Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматизації, комп'ютерно-інтегрованих технологій та систем
(повна назва)

Кафедра Комп'ютерно-інтегрованих, технологій, автоматизації, робототехніки та безпекової інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший(бакалаврський)
Розроблення мобільної системи моніторингу простоїв обладнання і контролю за виконанням завдань на виробництві
(тема)

Виконала:
здобувачка 4 року навчання,
групи АКТАКІТ-22-1
Олександра ДОРОФЄЄВА
(власне ім'я, прізвище)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва освітньої програми)

Керівник проф. Ірина СЕЗОНОВА
(посада, власне ім'я, прізвище)

Допускається до захисту
Завідувач кафедри _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2026 р.

Я, Дорофєєва Олександра Ігорівна, як здобувачка вищої освіти ХНУРЕ, розумію та підтримую політику закладу з академічної доброчесності. Я не надавала і не одержувала недозволеної допомоги під час підготовки кваліфікаційної роботи. Я не використовувала штучний інтелект під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів має містити посилання на відповідне джерело.

8 червня 2026 р.  Олександра ДОРОФЄЄВА

Харківський національний університет радіоелектроніки

Факультет Автоматизації, комп'ютерно-інтегрованих технологій та систем

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_____»_____ 2026 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачці Дорофєєвій Олександрі Ігорівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення мобільної системи моніторингу простоїв обладнання і контролю за виконанням завдань на виробництві
затверджена наказом університету від 04 06 2026 р. № 472 Ст
2. Термін подання здобувачкою роботи до екзаменаційної комісії 17 06 2026 р.
3. Вихідні дані до роботи
3.1 Середовища розробки: InelliJ IDEA, Android Studio
3.2 Мови програмування: Java для розробки API та Kotlin для розробки мобільної частини додатка
3.3 Реляційна база даних MySQL
3.4 Хмарні платформи: Google Gemeni Api, Google ML Kit
3.5 Середовище тестування Postman
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Аналіз предметної області
4.2 Проєктування мобільної інформаційної системи
4.3 Програмна реалізація серверної частини
4.4 Розробка мобільного Android-додатку
4.5 Тестування системи та аналіз результатів інтеграції
4.6 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Графічний демонстраційний матеріал у форматі powerpoint(*.ppt), 12 арк. ф. А4

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз технічного завдання	30.04.2026 – 03.05.2026	Виконано
2	Опрацювання технічної документації за темою	04.05.2026 – 06.05.2026	Виконано
3	Аналіз аналогів та обгрунтування стеку	07.05.2026 – 09.05.2026	Виконано
4	Проєктування архітектури системи та концептуальних моделей бази даних	10.05.2026 – 20.05.2026	Виконано
5	Розробка серверної частини	21.05.2026 – 28.05.2026	Виконано
6	Інтеграція штучного інтелекту	29.05.2026 – 30.05.2026	Виконано
7	Налаштування SMTP-клієнта	31.05.2026 – 03.06.2026	Виконано
8	Функціональне тестування запитів у середовищі Postman	04.06.2026 – 06.06.2026	Виконано
9	Оформлення пояснювальної записки	07.06.2026 – 15.06.2026	Виконано

Дата видачі завдання 27 04 2026 р.

Здобувач _____ Олександра ДОРОФЄЄВА
(підпис) (посада, власне ім'я, прізвище)

Керівник роботи _____ проф. Ірина СЕЗОНОВА
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра містить: 120 с., 6 табл., 8 рис., 2 дод., 23 джерела.

МОБІЛЬНА СИСТЕМА, МОНІТОРИНГ ПРОСТОЮ ОБЛАДНАННЯ, КОНТРОЛЬ ВИКОНАННЯ ЗАВДАНЬ, ШТУЧНИЙ ІНТЕЛЕКТ, OCR-СКАНУВАННЯ, JAVA, REST API, SPRING BOOT, MYSQL, GOOGLE GEMINI API, SMTP-CLIENT, GOOGLE ML KIT, ANDROID, KOTLIN, JETPACK COMPOSE.

Об'єкт розробки – процес моніторингу виробничих процесів, фіксації простоїв промислового обладнання та оперативного керування завданнями персоналу на підприємстві.

Предмет розробки – мобільна клієнт-серверна інформаційна система автоматизованого збору метрик ефективності, безконтактної ідентифікації обладнання та інтелектуального аналізу інцидентів.

Мета розробки – підвищення загальної ефективності використання виробничого обладнання шляхом розроблення і впровадження мобільно-серверного комплексу автоматизованого збору даних, безконтактного розпізнавання інвентарних номерів верстатів, контролю завдань операторів та генерації інтелектуальної ШІ-підказки інцидентів у реальному часі.

У результаті виконання кваліфікаційної роботи створено комплексну мобільну систему для моніторингу простоїв обладнання та контролю виконання завдань, яка складається з Android-додатка та REST API-сервера. Застосунок оптимізує рутинні процеси обліку роботи обладнання, надає інтелектуальний аналіз, спрощує комунікацію між виробництвом і керівництвом та мінімізує вплив людського фактора, забезпечуючи позитивний досвід користувачів.

ABSTRACT

The explanatory note to the bachelor's qualification thesis contains: 120 p., 6 tbl., 8 fig., 2 app., 23 sources.

MOBILE SYSTEM, EQUIPMENT DOWNTIME MONITORING, TASK CONTROL, ARTIFICIAL INTELLIGENCE, OCR SCANNING, JAVA, REST API, SPRING BOOT, MYSQL, GOOGLE GEMINI API, SMTP-CLIENT, GOOGLE ML KIT, ANDROID, KOTLIN, JETPACK COMPOSE.

The object of development – is the process of monitoring production processes, recording downtime of industrial equipment and operational management of personnel tasks at the enterprise.

Development subject – mobile client-server information system automated collection of performance metrics, contactless equipment identification and intelligent incident analysis.

The purpose of the development is to increase the overall efficiency of the use of production equipment by developing and implementing a mobile server complex of automated data collection, contactless recognition of machine inventory numbers, control of operator tasks and generation of intelligent AI prompts for incidents in real time.

As a result of the qualification work, a complex mobile system was developed for monitoring equipment downtime and task performance, consisting of an Android application and a REST API server. The developed application provides users with a positive interaction experience, optimizing routine accounting processes for equipment operation by enabling intelligent AI analysis, simplifying communication between production and management, and minimizing the influence of the human factor.

ЗМІСТ

Перелік скорочень....9
Вступ....11
1 Аналіз предметної області та існуючих рішень....13
1.1 Огляд методів підвищення ефективності виробничого обладнання. OEE, Lean....14
1.2 Застосування принципів теорії автоматичного керування при цифровізації диспетчерських служб....16
1.3 Аналіз існуючих програмних рішень для моніторингу виробництва та управління задачами операторів....18
1.4 Огляд мобільних технологій у промисловості. оптичне розпізнавання інвентарних номерів за допомогою технологій OCR....21
1.5 Обґрунтування вибору технологій для розробки системи....22
2 Проєктування мобільної інформаційної системи....26
2.1 Розробка архітектури програмного комплексу....27
2.2 Проєктування концептуальної та логічної моделей бази даних....30
2.3 Розробка REST API для взаємодії між мобільним додатком та сервером 31
2.4 Проєктування інтерфейсу користувача мобільної системи....35
3 Програмна реалізація системи....39
3.1 Реалізація серверної частини на базі Spring Boot....40
3.1.1 Налаштування ORM Hibernate....44
3.1.2 Інтеграція Large Language Model Google Gemini для аналізу простоїв....45
3.1.3 Реалізація SMTP-клієнта для надсилання звітів....46
3.2 Розробка мобільного android-додатку....49

3.2.1 Реалізація мережевої взаємодії з REST API.....52
3.2.2 Інтеграція модуля оптичного розпізнавання інвентарних номерів... 54
4 Тестування системи та аналіз результатів інфраструктурної інтеграції..... 57
4.1 Функціональне тестування REST API.....57
4.2 Інфраструктурна проблема SMTP-клієнта.....61
4.3 Охорона праці персоналу при експулуатації системи.....62
Висновки.....65
Перелік джерел посилання..... 67
Додаток А Лістинги ключових фрагментів програмного коду.....70
Додаток Б Супровідні та цифрові матеріали проєкту.....111
Додаток В Демонстраційний матеріал.....112

ПЕРЕЛІК СКОРОЧЕНЬ

ПЛК – програмований логічний контролер;

ШІ – штучний інтелект;

ACID – атомарність, узгодженість, ізольованість, довговічність – надійність і цілісність даних при використанні транзакцій у СУБД (Atomicity, Consistency, Isolation, Durability);

API – інтерфейс прикладного програмування (Application Programming Interface);

CRUD – базові операції управління даними в програмуванні: створення, читання, оновлення, видалення (Create, Read, Update, Delete);

DTO – об'єкт перенесення даних (Data Transfer Object);

ERP – планування ресурсів підприємства (Enterprise Resource Planning);

GUI – графічний інтерфейс користувача (Graphical User Interface);

HTML – мова розмітки гіпертексту (HyperText Markup Language);

HTTP – протокол передачі гіпертексту (Hypertext Transfer Protocol);

IIoT – промисловий інтернет речей (Industrial Internet of Things);

JPA – специфікація збереження даних (Jakarta Persistence API);

JSON – об’єктна нотація JavaScript (JavaScript Object Notation);

JVM – віртуальна машина Java (Java Virtual Machine);

JWT – вебтокен JSON (JSON Web Token);

MES – система управління виробничими процесами (Manufacturing Execution System);

MOM – управління виробничими операціями (Manufacturing Operations Management);

MVVM – архітектурний шаблон для розділення бізнес-логіки додатка та графічного інтерфейсу користувача (Model-View-ViewModel);

MySQL – система управління базами даних;

OCR – оптичне розпізнавання символів (Optical Character Recognition);

OEE – загальна ефективність обладнання (Overall Equipment Effectiveness);

OLE – зв'язування та впровадження об'єктів (Object Linking and Embedding);

OPC – відкритий стандарт комунікації платформ (Open Platform Communications);

ORM – об'єктно-реляційне відображення (Object-Relational Mapping);

PLM – управління життєвим циклом виробу (Product Lifecycle Management);

QR-код – код швидкого реагування (Quick Response code);

REST API – інтерфейс прикладного програмування на основі передачі представлення стану (Representational State Transfer Application Programming Interface);

SCM – управління ланцюгом постачання (Supply Chain Management);

SMTP – простий протокол передачі пошти (Simple Mail Transfer Protocol);

SQL – мова структурованих запитів (Structured Query Language);

XML – розширювана мова розмітки (eXtensible Markup Language).

ВСТУП

Системи моніторингу машин у реальному часі дають змогу швидко виявляти неефективність, оптимізувати виробничі процеси, скорочувати простої та підвищувати продуктивність. Доцільно об’єднати відстеження обладнання та прозорий контроль за завданнями на підприємстві, оскільки своєчасне ознайомлення з ключовими показниками гарантовано дозволить виявити проблеми та мінімізувати ризики поломок або простоїв.

Безпека є важливою ціллю у промисловості, і спеціалізована система моніторингу обладнання забезпечує розширений контроль. Відстеження критичних умов дозволяє своєчасно виявляти потенційні загрози та впроваджувати проактивні заходи для захисту обладнання і персоналу.

Індивідуальні рішення забезпечують цілеспрямований аналіз даних, що допомагає компаніям зменшити витрати на ремонти та енергію завдяки ранньому виявленню потреб у технічному обслуговуванні. Аналіз даних у реальному часі дозволяє скоротити експлуатаційні витрати та прискорити окупність інвестицій.

Спеціалізоване програмне забезпечення надає цінну інформацію для стратегічного управління, пропонуючи адаптовані звіти та дані про ефективність. Це дозволяє керівництву приймати обґрунтовані рішення щодо модернізації обладнання, оптимізації процесів і вдосконалення операцій. Такий підхід підтримує цифрову трансформацію та сприяє формуванню культури вдосконалення на основі даних.

Система моніторингу промислового обладнання забезпечує точне розуміння використання ресурсів, дозволяючи організаціям ефективно розподіляти активи. Це запобігає як недовикористанню, так і перевантаженню ресурсів, оптимізує планування виробництва та підвищує рентабельність інвестицій.

Об'єкт розробки – процес моніторингу виробничих процесів, фіксації

простоїв промислового обладнання та оперативного керування завданнями персоналу на підприємстві.

Предмет розробки – мобільна клієнт-серверна інформаційна система автоматизованого збору метрик ефективності, безконтактної ідентифікації обладнання та інтелектуального аналізу інцидентів.

Мета розробки – підвищення загальної ефективності використання виробничого обладнання шляхом розроблення і впровадження мобільно-серверного комплексу автоматизованого збору даних, безконтактного розпізнавання інвентарних номерів верстатів, контролю завдань операторів та генерації інтелектуальної ШІ-підказки інцидентів у реальному часі.

У результаті вирішено такі задачі, за для досягнення поставленої мети, як аналіз сучасних патернів побудови систем моніторингу та обґрунтування вибору технологічного стеку, проєктування моделі реляційної бази даних, розроблення загальної архітектури програмного комплексу, реалізація серверної частини на базі Spring Boot, розроблення нативного мобільного Android-додатку на основі патерну MVVM з реактивним керуванням станом інтерфейсу, інтеграція розпізнавання інвентарних номерів за допомогою комп’ютерного зору, впровадження модулю, який оснащений інтелектуальним аналізом простоїв, та налаштування SMTP-клієнту для розсилки емейл-звітів, проведення інтеграційного тестування розробленої мобільної системи.

Отримані результати відповідають Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», оскільки розроблене програмне рішення сприяє модернізації промислових потужностей, впровадженню передових цифрових технологій і підвищенню ефективності використання ресурсів на підприємствах.

Звіт з кваліфікаційної роботи оформлено згідно з ДСТУ 3008:2015 [1] та з використанням навчально-методичних матеріалів кафедри комп’ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інформації [2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

Загальна ефективність обладнання (OEE – Overall Equipment Effectiveness) – це показник для вимірювання результативності та продуктивності виробничих процесів або окремого елемента обладнання. Він показує, наскільки коректно обладнання використовується у виробництві товарів або наданні послуг [3].

На практиці OEE відповідає на питання: «Скільки часу, протягом якого наше обладнання відведено на виробництво, насправді є продуктивним?»

Розуміння загальної ефективності обладнання починається з трьох його фундаментальних компонентів: доступності, продуктивності та якості. Кожен компонент визначає окремий тип втрат, що впливає на продуктивність, і пропонує цінну інформацію про те, де найбільше потрібні покращення.

Доступність вимірює співвідношення фактичного часу роботи до запланованого часу виробництва [3]. Вона охоплює всі форми простою обладнання. Зазвичай дві основні форми – це планові та незаплановані. Виділяючи поширені причини втрати доступності, стає зрозуміло, що планова форма зупинки роботи – це переналаштування, налаштування машин, профілактичне обслуговування, а незапланована – це неочікувані збої обладнання, поломки або дефіцит матеріалів, що зупиняють виробництво та скорочують час.

Наприклад, пакувальна лінія може мати запланований 30-хвилинний перехід між продуктами, але часто стикається із затримками через відсутність інструментів або некомплектність матеріалів. Завдяки оцифруванню та стандартизації процесу перехідника, виробники можуть скоротити час переходу та мінімізувати запланований час простою, тим самим покращити доступність.

Продуктивність враховує втрати, пов'язані зі швидкістю. Існують не лише механічні уповільнення, а й проблеми, пов'язані з процесом, які знижують пропускну здатність. У типових втратах продуктивності винні мікрозупинки,

які через короткі переривання часто спричиняють порушення процесу роботи. Доцільно згадати про повільні цикли – обладнання постійно працює нижче своєї проєктної швидкості через неоптимальні налаштування або звички оператора.

Втрата якості вимірює дефектну продукцію, незалежно від того, чи її повністю бракують, чи вона потребує переробки. Це включає продукцію, яка не відповідає специфікаціям замовника, навіть якщо її можна переробити пізніше. Нестабільні умови процесу, такі як коливання температури або неправильні налаштування, а також неякісна сировина, що призводить до бракованої готової продукції, та помилки, допущені під час ручного втручання або переналаштування, вважаються поширеними джерелами втрати якості.

1.1 Огляд методів підвищення ефективності виробничого обладнання. OEE, Lean

Підвищення OEE приносить значні переваги організаціям у виробничому секторі, сприяючи зростанню продуктивності. Оптимізуючи використання обладнання, скорочуючи час простою й упорядковуючи операції, компанії ефективніше використовують ресурси та підвищують загальну ефективність. Поліпшення OEE часто знижує витрати та підвищує прибуток, оскільки скорочення простоїв дозволяє уникати затримок, а зменшення дефектів – браку, переробки й втрат матеріалів, що призводить до економії. Моніторинг і аналіз OEE забезпечують цінну інформацію щодо продуктивності обладнання, дозволяють приймати рішення на основі даних, визначати пріоритети інвестицій, ефективно розподіляти ресурси та впроваджувати оптимізацію процесів.

Проте покращення OEE потребує системного підходу. Вимірювання та відстеження обладнання, виробничих процесів, встановлення базового рівня та цілі для покращення. Максимізоване вирішення питання простоїв обладнання, тобто впровадження програми профілактичного обслуговування, щоб

мінімізувати поломки, план роботи з технічного обслуговування під час запланованих зупинок та оптимізація процесів переналаштування.

Важливо впроваджувати навчальні програми, щоб забезпечити операторам необхідні навички для ефективної роботи з обладнанням. Регулярні тренінги на тему зменшення втрати швидкості та час простою, позитивно впливають як на загальну продуктивність так і на психоемоційний, фізичний стан працівника, бо попередній розбір типових кейсів на виробництві на робочому місці формує чітке уявлення про порядок дій у критичних ситуаціях, що дозволяє зменшити рівень людського фактору. Віддалений контроль активів за допомогою датчиків та пристроїв Інтернету речей і комп'ютерного зору на периферії зменшує залежність від команд технічного обслуговування.

Заходи контролю якості – це ключ до виявлення першопричин дефектів. Технології комп'ютерного зору – виявлення аномалій, коригувальні дії – покращення якості продукції та зменшення відходів.

В майбутньому це посприяє розвитку культури постійного вдосконалення: впроваджуватимуться структуровані ініціативи з покращення, такі як заходи Кайдзен, проєкти Шість Сигма або методології бережливого виробництва, для стимулювання зусиль щодо постійного зростання [4]. Аналіз тенденцій виявляє закономірності для ухвалення обґрунтованих рішень щодо модернізації обладнання, оптимізації процесів або розподілу ресурсів.

Загалом, покращення показника OEE – це довгострокова справа, що вимагає відданості, співпраці та невпинної зосередженості на постійному вдосконаленні. Важливо залучати до процесу всіх зацікавлених сторін, від операторів до менеджерів, та відзначати успіхи на цьому шляху, щоб підтримувати мотивацію та залученість.

Останні тенденції підвищення ефективності виробничого обладнання стрімко поширюються. Інтеграція систем OEE з технологіями IIoT набирає обертів.

IIoT дозволяє збирати дані в режимі реального часу з датчиків обладнання, забезпечуючи точні та своєчасні вимірювання OEE [5]. Ця

інтеграція також сприяє прогнозованому технічному обслуговуванню, дистанційному моніторингу та прийняттю рішень на основі даних для оптимізації продуктивності обладнання.

Використання розширеної аналітики та штучного інтелекту в аналізі OEE розширюється. Алгоритми машинного навчання можуть аналізувати величезні обсяги даних, виявляти закономірності та розкривати приховані ідеї для оптимізації OEE. Прогнозна аналітика допомагає організаціям передбачати збої в роботі обладнання, оптимізувати графіки технічного обслуговування та підвищувати загальну ефективність.

Хмарні рішення OEE пропонують масштабованість, доступність і простоту впровадження. Організації можуть використовувати хмарні платформи для зберігання та обробки великих обсягів даних OEE, співпрацювати в режимі реального часу та отримувати доступ до аналітики й звітів з будь-якого місця, що спрощує дистанційний моніторинг і ухвалення рішень.

Додатки та інструменти візуалізації надають дані в режимі реального часу, а також панель інструментів продуктивності на мобільних пристроях. Це дозволяє операторам і менеджерам контролювати продуктивність обладнання, отримувати сповіщення та доступ до аналітики OEE в дорозі, що сприяє швидшому ухваленню рішень і швидкому реагуванню.

Стандартизація допомагає забезпечити узгодженість, дозволяє проводити бенчмаркінг і полегшує співпрацю та обмін знаннями між колегами в галузі.

1.2 Застосування принципів теорії автоматичного керування при цифровізації диспетчерських служб

З позицій сучасної теорії автоматичного керування будь-який виробничий процес промислового підприємства можна розглядати як складну, динамічну та стохастичну систему, що піддається постійному впливу зовнішніх і внутрішніх збурювальних факторів. При класичному підході до організації диспетчерських служб управління цехом він базується на розімкнутому контурі. Інформація про аварійні зупинки, технічні несправності або організаційні простої надходить із

великим запізненням через людський фактор, паперовий облік або усні доповіді персоналу. Така розімкнутість системи призводить до значного динамічного запізнення, збільшує час перехідних процесів і суттєво знижує загальну стійкість та ефективність функціонування всього об'єкта управління [6].

Цифровізація диспетчерських служб за допомогою клієнт-серверних рішень дозволяє трансформувати менеджмент у замкнуту систему автоматичного регулювання з безперервним зворотним зв'язком. У рамках розробленої системи об'єктом управління є технологічний комплекс промислового обладнання цеху разом із сукупністю операторів, які його обслуговують. Мобільний Android-додаток, оснащений модулем комп'ютерного зору та OCR-сканування, у цій структурі виконує функцію інтелектуального датчика зворотного зв'язку. Цей датчик фіксує первинну телеметрію, зчитує інвентарні номери верстатів і миттєво передає вектор помилки системи, тобто інформацію про розбіжність між плановим робочим станом і фактичним виникненням простою, на рівень сервера.

Серверна частина системи, реалізована на базі Spring Boot, є центральним керуючим пристроєм або цифровим логічним регулятором [6]. Головним завданням цього регулятора є мінімізація відхилення системи від заданої траєкторії, що математично виражається через максимізацію критерію загальної ефективності обладнання. Отримавши сигнал про збій, серверний регулятор миттєво обчислює необхідний керуючий вплив. Особливістю даної архітектури є використання великої мовної моделі Google Gemini API як стохастичного блоку інтелектуального аналізу, що на основі накопичених логів формує адаптивні підказки для оператора щодо усунення конкретної причини простою.

Сформований керуючий вплив передається назад на об'єкт управління через два канали зв'язку. Перший – безпосередньо на екрані мобільного додатка оператора у вигляді цільового оперативно-виробничого завдання з ШІ-підказкою. Другий – на електронну пошту майстра цеху через SMTP-клієнт у вигляді аналітичного повідомлення про порушення стабільності процесу. Персонал, виконуючи отримані вказівки та усуваючи несправність, виступає як

виконавчий механізм, що повертає промисловий верстат у штатний режим роботи. Таким чином, замикання контуру керування за допомогою інформаційних технологій мінімізує час непланових зупинок, забезпечує високу стійкість виробничого процесу до збоїв і гарантує оптимальне автоматизоване управління ресурсами підприємства в реальному часі.

1.3 Аналіз існуючих програмних рішень для моніторингу виробництва та управління задачами операторів

MES – це програмне рішення, яке використовується у виробництві для моніторингу та контролю виробничих процесів у цеху. В управлінні виробничими операціями MES слугує мостом між системами планування та контролю підприємства, такими як система планування ресурсів підприємства, і фактичними виробничими операціями.

Відстеження та документація процесу виготовлення продукції в режимі реального часу – основне призначення. MES збирає дані з різних джерел, включаючи машини, датчики та операторів, щоб надавати точну й актуальну інформацію про стан виробничої діяльності. Забезпечує видимість контролю виробничих процесів у режимі реального часу, дозволяючи зацікавленим сторонам контролювати операції, виявляти вузькі місця, мінімізувати простої та оперативно приймати обґрунтовані рішення [7].

Сприяючи оптимізованому плануванню та складанню графіків виробництва, MES-системи забезпечують ефективний розподіл ресурсів, балансування робочого навантаження та своєчасні поставки, що призводить до підвищення прибутковості. Вони також відіграють вирішальну роль у забезпеченні якості та дотриманні вимог. Забезпечують дотримання процедур контролю якості, моніторинг показників і збір даних у режимі реального часу. Завдяки своїй здатності керувати рівнями запасів, відстежувати рух матеріалів та допомагати забезпечувати своєчасну наявність матеріалів, MES-системи оптимізують управління запасами та мінімізують затримки у виробництві.

Крім того, MES-системи дозволяють ухвалювати рішення на основі даних і надають вичерпні та точні відомості про виробництво. Організації мають можливість постійно вдосконалювати процеси та оптимізувати використання ресурсів. MES-системи можуть допомогти особам, що приймають рішення, визначити загальну ефективність обладнання, широкий показник, який використовується для моніторингу ефективності виробництва.

Завдяки оптимізації робочих процесів, автоматизації завдань та забезпеченню зворотного зв'язку в режимі реального часу, MES-системи підвищують ефективність та продуктивність виробництва. Системи управління дозволяють відстежувати генеалогію, що є критично важливим для галузей зі суворими правилами, шляхом відстеження руху матеріалів і процесів протягом усього життєвого циклу інтелектуального виробництва.

Процес відбувається таким чином: першим кроком система збирає дані з джерел, включаючи машини, датчики, операторів та інші інформаційні системи, такі як системи ERP (Enterprise Resource Planning – планування ресурсів підприємства) або системи управління життєвим циклом продукції.

Ці дані можуть включати показники виробництва, стан машин, рівні запасів, а також показники якості. Зібрані дані обробляються та інтегруються в систему MES, створюючи комплексне уявлення про виробниче середовище. Ця інтеграція допомагає забезпечити MES точною та актуальною інформацією для роботи.

На основі виробничих замовлень, отриманих від систем планування вищого рівня, MES генерує графік виробництва. Цей графік враховує такі фактори, як пріоритети замовлень, доступні ресурси, потужності машин і доступність робочої сили.

Система призначає робочі замовлення операторам або робочим станціям на основі графіка. Вона надає операторам інструкції, специфікації та необхідну документацію, щоб допомогти їм виконувати свої завдання. Під час процесу відстежується хід кожного робочого замовлення, а статус незавершеного виробництва оновлюється в режимі реального часу.

Система взаємодіє з машинами та обладнанням у цеху для контролю їхнього стану, збору виробничих даних та обміну інформацією. Цієї інтеграції можна досягти за допомогою різних засобів, таких як датчики машин, інтерфейси програмованих логічних контролерів (ПЛК) або протоколи зв'язку, такі як OLE, що забезпечують зв'язування та вбудовування об'єктів, для керування процесами.

Дані про якість збираються під час виробництва, такі як вимірювання, перевірки та результати випробувань. Впровадження процедур контролю якості, запускає сповіщення або сповіщення про проблеми з якістю та записує інформацію, пов'язану з якістю, для аналізу та відстеження.

MES відстежує рух матеріалів і компонентів протягом усього виробничого процесу. Контролює рівні запасів, ініціює заявки на матеріали або поповнення запасів та допомагає забезпечити наявність правильних матеріалів у потрібний час і в потрібних кількостях.

Зібрані дані аналізуються для отримання аналітики та показників продуктивності в режимі реального часу, далі генеруються звіти, інформаційні панелі та візуалізації, які допомагають керівництву та операторам приймати обґрунтовані рішення та визначати області для покращення.

Система взаємодіє з іншими системами, такими як ERP, PLM або системи управління ланцюгами поставок. Така інтеграція дозволяє обмінюватися даними, синхронізувати інформацію та узгоджувати виробничі процеси із загальними бізнес-операціями.

Незважаючи на широку функціональність традиційних MES-систем, їх впровадження на підприємствах, особливо з різнотипним парком обладнання, пов'язане з низкою суттєвих обмежень. Висока вартість ліцензій та складність інтеграції з наявною інфраструктурою часто роблять повноцінні MES економічно недоцільними для малого та середнього бізнесу або окремих виробничих дільниць.

Архітектура класичних MES зазвичай базується на стаціонарних терміналах, що змушує операторів залишати робоче місце для введення даних.

Це призводить до затримок в отриманні інформації, зниження точності даних через людський фактор та виникнення інформаційних розривів між цехом і менеджментом.

Мобільна система моніторингу, яку пропонують до розробки, вирішує ці проблеми шляхом зміни парадигми збору даних.

Інтерфейс вводу даних перенесенно на мобільний пристрій, це дозволить фіксувати події простою, браку, виконання завдань безпосередньо в момент і в місці їх виникнення, що відповідає концепції MOM (Manufacturing Operations Management), яка розширює можливості MES, роблячи управління незалежним від пристрою та доступним у будь-якій точці цеху.

Використання стандартних мобільних пристроїв на базі Android замість спеціалізованих промислових панелей та контролерів значно знижує капітальні витрати на впровадження системи, а розробка мобільного додатку є не просто полегшеною версією MES, а цільовим рішенням проблеми у цифровізації виробництва, що забезпечує безбар'єрний збір даних та залучення персоналу до процесів постійного вдосконалення.

1.4 Огляд мобільних технологій у промисловості. Оптичне розпізнавання інвентарних номерів за допомогою технологій OCR

Оптичне розпізнавання символів відіграє ключову роль в управлінні інформацією, оскільки ця технологія допомагає уникнути ручного введення даних, яке забирає дорогоцінний час у співробітників, а також виникнення помилок. Коректно ідентифікує важливу інформацію та надалі успішно отримує з неї цінність під час роботи.

Як правило, дані, які потребують переведення в цифровий формат, скануються або фотографуються. Застосовується технологія оптичного розпізнавання символів. Інформація, що міститься в документі, зберігається у вигляді тексту з метаданими для пошуку, а отримані дані використовуються для запуску робочих процесів і покращення підтримки всіх процесів. Технологія

оптичного розпізнавання символів призначена для усунення необхідності ручного введення даних шляхом автоматичного розпізнавання буквених, цифрових знаків.

В цілому оптичне розпізнавання символів дає широкий спектр переваг, які можуть використовувати компанії всіх галузей і розмірів. Окрім підвищення ефективності пошуку даних, OCR також лежить в основі інших процесів, наприклад, машинного навчання. Машинне навчання відіграє важливу роль в управлінні документами: воно може розпізнавати рукописний текст та переводити його в цифровий формат та більш детально аналізувати. Це допоможе зменшити необхідність втручання в процес співробітників, а отже збільшить безпеку даних, та зменшити вірогідність виникнення помилок при обробці даних.

Відстежуваність і ідентифікація деталей – це найпоширеніший випадок використання. Камери можуть зчитувати індентифікатори на деталях під час проходження виробничої лінії, щоб переконатися, що потрібна деталь знаходиться у потрібному місці в потрібний час, зчитувати ідентифікаційні номери, штамповані на рамах автомобіля або серійні номери, викарбувані на оброблених деталях, навіть на складних поверхнях, таких як литий метал або відбивний пластик, маленькі коди символів на окремих напівпровідникових чіпах та інших компонентах на друкованій платі, щоб переконатися, що деталь встановлена правильно.

Як висновок, оптичне розпізнавання символів – це перший крок до запуску масштабніших проєктів автоматизації робочих процесів у бізнесі. Бо чимало важливих завдань, які виконують співробітники вручну, піддаються впливу людського чинника.

1.5 Обґрунтування вибору технологій для розробки системи

Вибір технологічного стеку для реалізації мобільної системи моніторингу базується на аналізі специфічних умов експлуатації на

виробництві, вимогах до надійності роботи в умовах нестабільного зв'язку та економічній доцільності впровадження.

Як цільову платформу для клієнтського додатка було обрано операційну систему Android. Цей вибір зумовлений передусім економічним фактором та апаратною доступністю. На відміну від iOS-пристроїв, ринок Android пропонує широкий спектр бюджетних смартфонів та спеціалізованих захищених промислових терміналів, що стійкі до ударів, пилу та вологи. Це дозволяє підприємству масштабувати систему на велику кількість робочих місць з мінімальними капітальними витратами.

Для розробки інтерфейсу додатка та впровадження серверної частини обрано мову програмування Kotlin, яка на сьогодні є офіційним стандартом Google для Android-розробки [8]. Вибір Kotlin обґрунтований її сучасними механізмами безпеки, зокрема, Null Safety – захист від помилок з пустими посиланнями. Це критично важливо для промислового софту, оскільки мінімізує ризик раптового закриття додатка під час зміни оператора, забезпечуючи високу стабільність роботи. Використання Jetpack Compose, майданчика для гнучкого створення інтерфейсів із декларативним підходом, дозволяє уникнути розробки застарілою XML-верстkою [9]. Також важливою перевагою є безшовна інтеграція з серверною частиною, написаною на базі Spring Boot (Java), оскільки ці мови програмування компілюються у байт-код для JVM, дозволяють використовувати еквівалентні структури даних, моделі серіалізації.

Для реалізації функції сканування інвентарних номерів обрано бібліотеку Google ML Kit. Ключовим аргументом є здатність працювати в режимі On-Device (офлайн). Принцип обробки даних наведено на рисунку 1.1.

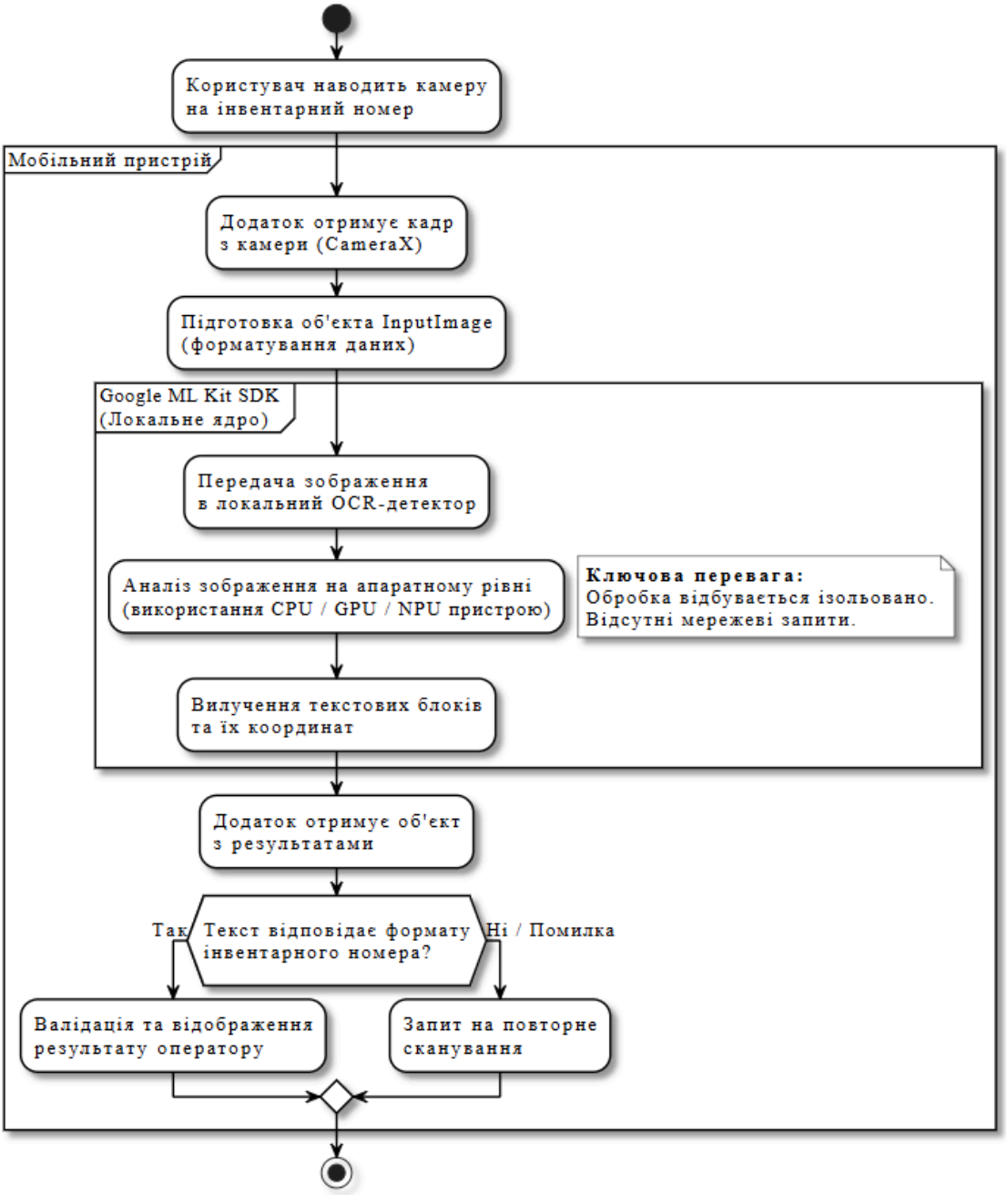


Рисунок 1.1 – Принцип роботи On-Device OCR (Google ML Kit)

В умовах виробничого цеху часто спостерігаються проблеми з покриттям Wi-Fi. Використання хмарних API було б ненадійним. ML Kit виконує обробку зображення безпосередньо на процесорі смартфона, що забезпечує миттєвий результат навіть за повної відсутності інтернету [10].

Для реалізації серверної частини обрано мову програмування Java з використанням фреймворку Spring Boot [11]. Цей вибір зумовлений високими вимогами до відмовостійкості та безпеки промислових систем.

По-перше, Java є мовою зі суворою типізацією, що дозволяє виявляти більшість помилок ще на етапі компіляції, гарантуючи стабільну роботу сервера в безперервному режимі. По-друге, екосистема Spring Boot дозволяє швидко побудувати масштабовану архітектуру мікросервісів та легко інтегруватися з корпоративними базами даних. По-третє, багатопотокова модель Java ефективно обробляє одночасні запити від великої кількості операторів та підключеного обладнання, що є критичним для моніторингу в реальному часі.

2 ПРОЄКТУВАННЯ МОБІЛЬНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

У процесі проєктування важливим є врахування кожного аспекту, оскільки фундаментальні характеристики визначають ключові області знань. Інженерний підхід передбачає застосування концепцій, що мінімізують суб'єктивність при виборі рішень, які виникають у ході розробки.

Визначення архітектурних характеристик є складним завданням через їхню чисельність і неявність, тому необхідно ідентифікувати та пріоритизувати їх до початку впровадження. Явні характеристики зазвичай легше визначити, оскільки вони прямо зазначені у вимогах до системи.

Значення архітектури програмного забезпечення – це вирішальний фактор загального успіху програмної системи. Продумана архітектура безпосередньо впливає на продуктивність, надійність та зручність обслуговування системи, полегшуючи розробникам управління складністю та досягнення високоякісних результатів.

Корректно спроєктована архітектура оптимізує розробку програмного забезпечення, надає послідовну структуру, що спрощує модифікації та додавання нових функцій. Вона також підвищує масштабованість та відмовостійкість, гарантуючи, що система може справлятися зі зростанням та коректно відновлюватися після збоїв.

Крім того, міцна архітектурна основа знижує експлуатаційні витрати та витрати на обслуговування, мінімізуючи потребу в значних переробках та забезпечуючи ефективні процеси розробки. По суті, інвестування в надійну архітектуру програмного забезпечення приносить дивіденди протягом усього життєвого циклу системи, підтримуючи як нагальні потреби, так і майбутні інновації.

2.1 Розробка архітектури програмного комплексу

Архітектура розроблювальної мобільної системи побудована на основі багаторівневого проєктування програмного забезпечення. Багаторівневий шаблон розділяє додаток на так звані горизонтальні шари, а шар відповідає за певну функцію.

У процесі планування програмного комплексу було обрано тришаровий шаблон, який розділяє систему на рівні презентації, бізнес-логіки та даних. Основною перевагою такого підходу є модульність, що дозволяє зосередитися на створенні або модифікації окремого рівня без необхідності враховувати деталі реалізації інших шарів. Це сприяє підвищенню швидкості розробки, ефективності виправлення помилок, розширюваності, а також забезпечує надійність і зручність експлуатації [12].

Рівень презентації, ще його називають веб, містить набір класів контролерів, які отримують REST API від клієнтів, запити інтегруються та обробляються шляхом виклику методів об’єктів на рівні бізнес логіки (сервісу), який викликає API зовнішніх систем, в додатку це надсилання електронних листів, та використовує моделі на рівні даних для CRUD операції (створення, читання, оновлення та видалення) з або до таблиці бази даних. Структурну схему архітектуру та взаємодію компонентів наведено на рисунку 2.1.

Клієнтська частина – нативний Android-застосунок, розроблений за допомогою обраної мови програмування Kotlin. Для створення інтерфейсу користувача використано Jetpack Compose – це фреймворк, що дозволяє створювати динамічні, адаптивні, різної складності інтерфейси.

Архітектурним шаблоном обрано MVVM (Model-View-ViewModel), який відокремлює питання інтерфейсу користувача від логіки програми – ViewModel та даних – Model.

Модель відображає дані та бізнес-логіку додатка, інкапсулює об’єкти даних та надає методи для обробки цих даних. Представлення, тобто View, спостерігає за змінами та перемальовує своє відображення.

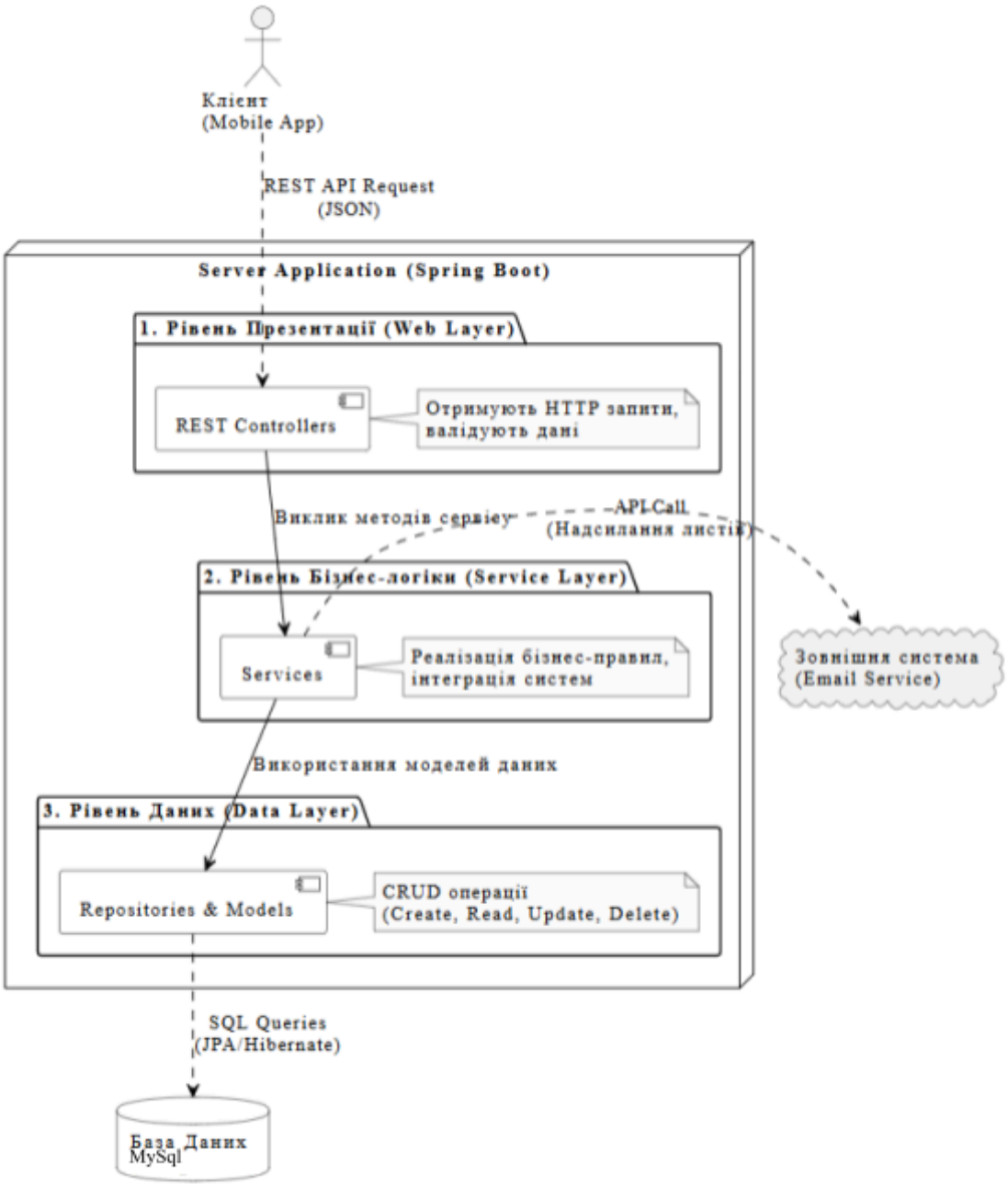


Рисунок 2.1 – Структурна схема тришарової архітектури серверної частини

Уся логіка змін делегована ViewModel, яка діє як посередник, надає дані та функції, до яких View прив’язується та реагує на взаємодії з користувачем [13].

MVVM часто спирається на механізми зв'язування даних, що надаються використовуваним фреймворком або бібліотекою. Зв'язування даних дозволяє

ViewModel повідомляти View про зміни в даних, усуваючи необхідність ручної синхронізації між View та ViewModel, що сприяє чіткому розподілу завдань, де кожен компонент відповідає за певний аспект програми [13]. Такий розподіл спрощує розуміння, підтримку та тестування кодової бази, заохочує повторне використання компонентів. Архітектуру застосунку наведено на рисунку 2.2.

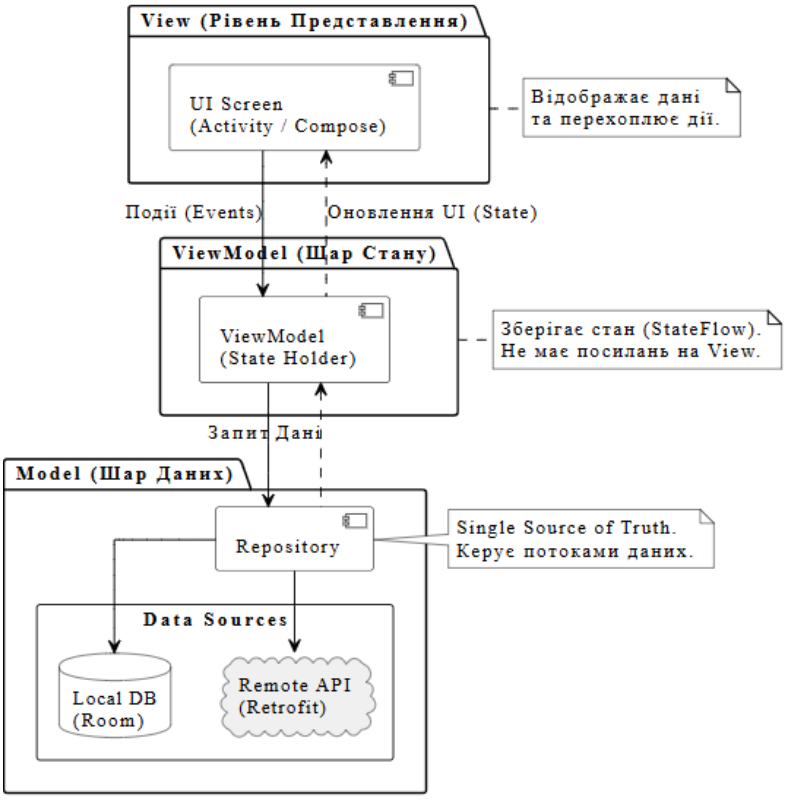


Рисунок 2.2 – Архітектура клієнтського застосунку на базі патерну MVVM

Ключовою особливістю програмного комплексу є гібридний підхід до збору даних, який поєднує ручне введення оператором і автоматизовані розрахунки на сервері. Мобільний клієнт фіксує дискретні події, такі як зміна статусу обладнання, початок виконання завдання або сканування QR-коду, а сервер трансформує ці події у аналітичні метрики в реальному часі. Така архітектура запобігає розсинхронізації часу, оскільки всі часові мітки генеруються або перевіряються сервером, а також знижує навантаження на мобільний пристрій, передаючи йому вже оброблені дані для відображення.

2.2 Проєктування концептуальної та логічної моделей бази даних

Проєктування бази даних визначає, як дані будуть зберігатися, і це має вирішальне значення, адже має значний вплив на те, як до бази даних здійснювати запити для читання даних або для оновлення даних. Існують два важливі поняття, які слід знати, коли йдеться про проєктування баз даних: моделі та схеми баз даних. Моделі баз даних – це високорівневі специфікації для структури бази даних. Реляційна модель, яка є найпопулярнішою, – це модель, що використовується для створення реляційних баз даних. Вона визначає рядки як записи, а стовпці – як атрибути. Вона вимагає таких правил – кожен рядок має унікальні ключі. Схема – це креслення, яке визначає конкретні таблиці, поля, зв'язки, індекси та представлення в базі даних.

Для системи було обрано саме реляційну модель даних, оскільки вона забезпечує строгу типізацію, підтримку транзакцій та чіткі зв'язки між сутностями, що є необхідним для систем обліку та моніторингу виробництва. На етапі концептуального та логічного моделювання було виділено ключові сутності системи та зв'язки між ними. База даних спроєктована за правилами нормалізації задля уникнення надлишковості даних та аномалій при оновленні. Фізична схема бази даних реалізована у середовищі системи управління базами даних MySQL. Вибір цієї СУБД обумовлений її високою продуктивністю при обробці запитів на читання, надійністю та повною підтримкою принципів ACID, що є критичним для забезпечення цілісності виробничих даних [15].

База даних складається з наступних основних таблиць. Користувачі (users) – зберігає інформацію про персонал, ключовий атрибут ідентифікації – табельний номер, який повинен бути унікальним, також таблиця містить посилання на індифікатор майстра, це потрібно для реалізації ієрархії «Майстер – Підлеглий». Рольова модель реалізована через поле role, значення ENUM: OPERATOR, MASTER.

Обладнання (equipment) – ключова сутність системи, яка ідентифікується за унікальним інвентарним номером та зберігає поточний технічний стан, дати

обслуговування та посилання на зображення, а також поточну активність без необхідності робити важкі вибірки з логів. Діаграму станів життєвого циклу наведено на рисунку 2.3.



Рисунок 2.3 – Діаграма станів життєвого циклу обладнання

Завдання – містить заплановані технічні завдання та пов'язує обладнання з відповідальним оператором. Завдання має планові та фактичні показники часу, що дозволяють аналізувати ефективність роботи.

Журнал подій документує дані про зміни станів обладнання для формування аналітичних звітів про простої, час налаштування та корисну роботу.

Користувач-Обладнання – таблиця з реалізацією зв'язку Many to Many, яка визначає, за яким обладнанням закріплений конкретний оператор.

Додатково для сутностей Equipment та EquipmentLog розроблено логіку переходів, щоб дати гарантію, – обладнання не може перебувати у двох станах одночасно, а кожен запис у логах має коректний час початку та завершення.

2.3 Розробка REST API для взаємодії між мобільним додатком та сервером

Для організації взаємодії сервера з клієнтською частиною обрано архітектурний стиль Representational State Transfer. REST – це стандартний

підхід для сучасних веб-сервісів, який надає такі властивості, як масштабованість, простота інтеграції та використання стандартних методів протоколу HTTP [15]. Базові HTTP-методи для взаємодії мобільного додатка та серверної частини наведені в таблиці 2.1.

Таблиця 2.1 – Основні HTTP-методи взаємодії з сервером

Назва методу	Призначення та функція
GET	Використовується для отримання (читання) даних із сервера.
POST	Використовується для надсилання та створення нових даних на сервері.
PUT	Використовується для повного оновлення (заміни) існуючих даних на сервері.
PATCH	Використовується для внесення часткових змін до існуючих даних на сервері.
DELETE	Використовується для видалення даних із сервера.

Завдяки клієнт-серверній архітектурі отримано чітке розмежування відповідальності. Сервер займається бізнес-логікою, доступом до бази даних та безпекою. Мобільний додаток відповідає виключно за відображення даних та взаємодію з користувачем, а сервер не зберігає сесію клієнта між запитами. Кожен HTTP-запит містить всю необхідну інформацію для його обробки, включаючи токен авторизації, що дозволяє легко масштабувати систему, при цьому використовуються стандартні HTTP-методи для керування ресурсами.

Обмін даними здійснюється у форматі JSON (JavaScript Object Notation), який цінується за компактність повідомлень та легку інтеграцію з мовою Kotlin на стороні клієнта. Для стандартизації відповідей сервера використовуються відповідні коди стану HTTP, призначення та опис яких наведені в таблиці 2.2.

Таблиця 2.2 – Основні коди відповідей HTTP-сервера

Код та статус	Опис значення
200 OK	Запит пройшов успішно, і сервер повернув відповідь.
201 Created	Запит пройшов успішно, і сервер створив нові дані в результаті запиту.
204 No Content	Запит пройшов успішно, але у відповіді немає даних для повернення.
400 Bad Request	Сервер не може розібрати запит через некоректність або недостатність інформації від клієнта.
401 Unauthorized	Запит вимагає автентифікації, а клієнт не надав дійсні облікові дані.
403 Forbidden	Запит був дійсним, але сервер відмовився його виконувати через відсутність прав (дозволів).
404 Not Found	Запитані дані або ресурс не існують на сервері.
500 Internal Server Error	Внутрішня помилка сервера: запит не може бути оброблений коректно.
503 Service Unavailable	Сервер тимчасово не може обробити запит через перевантаження або технічне обслуговування.

Серверна частина реалізована мовою з використанням фреймворку Spring Boot Web. Точки доступу згруповані у контролери, а для автоматичної генерації документації API використано бібліотеку OpenAPI [16].

Під час проєктування основними контролерами було виділено. Контролер обладнання забезпечує пошук обладнання за інвентарним номером, використовуючи сканування або ручний ввід, а також управління статусами. Метод зміни статусу є критичним для моніторингу, адже фіксує час початку та кінця простою або налаштування. Контролер завдань дозволяє операторам отримувати список своїх завдань, залишати коментарі, фіксувати фактичний час

виконання завдань. Контролер користувачів надає доступ до профілю користувача та списку закріпленого за ним обладнання.

Безпека API забезпечується механізмом JSON Web Token. При успішному вході сервер генерує підписаний токен, який клієнт додає до заголовка Authorization кожного наступного запиту [17]. Для розмежування доступу використовується рольова модель. Анотація @PreAuthorize обмежує доступ до критичних функцій, таких як реєстрація нових співробітників або пакетне завантаження даних. Права доступу для різних ролей наведені в таблиці 2.3.

Таблиця 2.3 – Матриця прав доступу до функцій API

Функція системи	OPERATOR	MASTER
Вхід у систему	+	+
Зміна статусу обладнання	+	+
Реєстрація нових користувачів	-	+
Пакетне додавання обладнання	-	+
Отримання звітів на Email	-	+

Окремим модулем стане контролер звітності ReportController, який реалізує складну бізнес-логіку збору статистики продуктивності та автоматизованої розсилки звітів на електронну пошту керівництва. З бази даних витягуються всі завдання, виконані користувачем за останній тиждень. ChartService створює візуалізацію, далі дані конвертуються у форматований HTML-документ, що містить таблиці виконаних завдань та завдань, що перебувають у роботі. Через EmailService та протокол SMTP сформований лист із вкладеним графіком надсилається на пошту майстра зміни. Такий же процес відбувається над формуванням звіту продуктивності обладнання за день, єдина відмінність полягає в тому що до даних про статус простою або налаштування обладнання додається коментар від оператора та аналіз штучного інтелекту, принцип роботи зображено на рисунку 2.4.

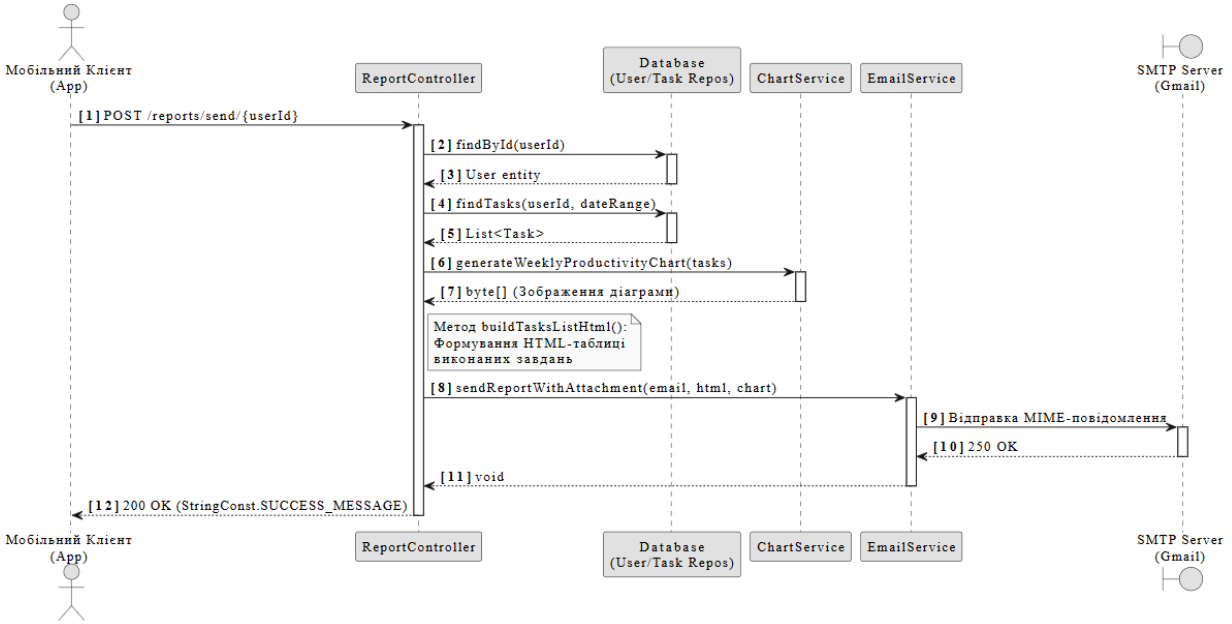


Рисунок 2.4 – Діаграма послідовності процесу генерації та відправки аналітичного звіту

Така архітектура API дозволить не лише забезпечити оперативну роботу мобільного додатка, а й автоматизувати адміністративні процеси, пов'язані зі звітністю та контролем ефективності виробництва.

2.4 Проєктування інтерфейсу користувача мобільної системи

При розробці графічного інтерфейсу (GUI – Graphical User Interface) мобільного клієнта ключовим орієнтиром стала дизайн-система Material Design 3 від Google. Це адаптивна система інструкцій та компонентів, що базується на відкритому вихідному коді та підтримує найкращі практики дизайну користувацького інтерфейсу та досвіду [18].

Вибір Material Design 3 обумовлений специфікою використання додатка в умовах виробництва, бо важливо використовувати чітку типографіку та контрастні кольори – це дозволить оператору зчитувати інформацію навіть при поганому освітленні або при запиленості екрану.

Інтерфейс спроєктовано з урахуванням закону ергономіки, згідно з яким час досягнення цілі залежить від її розміру, а використання стандартних

патернів, таких як нижня навігаційна панель (Bottom Navigation Bar), дозволяє користувачеві інтуїтивно перемикатися між контекстами без додаткового навчання [19]. Загальна карта переходів між екранами та логіка навігації в додатку наведені на рисунку 2.5.

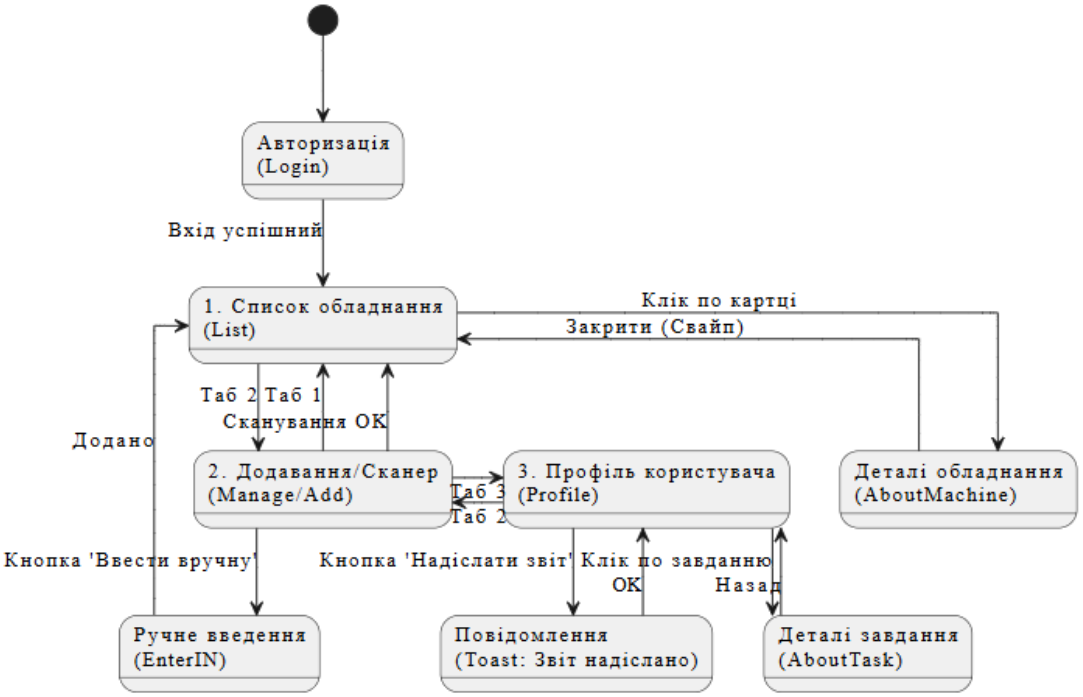


Рисунок 2.5 – Діаграма навігації та переходів між екранами мобільного додатку

Модуль моніторингу є основним робочим інструментом оператора і розподілений між двома вкладками навігації.

Першою вкладкою є список обладнання, який відображає перелік закріплених за оператором верстатів. Для реалізації списку використовуватиметься компонент LazyColumn, що забезпечує ефективний рендеринг великих масивів даних. Кожен елемент списку є карткою, що містить фото обладнання, назву, інвентарний номер та кольоровий індикатор статусу. Використана семантична кольорова схема: зелений – обладнання в роботі – статус RUNNING, червоний – простій або поломка – статус DOWNTIME, жовтий – налаштування – статус SETUP. А при натисканні на картку обладнання у списку відкривається панель. Для цього використано патерн Modal Bottom Sheet. На відміну від відкриття нового вікна, шторка зберігатиме

контекст, де користувач бачить, що він все ще знаходиться у списку, але взаємодіє з конкретним об'єктом. Тут оператор матиме змогу переглянути історію та змінити статус.

Друга вкладка – сканування та додавання обладнання. Цей екран відповідає за ідентифікацію нових об'єктів. Сканування реалізовуватиметься з використанням бібліотеки CameraX. Поверх зображення з камери накладається напівпрозорий шар з вирізом-рамкою – це візуальна підказка, яка підсвідомо змушує оператора розташувати QR-код у центрі камери. Також передбачено кнопку ручного введення, яка відкриває діалогове вікно (патерн Modal Bottom Sheet) для ведення інвентарного номера з клавіатури, якщо код пошкоджено.

Враховуючи, що на сучасних виробництвах часто використовується тьмяне освітлення, інтерфейс розроблено в темній кольоровій гамі. Темний фон знижує навантаження на очі, а акцентні кольори десатуровані. Мета приглушеної палітри: уникнення ефекту візуальної вібрації та надмірної яскравості на темному фоні.

Такий підхід не лише покращує ергономіку, а й дозволяє заощаджувати заряд батареї на пристроях з OLED-екранами, що є важливим для роботи протягом повної робочої зміни.

Модуль планування та адміністративні функції об'єднані на екрані профілю користувача. План робіт відображається під інформацією про користувача. Тут важлива візуальна ієрархія, бо заголовок акцентує увагу на назві завдання, а статус використовує кольорове кодування – “Назначено”, “В процесі”, “Виконано”. При натисканні на завдання з’явиться модальне вікно з детальним описом та кнопками дії, що дозволять почати або завершити роботу. Також є можливість залишити коментар. У нижній частині профілю розміщена кнопка, яка відповідатиме за надсилання звіту за зміну.

Для забезпечення якісного досвіду користувача процес супроводжується зворотним зв'язком. Наприклад, при натисканні кнопка блокується, а на екрані може відображатися індикатор завантаження,

Після успішної відправки з'являється спливаюче повідомлення (Toast), яке інформує користувача про результат. Архітектурні компоненти Material Design, використані для побудови користувацького дизайну, наведені в таблиці 2.4.

Таблиця 2.4 – Використання компонентів Material Design для побудови інтерфейсу

Компонент	Використання	Призначення
NavigationBar	Головний екран	Забезпечує перемикання між списком, сканером та профілем
ModalBottomSheet	Деталі завдання, деталі обладнання	Відображення додаткової інформації без втрати контексту основного екрану
Card	Списки завдань та обладнання	Групування пов'язаної інформації в окремі візуальні блоки.
OutlinedTextField	Введення номера вручну, логін	Введення тексту з чіткою візуалізацією стану фокусу
Toast	Відправка звітів	Тимчасове повідомлення про успішну операцію

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

Для ефективного вирішення поставлених інженерних завдань обрана мова програмування має бути архітектурно-незалежною, повністю об'єктно-орієнтованою та мати вбудовану підтримку для розробки розподілених клієнт-серверних додатків. Важливою умовою є здатність технології забезпечувати високу надійність обробки даних, багатопоточність та безпечне управління пам'яттю, що є критичним для безперебійної роботи серверних компонентів.

Платформа Java повною мірою задовольняє зазначені критерії та узгоджується з вимогами до сучасних корпоративних систем. Завдяки концепції віртуальної машини досягається високий рівень кросплатформності, а сувора типізація дозволяє виявляти помилки ще на етапі компіляції. Фундаментальні концепції Java є логічно структурованими, що мінімізує час на їх освоєння та дозволяє розробнику максимально швидко перейти до продуктивного написання коду без втрати якості.

Окрім самої мови, створення надійної серверної частини вимагає застосування інфраструктурних фреймворків. Для реалізації бекенду використовується екосистема Spring Boot, яка бере на себе рутинні завдання з конфігурації та впровадження залежностей (Dependency Injection). Це дозволяє абстрагуватися від низькорівневих деталей налаштування сервера та зосередити ресурси безпосередньо на реалізації бізнес-логіки та проєктуванні архітектури.

Завершальним елементом етапу вибору інструментарію є визначення середовища розробки та систем автоматизації збірки. Написання вихідного коду здійснюється за допомогою інтегрованого середовища IntelliJ IDEA, яке забезпечує глибокий статичний аналіз коду та розширені можливості рефакторингу. Для управління бібліотеками та процесом компіляції застосовується система збірки Gradle, що гарантує відтворюваність програмного продукту на різних етапах його життєвого циклу.

3.1 Реалізація серверної частини на базі Spring Boot

Серверна частина інформаційної системи розроблена на базі платформи Java з використанням фреймворку Spring Boot [11]. Реалізація проєкту жорстко орієнтована на ефективну обробку виробничих даних та швидку інтеграцію. Програмний код структуровано за класичним багаторівневим патерном, що розділяє систему на рівень маршрутизації, рівень бізнес-логіки та рівень доступу до даних. Такий підхід забезпечує дотримання принципу єдиної відповідальності та полегшує підтримку кодової бази, детальна структура пакетів наведена на рисунку 3.1.

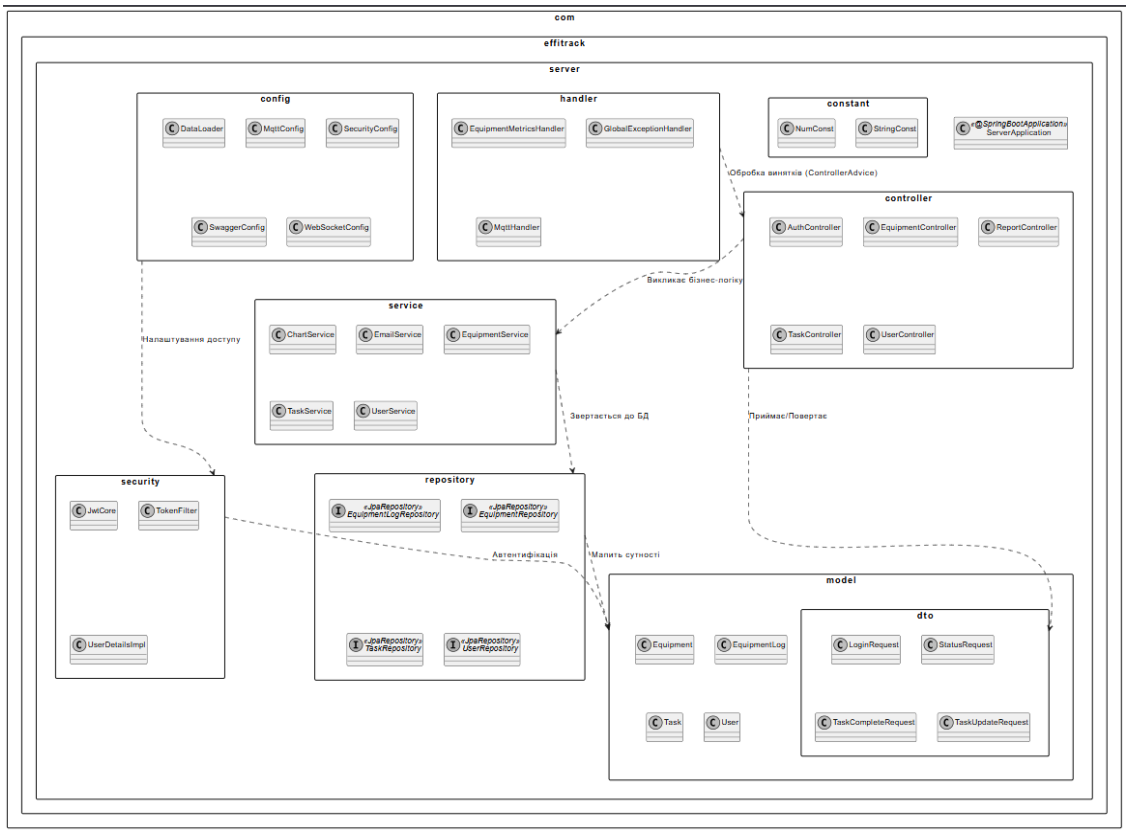


Рисунок 3.1 – Діаграма класів та пакетів серверної частини Spring Boot

Окрім базових архітектурних шарів, структура проєкту включає спеціалізовані пакети, що забезпечують відповідність системи стандартам корпоративних додатків. Пакет config містить класи конфігурації для налаштування безпеки, наприклад, SecurityConfig (Додаток А, додаток А.1),

автоматичної генерації документації API – SwaggerConfig, а також налаштування протоколів реального часу та Інтернету речей – WebSocketConfig, MqttConfig (Додаток А, додаток А.2 та А.3 відповідно), що розширює можливості системи для прямого моніторингу обладнання. Пакет handler містить клас GlobalExceptionHandler, який реалізує патерн глобального перехоплення винятків, гарантуючи, що мобільний клієнт завжди отримуватиме стандартизовані та зрозумілі відповіді про помилки замість системних збоїв, які називаються stack traces.

Система безпеки та розмежування прав доступу імплементована за допомогою пакета сек'юрності. Процес автентифікації базується на використанні стандарту JSON Web Token, за генерацію та валідацію якого відповідає клас JwtCore. Усі вхідні HTTP-запити проходять через спеціальний фільтр TokenFilter, який витягує токен із заголовка запиту, перевіряє його криптографічний підпис та ідентифікує користувача через сервіс UserDetailsImpl. Завдяки цьому сервер функціонує за безстановою (stateless) моделлю, не зберігаючи жодних даних про активні сесії у своїй пам'яті, що мінімізує споживання ресурсів та підвищує загальну безпеку інформаційної системи. Програмна реалізація компонентів безпеки наведена у Додатку А, додатки А.4 та А.5 відповідно.

Для детального розуміння механізму захисту програмного інтерфейсу розроблено діаграму послідовності процесу автентифікації та доступу до ресурсів. Як показано на рисунку 3.2, клієнт ініціює запит з обліковими даними, які проходять перевірку у сервісному шарі. Після успішної верифікації у базі даних компонент JwtCore генерує унікальний криптографічний токен. Під час кожного наступного звернення до закритої частини API спеціальний фільтр перехоплює запит, перевіряє підпис токена та ідентифікує оператора ще до того, як запит досягне контролера з бізнес-логікою.

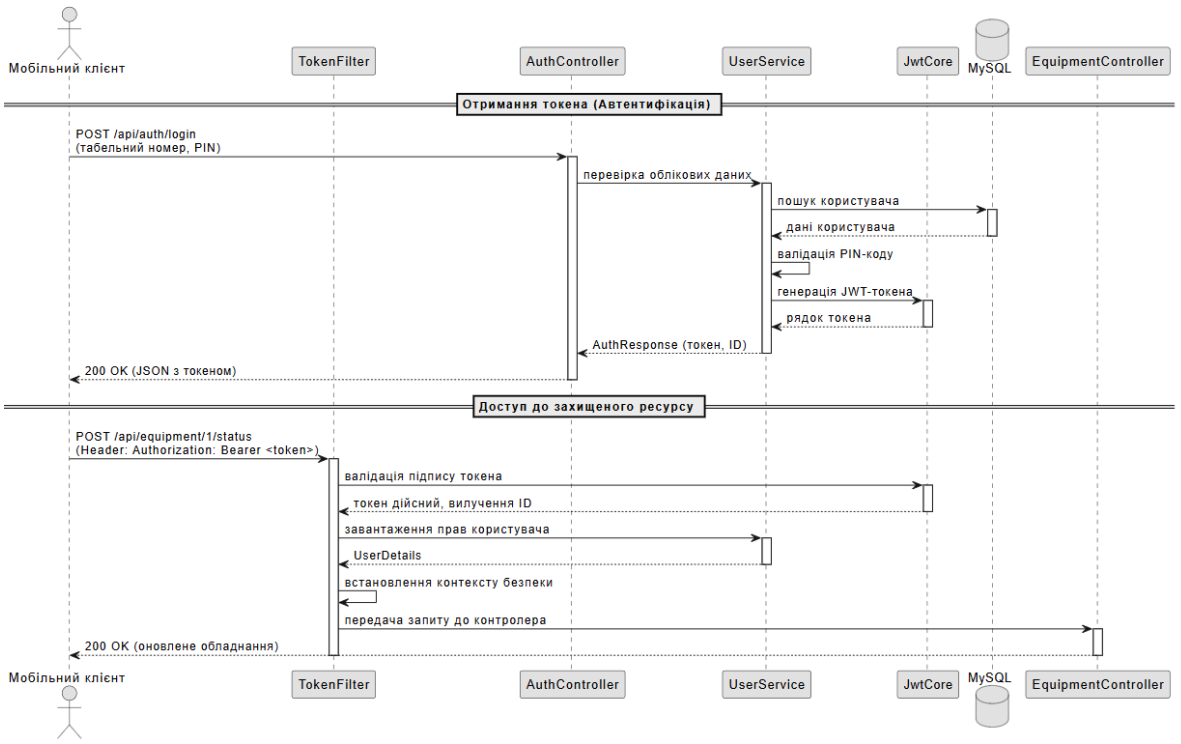


Рисунок 3.2 – Робота пакетів security та класу JwtCore

Уся складна бізнес-логіка системи зосереджена у сервісному шарі, що дозволяє ізолювати правила предметної області від механізмів доставки даних. Класи у пакеті сервісу, наприклад, EquipmentService (Додаток А, додаток А.6), TaskService (Додаток А, додаток А.7), відповідають не лише за базові CRUD-операції, а й за виконання комплексних транзакцій. Наприклад, операція зміни статусу обладнання передбачає перевірку валідності переходу, автоматичну фіксацію часових міток, генерацію нового запису у таблицю журналу EquipmentLog (Додаток А, додаток А.8) та перерахунок накопиченого часу простоїв чи корисної роботи. Завдяки використанню анотації @Transactional на рівні сервісів, фреймворк Spring гарантує транзакційну цілісність, якщо під час виконання багатоетапної операції виникне збій, усі зміни будуть скасовані, і база даних залишиться у консистентному стані.

API реалізовано за допомогою REST-контролерів, розміщених у пакеті controller. Кожен контролер використовує набір анотацій Spring Web: @RestController, @RequestMapping, @PostMapping тощо для маршрутизації HTTP-запитів до відповідних методів сервісу. Для забезпечення гнучкості

системи контролери упаковують результати виконання у клас ResponseEntity з пакета Spring Framework. Це дозволяє серверу точно керувати HTTP-статусами відповідей, що є ключовою вимогою до побудови правильної REST-архітектури. До прикладу, наведено реалізацію REST-контролера обладнання у Додатку А, додаток А.10.

Важливою архітектурною особливістю розробленої системи є підтримка комунікації в реальному часі, що критично для промислових систем моніторингу. Оскільки класична модель "запит-відповідь" протоколу HTTP не підходить для миттєвого отримання телеметрії, архітектуру було розширено технологіями MQTT та WebSockets.

Клас MqttConfig забезпечує підключення до брокера повідомлень для збору метрик або тригерів безпосередньо від апаратних датчиків на виробничій лінії. Отримані дані проходять через MqttHandler і за допомогою WebSocketConfig миттєво транслюються підключеним клієнтам. Це перетворює серверне API з пасивного сховища даних на реактивну подієво-орієнтовану систему.

Так як під час розробки промислове обладнання було відсутнє, а сторонніх API для отримання даних з обладнання не було знайдено, класи MqttHandler (Додаток А, додаток А.11) та WebSocketConfig, MqttConfig, лістинги, які були згадані вище, наводяться як каркас можливої реалізації.

Для забезпечення функціонування у проєкті розроблено класи, які відповідають за симуляцію метриків, а саме EquipmentSimulatorService (Додаток А, додаток А.12) та EquipmentMetricsHandler (Додаток А, додаток А.13). Сервер симуляції обладнання відповідає за генерацію причини зупинки нормальної роботи верстатів, а EquipmentMetricsHandler рахує час кожного простою у реальному часі, який триває поки оператор вручну не змінить статус у мобільному додатку, або поки EquipmentSimulatorService знову не переведе обладнання у стан штатної роботи.

Для забезпечення стабільної роботи мобільних клієнтів реалізовано механізм глобальної обробки винятків. Клас GlobalExceptionHandler (Додаток

А, додаток А.14), позначений анотацією @ControllerAdvice, перехоплює всі помилки на рівні додатка. Замість стандартного HTML-стеку помилок сервера цей обробник формує і повертає клієнту єдиний стандартизований JSON-об'єкт із кодом помилки та зрозумілим текстовим повідомленням. Додатково, для спрощення етапу інтеграції з клієнтською частиною, розроблений API автоматично документується.

SwaggerConfig (Додаток А, додаток А.15) генерує інтерактивну веб-специфікацію OpenAPI, де докладно описані всі доступні ендпоінти, формати запитів та схеми авторизації, що значно пришвидшує процес тестування та налагодження системи.

3.1.1 Налаштування ORM Hibernate

Взаємодія з базою даних побудована на основі специфікації JPA та ORM-фреймворку Hibernate [20]. Сутності предметної області, такі як User, Equipment, Task та EquipmentLog, локалізовані у пакеті моделей.

Використання технології об'єктно-реляційного відображення дозволило спроєктувати таблиці бази даних у вигляді Java-класів, абстрагуючись від прямого написання низькорівневих SQL-запитів.

Для виконання операцій з базою даних створено інтерфейси в пакеті репозиторію, які успадковують JpaRepository, що забезпечує автоматичну генерацію базових запитів. У Додатку А, додаток А.16, наведено приклад програмної реалізації сутності обладнання з відповідними анотаціями мапінгу.

Для оптимізації мережевого обміну та приховування внутрішньої структури доменних моделей передача інформації між сервером та мобільним клієнтом здійснюється через спеціалізовані об'єкти запитів, згруповані у підпакеті model.dto, наприклад, LoginRequest, TaskUpdateRequest. Це гарантує, що клієнт отримує та відправляє лише необхідні дані – DTO (Data Transfer Object). Фрагмент вхідного коду DTO наведено у Додатку А, додаток А..17.

3.1.2 Інтеграція Large Language Model Google Gemini для аналізу простоїв

Футуристична концепція ШІ сьогодні перетворилася на один із компонентів сучасних мобільних додатків. Інтеграція інструментів, обладнаних штучним інтелектом, не обов’язкова, проте є одним із ключів для підтримки конкурентоспроможності, що покращує досвід користувача та сприяє ефективності роботи.

Використання таких програмних рішень на виробництві дозволить автоматизувати виконання завдань, аналізувати великий обсяг даних, до прикладу, аналізувати простої обладнання.

Під час розробки програмного комплексу було вирішено інтегрувати Large Language Model – офіційний інструмент Google Gemini API [21]. Першим кроком є впровадження безпечного доступу до запитів. Для цього в кабінеті Google Cloud Console (Google AI Studio) було згенеровано API-ключ, який додано до файлу з властивостями. Оскільки поточний план є базовим, обрана модель Gemini 1.5 Flash дозволяє надсилати до 15 запитів на хвилину, що повністю задовольняє потреби для початкової роботи застосунку.

Архітектурна реалізація клієнта розроблена на рівні серверного шару. Метою впровадження стала автоматизація первинного технічного аналізу простою обладнання. Компонент AiAnalysisService (Додаток А, додаток А.18) – клієнтський сервер, що виконує HTTP-взаємодію з хмарою API Google. Сервер відповідає за формування запитів і десеріалізацію отриманих відповідей. Результат виконання, отриманий від ШІ, використовується у вже раніше згадуваному EquipmentService. Якщо обладнання зупиняється, система збирає дані з активного логу простою. Це передбачає запис системної причини та текстового коментаря, який оператор вводить вручну через мобільний додаток. Дані інтегруються із системною інструкцією, а саме константою AI_SYSTEM_PROMPT, яка робить запит до моделі Gemini, щоб вона виступила як фахівець з ремонту обладнання. Необхідно надати коротку відповідь українською мовою у 2–3 реченнях. Результат базується на трьох основних

критеріях: орієнтований час ремонту, оцінка вартості витрат, рекомендація для майстра.

Основна цінність впровадження ШІ-інтеграції полягає в тому, що створений аналіз слугує інтелектуальною підказкою для оператора в режимі реального часу. Замість того, щоб чекати на майстра або інженера первинної діагностики, оператор одразу на своєму мобільному пристрої отримує рекомендацію. Це дозволяє оператору оцінити масштаб поломки, зрозуміти орієнтовний час та виконати першочергові кроки для усунення несправності. Наприклад, перезапустити модуль, перевірити та відрегулювати механічні люфти між гвинтовим стрижнем та кріпленням.

Такий підхід мінімізує людський фактор, допомагає оператору швидше прийняти ефективне рішення безпосередньо на робочому місці та значно скорочує загальний час простою обладнання.

3.1.3 Реалізація SMTP-клієнта для надсилання звітів

Один з найпоширеніших способів надсилати електронні листи надійно та стандартизовано – це SMTP (Simple Mail Transfer Protocol). SMTP – простий протокол передачі пошти, який автентифіковано надислає листи клієнту від сервера, і навпаки, або від поштового шлюза [21]. Щоб налаштувати автоматизацію зовнішніх комунікацій у розроблюваній системі моніторингу, додано функцію надсилання звітів. Ця функція генерує підсумковий аналітичний репорт з графіками та ділиться поштою з керівництвом або технічною службою відповідно до вимог виробничого процесу.

Spring Boot надає потужний інструмент для реалізації поштового протоколу – spring-boot-starter-mail. Стартер базований на специфікації Jakarta Mail, раніше – JavaMail API, а JavaMailSender – головний інструмент для роботи з поштовим сервером. Інтерфейс для надсилання листів ін'єктується в серевісний рівень застосунка за рахунок анотації @Autowired.

У класі EmailService (Додаток А, додаток А.19) інкапсульована бізнес-логіка формування та відправки звітів. Так як звіти включають HTML-верстку таблиць та вкладення у вигляді графіків, а це складна структура, замість базового SimpleMailMessage у коді використовується MimeMessage. Mime – це стандарт, який розширює можливості електронної пошти для передачі не лише тексту, а й іншого контенту, наприклад, вкладені файли, відправка листів з форматуванням, обробка мультимедійного контенту. Етапи відправки звітності у мобільному додатку наведено в таблиці 3.1

Таблиця 3.1 – Етапи відправки звітності

Етап алгоритму	Метод	Призначення та деталі реалізації
Ініціалізація відправника	helper.setFrom(fromEmail)	Фіксує пошту системи з файлу конфігурації для успішного проходження верифікації.
Заповнення метаданих	helper.setTo(to) helper.setSubject(subject)	Встановлює тему та отримувача. Email майстра визначається динамічно.
Формування тіла листа	helper.setText(body, true)	Приймає згенерований HTML-код. Параметр true активує рендеринг листа як вебсторінки з тегами та стилями.
Вкладення графіків	helper.addAttachment(...)	Конвертує масив байтів діаграми у ресурс ByteArrayResource та прикріплює до листа.
Транспорт даних	mailSender.send(message)	Ініціює SMTP-сесію, виконує безпечний конект з сервером Google та здійснює відправку пакета.

HTTP POST-запит від мобільного застосунку обробляє компонент ReportController (Додаток А, додаток А.20), який запускає збір даних і передає їх до модуля ChartService (Додаток А, додаток А.21) для візуалізації. У підсумку створюється структуроване поштове повідомлення. HTML-лист «Звіт за зміну» з вкладеним графіком продуктивності оператора за тиждень наведено на рисунках 3.3 і 3.4. Інтерфейс листа «Звіт по обладнанню» з інтегрованим AI-аналізом і графіком ефективності верстата за зміну показано на рисунках 3.5 і 3.6.

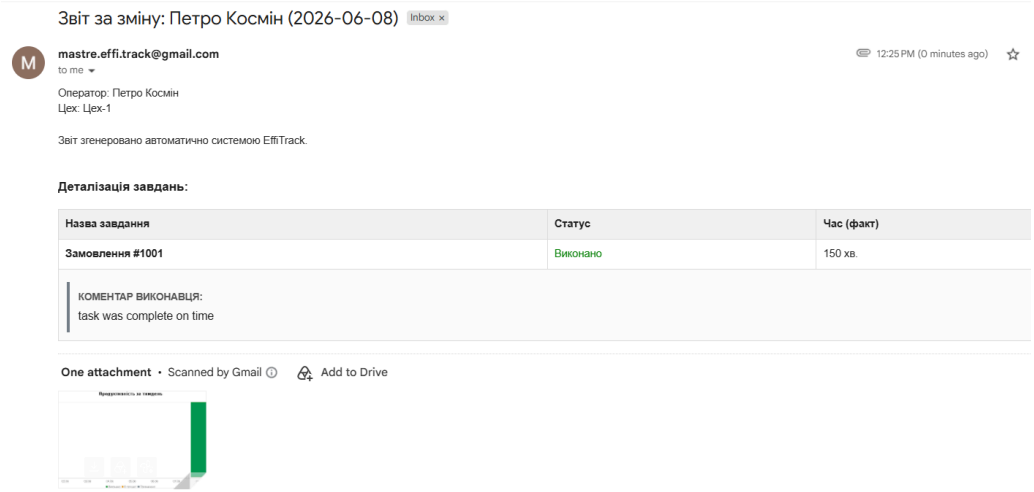


Рисунок 3.3 – Згенерований HTML-листа “Звіт за зміну”

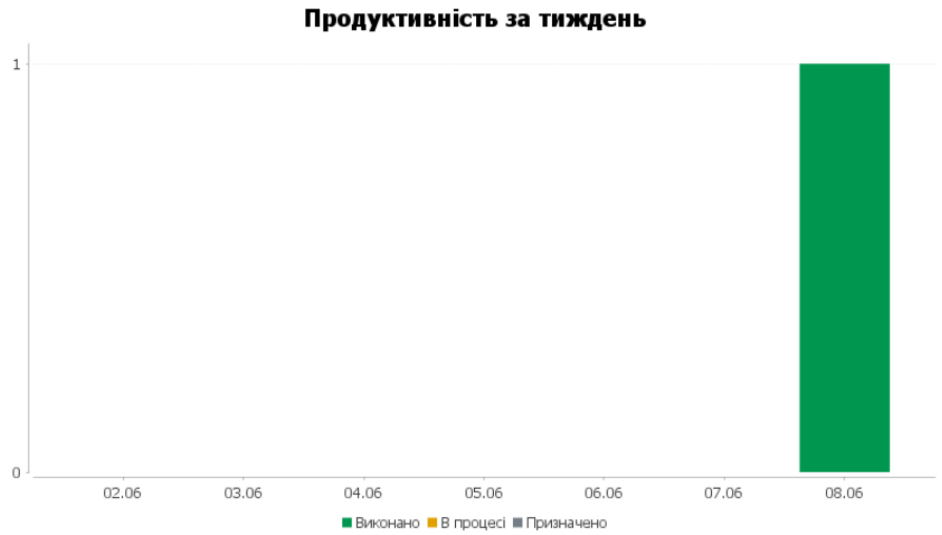
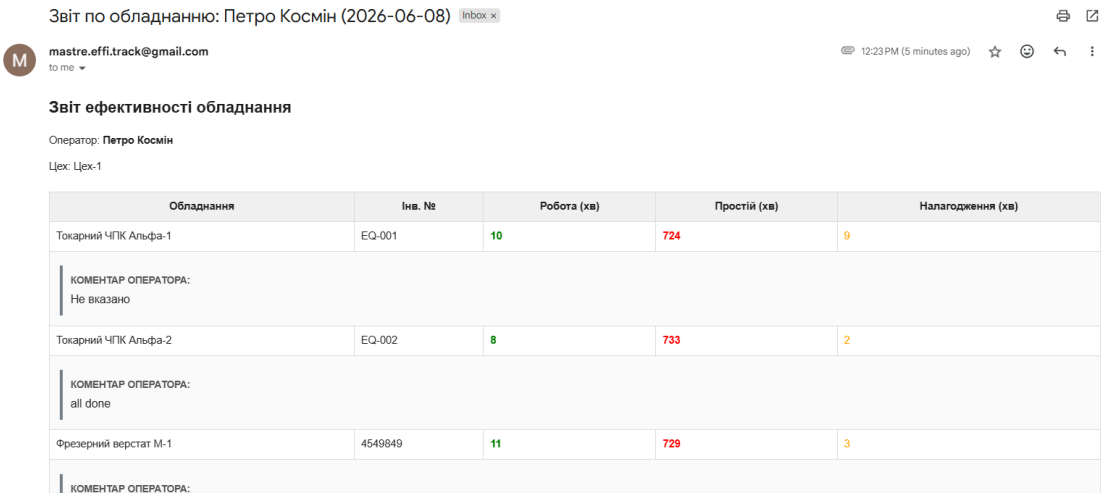


Рисунок 3.4 – Графік продуктивності за тиждень



Звіт по обладнанню: Петро Космін (2026-06-08) Inbox ×

mastre.effi.track@gmail.com
to me

12:23 PM (5 minutes ago)

Звіт ефективності обладнання

Оператор: **Петро Космін**

Цех: Цех-1

Обладнання	Інв. №	Робота (хв)	Простій (хв)	Налагодження (хв)
Токарний ЧПК Альфа-1	EQ-001	10	724	9
КОМЕНТАР ОПЕРАТОРА: Не вказано				
Токарний ЧПК Альфа-2	EQ-002	8	733	2
КОМЕНТАР ОПЕРАТОРА: all done				
Фрезерний верстат М-1	4549849	11	729	3
КОМЕНТАР ОПЕРАТОРА:				

Рисунок 3.5 – Згенерований HTML-листа “Звіт по обладнанню”

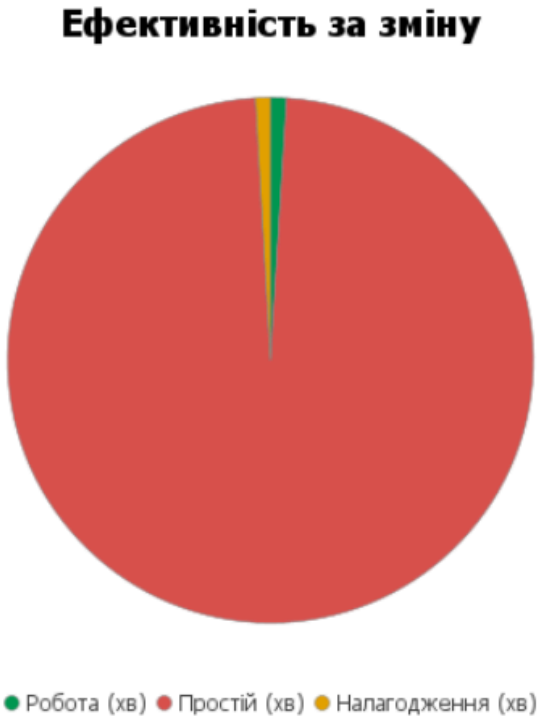


Рисунок 3.6 – Графік ефективності за зміну вкладений до звіту по обладнанню

3.2 Розробка мобільного Android-додатку

Клієнтська частина мобільної системи розроблена у вигляді нативного Android застосунку, мовою програмування обрано Kotlin, яка є сучасним стандартом для написання софту під операційну систему андроїд [6]. Такий

підхід забезпечить високу безпеку типів, асинхронне та неблокуюче програмування за допомогою легковагових конструкцій – корутин.

Додаток спроєктований з використанням архітектурного патерну MVVM (Model-View-ViewModel). Патерн дозволяє чітко відокремити бізнес-процеси від роботи з даними та управління станами візуального відображення [11]. Взаємодія компонентів на клієнтській стороні MVVM відображена на рисунку 3.7.

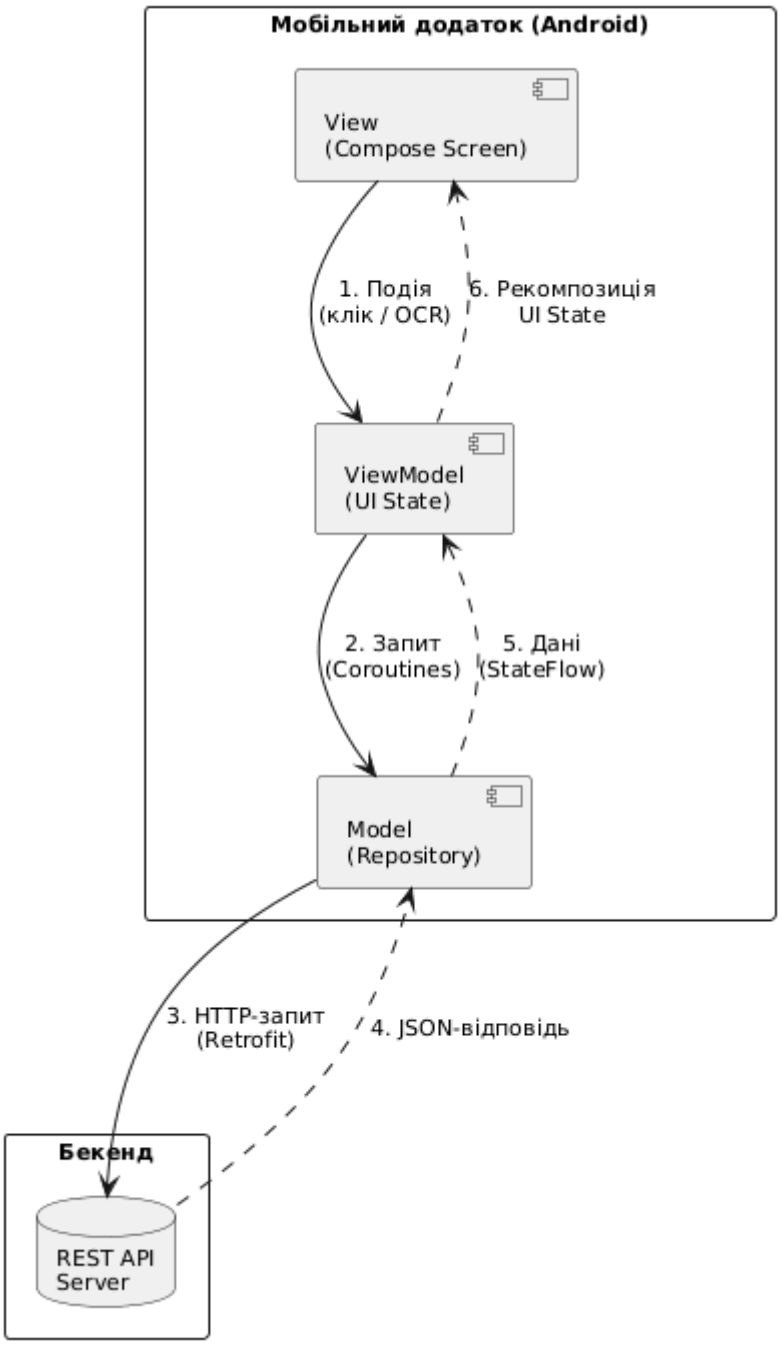


Рисунок 3.7 – Взаємодія компонентів у Android-додатку

Графічний інтерфейс користувача побудовано за допомогою сучасного декларативного фреймворку Jetpack Compose та компонентів Material Design 3 [7]. На відміну від класичного імперативного підходу з використанням XML-макетів, де вручну потрібно синхронізовувати стан даних із елементами інтерфейсу, Compose дозволяє описувати інтерфейс за допомогою composable-функцій, які автоматично оновлюються при зміні пов'язаного стану.

Це значно скорочує обсяг шаблонного коду та спрощує створення складних динамічних екранів, таких як списки обладнання з фільтрацією або модальні нижні панелі для перегляду деталей завдань. Візуальне відображення екранів застосунку та елементів навігації наведено в Додатку Б. Перелік ключових бібліотек, використаних при розробці додатка, наведено у таблиці 3.2.

Таблиця 3.2 – Основні бібліотеки та інструментальні засоби мобільного додатку

Назва бібліотеки	Призначення
Jetpack Compose (UI, Material 3)	Створення сучасного декларативного інтерфейсу користувача з використанням компонентів дизайну Material Design 3.
Retrofit 2	Забезпечення клієнт-серверної взаємодії, виконання REST API запитів.
Moshi	Серіалізація та десеріалізація даних у форматі JSON для обміну інформацією із сервером.
Navigation Compose	Організація маршрутизації, управління стеком навігації та переходами між екранами додатка.
Accompanist Permissions	Управління системними дозволами (наприклад, запит доступу до камери) безпосередньо у декларативному UI Compose.

Застосування однонаправленого потоку даних для керування станом інтерфейсу є ключовою архітектурною особливістю розробленого застосунку. Відповідні класи ViewModel містять реактивні потоки MutableStateFlow, які інкапсулюють усі змінні стану, включаючи прапорці завантаження, списки завдань та повідомлення про помилки [13].

Компоновані функції приймають лише поточний зріз даних як об'єкт State та відображають його на екрані; вони є чистими та не мають власного прихованого стану. ViewModel отримує подію у вигляді зворотного виклику, коли користувач взаємодіє з інтерфейсом (натискає кнопку, сканує код), що змінює стан та ініціює ізольовану рекомпозицію інтерфейсу користувача. Розсинхронізація між візуальним відображенням та бізнес-логікою повністю усувається за допомогою цього підходу.

Передача даних між різними частинами системи здійснена за допомогою бібліотеки Navigation Compose. В межах додатку створено особливий навігаційний граф NavHost (Додаток А, додаток А.22) який визначає шляхи для кожного екрану авторизації головного меню вікна сканування та аналітичних звітів. Для того щоб забезпечити типову безпеку і уникнути можливих помилок при передачі аргументів особливо унікального ідентифікатора верстата з екрану сканера до екрану деталей простою використано механізм безпечної навігації Safe Args. Керування життєвим циклом екранів разом із збереженням історії навігації наприклад при натисканні кнопки Назад повністю передане операційній системі За допомогою вбудованого NavController.

3.2.1 Реалізація мережевої взаємодії з REST API

Зв'язок між мобільним клієнтом та сервером інформаційної системи побудований на архітектурному стилі REST API з використанням інструментів бібліотеки Retrofit 2. Програмний інтерфейс для взаємодії від імені клієнта Android Характеризується через централізований інтерфейс Kotlin з назвою ApiService (Додаток А, додаток А.23). У цьому інтерфейсі кожна точка сервера

відповідає певній функції яка забезпечує асинхронне виконання запитів у фонових корутинах без перешкоджання основному потоку роботи користувацького інтерфейсу. Опис маршрутів включає процедури що стосуються перевірки операторів призначення обладнання пошуку запитів для верстатів на основі номерів запасу оновлення статусу простоїв і отримання списків пов'язаних із виробничими завданнями.

Однотонний об'єкт RetrofitClient (Додаток А, додаток А.24) розроблений для централізації конфігурації мережевого рівня шляхом керування екземпляром Retrofit. Конфігурація клієнта включає конвертер Moshi для автоматичної серіалізації та десеріалізації об'єктів передачі даних на основі типобезпечних характеристик мови Kotlin. Конфігурація мережевого клієнта OkHttpClient встановлена на часове обмеження 30 секунд для операцій підключення, читання та запису, щоб забезпечити промисловим застосункам безперервну роботу в мережі, водночас надаючи буфер для запобігання зависанню або виходу застосунків з ладу під час збоїв у мережі. Збірка клієнта включає плагін-модуль реєстрації трафіку HttpLoggingInterceptor, який надає розробникам API інформацію для налагодження структури HTTP-пакета під час розробки на рівні деталізації.

У процесі проєктування клієнтського рівня особливий акцент було зроблено на безпеці та стійкості застосунку до критичних помилок мережі. Особлива увага була приділена спеціальній функції, відомій як AuthInterceptor, яка реалізує інтерфейс Interceptor у бібліотеці OkHttp. Основне призначення Interceptor – автоматизувати процес аутентифікації. Interceptor динамічно отримуватиме доступ до збереженого JSON Web Token із сеансу користувача, і якщо JWT знайдено, Interceptor додасть цей JWT до заголовка Authorization: Bearer у всіх вихідних запитах.

Крім автоматизації авторизації, в методі intercept класу AuthInterceptor є дворівнева система обробки помилок і контролю мережі. Системні помилки введення-виведення обробляються за допомогою блоку try-catch, що перехоплює винятки відсутності фізичного зв'язку. Метод isNetworkAvailable

перевіряє доступні способи передачі даних за допомогою системного сервісу ConnectivityManager. Якщо на пристрої немає інтернету, додаток перенаправляє користувача на екран статусів з повідомленням про відсутність мережі. Якщо зв'язок є, але сервер не відповідає, з'являється повідомлення про тимчасову непрацездатність серверної частини, а клієнту повертається штучний HTTP-код 504 (No Internet Connection).

Обробка критичних статусів відповідей сервера: інтерцептор аналізує HTTP-коди відповідей у реальному часі. Якщо отримуються коди 401 або 403 (закінчення терміну дії токена або відсутність прав доступу), об'єкт UserSession автоматично очищає локальні дані користувача. Головний потік через Handler(Looper.getMainLooper()) відображає інформаційне спливаюче вікно (Toast), а навігаційний стек скидається для примусової переадресації на екран авторизації MainActivity. При отриманні відповідей з кодом 500 і вище виконується автоматичне перенаправлення на екран відображення помилок сервера за допомогою методу redirectToStatusScreen.

Така реалізація мережевого шару гарантує високу відмовостійкість мобільної системи, захищає додаток від аварійних завершень під час втрати сигналу в закритих приміщеннях і забезпечує прозору логіку управління сесіями користувачів.

3.2.2 Інтеграція модуля оптичного розпізнавання інвентарних номерів

Щоб зменшити вплив людини, пришвидшити процес фіксації простоїв промислового обладнання та повністю виключити помилки при ручному введенні даних, у мобільний додаток інтегрували підсистему безконтактного розпізнавання об'єктів. Модуль працює на основі технології оптичного розпізнавання символів. Функціональність підсистеми реалізована завдяки взаємодії сучасного низькорівневого API AndroidX CameraX та алгоритмів нейромережевого розпізнавання тексту з бібліотеки Google ML Kit [10].

Технологічний процес сканування та ідентифікації інвентарних номерів у реальному часі починається з динамічного керування дозволами платформи.

Доступ до апаратних ресурсів камери є важливою операцією з точки зору безпеки Android. У декларативному інтерфейсі Jetpack Compose за допомогою інструментів Accompanist Permissions реалізовано інтерактивний запуск перевірки прав на наявність дозволу системи. Якщо дозвіл відсутній, користувачу показується модальне вікно з поясненням необхідності доступу. Це відповідає принципам створення сучасного користувацького досвіду.

Наступний етап – конфігурація життєвого циклу камери та відеопотоку. CameraX прив'язується до життєвого циклу екрана додатка, що запобігає витокам оперативної пам'яті. Це автоматично звільняє апаратні ресурси при згортанні програми. Стрімінг відеопотоку здійснюється через об'єкт PreviewView, який інтегрується в декларативне дерево Compose за допомогою мосту AndroidView. Для паралельної обробки візуальної інформації створюється конвеєр аналізу кадрів ImageAnalysis. Він налаштований на режим ізоляції та обробки останнього доступного кадру для забезпечення стабільної частоти кадрів і плавності роботи інтерфейсу.

Локальний аналіз графічних даних виконується шляхом конвертації кожного дискретного кадру у внутрішній сумісний формат InputImage. Це враховує поточний кут повороту пристрою у просторі, і кадр надсилається на обробку локальному екземпляру класу TextRecognizer. Важливою перевагою обраного інструменту Google ML Kit є те, що розпізнавання виконується безпосередньо на графічному або центральному процесорі мобільного пристрою. Локальний запуск нейромережі повністю усуває необхідність передачі великого медіа-трафіку на віддалений сервер. Це гарантує миттєву швидкість аналізу і забезпечує безперебійну роботу системи в ізольованих або екранованих залізобетонних конструкціях промислових цехів підприємства.

На фінальному етапі здійснюється фільтрація та валідація результатів за допомогою регулярних виразів, оскільки сирий масив текстових блоків, розпізнаний нейромережею, містить багато стороннього інформаційного шуму у вигляді маркувань на обладнанні, інструкцій або серійних номерів інших елементів. Для виділення цільового інвентарного номера верстата розроблено

алгоритм фільтрації на основі регулярних виразів (Regex), який аналізує текстовий потік за специфічною маскою, що відповідає внутрішньому стандарту кодування основних засобів на конкретному виробництві. Як тільки розпізнаний рядок успішно проходить валідацію за регулярним виразом, додаток генерує тактильний вібровідгук для інформування оператора про успішне зчитування, призупиняє та закриває відеопотік камери для заощадження заряду батареї, автоматично підставляє валідований інвентарний номер у пошукову модель архітектурного шару ViewModel та ініціює асинхронний REST-запит до ендпоінту сервера для миттєвого отримання актуальних даних про знайдений верстат, його поточний статус та закріплені за ним оперативно-виробничі завдання.

4 ТЕСТУВАННЯ СИСТЕМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ ІНФРАСТРУКТУРНОЇ ІНТЕГРАЦІЇ

Функціональне тестування компонентів інформаційної системи, експериментальна верифікація інтеграційних процесів і оцінка ефективності впровадження мобільного клієнта та серверної інфраструктури у виробниче середовище становлять основні етапи розробки. На етапі інженерного проєктування необхідно підтвердити відповідність програмного забезпечення технічним вимогам щодо швидкості обробки транзакцій, безпеки передавання даних і відмовостійкості архітектурних модулів. Комплексне тестування дозволяє своєчасно виявити та усунути архітектурні дефекти, оптимізувати мережеві запити та підтвердити готовність системи до промислової експлуатації в умовах виробництва.

Експериментальний аналіз і розгортання системи охоплюють кілька послідовних рівнів перевірки: від ізольованого тестування програмних інтерфейсів до інтеграційного моделювання поведінки всієї системи керування. На початковому етапі здійснюється функціональне тестування кінцевих точок сервера із застосуванням автоматизованих інструментів для перевірки логіки автентифікації, коректності серіалізації даних за допомогою конвертера Moshi та стабільності обробки запитів модулем Spring Boot. Додатково досліджується інтеграція мобільного додатка з локальними нейромережами комп'ютерного зору та хмарним сервісом Google Gemini API. Такий підхід забезпечує кількісну оцінку приросту продуктивності операторів і дає змогу сформулювати висновки щодо економічної та технологічної доцільності впровадження програмного продукту у виробничих процесах.

4.1 Функціональне тестування REST API

Функціональне тестування програмного комплексу є ключовим етапом верифікації, що гарантує коректну роботу серверної частини, обробку

бізнес-логіки та надійність інтеграції між мобільним додатком, штучним інтелектом і базою даних. Для автоматизованого тестування HTTP-запитів використовували середовище Postman. Тестування проводилося шляхом надсилання запитів до REST-сервера на Spring Boot з подальшим аналізом JSON-відповідей, перевіркою HTTP-статусів і контролем відповідності даних у MySQL [11].

Модуль автентифікації операторів було протестовано через кінцеву точку /api/auth/login. Для перевірки застосовувалися POST-запити з валідними та невалідними даними облікового запису. У разі коректних даних сервер повертав HTTP 200 і JWT-токен, рисунок 4.1. За некоректних даних сервер коректно обробляв винятки безпеки, повертаючи контрольовану помилку з HTTP 401, рисунок 4.2, що підтверджує надійність захисту від несанкціонованого доступу.

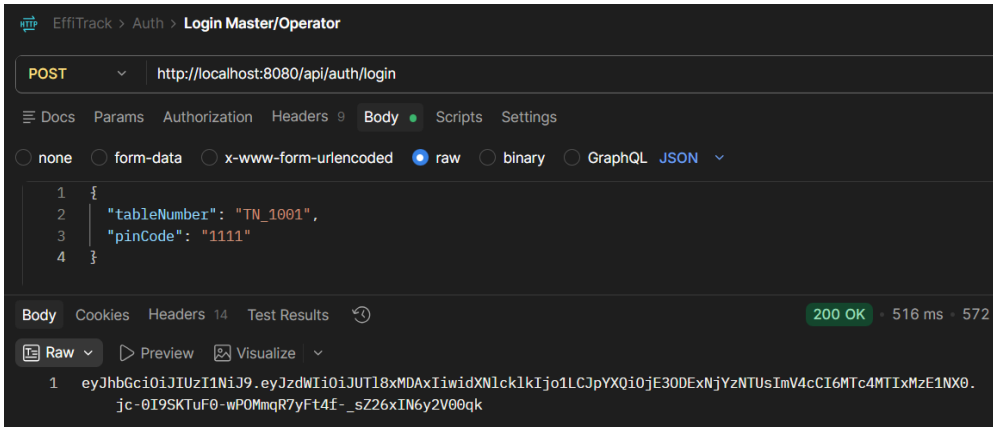


Рисунок 4.1 – Автентифікація з валідними даними

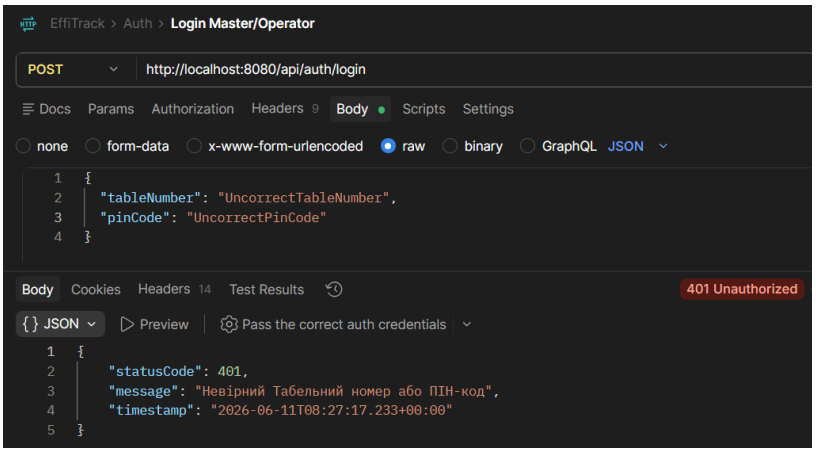


Рисунок 4.2 – Автентифікація з некоректними даними

GET-запити для отримання профілю користувача, закріпленого обладнання та списків завдань були протестовані з використанням унікального ідентифікатора оператора як параметра шляху. Тестування в Postman підтвердило, що Hibernate коректно виконує вибірку з бази даних, трансформує зв'язки MySQL у Kotlin-класи та повертає структуровані JSON-масиви без втрати цілісності даних, рисунок 4.3 [16].

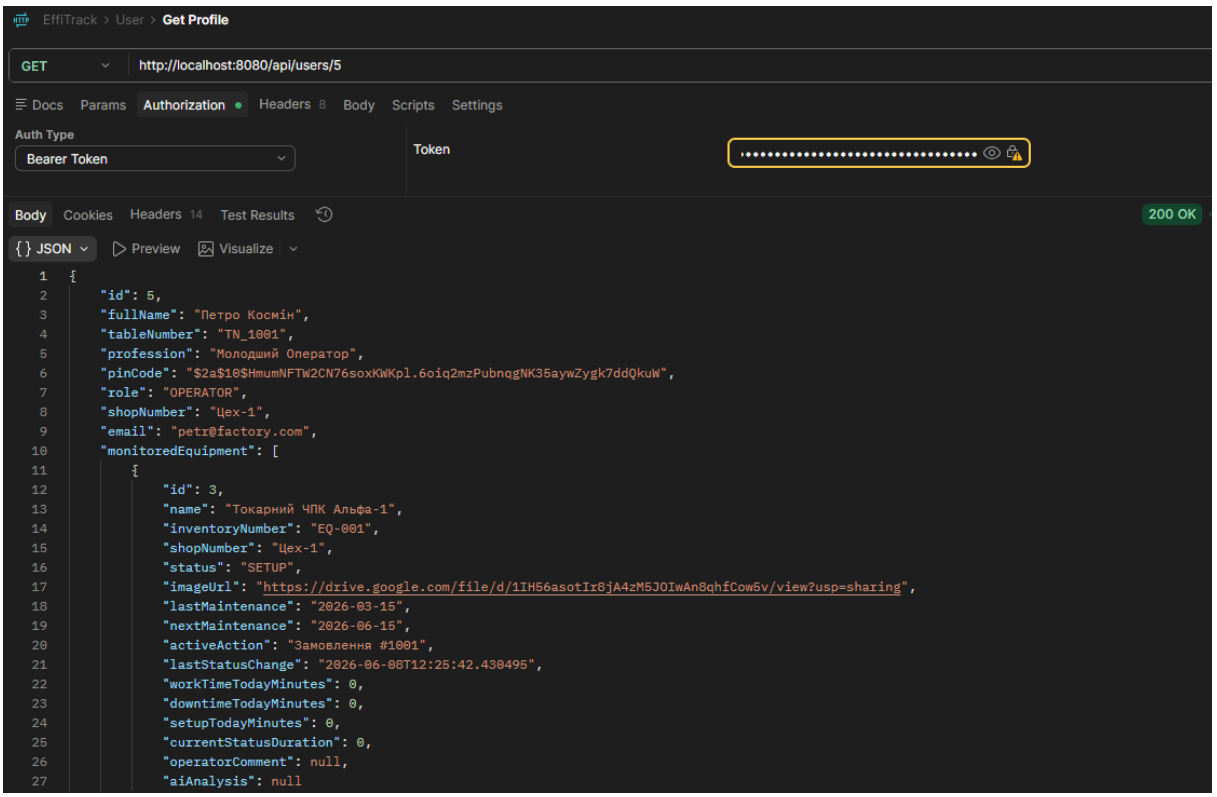


Рисунок 4.3 – Отримання профілю користувача

Інтеграційне тестування модуля комп'ютерного зору мобільного додатка проводилося з особливою увагою до перевірки алгоритмів Google ML Kit і CameraX. Для тестування використовувалися зображення інвентарних номерів за різних умов освітлення, кутів і ступенів зношеності маркування. Результати підтвердили високу точність розпізнавання символів і коректну роботу регулярного виразу для виділення цільового інвентарного номера. Процес

успішного сканування інвентарного у мобільному додатку наведено на рисунку 4.4.

Функціональне тестування інтеграції з хмарним сервісом Google Gemini API проводилася паралельно з іншими тестами.

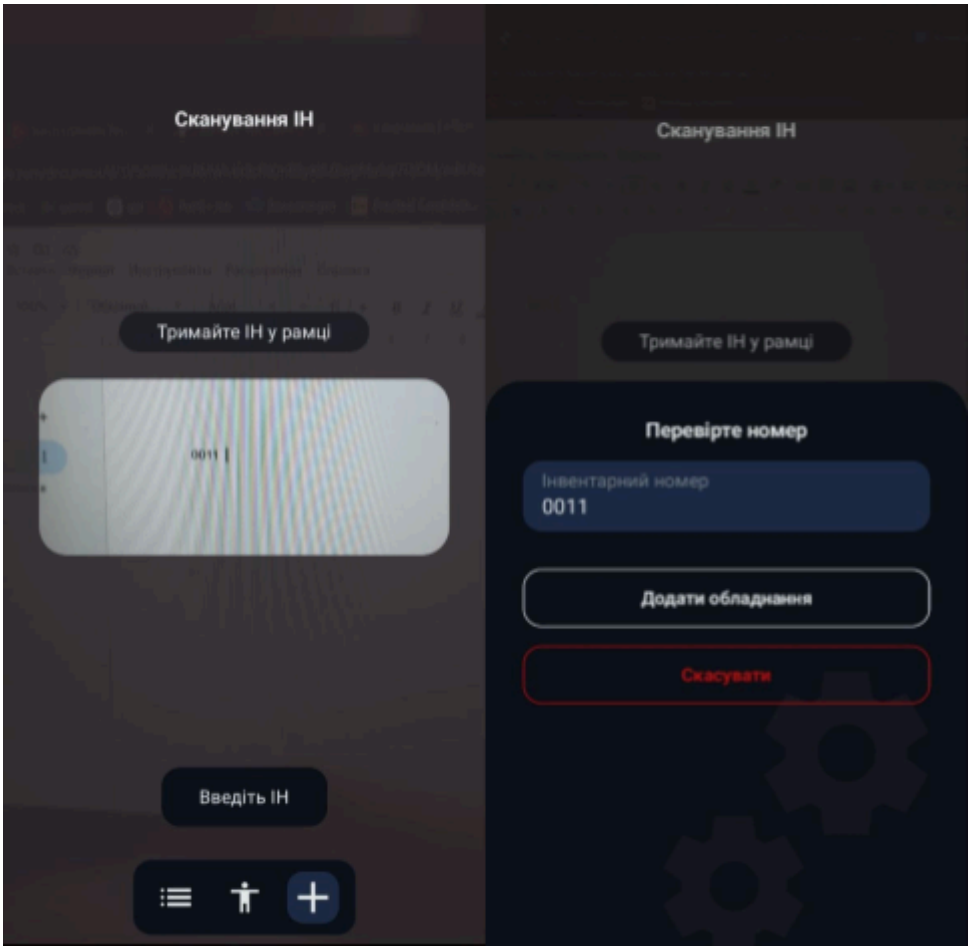


Рисунок 4.4 – Сканування інвентарного номера

Під час моделювання запиту типу Post з оновлення статусу обладнання з об'єктом EquipmentStatusUpdate сервер успішно ініціював звернення до нейромережі. Тестування включало аналіз швидкості обробки промптів і релевантності відповідей штучного інтелекту. Результати засвідчили, що Gemini API коректно інтерпретує коди помилок і логи простоїв, генеруючи чіткі підказки для оператора, які сервер трансформує у JSON для мобільного клієнта та дублює через SMTP на електронну пошту майстра.

Стрес-тестування логіки роботи клієнтського перехоплювача AuthInterceptor при отриманні помилкових статусів від сервера було проведено

на завершальному етапі. За допомогою імітаційних серверів Postman Mock підтверджено, що мобільний додаток коректно обробляє коди 401, 403 і 500, рисунок 4.5, не викликає аварійних завершень і перенаправляє користувача на екран авторизації або повідомлення про мережеву помилку.

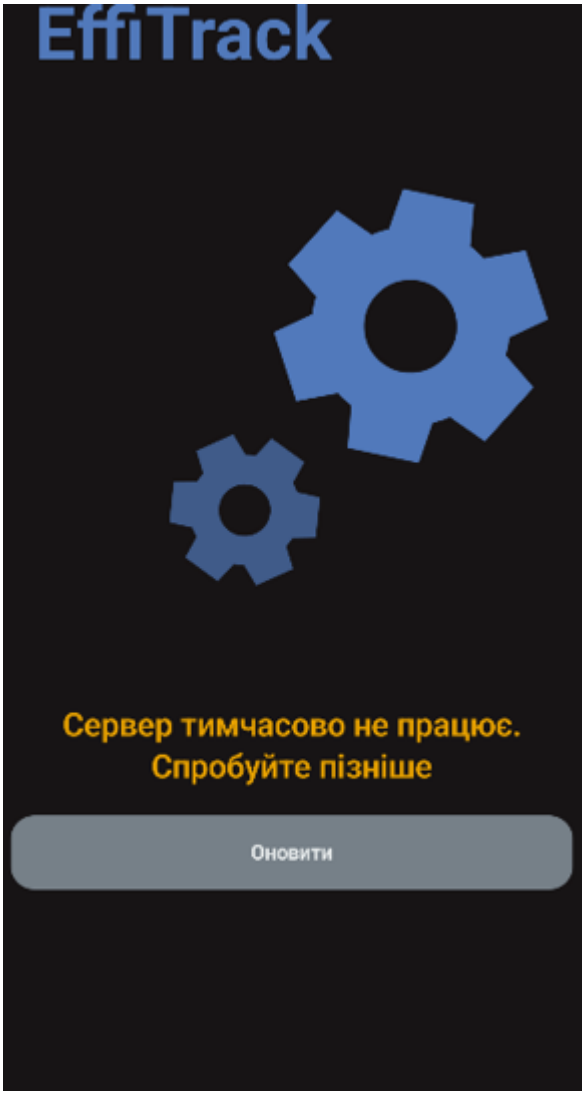


Рисунок 4.5 – Перенаправлення на екран помилки

4.2 Інфраструктурна проблема SMTP-клієнта

Тестування підсистеми SMTP-клієнта, яка автоматично генерує та надсилає електронні звіти про простої обладнання майстру цеху, стало важливим етапом інтеграційного контролю якості. У ході тестування було виявлено критичний інфраструктурний збій: у локальному середовищі localhost

відправка листів здійснювалася успішно, проте після розгортання на віддаленій платформі Render транзакції завершувалися помилкою тайм-ауту з'єднання MailSendException [22].

Первинна гіпотеза передбачала, що проблема виникає через надмірне навантаження під час генерації великих емейл-повідомлень із графічними елементами та фотографіями. Локальний комп'ютер обробляв ці файли швидко, тоді як обмежені ресурси хмарного сервера могли спричинити труднощі. Аналіз відповідних форумів підтвердив цю можливість. Проте оптимізація алгоритму формування листів і стиснення зображень проблему не усунула: помилка тайм-ауту залишалася.

Для подальшого аналізу було повторно вивчено технічну документацію та параметри безпеки мережевих протоколів [22]. Наступним кроком стало перенесення налаштувань автентифікації у змінні оточення Render, переведення SMTP-клієнта на порт 465 та активація SSL/TLS із використанням App Password. Проте навіть після впровадження цих змін з'єднання з поштовим сервером не встановлювалося.

Остаточний аналіз інфраструктури хмарного хостингу засвідчив, що причиною проблеми є обмеження безкоштовного тарифу Render. Згідно з політикою безпеки, на безкоштовних серверах заблоковано вихідний SMTP-трафік для запобігання спаму. Програмний модуль функціонує коректно, і для вирішення проблеми необхідно перейти на виділений серверний тариф Render.

4.3 Охорона праці персоналу при експлуатації системи

Вимоги до охорони праці поширюються на всіх суб’єктів на виробництві незалежно від форм власності, організаційно-правової форми і видів діяльності. Установлюються мінімальні вимоги безпеки та захисту здоров’я під час здійснення роботи, пов’язаної з використанням екранних пристроїв незалежно від їхнього типу та моделі [23].

4.3.1 Електробезпека

Після закінчення роботи екранні пристрої слід відключати від електричної мережі. У разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі. Не допускається самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження. Не допускається працювати з екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності.

4.3.2 Ергомічні вимоги до робочо місця

Робочі місця мають бути спроєктовані так, щоб працівники мали простір для зміни робочого положення та рухів, а організація робочого місця має забезпечувати відповідність усіх елементів робочого місця ергономічним, антропологічним та психофізіологічним вимогам. Робочий стіл або робоча поверхня повинна бути достатнього розміру та мати поверхню з низькою відбивною здатністю. Робоче крісло має бути стійким і дозволяти працівнику легко рухатися та займати зручне положення, сидіння має регулюватися по висоті. Спинка сидіння має регулюватися за висотою та нахилом.

4.3.3 Освітлення робочого місця

Освітлення робочого місця має створювати відповідний контраст між екраном і навколишнім середовищем. Екран не має відблискувати або відбивати світло, щоб не викликати дискомфорту працівника.

4.3.4 Вимоги до екранних пристроїв

Екранні пристрої не мають бути джерелом ризику для працівників. Усе випромінювання, крім видимої частини електромагнітного спектра, має бути зведене до незначного рівня. Символи на екрані мають бути чіткими та відповідного розміру. Зображення на екрані має бути стабільним, без миготінь; яскравість та контрастність повинні легко регулюватися. Екран має легко повертатися та нахилятися. За необхідності може використовуватися окрема підставка або регульований стіл для розміщення екрана.

4.3.5 Навчання та інформування працівників

Працівники повинні бути ознайомлені під підпис з умовами праці та небезпечними і шкідливими факторами. Роботодавець забезпечує навчання та перевірку знань з охорони праці. Навчання проводиться до початку роботи та у випадках модифікації обладнання. Роботодавець організовує регламентовані перерви для відпочинку працівників.

ВИСНОВКИ

У кваліфікаційній роботі розглянуто актуальну проблему автоматизації моніторингу простоїв промислового обладнання та оперативного управління завданнями персоналу із використанням сучасних клієнт-серверних і мобільних технологій, придатних для впровадження на підприємствах.

Аналіз існуючих рішень для моніторингу виробництва показав, що більшість систем або надто дорогі для впровадження або мають значні затримки через паперовий чи напівавтоматичний облік. Це підкреслює потребу у гнучкій, надійній та відмовостійкій інформаційній системі. Запропонований програмний комплекс забезпечує миттєву безконтактну ідентифікацію верстатів, фіксацію часу та причин зупинок ліній, а також оперативне інформування відповідальних осіб.

У межах проєкту реалізовано систему на основі трирівневої архітектури. Серверна частина побудована з використанням Spring Boot, ORM Hibernate та MySQL, що забезпечує стабільну обробку запитів REST API і надійне збереження даних у режимі постійної роботи. Мобільний Android-додаток створено на Jetpack Compose з інтеграцією Google ML Kit та CameraX для оптичного розпізнавання інвентарних номерів безпосередньо на пристрої. Це значно скорочує час реєстрації збоїв порівняно з ручним пошуком у журналах.
Інноваційною складовою проєкту є інтеграція сервера з хмарним сервісом Google Gemini API. Штучний інтелект автоматично аналізує коди помилок обладнання та генерує для операторів контекстні підказки щодо ремонту, що прискорює усунення несправностей. Під час тестування у хмарі Render виявлено обмеження безкоштовного тарифу через блокування SMTP-трафіку JavaMailSender, що обґрунтовує необхідність переходу на платний серверний план. Аналіз охорони праці та інтерфейсу Material Design 3 підтвердив ергономічність і безпеку використання додатка операторами біля верстатів.

У результаті створено універсальну інформаційну систему, яка легко налаштовується, масштабується та інтегрується в інфраструктуру підприємства.

Застосунок забезпечує миттєву ідентифікацію обладнання, автоматичне формування черги завдань і використовує штучний інтелект для оперативного усунення технічних аварій. Використання системи оптимізує робочий час і підвищує ефективність обладнання, відповідаючи сучасним вимогам до надійності, відмовостійкості та безпеки персоналу. Результати дослідження підтверджують, що сучасні мобільні технології та сервіси штучного інтелекту дозволяють створити ефективну систему диспетчеризації виробництва, яка мінімізує організаційні втрати, усуває вплив людського фактора та оптимізує робочий час. Рішення особливо актуальне для малих і середніх підприємств, де впровадження складних MES-систем є економічно недоцільним, а бюджет не дозволяє закуповувати нове обладнання чи інтегрувати дорогі апаратні рішення. Система забезпечує цифровізацію наявного технічного парку без капітальних інвестицій, використовуючи лише доступні мобільні пристрої операторів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008–15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. Введ. 2015–06–22. К. Держстандарт України, 2017. – 29 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти денної і заочної форми навчання спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» освітньої програми «Автоматизація та комп’ютерно-інтегровані технології» / Упоряд.: І.Ш. Невлюдов, О.І. Филипенко, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2023. 64 с.

3. Ефективність обладнання (OEE). База знань IT-Enterprise. URL: https://www.it.ua/knowledge-base/technology-innovation/oee (дата звернення: 02.05.2026).

4. Tanha N. I., Sumi S. S., Rahman M. T., Azad M. A. Optimizing Manufacturing Efficiency through Six Sigma: A Comprehensive Approach in Industrial Engineering. Saudi Journal of Engineering and Technology. 2025. Vol. 10, Issue 3. URL: https://www.researchgate.net/publication/390580113_Optimizing_Manufacturing_Efficiency_through_Six_Sigma_A_Comprehensive_Approach_in_Industrial_Engineering#read (дата звернення: 05.05.2026).

5. What is the IIoT? Industrial Internet of Things: Technical features, Functionality and Advantages. Industrial Automation & IT Knowledge Base. URL: https://www.hilscher.com/de/service-support/glossary/iiot (дата звернення: 05.05.2026).

6. Теорія автоматичного управління (збірник задач): навч. посіб. для студентів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / І. Ш. Невлюдов, О. В. Токарєва; Харків. нац. ун-т радіоелектроніки. Харків : Панов А. М., 2020. 242 с.

7. A guide to manufacturing execution system (MES). Medium. URL: https://medium.com/@erpinformation/a-guide-to-manufacturing-execution-system-mes-50966a6d3c3d (дата звернення: 06.05.2026).

8. Kotlin for Android. Kotlin Documentation Official Website. URL: https://kotlinlang.org/docs/android-overview.html (дата звернення: 08.05.2026).

9. Jetpack Compose Framework Architecture. Android Developers Official Documentation. URL: https://developer.android.com/jetpack/androidx/releases/compose (дата звернення: 08.05.2026).

10. Recognize text in images with ML Kit on Android. Google Developers Official Documentation. URL: https://developers.google.com/ml-kit/vision/text-recognition/v2/android (дата звернення: 09.05.2026).

11. Spring Boot Reference Documentation. Spring Framework Projects Official Website. URL: https://docs.spring.io/spring-boot/documentation.html (дата звернення: 09.05.2026).

12. Web Application Architecture: Common web application architectures. Microsoft Learn Official Documentation. URL: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures (дата звернення: 12.05.2026).

13. Guide to app architecture: UI Layer and MVVM pattern in Android. Android Developers Official Documentation. URL: https://developer.android.com/topic/architecture/ui-layer (дата звернення: 14.05.2026).

14. MySQL Reference Manual: Transactional Integrity and ACID Compliance. Oracle / MySQL Documentation. URL: https://dev.mysql.com/doc/refman/8.0/en/mysql-acid.html (дата звернення: 17.05.2026).

15. Architectural Styles and the Design of Network-based Software Architectures (REST). UC Irvine. URL:

https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (дата звернення: 19.05.2026).

16. OpenAPI 3 library for Spring Boot. Springdoc OpenAPI Official Documentation. URL: https://springdoc.org/ (дата звернення: 23.05.2026).

17. Spring Security Reference: Token-based Authentication with JWT. Spring Framework Projects Official Website. URL: https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/jwt.html (дата звернення: 25.05.2026).

18. Material Design 3 Specifications: Design system for Android. Google Material Design Official Website. URL: https://m3.material.io/ (дата звернення: 26.05.2026).

19. Design layouts and navigation patterns in Jetpack Compose. Android Developers Official Documentation. URL: https://developer.android.com/develop/ui/compose/layouts (дата звернення: 27.05.2026).

20. Jakarta Persistence (JPA) Specification and Hibernate ORM Core. Hibernate Official Documentation. URL: https://hibernate.org/orm/documentation/ (дата звернення: 28.05.2026).

21. Gemini API Overview and Developer Guides. Google AI Studio Documentation. URL: https://ai.google.dev/gemini-api/docs (дата звернення: 30.05.2026).

22. Jakarta Mail API Specification for Email Systems (SMTP). Eclipse Project for Jakarta Mail. URL: https://eclipse-ee4j.github.io/mail/ (дата звернення: 02.06.2026).

23. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями : Наказ Міністерства соціальної політики України від 14.02.2018 № 207. База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 10.06.2026).

24. Комплекс навчально-методичного забезпечення навчальної дисципліни «Безпека праці в індустрії ІТ-технологій» підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [http://catalogue.nure.ua/knmz] / ХНУРЕ; розроб.: Т. Є. Стиценко, Г. В. Пронюк, Н. М. Сердюк. – Харків, 2017. – 122 с.