники відстежують історію розвитку подій та ефективність прийнятих заходів.

Логічно життєвий цикл сповіщення починається в алгоритмічному модулі. Після кожного оновлення температури датчика обчислюються індекси стану температури й динаміки з урахуванням індивідуальних порогів для культури, силоса, підвіски та самого датчика. Якщо індекс виходить за межі нормального діапазону або змінюється клас стану (наприклад, із «норма» на «попереджувальний» чи з «попереджувальний» на «аварійний»), формується подія. Вона містить тип (зміна статусу, критичний рівень динаміки, відновлення норми), рівень пріоритету, посилання на силос, підвіску й конкретний датчик, часову мітку та текстове повідомлення, придатне для відображення в інтерфейсі. Ця подія записується до таблиці Alert у базі даних і паралельно публікується у брокер повідомлень, з якого її забирає сервіс оповіщень.

Сервіс оповіщень відповідає за доставку подій до користувача та керування їхнім станом. Для інтерактивного інтерфейсу використовується канал WebSocket або server-sent events: як тільки з’являється нове сповіщення, воно миттєво потрапляє до браузера без необхідності періодичного опитування сервера. На цьому етапі виконується додаткова фільтрація за правами доступу: користувачі одного холдингу бачать лише події по своїх підприємствах, а всередині підприємства – лише ті силоси, до яких у них є доступ згідно з моделлю Permission & Authority. Для кожної підсистеми (термометрія, TOiP тощо) події групуються окремо, що дозволяє оператору зосередитися саме на температурних тривогах чи, навпаки, на подіях, пов’язаних із ремонтом обладнання.

На рівні інтерфейсу модуль представлений компактним «віджетом сповіщень» у верхній панелі та окремим екраном історії (рисунки 3.13 та 3.14). Віджет показує дві вкладки – «Термометрія» та «TOiP» – з лічильниками непрочитаних повідомлень. У списку відображаються останні події за температурою: «Зміна статусу: Аварійний», «Зміна статусу: Попереджувальний», «Зміна статусу: Норма», «Зміна статусу: Нижній», «Критичний рівень динаміки температури». Кожному типу відповідає власна піктограма та колір (червоний для

аварійних, жовтий для попереджувальних, зелений для нормалізації, синій для зниженого рівня), що дозволяє з першого погляду оцінити характер проблеми [29].

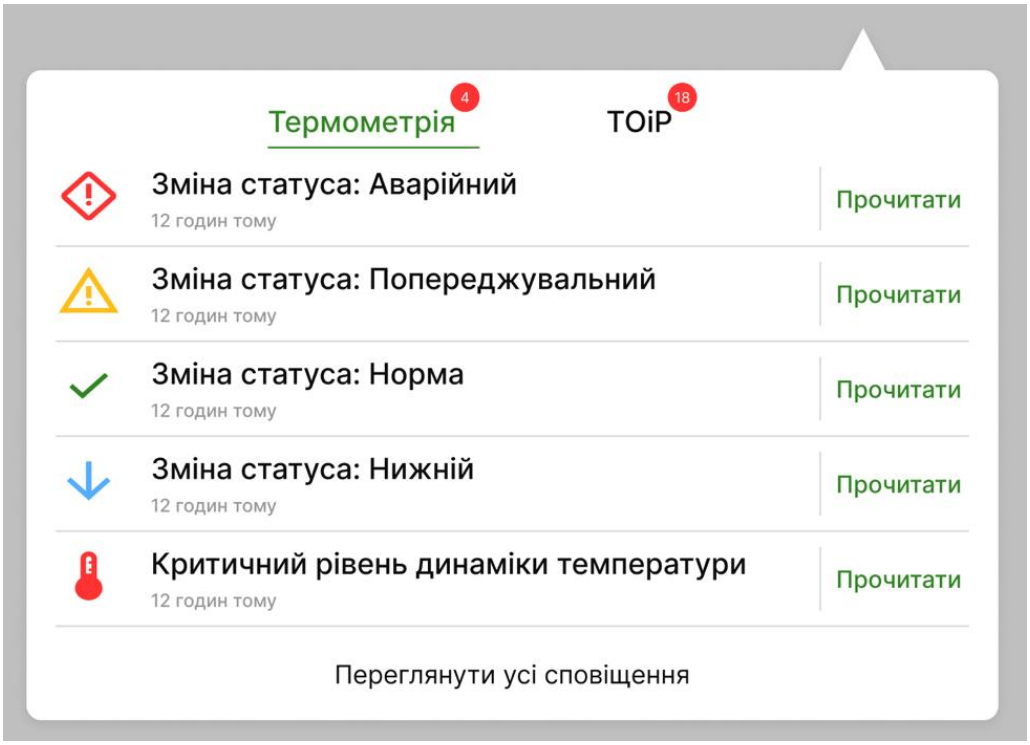


Рисунок 3.13 – Віджет сповіщень модуля термометрії

Кнопка «Прочитати» поруч із кожним рядком позначає подію як переглянуту, зменшуючи лічильник непрочитаних сповіщень; при цьому запис залишається в історії та може бути проаналізований пізніше.

Розгорнута історія сповіщень термометрії відображається на окремому екрані (рисунок 3.14). Тут усі події згруповані блоками за часом виникнення, для кожної вказано тип події, а також контекст: силос, підвіска, номер датчика, поточний статус або величина дельти температури. Колірні виділення повторюють шкалу, прийняту в основному інтерфейсі моніторингу. Користувач може перемикати період відображення (сьогодні, тиждень, довільна дата), позначати події як прочитані, а також виконувати дії над групою – наприклад, приховати всі сповіщення по вже оброблених датчиках. Додатково передбачені кнопки «Друкувати» та «Завантажити», які формують документований звіт по

сповіщеннях за обраний період у форматі, придатному для збереження в архіві або передачі контролюючим органам.

З технічної точки зору стан сповіщення в системі описується кількома ознаками: «нове», «прочитане», «закрите/вирішене». При першому показі події конкретному користувачу створюється зв’язок у таблиці користувацьких статусів, де позначається, чи переглядав він цю подію. Таким чином один і той самий запис в Alert може бути «новим» для одного оператора та вже «прочитаним» для іншого. При необхідності подія може бути пов’язана з записом у модулі TOiP (створення заявки на огляд або ремонт), і тоді її статус оновлюється автоматично при зміні стану відповідної заявки. Це забезпечує прозоре керування життєвим циклом подій від моменту виявлення проблеми до її усунення.

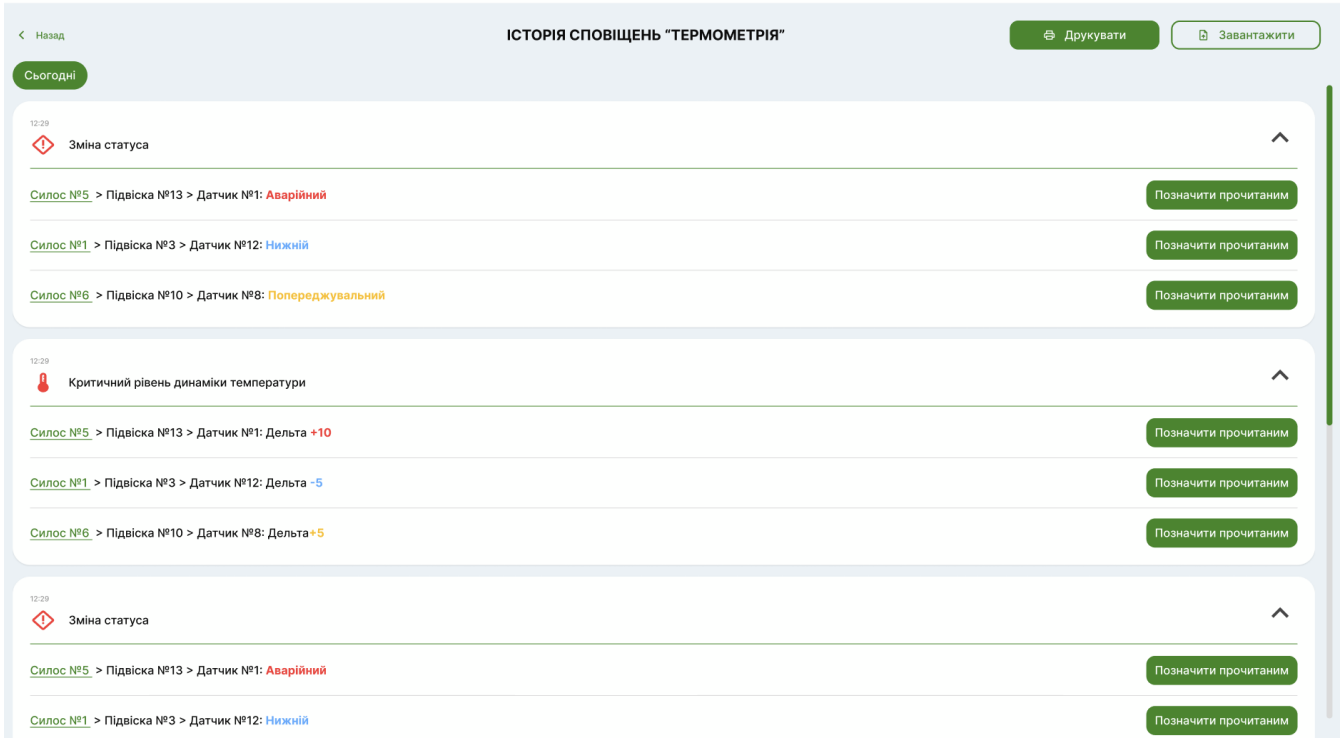


Рисунок 3.14 – Екран історії сповіщень термометрії з деталізацією по силосах і датчиках

Модуль оповіщень інтегрований із інтерфейсами рівня «Елеватор» і «Силос». Клік по сповіщенню відкриває картку відповідного силоса і автоматично підсвічує проблемний датчик у 2D/3D-візуалізації, дозволяючи оператору одразу перейти від

текстового повідомлення до просторової картини розподілу температури. У зворотному напрямку, з екрану силоса, користувач може відкрити список подій лише по цьому силосу, щоб відстежити, як змінювався його стан у часі. Така двостороння інтеграція робить роботу оператора більш ефективною: замість ручного пошуку «де саме перегрів», він отримує готовий сценарій — сповіщення → перехід до силоса → аналіз ситуації → фіксація результату.

Рольова модель доступу впроваджена і в модулі оповіщень. Оператори мають доступ до перегляду подій і відмітки їх як прочитаних; менеджери даних можуть додатково конфігурувати правила групування та фільтрації сповіщень; адміністратори – налаштовувати глобальні політики, наприклад, які типи подій вимагають обов’язкової реакції, які дублюються на електронну пошту або в зовнішні системи. Це дозволяє адаптувати поведінку модуля під специфіку конкретного підприємства або холдингу, не змінюючи основних алгоритмів аналізу.

Таким чином модуль оповіщень та керування подіями забезпечує оперативне інформування персоналу про зміни стану зернової маси, підтримує наскрізний життєвий цикл подій і тісно інтегрується з іншими компонентами автоматизованої системи моніторингу. Він перетворює великі потоки телеметрії та обчислених індексів на структурований потік дійових сигналів, які можуть бути швидко опрацьовані операторами й використані для прийняття технологічних рішень.

### 3.7 Висновки до розділу 3

У третьому розділі реалізовано повний програмний контур автоматизованої системи моніторингу – від приймання телеметрії до візуалізації, звітів та сповіщень. Розроблені алгоритми аналізу температури й рівня заповнення силосів забезпечують оцінку поточного стану зернової маси, виявлення аномалій та розрахунок орієнтовної маси культури з урахуванням геометрії силоса й насипної густини. Це дозволяє не лише фіксувати проблеми перегріву, а й використовувати

дані для технологічних рішень щодо аерації, сушіння та оптимального завантаження.

Модуль зберігання та попередньої обробки даних побудовано на зв’язці контролери – брокер повідомлень – серверна частина – СУБД, із застосуванням валідації, фільтрації шумів, кешування та оптимізованих структур БД. Такий підхід відповідає вимогам до масштабованості, мультихмарного розгортання та роботи з великою кількістю датчиків, задекларованим у технічній документації системи. Користувацький інтерфейс реалізує багаторівневу візуалізацію «елеватор – силос – підвіска – датчик» у 2D, 3D та табличному режимах, що забезпечує як оглядову, так і детальну аналітику. Інструмент інтерактивного редагування дашборду та інтерфейс індивідуальних налаштувань температур дають можливість технологам гнучко адаптувати систему під конкретний об’єкт, культуру та сценарій експлуатації без зміни програмного коду.

Модуль формування звітів забезпечує фіксацію стану силосів на вибраний момент часу та дає змогу зберігати, переглядати, завантажувати й друкувати сформовані документи. Наявність історії звітів та їх параметрів створює основу для аудиту режимів зберігання й підготовки регламентної документації. Модуль оповіщень перетворює зміни індексів температури та їх динаміки на структуровані події з різними рівнями пріоритету й каналами доставки, що дозволяє операторам оперативно реагувати на відхилення без постійного ручного контролю.

У сукупності розроблені програмні модулі демонструють, що система здатна працювати з великими потоками вимірювань у режимі, наближеному до реального часу, забезпечувати стійке зберігання даних, наочну візуалізацію, гнучке налаштування порогів і інтерфейсів, а також повноцінну подієву модель та звітність. Це створює технічне підґрунтя для подальшого розвитку – розширення набору датчиків, інтеграції з іншими підсистемами елеватора та впровадження прогнозних сервісів на основі методів аналізу даних й штучного інтелекту.

# 4 ЕКОНОМІЧНИЙ РОЗДІЛ

## 4.1 Технологічний аудит розробленої автоматизованої системи моніторингу

Розроблена в магістерській роботі автоматизована система моніторингу умов зберігання зернових культур у силосах призначена для багатоканального контролю температури, вологості, рівня зерна та концентрації $CO_2$, а також для аналізу та прогнозування зміни цих параметрів у часі. Система реалізує багаторівневу візуалізацію даних (рівні «підприємство – силос – підвіска – датчик») у 2D, 3D та табличному режимах, забезпечує формування звітів і надсилання аварійних оповіщень користувачам.

Архітектурно система побудована за багаторівневим принципом, який включає: рівень збору даних (контролери, термопідвіски та додаткові датчики), рівень обробки (фільтрація, нормалізація, аналітичні алгоритми, оцінка коливань температури), рівень зберігання (багаторівнева БД із можливістю використання реляційних і NoSQL-сховищ), рівень API та інтеграції (REST/GraphQL-сервіси), рівень візуалізації веб-інтерфейс у хмарній інфраструктурі, рівень безпеки та управління подіями.

Технологічний стек включає Java 17+ для серверної частини, Spring Boot для реалізації мікросервісів та REST API, PostgreSQL / H2 для зберігання даних, засоби контейнеризації та хмарного розгортання, а також бібліотеку three.js для реалізації 3D-візуалізації температурного поля силосів.

Особливістю розробленої системи є:

- підтримка роботи в хмарній конфігураціях з можливістю синхронізації даних;

- інтеграція з існуючими системами збору даних (у т.ч. OpenSCADA) і промисловими контролерами за протоколами Modbus RTU/TCP, OPC UA тощо;

- реалізація алгоритмів оцінювання коливань температури на основі коефіцієнтів варіації та добових сегментів часових рядів;

- підтримка гнучкої конфігурації структури елеватора (довільна кількість силосів, підвісок і датчиків) та інструмент інтерактивного редагування дашборду на рівні «Елеватор» і «Силос»;

- модуль формування звітів з можливістю вибору силосів, періоду, формату представлення даних та експорту;

- модуль керування оповіщеннями (електронна пошта, месенджери, внутрішні push-сповіщення).

У порівнянні з класичною термометричною системою зразка 2014 року, що реалізована за принципами «Industry 3.0» і має обмежені можливості щодо інтеграції та прогнозної аналітики, розроблена система забезпечує: SaaS-модель на базі багатохмарної платформи, розширені сервіси аналізу та прогнозування, 3D-візуалізацію, мобільний доступ і розвинену систему сповіщень.

Виходячи з аналізу ринку, в Україні експлуатується понад 1000–1300 зернових елеваторів, більшість з яких оснащені морально застарілими системами термометрії, що створює значний потенціал для впровадження більш сучасних рішень.

Для формалізованої оцінки комерційного потенціалу розробленої системи використано методику експертного оцінювання, аналогічну тій, що застосовується у типових економічних розділах магістерських робіт, але адаптовану до предметної області моніторингу зерносховищ.

Запропонована система орієнтована на поетапне впровадження та може інтегруватися з існуючою інфраструктурою елеваторів без необхідності повної заміни обладнання. Це знижує вхідні витрати для підприємств і робить перехід до сучасних цифрових рішень менш ризикованим з організаційної та фінансової точок зору.

Практичне застосування розробленої системи дозволяє підвищити надійність контролю умов зберігання зерна, зменшити ймовірність виникнення аварійних ситуацій та втрат продукції, а також забезпечити більш обґрунтоване прийняття управлінських рішень на основі накопичених даних моніторингу.

Для встановлення комерційного потенціалу системи було запрошено 3-х експертів-фахівців: Я.А. Кулик, В.В. Гармаш, М.В. Барабан.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу будь-якої розробки і їх бальна оцінка

| Критерії оцінювання та бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| Технічна здійсненність концепції: | | | | | |
| 1 | Достовірність концепції не підтверджена | Концепція підтверджена експертними висновками | Концепція підтверджена розрахунками | Концепція перевірена на практиці | Перевірено роботоздатність продукту в реальних умовах |
| Ринкові переваги (недоліки): | | | | | |
| 2 | Багато аналогів на малому ринку | Мало аналогів на малому ринку | Кілька аналогів на великому ринку | Один аналог на великому ринку | Продукт не має аналогів на великому ринку |
| 3 | Ціна продукту значно вища за ціни аналогів | Ціна продукту дещо вища за ціни аналогів | Ціна продукту приблизно дорівнює цінам аналогів | Ціна продукту дещо нижче за ціни аналогів | Ціна продукту значно нижче за ціни аналогів |
| 4 | Технічні та споживчі властивості продукту значно гірші, ніж в аналогів | Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів | Технічні та споживчі властивості продукту на рівні аналогів | Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів | Технічні та споживчі властивості продукту значно кращі, ніж в аналогів |
| Ринкові перспективи | | | | | |
| 5 | Експлуатаційні витрати значно вищі, ніж в аналогів | Експлуатаційні витрати дещо вищі, ніж в аналогів | Експлуатаційні витрати на рівні експлуатаційних витрат аналогів | Експлуатаційні витрати трохи нижчі, ніж в аналогів | Експлуатаційні витрати значно нижчі, ніж в аналогів |
| 6 | Ринок малий і не має позитивної динаміки | Ринок малий, але має позитивну динаміку | Середній ринок з позитивною динамікою | Великий стабільний ринок | Великий ринок з позитивною динамікою |
| 7 | Активна конкуренція великих компаній на ринку | Активна конкуренція | Помірна конкуренція | Незначна конкуренція | Конкурентів немає |

| Практична здійсненність | | | | | |
|---|---|---|---|---|---|
| 8 | Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї | Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців | Необхідне незначне навчання фахівців та збільшення їх штату | Необхідне незначне навчання фахівців | Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї |

Продовження таблиці 5.1

| Критерії оцінювання та бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| 9 | Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні | Потрібні незначні фінансові ресурси. Джерела фінансування відсутні | Потрібні значні фінансові ресурси. Джерела фінансування є | Потрібні незначні фінансові ресурси. Джерела фінансування є | Не потребує додаткового фінансування |
| 10 | Необхідна розробка нових матеріалів | Потрібні матеріали, що використовуються у військово-промисловому комплексі | Потрібні дорогі матеріали | Потрібні досяжні та дешеві матеріали | Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві |
| 11 | Термін реалізації ідеї більший за 10 років | Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років | Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років |
| 12 | Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту | Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу | Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу | Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту | Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту |

Наведена в таблиці 5.1 система критеріїв дозволяє комплексно оцінити комерційний потенціал програмно-апаратних розробок з урахуванням як технічних, так і ринкових аспектів. Запропоновані критерії охоплюють рівень технічної зрілості концепції, наявність конкурентних переваг, стан і динаміку ринку, а також практичну здійсненність реалізації ідеї з огляду на кадрові, фінансові та матеріальні ресурси. Використання п’ятибальної шкали оцінювання забезпечує можливість порівняння різних розробок між собою та формування узагальненої кількісної оцінки. Такий підхід спрощує прийняття рішень щодо доцільності подальшого розвитку, інвестування або впровадження розробки у виробничу діяльність, а також дозволяє виявити найбільш проблемні аспекти, що потребують додаткового опрацювання.

Результати оцінки комерційного потенціалу розробки наведено в таблиці 5.2.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробленої нами системи обслуговування клієнтів в сфері стоматології за допомогою агентів штучного інтелекту (Шкала оцінювання «0»-«1»-«2»-«3»-«4»)

| Критерії | Ініціали, прізвище експертів | | |
|---|---|---|---|
| | Я.А. Кулик | В.В. Гармаш | М.В. Барабан |
| | Бали, що їх виставили експерти: | | |
| 1 | 4 | 4 | 4 |
| 2 | 3 | 4 | 3 |
| 3 | 4 | 3 | 3 |
| 4 | 3 | 3 | 4 |
| 5 | 4 | 4 | 4 |
| 6 | 3 | 3 | 3 |
| 7 | 4 | 4 | 4 |
| 8 | 3 | 4 | 3 |
| 9 | 4 | 3 | 4 |
| 10 | 4 | 4 | 3 |
| 11 | 3 | 3 | 3 |
| 12 | 4 | 3 | 4 |
| Сума балів | $СБ_1 = 43$ | $СБ_2 = 42$ | $СБ_3 = 41$ |
| Середньоарифметична сума балів $\overline{СБ}$ | $\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{43+42+41}{3} = \frac{126}{3} = 42,00$ | | |

Для встановлення комерційного потенціалу розробленої системи автоматизації моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі звернемося до рекомендацій, які наведено в таблиці 5.3.

Таблиця 5.3 – Рівні комерційного потенціалу будь-якої наукової розробки

| Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів | Рівень комерційного потенціалу розробки |
|---|---|
| 0 – 10 | Низький |
| 11 – 20 | Нижче середнього |
| 21 – 30 | Середній |
| 31 – 40 | Вище середнього |
| 41 – 48 | Високий |

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 42,00 балів (із максимально можливих 48-ми балів), то це свідчить, що розроблена автоматизації моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі має рівень комерційного потенціалу, який можна вважати «високим».

Така оцінка обумовлена тим, що впроваджені в систему інноваційні рішення – багаторівнева 2D/3D-візуалізація стану силосів, аналітична обробка часових рядів температури та вологості, інтеграція з хмарною інфраструктурою та гнучкий механізм оповіщень – відкривають нові можливості для підвищення надійності контролю та зниження втрат зерна під час зберігання. Подальший розвиток системи, зокрема розширення функцій прогнозної аналітики та інтеграція з модулями оптимізації вентиляції й сушіння, може стати ключовим фактором підвищення енергетичної ефективності роботи елеваторів та зміцнення конкурентоспроможності підприємств зернозберігання.

## 4.2 Розрахунок витрат на розроблення системи автоматизації моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі

Під час розробки були зроблені такі витрати:

А). Основна заробітна плата ($Z_o$) розробників (дослідників тощо), яка визначається за формулою:

$$Z_o = \sum_{i=1}^{n} \frac{M_i}{T_p} \cdot t_i, \quad (5.1)$$

де ($M_i$) – місячний посадовий оклад і-го розробника, грн;

$T_p$ – кількість робочих днів у місяці;

t – кількість днів участі і – го виконавця в проєкті;

(n) – кількість учасників

Розрахунки основної заробітної плати учасників наведено в таблиці 5.4.

Таблиця 5.4 – Основна заробітна плата учасників розроблення системи

| Найменування посади виконавця | Місячний посадовий оклад, грн | Оплата за робочий день, грн | Число днів (годин) роботи | Витрати на оплату праці, грн |
|---|---|---|---|---|
| 1. Науковий керівник магістерської роботи | 25000 | 1086 | 4 години | 3600 |
| 2. Здобувач-магістрант (виконавець) | 12000 | 522 | 60 днів | 31304 |
| 3. Консультант з економічної частини | 20000 | 870 | 0,5 днів | 435 |
| 4. Галузевий консультант (елеваторне обладнання) | 28000 | 1218 | 0,5 днів | 609 |
| Загалом | | | | $Z_o$ = 35971 грн |

Б). Додаткова заробітна плата $Z_o$ розробників (дослідників тощо) розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$Z_d = a_d \cdot Z_o, \quad (5.2)$$

Приймемо, що α = 0,1. Тоді для нашого випадку отримаємо:

$$Z_o = 0{,}10 \times 35971 \approx 3597 \text{ грн.}$$

В). Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (5.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. В 2025 році ставка β = 22%. Тоді:

$НЗН_{зп} = (35971 + 3597) \times 0{,}22 \approx 8705$ грн.

Г). Амортизація основних засобів А, які використовувались під час виконання магістерської кваліфікаційної роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн}, \quad (5.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань.

Приймемо, що $Н_а$ = (2,5...25)%;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 5.5.

Таблиця 5.5 – Розрахунок амортизаційних відрахувань

| Найменування обладнання, приміщень тощо | Балансова вартість, грн. | Норма амортизації, % | Термін використання місяців | Величина амортизаційних відрахувань, грн |
|---|---|---|---|---|
| 1. Комп'ютерна техніка, обладнання тощо | 60000 | 25 | 2,8 (при 75% використанні) | 2812,5 ≈ 2813 |
| 2. Приміщення університету, факультету, кафедри | 16000 | 5 | 2,8 (при 50% використанні) | 100 |
| Всього | | | | А = 2913 грн |

Д). Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \quad \text{грн,} \qquad (5.5)$$

де $Н_i$ – витрати матеріалу *і*-го найменування, кг; $Ц_i$ – вартість матеріалу *і*-го найменування; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *і*-го найменування; $Ц_в$ – ціна відходів матеріалу *і*-го найменування; n – кількість видів матеріалів.

Е). Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \quad \text{грн,} \qquad (5.6)$$

де $Н_i$ – кількість комплектуючих *і*-го виду, шт.; $Ц_i$ – ціна комплектуючих *і*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Витрати на матеріали та витратні ресурси (папір, витратні матеріали для друку, канцтовари) для оформлення магістерської кваліфікаційної роботи та підготовки проєктної документації округлено приблизно 800 грн.

Ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \qquad (5.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2025 р. В ≈ 4,8 грн/кВт;

П – установлена потужність обладнання, кВт; П = 1,28 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 235 годин;

$К_п$ – коефіцієнт використання потужності; $К_п$ < 1 = 0,80.

$К_д$ – коефіцієнт корисної дії, $К_д$ = 0,75.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \times П \times Ф \times К}{К_d} = \frac{4{,}8 \times 1{,}28 \times 235 \times 0{,}8}{0{,}75} = 1540 \text{ грн.}$$

И). Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0{,}5....3) \times З_о. \quad (5.8)$$

Для нашого випадку отримаємо:

$$В_{інш} = 0{,}50 \times 35971 = 17986 \text{ грн.}$$

К). Сума всіх попередніх статей витрат складає витрати на виконання нашої магістерської роботи безпосередньо розробником-магістрантом – В.

$$В = 35971 + 3597 + 8705 + 2913 + 800 + 1540 + 17986 = 71512 \text{ грн.}$$

Л). Загальні витрати на розробку та можливу комерціалізацію виконаної системи розраховуються за формулою:

$$В_{заг} = \frac{В}{\beta}, \quad (5.9)$$

де $\beta$ – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи.

Оскільки розробка на цей момент часу практично готова до впровадження, то можна прийняти, що, $\beta \approx 0{,}95$.

Тоді: $В_{заг} = \frac{71512}{0{,}95} = 75275$ грн або приблизно 75 тисяч грн.

Прогнозовані загальні витрати на розробку та можливу комерціалізацію розробленої системи становлять приблизно 75 тисяч грн.

## 4.3 Розрахунок економічного ефекту від можливої комерціалізації розробленої системи автоматизації моніторингу

Проведене дослідження ринку систем моніторингу умов зберігання зерна показало, що розроблена автоматизована система моніторингу температури, вологості, рівня заповнення силосів та концентрації $CO_2$ дає змогу суттєво підвищити ефективність роботи технологічного та чергового персоналу елеватора, знизити ризик самозігрівання зерна та зменшити втрати від псування продукції. Завдяки багаторівневій 2D/3D-візуалізації, автоматизованим алгоритмам аналізу часових рядів та гнучкій системі оповіщень скорочується час реагування на відхилення параметрів зберігання, оптимізується використання вентиляційного та сушильного обладнання, що безпосередньо впливає на економічні результати діяльності підприємства.

Аналіз місткості ринку аналогічних систем показує, що в Україні функціонує значна кількість зернових елеваторів і великих зерносховищ, для яких актуальним є модернізація існуючих термометричних комплексів. За експертними оцінками, потенційна кількість впроваджень сучасних систем моніторингу може становити орієнтовно (80…110) об’єктів щороку. Для подальших розрахунків приймемо, що базовий річний обсяг реалізації програмно-апаратних комплексів підприємством-розробником до впровадження нової системи становить 70 систем на рік. Можна також очікувати зростання попиту на запропоновану систему щонайменше протягом трьох років після початку її серійної реалізації.

Отже, якщо впровадження розробленої системи буде розпочато з 1 січня 2026 року, то економічні результати від її комерціалізації проявлятимуться протягом 2026, 2027 та 2028 років.

Прогноз зростання обсягів реалізації порівняно з базовим роком можна прийняти таким:

а) 2026 р. – приблизно +10 комплектів (відносно базового року);

б) 2027 р. – +20 комплектів;

в) 2028 р. – +30 комплектів.

Економічний ефект від можливої комерціалізації розробленої автоматизованої системи моніторингу пояснюється її суттєво кращими функціональними можливостями порівняно з типовими рішеннями. Нині аналіз ринку показує, що середня вартість подібних програмно-апаратних комплексів для елеваторів становить близько 150 тис. грн за систему в базовій конфігурації.

Оскільки запропонована система забезпечує розширений функціонал (багаторівнева візуалізація, прогнозна аналітика, інтеграція з хмарною інфраструктурою та зовнішніми сервісами), її доцільно реалізовувати за середньою ціною близько 160 тис. грн, тобто на 10 тис. грн дорожче за типові аналоги.

Тоді можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від комерціалізації розробленої автоматизованої системи моніторингу умов зберігання зернових культур у силосах, тобто від виведення її на ринок, становитиме:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (5.10)$$

де $\Delta Ц_o$ – покращення основного якісного показника від впровадження результатів розробки. Для нашого випадку це приріст ціни реалізації одного програмно-апаратного комплексу системи моніторингу $\Delta Ц_o = 160 - 150 = +10$ тисяч грн;

N – основний кількісний показник, Для нашого випадку це приріст ціни реалізації одного програмно-апаратного комплексу системи моніторингу N = 70 шт.;

$\Delta N$ – покращення основного кількісного показника від впровадження результатів розробки. Таке покращення основного кількісного показника становитиме по роках, у 2026 році – + 10 шт., у 2027 році + 20 шт., у 2028 році +30 шт. (відносно базового 2025 року);

$Ц_о$ – основний якісний показник (тобто ціна), який являє собою ціну реалізації нашої розробки після її виведення на ринок, грн; $Ц_о$= 160 тисяч грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для нашого випадку n = 3;

$\lambda$ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 1/1{,}2 = 0{,}8333$;

$\rho$ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $\rho$= (0,2...0,5); візьмемо $\rho$= 0,5;

$\upsilon$ – ставка податку на прибуток підприємств. Станом на 2025 році $\upsilon$ = 18%.

У 2026-2027 роках будемо очікувати, що ця ставка залишиться на тому ж рівні: $\upsilon$ = 18%.

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливої комерціалізації розробки (2026 рік) становитиме:

$$\Delta P_1 = (10 \cdot 70 + 160 \cdot 10) \cdot 0{,}8333 \cdot 0{,}5 \cdot 0{,}82 = 2\,300 \cdot 0{,}8333 \cdot 0{,}5 \cdot 0{,}82$$
$$\approx 785{,}8 \text{ тис. грн}$$

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливої комерціалізації розробки протягом другого (2026 р.) року становитиме:

$$\Delta P_2 = (10 \cdot 70 + 160 \cdot 20) \cdot 0{,}8333 \cdot 0{,}5 \cdot 0{,}82 = 3900 \cdot 0{,}8333 \cdot 0{,}5 \cdot 0{,}82$$
$$\approx 1\,332{,}44 \text{ тисяч грн}$$

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від можливої комерціалізації розробки протягом третього (2027 р.) року становитиме:

$$\Delta P_3 = (10 \cdot 70 + 160 \cdot 30) \cdot 0{,}8333 \cdot 0{,}5 \cdot 0{,}82 = 5\,500 \cdot 0{,}8333 \cdot 0{,}5 \cdot 0{,}82 \approx 1\,879{,}09 \text{ тис. грн}$$

Приведена вартість зростання для потенційного інвестора всіх чистих прибутків від можливої комерціалізації розробки становитиме:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої нашої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для нашого випадку т = 3 роки;

$\tau$ – ставка дисконтування. Приймемо $\tau$ = 0,10 (10%);

t – період часу від моменту початку розроблення системи до моменту отримання можливих чистих прибутків від її впровадження і виведення на ринок.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації розробки, складе:

$$PP = \frac{785{,}8}{1{,}1^2} + \frac{1\,332{,}44}{1{,}1^3} + \frac{1\,879{,}09}{1{,}1^4} \approx 2933{,}95 \text{ тис. грн}$$

Теперішня вартість інвестицій PV, що мають бути вкладені в комерціалізацію розробки: $PV = К \times В_{заг} = (1{,}0…5{,}0) \times В_{заг}$.

Для нашого випадку приймемо, що:

PV = (1,0…5,0) × 95 = 5 × 75= 375 тисяч грн.

Розраховуємо абсолютний ефект від можливих вкладених інвестицій $Е_{абс}$.

$$Е_{абс} = ПП – PV, \quad (5.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для інвестора від можливої комерціалізації розробки, грн.

Абсолютний ефект від можливої комерціалізації розробки (при прогнозованому ринку збуту) за три роки складе:

$Е_{абс} = 2933,95 – 375 = 2558,95$ тисяч грн.

Оскільки $Е_{абс} > 0$, то комерціалізація розробки може бути доцільною.

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій (коштів):

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (5.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс} = 2558,95$ тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 375 тисяч грн;

$Т_ж$ – життєвий цикл розробки, роки.

$Т_ж = 4$ роки (2025-й, 2026-й, 2027-й, 2028-й роки)

Для нашого випадку отримаємо:

$$E_v = \left(\frac{2558,95}{375}\right)^{\frac{1}{4}} - 1 = 0,6725 \approx 67,25\%$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією розробки.

Мінімальна дохідність $\tau_{мін}$ визначається за формулою:

$$\tau_{міп} = d + f, \quad (5.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d = (0,10...0,12), а окремі банки встановлюють ставку за депозитними операціями на рівні до 18%. Приймемо, що $\tau$ = 15%.

f – показник, що характеризує ризикованість вкладень; f = (0,05...0,30).

Приймемо, що f = 30%, тобто f = 0,3.

Тоді для нашого випадку отримаємо:

$\tau_{мін}$ = 0,18 + 0,30 = 0,48 або $\tau_{мін}$ = 48%.

Оскільки величина $E_в$ = 67,25% > $\tau_{мін}$ = 48%, то потенційний інвестор у принципі може бути зацікавлений у комерціалізації розробленої автоматизованої системи моніторингу.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленої системи автоматизації системи моніторингу.

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{\overline{E_В}} = \frac{1}{0,67} \approx 1,49 \text{ років} \approx 1,6 \text{ рік} < 3 \text{ років} \quad (5.15)$$

що додатково підтверджує економічну привабливість та швидку окупність проєкту впровадження автоматизованої системи моніторингу умов зберігання зернових культур у силосах.

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю (таблиця 5.6):

Таблиця 5.6 - Результати виконаної економічної частини магістерської кваліфікаційної роботи

| Показники | Задані у ТЗ | Досягнуті у магістерській кваліфікаційній роботі | Висновок |
|---|---|---|---|
| 1. Витрати на розробку | Не більше | 75 тисяч грн | Досягнуто |

| | | | |
|---|---|---|---|
| | 80 тисяч грн | | |
| 2. Абсолютний ефект від впровадження розробки, тисяч грн | Не менше 2000 тисяч грн (за три роки) | 2558,95 тисячі грн (при 10% інфляції) | Виконано |
| 3. Внутрішня дохідність інвестицій, % | не менше 48,0% | 67,25% | Виконано |
| 4. Термін окупності інвестицій, роки | до 3 років | 1,6 року | Виконано |

Таким чином, проведені розрахунки, які визначені у технічному завданні підтверджують економічну доцільність розроблення та комерціалізації автоматизованої системи моніторингу умов зберігання зернових культур у силосах.

# ВИСНОВКИ

У магістерській кваліфікаційній роботі розв’язано науково-прикладну задачу розроблення автоматизованої системи моніторингу умов зберігання зернових культур у силосах з багаторівневою візуалізацією, аналізом температурних режимів та підтримкою мультихмарної інфраструктури. Актуальність роботи зумовлена зростанням тривалості зберігання зерна, підвищенням вимог до якості продукції, необхідністю мінімізації втрат від саморозігрівання в силосах, а також сучасними викликами для аграрної галузі України, пов’язаними з логістичними обмеженнями й потребою в більш точному управлінні складською інфраструктурою.

На першому етапі проведено аналіз предметної області й існуючих технічних рішень для контролю стану зерна в силосах. Розглянуто промислові системи термометрії, SCADA-рішення та окремі хмарні сервіси моніторингу. Показано, що більшість із них орієнтовані або на локальний рівень (у межах одного підприємства), або на вузькоспеціалізований функціонал (лише температурний контроль без глибокої аналітики, без гнучких інтерфейсів та інтеграції на рівні холдингу). Виявлено обмеження таких систем: недостатня масштабованість, обмежені можливості зберігання та аналітики історичних даних, слабка підтримка хмарних і мультихмарних сценаріїв, а також слабка інтеграція з сучасними веб-інтерфейсами та засобами візуалізації. На основі цього обґрунтовано доцільність створення нової автоматизованої системи з акцентом на багаторівневу архітектуру, розширену аналітику та сучасні засоби візуалізації.

У роботі спроєктовано архітектуру автоматизованої системи моніторингу, що включає рівні збору даних, попередньої обробки, централізованого зберігання, прикладної логіки, візуалізації й оповіщень. На нижньому рівні передбачено взаємодію з контролерами термометрії та іншими вимірювальними пристроями; дані надходять до брокера повідомлень, далі – до серверних сервісів, які реалізують валідацію, перетворення форматів та запис у базу даних. Середній рівень

представлений модулем зберігання та попередньої обробки даних на основі реляційної СУБД, оптимізованої для роботи з часовими рядами та великою кількістю датчиків. Верхній рівень складають веб-інтерфейс, модуль формування звітів, інструменти 2D/3D-візуалізації та підсистема оповіщень, що забезпечує взаємодію з оператором і технологом у режимі, наближеному до реального часу. Закладено можливість мультихмарного розгортання, коли окремі компоненти системи можуть працювати у кількох хмарних середовищах із використанням реплікації даних та механізмів резервування.

Розроблено алгоритмічне забезпечення аналізу температурних режимів і рівня заповнення силоса. Запропоновано використання індексів рівня температури й індексів динаміки, що дозволяє оцінювати не лише поточний стан, а й характер зміни температури у часі. Сформульовано критерії підтвердження аномальних змін на основі порівняння показань датчиків із пороговими значеннями для конкретної культури, класу зерна та партії. Окремо розроблено методику оцінювання рівня заповнення й орієнтовної маси зерна на основі вертикального профілю температур і геометричної моделі силоса. Для цього використано інформацію про висоти розташування датчиків на термопідвісках, результати класифікації датчиків на «занурені в зерно» та «у повітрі», а також відому залежність об’єму силоса від висоти заповнення. На основі цієї моделі реалізовано обчислення об’єму зернової маси й маси зерна за насипною густиною. Таким чином, система перетворює сирі температурні вимірювання на набір інтерпретованих показників, придатних для прийняття технологічних рішень.

У роботі спроєктовано структуру бази даних, яка безпосередньо відображає предметну область. Забезпечено можливість пов’язувати окремі вимірювання з конкретними об’єктами (силосами, підвісками, датчиками), користувачами й конфігураціями порогів, а також формувати гнучкі запити для аналітики. Передбачено індексування за часом та ідентифікаторами датчиків, що забезпечує прийнятний час відгуку при запитах до великих обсягів історичних даних. Структура БД підтримує розділення даних по підприємствах і холдингах, що важливо для використання системи в умовах групи компаній.

Реалізовано багаторівневий користувацький інтерфейс. На рівні «елеватор» користувач отримує оглядовий дашборд із переліком силосів, агрегованими температурними індикаторами, оцінками рівнів заповнення та основними тривожними сигналами. На рівні «силос» забезпечено 2D та 3D-відображення розподілу температури й рівня зерна, доступ до детальної інформації по термопідвісках та датчиках, перегляд графіків часових рядів. Окремий інструмент інтерактивного редагування дашбордів дозволяє користувачам із відповідними правами формувати власні набори віджетів, змінювати їх розташування та розміри, а також зберігати конфігурації дашбордів для різних ролей (оператор, технолог, менеджер). Для індивідуальних температурних налаштувань реалізовано інтерфейс, у якому задаються порогові значення температури та її динаміки для різних культур, класів зерна та окремих силосів, підвісок і датчиків.

Модуль формування звітів забезпечує фіксацію стану силосів на обраний момент часу або за період. Реалізовано матричні звіти по термопідвісках і датчиках із кольоровою індикацією стану, а також журнал сформованих звітів із можливістю повторного перегляду, експорту й друку. Модуль оповіщень та керування подіями перетворює зміни індексів і станів датчиків на структуровані сповіщення з розмежуванням за рівнями, фіксацією часу, джерела та користувацьких статусів. Завдяки інтеграції модуля оповіщень з інтерфейсами рівня «елеватор» і «силос» користувач може швидко перейти від текстового сповіщення до просторової візуалізації проблемної зони.

У роботі виконано економічне обґрунтування доцільності впровадження розробленої системи. На основі вихідних даних щодо вартості розробки, експлуатації, обсягів зберігання та орієнтовних втрат від псування зерна оцінено економічний ефект від впровадження. Розрахунки показали, що орієнтовні витрати на розробку та впровадження системи становлять до 80 тис. грн, тоді як абсолютний економічний ефект від зменшення втрат та оптимізації режимів зберігання може досягати 2558 тис. грн за три роки експлуатації. Внутрішня норма дохідності інвестицій у впровадження системи становить не менше 48 %, а розрахунковий термін окупності становить лише 1,6 року. Такі числові результати

свідчать про фінансову доцільність проєкту для типового підприємства зернової галузі та підтверджують перспективність впровадження системи в промислову експлуатацію.

Практичні результати мають значне прикладне значення для елеваторних підприємств і холдингів. Система дозволяє зменшити ризики саморозігрівання та псування зерна завдяки більш ранньому виявленню перегрівів і аномалій, підвищити прозорість і керованість процесів зберігання, знизити навантаження на персонал, який раніше був вимушений покладатися переважно на ручний контроль та несистематизовані джерела інформації. Створене рішення формує єдине інформаційне середовище для моніторингу, аналізу та прийняття рішень, що є важливим кроком до цифрової трансформації елеваторної інфраструктури.

У роботі поставлену задачу розроблення й дослідження автоматизованої системи моніторингу умов зберігання зерна в силосах з багаторівневою візуалізацією, модулем звітності та оповіщень, а також підтримкою мультихмарної інфраструктури вирішено. Сформульовано та реалізовано алгоритми аналізу температурних режимів і рівня заповнення, спроєктовано структуру бази даних, реалізовано серверні сервіси й веб-інтерфейс, а також виконано кількісну оцінку економічної ефективності. Отримані результати підтверджують досягнення мети магістерської роботи й виконання всіх основних завдань дослідження.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. An Overview of Modern Grain Farming URL: https://modernfarmers.ca/an-overview-of-modern-grain-farming/ (дата звернення 18.06.2025).

2. IoT-Based Real-Time Crop Drying and Storage Monitoring System URL: https://onlinelibrary.wiley.com/doi/10.1155/2023/4803000(дата звернення 18.06.2025).

3. Long-term Grain Storage Requires Good Management URL: https://cropwatch.unl.edu/2017/long-term-grain-storage-requires-good-management/ (дата звернення 18.06.2025).

4. Seeking end to loss and waste of food along production chain URL: https://www.fao.org/in-action/seeking-end-to-loss-and-waste-of-food-along-production-chain/en/ (дата звернення 18.06.2025).

5. Multi-Cloud Architecture Guide URL: https://www.backblaze.com/blog/multi-cloud-strategy-architecture-guide/ (дата звернення 18.06.2025).

6. HARDWARIO and GrainLink: Revolutionizing Grain Storage with IoT Technology URL: https://www.hardwario.com/news/hardwario-and-grainlink-revolutionizing-grain-storage-with-iot-technology (дата звернення 18.06.2025).

7. Why Grain Storage Automation? URL:https://opisystems.com/why-grain-storage-automation/ (дата звернення 18.06.2025).

8. A review of three-dimensional vision techniques in food and agriculture applications URL: https://www.sciencedirect.com/science/article/pii/S2772375523000898 (дата звернення 18.06.2025).

9. Visualizing IoT Data with Web App: Enhancing Real-Time Analytics URL: https://thinglabs.io/visualizing-iot-data-with-web-app (дата звернення 18.06.2025).

10. Методи оптимізації react reconciliation алгоритму для підвищення ефективності клієнтської частини веб-додатків / Фоучек В.О., Богач І.В. // Матеріали LIV Всеукраїнській науково-технічній конференції факультету

інтелектуальних інформаційних технологій та автоматизації (2025), Вінниця, 2025. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24513 (дата звернення 18.06.2025).

11. Grain Storage URL: https://www.fao.org/4/s1250e/S1250E0w.htm

12. Equilibrium Moisture Content Charts for Grain Storage Management URL: https://pami.ca/wp-content/uploads/2021/10/Equilibrium-Moisture-Content-Charts-for-Grain-Storage-Management_rev2.pdf (дата звернення 18.06.2025

13. Temperature Monitoring URL: https://entomology.k-state.edu/doc/finished-chapters/s156-ch-23-temperature-monitoring.pdf (дата звернення 18.06.2025)

14. Departments Of Biological And Agricultural Engineering And Entomology – Storing Wheat URL: https://bookstore.ksre.ksu.edu/pubs/storing-wheat_MF855.pdf (дата звернення 18.06.2025)

15. Effect of Temperature Sensor Numbers and Placement on Aeration Cooling of a Stored Grain Mass Using a 3D Finite Element Model URL: https://www.mdpi.com/2077-0472/11/3/231 (дата звернення 18.06.2025)

16. Programmable Resolution 1-Wire Digital Thermometer URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf (дата звернення 18.06.2025)

17. Top 10 Grain Storage Management Tips Grain Storage - Joe Harner URL: https://entomology.k-state.edu/extension/stored-product-pests/stored-grain/topten.html (дата звернення 18.06.2025)

18. MQTT Version 5.0 URL: https://www.oasis-open.org/standard/mqtt-v5-0-os/ (дата звернення 18.06.2025)

19. How to Define Zones and Conduits URL: https://gca.isa.org/blog/how-to-define-zones-and-conduits (дата звернення 18.06.2025)

20. Revolutionizing Smart Storage URL: https://opisystems.com/ (дата звернення 18.06.2025)

21. Державне науково - виробниче підприємство «Ельдорадо» URL: http://dnvpeldorado.com/index.php/ua/ (дата звернення 18.06.2025)

22. Controlling the Temperature and Humidity of Grain, Stored in Silos URL: https://tmsgrup.com/article/controlling-the-temperature-and-humidity-of-grain-stored-in-silos/ (дата звернення 25.10.2025)

23. Temperature Humidity Index: what you need to know about it URL: https://www.pericoli.com/en/temperature-humidity-index-what-you-need-to-know-about-it/ (дата звернення 25.10.2025)

24. Research on Grain Temperature Detection Based on Rational Sound-Source Signal URL: https://www.mdpi.com/2077-0472/15/10/1035 (дата звернення 25.10.2025) (дата звернення 25.10.2025)

25. Best Practices for Scaling RabbitMQ URL: https://scalegrid.io/blog/scaling-rabbitmq/ (дата звернення 25.10.2025)

26. Time series visualization and dashboards URL: https://grafana.com/docs/grafana/latest/visualizations/panels-visualizations/visualizations/time-series/ (дата звернення 25.10.2025)

27. Kong Inc. Microservices and Multi-Cloud Building Cross-Cloud Harmony (дата звернення 25.10.2025)

28. Mastering React Monitoring - Best Practices and Tools URL: https://signoz.io/guides/react-monitoring/ (дата звернення 25.10.2025)

29. Mastering React Toast Notifications: A Comprehensive Guide URL: https://blog.stackademic.com/mastering-react-toast-notifications-a-comprehensive-guide-92f5bb2c5179

# ДОДАТКИ

**Додаток А (обов'язковий)**

**Технічне завдання**

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«17» жовтня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
**«АВТОМАТИЗАЦІЯ МОНІТОРИНГУ УМОВ ЗБЕРІГАННЯ ЗЕРНОВИХ КУЛЬТУР У МУЛЬТИХМАРНІЙ ІНФРАСТРУКТУРІ»**
08-31.МКР.013.02.000 ТЗ

Керівник роботи
к.т.н., доц. каф. АІІТ
Ілона БОГАЧ
«16» жовтня 2025 р.

Виконавець:
ст. гр. 1АКІТР-24М
Володимир ФОУЧЕК
«16» жовтня 2025 р.

Вінниця ВНТУ 2025

## 1. Назва та галузь застосування

Магістерська кваліфікаційна робота: «Автоматизація моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі». Галузь застосування – агропромисловий комплекс.

## 2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету № 313 від 24 вересня 2025 року та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій»

## 3. Мета та призначення розробки

Метою роботи є розроблення програмного забезпечення для автоматизованої системи моніторингу стану зернових культур у силосах з підтримкою мультихмарної інфраструктури, яке забезпечить:

- безперервний автоматичний збір телеметрії від контролерів (температура, вологість, рівень заповнення, $CO_2$ тощо) з використанням промислових протоколів та брокера повідомлень.
- надійне зберігання, попередню обробку та агрегування великих обсягів вимірювальних даних у реляційній базі даних з можливістю масштабування та реплікації у кількох хмарних середовищах.
- реалізацію алгоритмів аналізу температурних режимів та рівня заповнення силосів (індекси температури та її динаміки, оцінка рівня та маси зерна на основі геометрії силоса та насипної густини).
- інтерактивну 2D/3D-візуалізацію стану силосів, термопідвісок і датчиків, а також формування звітів та оповіщень про небезпечні відхилення параметрів у режимі, наближеному до реального часу

Призначення системи: забезпечення ефективного, наочного та надійного моніторингу умов зберігання зерна на елеваторах, зниження ризиків псування продукції за рахунок своєчасного виявлення перегрівів і аномалій, підтримка

прийняття технологічних рішень (аерація, сушіння, перерозподіл маси), а також можливість розгортання та масштабування рішення в мультихмарному середовищі холдингу.

## 4. Джерела розробки

1. 3D Data Visualization with React and Three.js URL: https://medium.com/cortico/3d-data-visualization-with-react-and-three-js-7272fb6de432 (дата звернення 18.06.2025)
2. Clean Architecture in React URL: https://alexkondov.com/full-stack-tao-clean-architecture-react/ (дата звернення: 18.06.2025)
3. Grafana React Components URL: https://developers.grafana.com/ui/latest/index.html (дата звернення: 18.06.2025)

## 5. Показники призначення

### 5.1 Основні технічні характеристики системи

**Функціональні можливості:**

- Модуль збору та доставки даних: Прийом телеметрії від контролерів термометрії та інших вимірювальних пристроїв; формування повідомлень з вимірюваннями; передача даних через брокер повідомлень (RabbitMQ або аналог) до серверної частини за протоколом AMQP; базова валідація структури пакета.
- Модуль зберігання та попередньої обробки даних: Збереження вимірювань у реляційній СУБД (PostgreSQL); індексування та розділення таблиць за часом; кешування найчастіше використовуваних даних; фільтрація шумів та аномалій; агрегування до хвилинних/годинних/добових значень для аналітики та побудови графіків.
- Модуль алгоритмічного аналізу температури та рівня заповнення: Обчислення абсолютних температурних індексів, індексів динаміки температури, критеріїв виявлення аномалій; визначення локального рівня заповнення по кожній термопідвісці та узагальненого рівня по силосу; орієнтовний розрахунок об’єму та маси зерна з урахуванням геометрії силоса

й насипної густини культури.

- Модуль індивідуальних температурних налаштувань: Конфігурування порогових значень температури та її динаміки за культурами й класами зерна; налаштування локальних порогів для окремих силосів, термопідвісок і датчиків; збереження конфігурацій у базі даних та їх застосування в алгоритмах аналізу.
- Модуль візуалізації та дашбордів: Відображення стану елеватора на оглядовому екрані; 2D-та 3D-візуалізація силоса з проєкцією датчиків; графіки часових рядів температури та індексів; інструмент інтерактивного редагування дашбордів.
- Модуль оповіщень та керування подіями: Генерація сповіщень при переході датчика/силоса в попереджувальний або аварійний стан, при критичній динаміці температури або відновленні норми; віджет сповіщень і журнал історії подій; можливість інтеграції з зовнішніми каналами (Email/SMS/месенджери).
- Модуль формування звітів: Створення звітів по температурі та стану силосів на заданий момент часу або за період; матричні звіти по термопідвісках і датчиках; збереження параметрів сформованих звітів; перегляд, друк та експорт у поширені формати.
- Модуль аутентифікації, авторизації та керування доступом: Керування обліковими записами користувачів; розмежування прав доступу за ролями (адміністратор, технолог, оператор, менеджер холдингу); прив’язка користувачів до конкретних підприємств/елеваторів; аудит дій користувачів.
- Модуль адаптації до мультихмарної інфраструктури: Підтримка розгортання серверних компонентів у кількох хмарних середовищах; реплікація бази даних; використання брокера повідомлень та сервісів зберігання, розгорнутих у різних хмарах.

## 5.2Мінімальні системні вимоги

**Апаратне забезпечення:**

- Процесор: 4-ядерний CPU з тактовою частотою ≥ 2.4 GHz.

- Оперативна пам’ять: не менше 8 ГБ.
- Місце на диску: від 50 ГБ для системного ПЗ, СУБД, журналів і історичних даних (з можливістю подальшого розширення або винесення архіву на окремі диски/сховища).
- Мережа: стабільне інтернет-з’єднання з пропускною здатністю ≥ 10-20 Mbps для обміну з хмарними сервісами та клієнтами веб-інтерфейсу.

**Програмне забезпечення:**

- Операційна система: Linux (Ubuntu 20.04+), Windows 10/11 або macOS 10.15+;
- Платформа для серверної логіки: Java 17+;
- СУБД: PostgreSQL 13+;
- Брокер повідомлень: RabbitMQ 3.9+ або сумісний;
- Засіб управління міграціями БД: Liquibase або аналог;
- Опціонально: Redis або інше in-memory сховище для кешування; Grafana чи інший інструмент для побудови часових графіків.

**Клієнтська частина:**

- Сучасний веб-браузер (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) з підтримкою HTML5, CSS3, JavaScript ES6+;
- Роздільна здатність екрана не нижче 1280×720 (рекомендовано 1920×1080 для зручної роботи з 2D/3D-візуалізацією та дашбордами).

## 5.3 Вхідні дані

- Телеметричні дані від вимірювальних пристроїв: Пакети вимірювань від контролерів термометрії та інших датчиків (температура, вологість, рівень, $CO_2$ тощо), що надходять через брокер повідомлень у внутрішньому форматі (JSON/бінарні структури).
- Конфігураційні дані структури елеватора: Опис підприємств, силосів, термопідвісок, датчиків: назви та типи силосів, геометричні параметри (висота, радіус, тип днища), координати розташування на схемі; структура підвісок (кільця, кількість підвісок, висоти розташування датчиків); типи датчиків,

робочі діапазони, заводські номери.

- Довідники культур та температурних режимів: Перелік культур і їхніх класів; порогові температури та пороги динаміки .
- Параметри користувачів та прав доступу: Облікові записи користувачів (ПІБ, логін, контактні дані, роль, прив'язка до підприємства/холдингу); налаштування дашбордів, списки улюблених силосів, індивідуальні фільтри.

**5.4 Результати роботи програми**

- Безперервний моніторинг та збереження даних: Автоматичне приймання та запис телеметрії від усіх підключених контролерів у базу даних з дискретністю, заданою конфігурацією (цільовий інтервал оновлення дашборду – до 1 секунди при нормальному навантаженні).
- Розрахунок індексів та оцінка стану: абсолютних температурних індексів; індексів динаміки температури; оціненого рівня заповнення; орієнтовної маси зерна; присвоєння станів згідно з налаштованими порогами.
- Візуальне подання стану елеватора: Формування інтерактивних 2D/3D-представлень силосів з кольоровою індикацією температурних полів та рівня заповнення; відображення часових графіків по вибраних датчиках; побудова дашбордів з набором віджетів для різних ролей користувачів.
- Автоматизовані сповіщення та історія подій: Генерація та реєстрація сповіщень при виході параметрів за допустимі межі або при аномальній динаміці; відображення поточних сповіщень у віджеті; ведення історії подій за обрані періоди з можливістю фільтрації, експорту та друку.
- Формування звітів: Створення звітів про температурний стан силосів за заданий момент часу/період; матричні звіти по термопідвісках та датчиках; збереження параметрів звітів, повторний перегляд та експорт у файл.

## 6. Економічні призначення

До економічних показників входять:

- витрати на розробку – не більше 80 тис. грн.
- абсолютний ефект від впровадження розробки – не менше 2000 тис. грн за три роки

- внутрішня дохідність інвестицій – не менше 48 %
- термін окупності – до 3-х років

## 7. Стадії розробки

1. Розділ 1 «Аналіз предметної галузі» має бути виконаний до 15.10.2025 р.
2. Розділ 2 «Проектування архітектури та огляд технологій розробки» має бути виконаний до 30.10.2025 р.
3. Розділ 3 «Реалізація автоматизованої системи моніторингу» має бути виконаний до 20.11.2025 р.
4. Розділ 4 «Економічний розділ» має бути виконаний до 01.12.2025 р.

## 8. Порядок контролю та приймання

1. Рубіжний контроль провести до 14.11.2025.
2. Попередній захист магістерської кваліфікаційної роботи провести до 02.12.2025.
3. Захист магістерської кваліфікаційної роботи провести в період з 15.12.2025 до 19.12.2025.

**Додаток Б**

**(обов’язковий)**

# ІЛЮСТРАТИВНА ЧАСТИНА

АВТОМАТИЗАЦІЯ МОНІТОРИНГУ УМОВ ЗБЕРІГАННЯ ЗЕРНОВИХ КУЛЬТУР У МУЛЬТИХМАРНІЙ ІНФРАСТРУКТУРІ

# Діаграма сутностей та зв’язків


Рисунок Б.1 – Діаграма сутностей та зв’язків

**Архітектура клієнт-серверної системи збору, оброблення й візуалізації телеметрії**

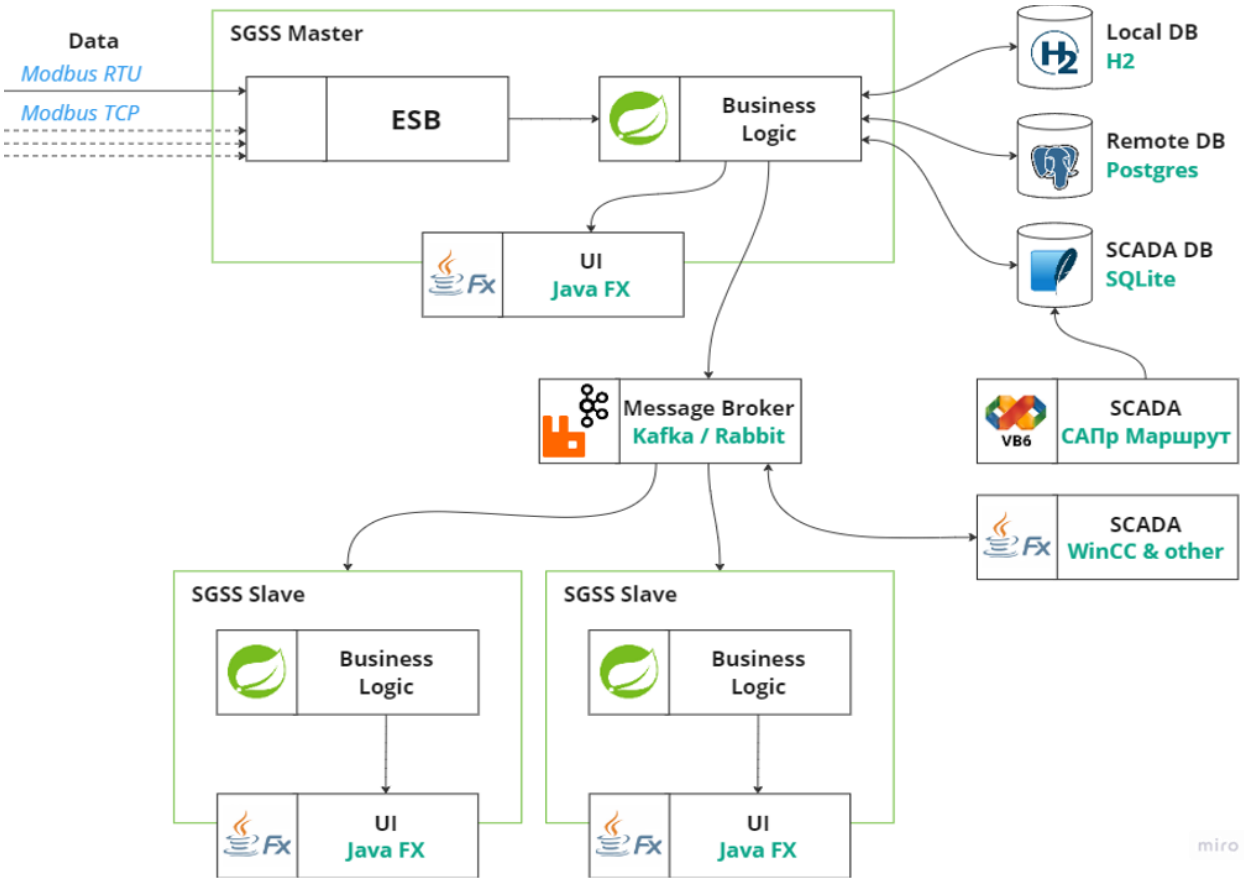


Рисунок Б.2 – Діаграма клієнт-серверного системного збору, оброблення й візуалізації телеметрії

**Ієрархія прав доступу в системі**

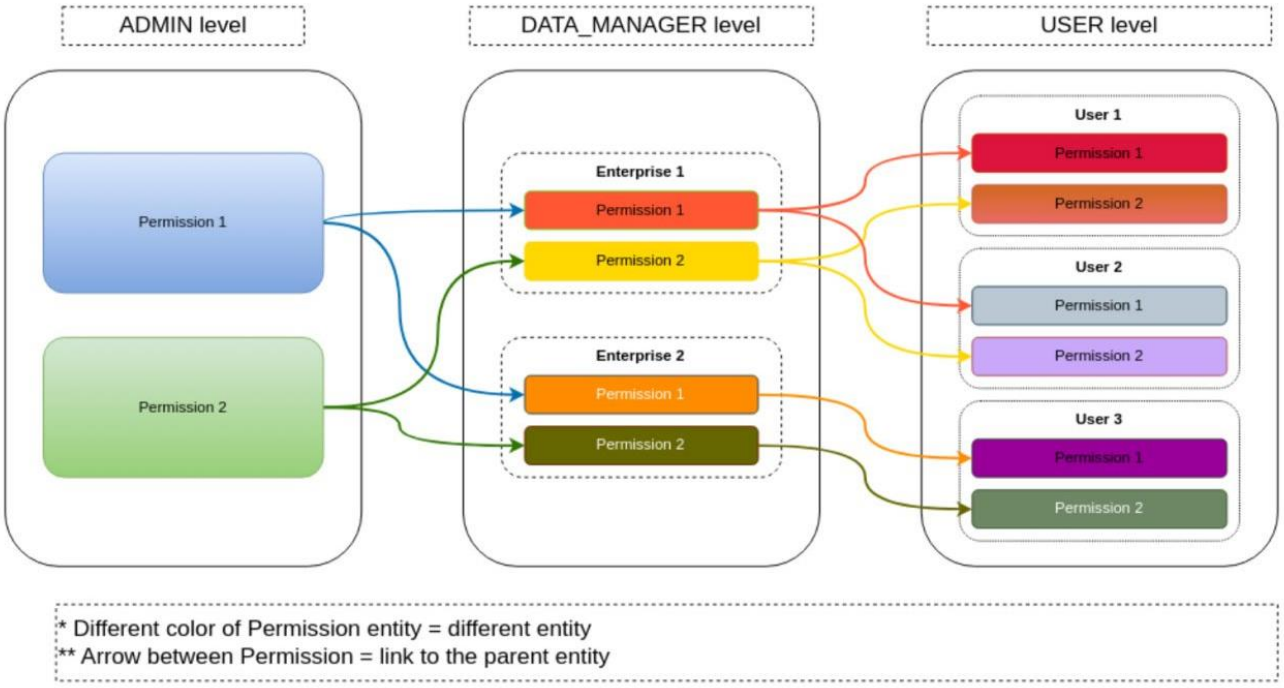


Рисунок Б.3 – Діаграма прав доступу в системі

**Ієрархія розмежування доступу: холдинг – підприємство – посади користувачів**

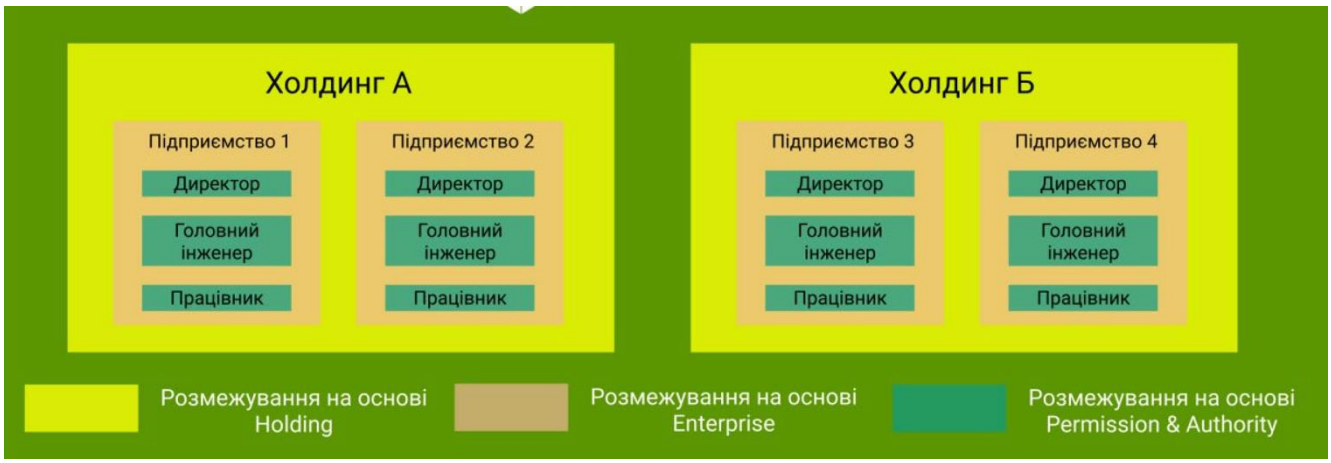


Рисунок Б.4 – Ієрархія розмежування доступу: холдинг – підприємство – посади користувачів

**Інтерфейс графічного налаштування термопідвісок у силосі**

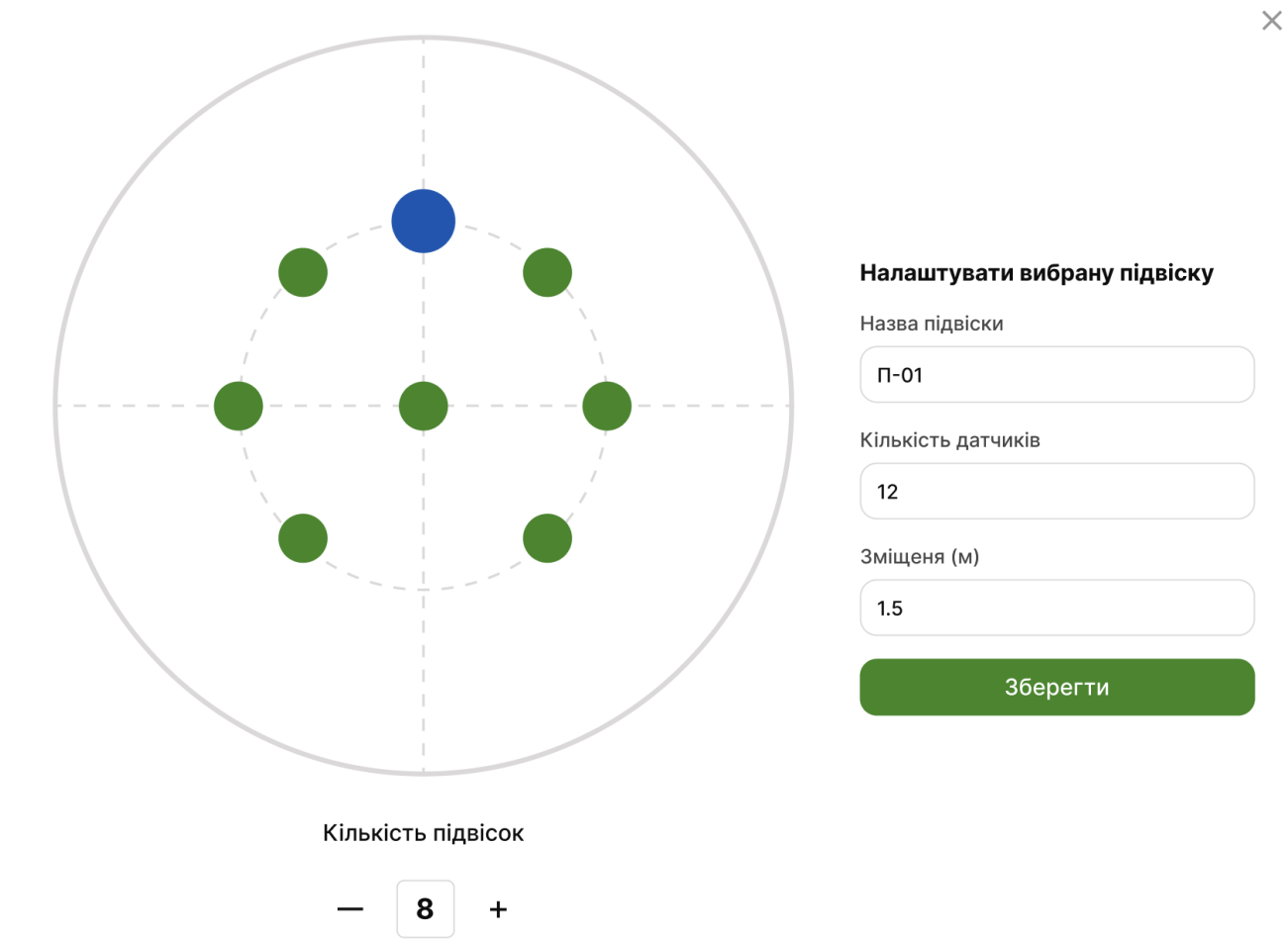


Рисунок Б.5 – Інтерфейс графічного налаштування термопідвісок у силосі

**Супутникове зображення майданчика з силосами**

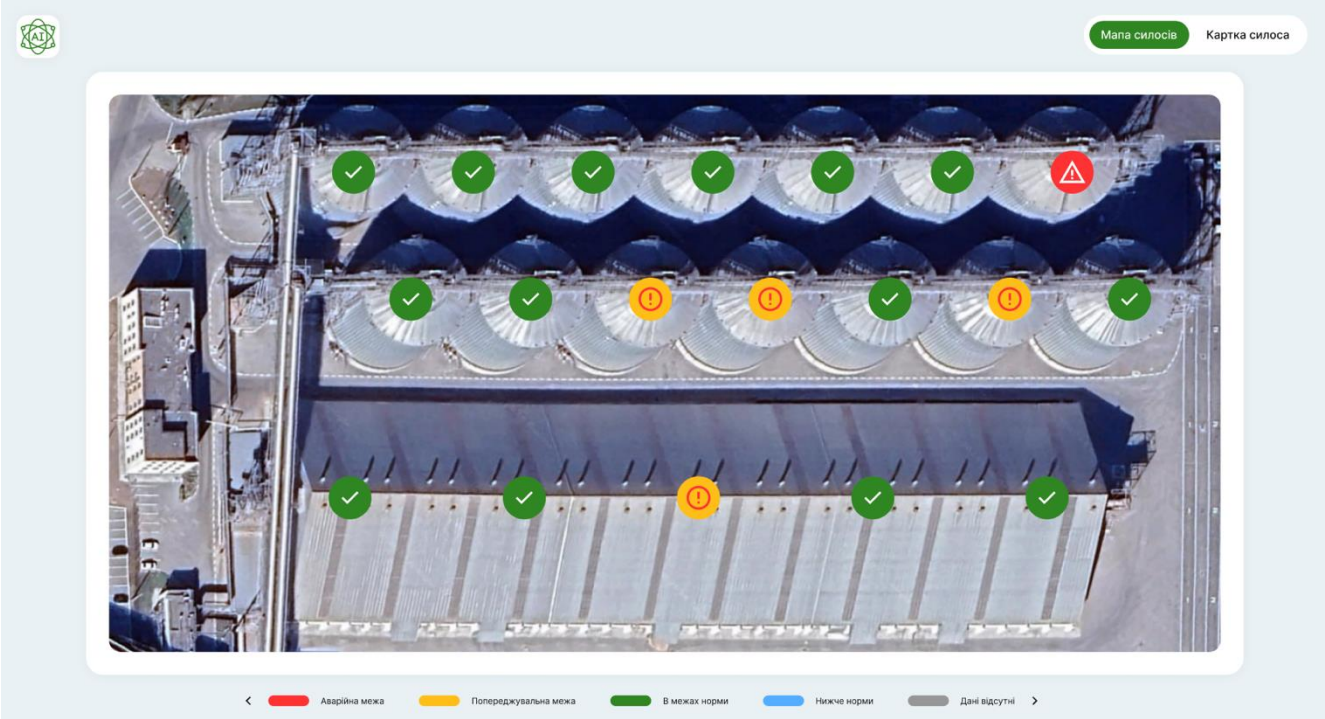


Рисунок Б.6 – Супутникове зображення майданчика з силосами

Додаток В (обов’язковий)

Лістинг основних функцій програми

```
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import java.io.Serializable;
import java.time.Instant;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.*;
import javax.persistence.Entity;
import javax.persistence.Table;

import org.hibernate.annotations.*;
import org.hibernate.annotations.Cache;
import org.springframework.data.annotation.CreatedBy;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedBy;
import org.springframework.data.annotation.LastModifiedDate;

import static ms.gtcs.config.Constants.*;

/**
 * A GrainBin.
 */
@Entity
@Table(name = "grain_bin")
@SQLDelete(sql = GRAIN_BIN_DELETE_STATEMENT)
@FilterDef(name = GRAIN_BIN_DELETED_FILTER_NAME, parameters =
@ParamDef(name = DELETE_PARAMETER_DEFINITION, type =
DELETE_PARAMETER_TYPE))
@Filter(name = GRAIN_BIN_DELETED_FILTER_NAME, condition =
DELETE_FILTER_CONDITION)
@Cache(usage = CacheConcurrencyStrategy.NONE)
@SuppressWarnings("common-java:DuplicatedBlocks")
public class GrainBin extends AbstractAuditingEntity<Long> implements Serializable
{

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"grainBinSG")
    @SequenceGenerator(name = "grainBinSG", allocationSize = 1)
    @Column(name = "id")
    private Long id;
```

```
    @Column(name = "name")
    private String name; // example: `S1` (from `DBAVl_export_S1_P2_3`)

    @Column(name = "display_name")
    private String displayName;

    @Enumerated(EnumType.STRING)
    @Column(name = "bin_type")
    private GrainBinType binType;

    @Column(name = "enabled_calculation")
    private Boolean enabledCalculation;

    @Column(name = "enabled_ui")
    private Boolean enabledUI;

    @Column(name = "x_position")
    private Double xPosition;

    @Column(name = "y_position")
    private Double yPosition;

    @Column(name = "radius")
    private Double radius;

    @Column(name = "container_width")
    private Double containerWidth;

    @Column(name = "container_height")
    private Double containerHeight;

    @Column(name = "annotations")
    private String annotations;

    @OneToMany(mappedBy = "grainBin")
    @Cache(usage = CacheConcurrencyStrategy.READ_WRITE)
    @JsonIgnoreProperties(value = { "sensors", "stateHistories", "notifications",
"measurementLimit", "grainBin" }, allowSetters = true)
    private Set<ThermalSuspension> thermalSuspensions = new HashSet<>();

    @OneToMany(mappedBy = "grainBin")
    @Cache(usage = CacheConcurrencyStrategy.READ_WRITE)
    @JsonIgnoreProperties(value = { "sensor", "thermalSuspension", "grainBin" },
allowSetters = true)
```

```
    private Set<StateHistory> stateHistories = new HashSet<>();

    @OneToMany(mappedBy = "grainBin")
    @Cache(usage = CacheConcurrencyStrategy.READ_WRITE)
    @JsonIgnoreProperties(
        value = {
            "user",
            "measurementLimit",
            "measurementSetting",
            "sensor",
            "thermalSuspension",
            "grainBin",
            "grainCropsType",
            "temperatureColour",
        },
        allowSetters = true
    )
    private Set<Notification> notifications = new HashSet<>();

    @JsonIgnoreProperties(value = { "grainType", "thermalSuspension", "grainBin",
"notifications" }, allowSetters = true)
    @OneToOne(mappedBy = "grainBin")
    private MeasurementLimit measurementLimit;

    @ManyToOne
    @JsonIgnoreProperties(
        value = { "grainBins", "colorsTemperatures", "notifications", "measurementLimit",
"sortType", "classType" },
        allowSetters = true
    )
    private GrainCropsType grainType;

    // jhipster-needle-entity-add-field - JHipster will add fields here

    public Long getId() {
        return this.id;
    }

    public GrainBin id(Long id) {
        this.setId(id);
        return this;
    }

    public void setId(Long id) {
        this.id = id;
```

```
    }

    public String getName() {
        return this.name;
    }

    public Set<ThermalSuspension> getThermalSuspensions() {
        return this.thermalSuspensions;
    }

    public void setThermalSuspensions(Set<ThermalSuspension> thermalSuspensions) {
        if (this.thermalSuspensions != null) {
            this.thermalSuspensions.forEach(i -> i.setGrainBin(null));
        }
        if (thermalSuspensions != null) {
            thermalSuspensions.forEach(i -> i.setGrainBin(this));
        }
        this.thermalSuspensions = thermalSuspensions;
    }

    public GrainBin thermalSuspensions(Set<ThermalSuspension> thermalSuspensions)
{
        this.setThermalSuspensions(thermalSuspensions);
        return this;
    }

    public GrainBin addThermalSuspensions(ThermalSuspension thermalSuspension) {
        this.thermalSuspensions.add(thermalSuspension);
        thermalSuspension.setGrainBin(this);
        return this;
    }

    public GrainBin removeThermalSuspensions(ThermalSuspension
thermalSuspension) {
        this.thermalSuspensions.remove(thermalSuspension);
        thermalSuspension.setGrainBin(null);
        return this;
    }

    public Set<StateHistory> getStateHistories() {
        return this.stateHistories;
    }

    public void setStateHistories(Set<StateHistory> stateHistories) {
        if (this.stateHistories != null) {
```

```
        this.stateHistories.forEach(i -> i.setGrainBin(null));
    }
    if (stateHistories != null) {
        stateHistories.forEach(i -> i.setGrainBin(this));
    }
    this.stateHistories = stateHistories;
}

public GrainBin stateHistories(Set<StateHistory> stateHistories) {
    this.setStateHistories(stateHistories);
    return this;
}

public GrainBin addStateHistory(StateHistory stateHistory) {
    this.stateHistories.add(stateHistory);
    stateHistory.setGrainBin(this);
    return this;
}

public GrainBin removeStateHistory(StateHistory stateHistory) {
    this.stateHistories.remove(stateHistory);
    stateHistory.setGrainBin(null);
    return this;
}

public Set<Notification> getNotifications() {
    return this.notifications;
}

public void setNotifications(Set<Notification> notifications) {
    if (this.notifications != null) {
        this.notifications.forEach(i -> i.setGrainBin(null));
    }
    if (notifications != null) {
        notifications.forEach(i -> i.setGrainBin(this));
    }
    this.notifications = notifications;
}

public GrainBin notifications(Set<Notification> notifications) {
    this.setNotifications(notifications);
    return this;
}

public GrainBin addNotifications(Notification notification) {
```

```
        this.notifications.add(notification);
        notification.setGrainBin(this);
        return this;
    }

    public GrainBin removeNotifications(Notification notification) {
        this.notifications.remove(notification);
        notification.setGrainBin(null);
        return this;
    }

    public MeasurementLimit getMeasurementLimit() {
        return this.measurementLimit;
    }

    public void setMeasurementLimit(MeasurementLimit measurementLimit) {
        if (this.measurementLimit != null) {
            this.measurementLimit.setGrainBin(null);
        }
        if (measurementLimit != null) {
            measurementLimit.setGrainBin(this);
        }
        this.measurementLimit = measurementLimit;
    }

    public GrainBin measurementLimit(MeasurementLimit measurementLimit) {
        this.setMeasurementLimit(measurementLimit);
        return this;
    }

    public GrainCropsType getGrainType() {
        return this.grainType;
    }

    public void setGrainType(GrainCropsType grainCropsType) {
        this.grainType = grainCropsType;
    }

    public GrainBin grainType(GrainCropsType grainCropsType) {
        this.setGrainType(grainCropsType);
        return this;
    }
/**
 * REST controller for managing
 */
```

```
@RestController
@RequestMapping("/api")
public class GrainBinResource {

    private final Logger log = LoggerFactory.getLogger(GrainBinResource.class);

    private static final String ENTITY_NAME = "grainBin";

    @Value("${jhipster.clientApp.name}")
    private String applicationName;

    private final GrainBinService grainBinService;

    private final GrainBinRepository grainBinRepository;

    public GrainBinResource(GrainBinService grainBinService, GrainBinRepository
grainBinRepository) {
        this.grainBinService = grainBinService;
        this.grainBinRepository = grainBinRepository;
    }

    /**
     * {@code POST  /grain-bins} : Create a new grainBin.
     *
     * @param grainBinDTO the grainBinDTO to create.
     * @return the {@link ResponseEntity} with status {@code 201 (Created)} and with
body the new grainBinDTO, or with status {@code 400 (Bad Request)} if the grainBin
has already an ID.
     * @throws URISyntaxException if the Location URI syntax is incorrect.
     */
    @PostMapping("/grain-bins")
    public ResponseEntity<GrainBinDTO> createGrainBin(@RequestBody
GrainBinDTO grainBinDTO) throws URISyntaxException {
        log.debug("REST request to save GrainBin : {}", grainBinDTO);
        if (grainBinDTO.getId() != null) {
            throw new BadRequestAlertException("A new grainBin cannot already have an
ID", ENTITY_NAME, "idexists");
        }
        GrainBinDTO result = grainBinService.update(grainBinDTO);
        return ResponseEntity
            .created(new URI("/api/grain-bins/" + result.getId()))
            .headers(HeaderUtil.createEntityCreationAlert(applicationName, true,
ENTITY_NAME, result.getId().toString()))
            .body(result);
    }
```

```
    /**
     * {@code POST  /grain-bins/chart-data} : Load chart data for top 3 sensors of a grain bin.
     *
     * @param request containing grainBinDTO, startTimestamp, and endTimestamp
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with body the chart data,
     * or with status {@code 400 (Bad Request)} if the request is not valid.
     */
    @PostMapping("/grain-bins/chart-data")
    public ResponseEntity<List<ChartDataDTO>> loadChartData(@RequestBody ChartDataRequestDto request) {
        log.debug("REST request to load chart data for GrainBin: {} from {} to {} with time period: {}",
            request.getGrainBinDTO().getName(), request.getStartTimestamp(), request.getEndTimestamp(), request.getTimePeriod());

        try {
            List<ChartDataDTO> chartData = grainBinService.loadChartDataForTopThreeSensors(
                request.getGrainBinDTO(),
                request.getStartTimestamp(),
                request.getEndTimestamp(),
                request.getTimePeriod()
            );
            return ResponseEntity.ok(chartData);
        } catch (IllegalArgumentException e) {
            log.error("Invalid request for chart data: {}", e.getMessage());
            throw new BadRequestAlertException(e.getMessage(), ENTITY_NAME, "invalidrequest");
        }
    }

    /**
     * {@code PUT  /grain-bins/:id} : Updates an existing grainBin.
     *
     * @param id the id of the grainBinDTO to save.
     * @param grainBinDTO the grainBinDTO to update.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with body the updated grainBinDTO,
     * or with status {@code 400 (Bad Request)} if the grainBinDTO is not valid,
     * or with status {@code 500 (Internal Server Error)} if the grainBinDTO couldn't be updated.
     * @throws URISyntaxException if the Location URI syntax is incorrect.
```

```
    */
   @PutMapping("/grain-bins/{id}")
   public ResponseEntity<GrainBinDTO> updateGrainBin(
      @PathVariable(value = "id", required = false) final Long id,
      @RequestBody GrainBinDTO grainBinDTO
   ) throws URISyntaxException {
      log.debug("REST request to update GrainBin : {}, {}", id, grainBinDTO);
      if (grainBinDTO.getId() == null) {
         throw new BadRequestAlertException("Invalid id", ENTITY_NAME, "idnull");
      }
      if (!Objects.equals(id, grainBinDTO.getId())) {
         throw new BadRequestAlertException("Invalid ID", ENTITY_NAME,
"idinvalid");
      }

      if (!grainBinRepository.existsById(id)) {
         throw new BadRequestAlertException("Entity not found", ENTITY_NAME,
"idnotfound");
      }

      GrainBinDTO result = grainBinService.update(grainBinDTO);
      return ResponseEntity
         .ok()
         .headers(HeaderUtil.createEntityUpdateAlert(applicationName, true,
ENTITY_NAME, grainBinDTO.getId().toString()))
         .body(result);
   }

   /**
    * {@code PATCH  /grain-bins/:id} : Partial updates given fields of an existing
grainBin, field will ignore if it is null
    *
    * @param id the id of the grainBinDTO to save.
    * @param grainBinDTO the grainBinDTO to update.
    * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body the updated grainBinDTO,
    * or with status {@code 400 (Bad Request)} if the grainBinDTO is not valid,
    * or with status {@code 404 (Not Found)} if the grainBinDTO is not found,
    * or with status {@code 500 (Internal Server Error)} if the grainBinDTO couldn't be
updated.
    * @throws URISyntaxException if the Location URI syntax is incorrect.
    */
   @PatchMapping(value = "/grain-bins/{id}", consumes = { "application/json",
"application/merge-patch+json" })
   public ResponseEntity<GrainBinDTO> partialUpdateGrainBin(
```

```
        @PathVariable(value = "id", required = false) final Long id,
        @RequestBody GrainBinDTO grainBinDTO
    ) throws URISyntaxException {
        log.debug("REST request to partial update GrainBin partially : {}, {}", id,
grainBinDTO);
        if (grainBinDTO.getId() == null) {
            throw new BadRequestAlertException("Invalid id", ENTITY_NAME, "idnull");
        }
        if (!Objects.equals(id, grainBinDTO.getId())) {
            throw new BadRequestAlertException("Invalid ID", ENTITY_NAME,
"idinvalid");
        }

        if (!grainBinRepository.existsById(id)) {
            throw new BadRequestAlertException("Entity not found", ENTITY_NAME,
"idnotfound");
        }

        Optional<GrainBinDTO> result = grainBinService.partialUpdate(grainBinDTO);

        return ResponseUtil.wrapOrNotFound(
            result,
            HeaderUtil.createEntityUpdateAlert(applicationName, true, ENTITY_NAME,
grainBinDTO.getId().toString())
        );
    }

    /**
     * {@code GET  /grain-bins} : get all the grainBins.
     *
     * @param pageable the pagination information.
     * @param filter the filter of the request.
     * @param isDeleted the softdelete flag.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and the list of
grainBins in body.
     */
    @GetMapping("/grain-bins")
    public ResponseEntity<List<GrainBinDTO>> getAllGrainBins(
        @org.springdoc.api.annotations.ParameterObject Pageable pageable,
        @RequestParam(required = false) String filter,
        @RequestParam(defaultValue = "true") Boolean eager,
        @RequestParam(name = "isDeleted", defaultValue = "false") Boolean isDeleted
    ) {
        if ("measurementlimit-is-null".equals(filter)) {
```

```
            log.debug("REST request to get all GrainBins where measurementLimit is
null");
            return new
ResponseEntity<>(grainBinService.findAllWhereMeasurementLimitIsNull(),
HttpStatus.OK);
        }
        log.debug("REST request to get a page of GrainBins");
        Page<GrainBinDTO> page = grainBinService.findAll(eager, pageable, isDeleted);
        HttpHeaders headers =
PaginationUtil.generatePaginationHttpHeaders(ServletUriComponentsBuilder.fromCurr
entRequest(), page);
        return ResponseEntity.ok().headers(headers).body(page.getContent());
    }
    /**
     * {@code GET  /grain-bins/:id} : get the "id" grainBin.
     *
     * @param id the id of the grainBinDTO to retrieve.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body the grainBinDTO, or with status {@code 404 (Not Found)}.
     */
    @GetMapping("/grain-bins/{id}")
    public ResponseEntity<GrainBinDTO> getGrainBin(@PathVariable Long id) {
        log.debug("REST request to get GrainBin : {}", id);
        Optional<GrainBinDTO> grainBinDTO = grainBinService.findOne(id);
        return ResponseUtil.wrapOrNotFound(grainBinDTO);
    }
    /**
     * {@code DELETE  /grain-bins/:id} : delete the "id" grainBin.
     *
     * @param id the id of the grainBinDTO to delete.
     * @return the {@link ResponseEntity} with status {@code 204 (NO_CONTENT)}.
     */
    @DeleteMapping("/grain-bins/{id}")
    public ResponseEntity<Void> deleteGrainBin(@PathVariable Long id) {
        log.debug("REST request to delete GrainBin : {}", id);
        grainBinService.delete(id);
        return ResponseEntity
            .noContent()
            .headers(HeaderUtil.createEntityDeletionAlert(applicationName, true,
ENTITY_NAME, id.toString()))
            .build();
    }
}
```

Додаток Г (обов'язковий)

**ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ**

Назва роботи: «Автоматизація моніторингу умов зберігання зернових культур у мультихмарній інфраструктурі»

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра АІІТ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 0.57 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Бісікало О.В., зав. каф. АІІТ _____ (підпис)
(прізвище, ініціали, посада)

Овчинников К.В., доц. каф. АІІТ _____ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку _____ Маслій Р.В.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Богач І.В., доц. каф. АІІТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Фоучек В.О.
(підпис) (прізвище, ініціали)