Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розроблення системи автоматизації для технічного обслуговування обладнання приладобудівного підприємства
(тема)

Виконав:
здобувач 3 року навчання,
(скорочений строк навчання)
групи АКТАКІТу-22-1
Руслан ШАТАЛЮК
(власне ім'я, прізвище)

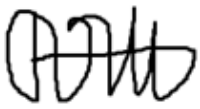
Спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма автоматизація та комп'ютерно-інтегровані технології
(повна назва освітньої програми)
Керівник доц. Артем БРОННІКОВ
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (прізвище, ініціали)

2025 р.

Я , Шаталюк Руслан Русланович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

" 25 " червня 2025 р.  Руслан ШАТАЛЮК

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики та комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегровані технології, автоматизація та робототехніка
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма автоматизація та комп'ютерно-інтегровані технології
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____20 25 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Шаталюку Руслану Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення системи автоматизації для технічного обслуговування обладнання приладобудівного підприємства
затверджена наказом університету від 21 травня 2025 р. № 405 Ст
2. Термін подання студентом роботи до екзаменаційної комісії _____
3. Вихідні дані до роботи C# (.NET Framework), MySQL, Windows Forms, MySQL Workbench, EER-діаграма, модуль авторизації, модуль заявок, модуль технічного обслуговування, модуль керування обладнанням, модуль повідомлень, модуль генерації звітів
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз існуючих автоматизованих систем для забезпечення технічного обслуговування обладнання на приладобудівних підприємствах
4.3.Вибір та обґрунтування технічних рішень для автоматизованої системи
4.4 Розробка програмного забезпечення для технічного обслуговування обладнання
4.5 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) слайди у форматі Power Point у кількості 11 слайдів з розширенням .pptx

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	29.04.2025	виконано
2	Опрацювання літератури за темою роботи.	08.05.2025	виконано
3	Аналіз існуючих автоматизованих систем для забезпечення технічного обслуговування обладнання на приладобудівних підприємствах	16.05.2025	виконано
4	Вибір та обґрунтування технічних рішень для автоматизованої системи	23.05.2025	виконано
5	Розробка програмного забезпечення для технічного обслуговування обладнання	30.05.2025	виконано
6	Подання роботи на перевірку Інтернет-сервісом StrikePlagiarism		
7	Оформлення пояснювальної записки		
8	Подання роботи на рецензію		
9	Подання роботи на підпис зав. кафедри		
10	Подання кваліфікаційної роботи в ЕК		

Дата видачі завдання 28 квітня 2025 р.

Студент _____ Руслан ШАТАЛЮК
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Артем БРОННІКОВ
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 87 с., 38 рис., 16 таблиці, 20 джерел, 4 додатки.

ТЕХНІЧНЕ ОБСЛУГОВУВАННЯ, ПРИЛАДОБУДІВНЕ ПІДПРИЄМСТВО, C#, .NET FRAMEWORK, WINDOWS FORMS, MYSQL, БАЗА ДАНИХ

Об’єкт розробки – процес організації та керування технічним обслуговуванням промислового обладнання на приладобудівному підприємстві.

Предмет розробки – автоматизована інформаційна система для управління технічним обслуговуванням обладнання на підприємстві, що включає інструменти для обліку та моніторингу ремонтними роботами.

Мета роботи – покращення процесів технічного обслуговування обладнання на приладобудівному підприємстві за рахунок розробки програмного забезпечення.

У кваліфікаційній роботі було проаналізовано вже наявні програмні вироби на ринку, визначено мову програмування та вид бази даних також було розроблено програмний застосунок для 3 акторів на приладобудівному підприємстві.

Також, отримані результати роботи можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», а саме п. 9.4: «Підвищення рівня промислової продуктивності шляхом модернізації обладнання і впровадження інноваційних технологій у виробничі процеси, зокрема забезпечення ефективного технічного обслуговування і ремонту промислового устаткування на підприємствах».

ABSTRACT

Explanatory note: 87 p., 38 fig., 16 table, 20 sources, 4 appendices.

TECHNICAL MAINTENANCE, INSTRUMENT MANUFACTURING ENTERPRISE, C#, .NET FRAMEWORK, WINDOWS FORMS, MYSQL, DATABASE

The object of development is the process of organizing and managing the technical maintenance of industrial equipment at an instrument-making enterprise.

The subject of development is an automated information system for managing the technical maintenance of equipment at the enterprise, which includes tools for accounting and monitoring repair work.

The purpose of the work is to improve the processes of equipment maintenance at an instrument-making enterprise through software development.

In the qualification work, existing software products on the market were analyzed, the programming language and type of database were determined, and a software application was also developed for 3 actors at an instrument-making enterprise.

Additionally, the results of this work can be related to Sustainable Development Goal 9, “Industry, Innovation, and Infrastructure,” namely target 9.4: “Increase industrial productivity through modernization of equipment and implementation of innovative technologies in production processes, including ensuring effective maintenance and repair of industrial machinery at enterprises.”

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АФЧХ – амплітудно-фазова частотна характеристика;

дашборди – інформаційні панелі;

ДСТУ – державний стандарт України;

IDEF0 – метод функціонального моделювання;

ПЗ – програмне забезпечення;

патчі – оновлення або виправлення програмного забезпечення;

СА – система автоматизації;

.NET Framework – програмна платформа від Microsoft для розробки і виконання застосунків;

ARM (Advanced RISC Machines) – ARM-архітектура мікропроцесорів із зниженим набором команд;

CMMS (Computerized Maintenance Management System) – комп’ютеризована система управління обслуговуванням;

ERP (Enterprise Resource Planning) – система планування ресурсів підприємства;

IBM – міжнародна корпорація з виробництва програмного та апаратного забезпечення;

IBM Maximo – система управління активами від IBM;

IDEF0 – метод функціонального моделювання;

MySQL – система керування базами даних з відкритим кодом;

SAP – системи, застосування та продукти в обробці даних;

SAP EAM – управління основними фондами підприємства в SAP;

SQL (Structured Query Language) – мова структурованих запитів для роботи з базами даних.

ВСТУП

На сучасному етапі розвитку промисловості особливого значення набуває не лише виробництво продукції, але й ефективне управління технічним обслуговуванням обладнання, що задіяне у технологічному процесі. Приладобудівні підприємства, як правило, мають широкий спектр обладнання – від простих верстатів до високоточних автоматизованих ліній, функціонування яких безперервно впливає на якість та своєчасність виконання виробничих планів. Втрата працездатності хоча б однієї одиниці такого обладнання може призвести до затримки у виробництві, порушення логістичних ланцюгів, а іноді й до значних фінансових втрат.

В умовах, коли значна частина управлінських і виробничих процесів вже є автоматизованою, дивно виглядає ситуація, коли планування та контроль за станом обладнання ще досі здійснюються вручну або із використанням застарілих таблиць. Це створює вразливість до людського фактору, ускладнює швидке реагування на аварійні ситуації, а також унеможливлює якісний аналіз технічного стану машин у довгостроковій перспективі.

Мета роботи – покращення процесів технічного обслуговування обладнання на приладобудівному підприємстві за рахунок розробки програмного забезпечення.

Об’єкт розробки – процес організації та керування технічним обслуговуванням промислового обладнання на приладобудівному підприємстві.

Предмет розробки – автоматизована інформаційна система для управління технічним обслуговуванням обладнання на підприємстві, що включає інструменти для обліку та моніторингу ремонтними роботами.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– здійснити аналіз наявних інструментальних засобів для реалізації автоматизованої системи, а також обґрунтувати вибір мови програмування, середовища розробки та технічної платформи;

– розробити функціональну декомпозицію системи, представивши її у вигляді ієрархічної структури та моделей у нотації IDEF0 з деталізацією ключових функціональних блоків;

– провести інженерні та техніко-економічні розрахунки, необхідні для вибору оптимальних технічних рішень з урахуванням витрат і очікуваної ефективності системи;

– застосувати елементи теорії автоматичного управління для аналізу динамічних характеристик системи, зокрема визначити передавальну функцію та оцінити її стійкість;

– оформити пояснювальну записку згідно методичними вказівками з підготовки й оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп’ютерно-інтегровані технології освітньої програми «Автоматизація та комп’ютерно-інтегровані технології» [1] та ДСТУ 3008:2015 [2].

Дана кваліфікаційна робота пройшла апробацію у журналі ADED-2025(1)[3].

1 АНАЛІЗ ІСНУЮЧИХ АВТОМАТИЗОВАНИХ СИСТЕМ ДЛЯ ЗАБЕЗПЕЧЕННЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ ОБЛАДНАННЯ НА ПРИЛАДОБУДІВНИХ ПІДПРИЄМСТВАХ

1.1 Загальна характеристика процесу технічного обслуговування виробничого обладнання

Технічне обслуговування (ТО) виробничого обладнання є важливим компонентом загальної системи експлуатації машин та механізмів на підприємстві. Його основною метою є забезпечення надійної, безперебійної та безпечної роботи технічних засобів, що застосовуються у виробничому процесі. У межах сучасних приладобудівних підприємств, де точність, стабільність і безперервність процесів мають ключове значення, технічне обслуговування набуває особливої ваги.

Процес технічного обслуговування (рис. 1.1) включає в себе комплекс заходів, спрямованих на запобігання поломкам, виявлення прихованих дефектів, збереження технічних характеристик обладнання на рівні, який відповідає вимогам технологічного процесу. До основних видів ТО належать планово-попереджувальне, поточне, капітальне обслуговування та аварійний ремонт. Під планово-попереджувальним ТО розуміється обслуговування обладнання у встановлені терміни відповідно до його експлуатаційної документації з метою недопущення виникнення несправностей [4].

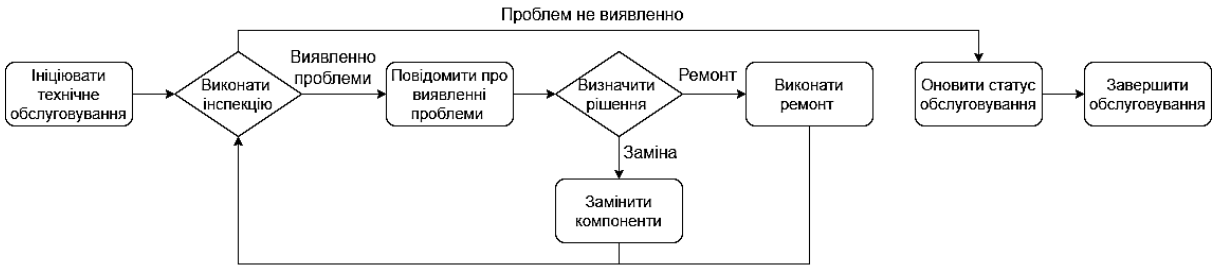


Рисунок 1.1 – Структурна схема етапів технічного обслуговування

Відповідно до державного стандарту ДСТУ 2861-94, технічне обслуговування є видом технічної експлуатації, який включає в себе контроль технічного стану, змазування, очищення, підтягування з'єднань, регулювання, заміну робочих рідин, фільтрів та інших елементів, з метою забезпечення працездатності й справності устаткування [5].

Сучасні реалії функціонування приладобудівного підприємства демонструють потребу не просто у періодичному обслуговуванні обладнання, а у створенні цілісної стратегії його підтримки в робочому технічному стані. Це передбачає не тільки суворе дотримання регламентів, а й врахування реального режиму експлуатації, специфіки технологічного процесу, інтенсивності навантажень, умов навколишнього середовища, а також професіоналізму обслуговуючого персоналу.

Варто зауважити, що в умовах зростаючої складності виробничих систем, ефективність технічного обслуговування дедалі більше залежить від інтеграції інформаційних технологій у цей процес. Надійний контроль за експлуатацією обладнання можливий лише за наявності повної та актуальної інформації про технічний стан кожного елемента, що, у свою чергу, вимагає впровадження автоматизованих систем обліку й аналізу стану машин. Такі системи мають здатність не тільки накопичувати статистичні дані, а й здійснювати прогнозування на основі аналітичних моделей, побудованих із використанням методів машинного навчання та математичної статистики.

Інтеграція концепції Total Productive Maintenance (TPM) – тотального продуктивного обслуговування – свідчить про глибоку трансформацію підходу до ТО (рис. 1.2). На відміну від класичного розподілу відповідальності, TPM передбачає активну участь не лише обслуговуючого персоналу, але й операторів обладнання, керівного складу, а подекуди й аналітичних відділів. Впровадження TPM сприяє підвищенню усвідомленості кожного працівника щодо стану обладнання, зниженню кількості аварійних зупинок, зменшенню непродуктивних витрат часу та ресурсів, а також збільшенню загальної продуктивності підприємства [6].

Рисунок 1.2 – Концепція Total Productive Maintenance

Поряд із впровадженням концепції TPM дедалі більшої популярності набуває ще один підхід – так зване предиктивне технічне обслуговування, або Predictive Maintenance (PdM). Якщо простими словами, це коли обладнання не обслуговується за чітким графіком, як зазвичай, а лише тоді, коли дійсно виникає потреба. Такий підхід став можливим завдяки новим технологіям: сучасні датчики, підключені до Інтернету речей (IoT), збирають дані про роботу обладнання, а потім ці дані аналізуються за допомогою спеціальних алгоритмів і навіть нейромереж. Система виявляє закономірності й може заздалегідь попередити, що певний вузол скоро вийде з ладу. Тож ремонт проводиться тоді, коли він реально потрібен, а не «про всяк випадок», що дозволяє зекономити ресурси і уникнути несподіваних простоїв [7].

Така точність в обслуговуванні була б неможливою без активного впровадження цифрових технологій. Сьогодні технічне обслуговування на передових підприємствах уже не виглядає як традиційне «огляд-змазування-заміна», а все частіше перетворюється на динамічну цифрову систему. В умовах переходу до Індустрії 4.0 обслуговування обладнання відбувається у повній взаємодії з автоматизованими платформами, які здатні у реальному часі збирати,

обробляти і передавати дані про стан кожного компонента. Наприклад, начальник цеху може зі свого комп’ютера або навіть смартфона переглянути, які вузли працюють у критичному режимі, де знижується тиск, а де підвищується температура, і миттєво ухвалити рішення. Це значно підвищує оперативність реагування і дозволяє уникати аварійних зупинок.

Особливо це важливо для приладобудівних підприємств, де ціна обладнання дуже висока, а технологічний процес часто не допускає навіть незначних відхилень. У таких умовах автоматизація не просто полегшує роботу, а фактично стає гарантією стабільності та безпеки всього виробництва. Окрім зменшення навантаження на персонал, цифрові системи дозволяють накопичувати цінний досвід – створюються детальні історії ремонтів, фіксуються типові поломки, відстежується частота збоїв, а сам графік обслуговувань формується не вручну, а на основі аналізу фактичного стану. Усе це робить процес не лише зручнішим, а й набагато ефективнішим.

1.2 Огляд сучасних підходів до автоматизації технічного обслуговування

У сучасних умовах інтенсивного розвитку промислового виробництва, зокрема в галузі приладобудування, підтримка безперервної роботи технологічного обладнання стала критично важливим чинником забезпечення стабільного функціонування підприємств. Враховуючи зростаючу складність конструкції верстатів, використання численних електромеханічних та електронних компонентів, а також підвищені вимоги до точності та швидкості обробки деталей, роль технічного обслуговування (ТО) значно зросла. З позиції інженера, що безпосередньо займається експлуатацією та обслуговуванням виробничих систем, стає очевидною необхідність переходу від застарілих ручних методів обліку і реагування на несправності до впровадження автоматизованих систем управління ТО, які дозволяють прогнозувати відмови й мінімізувати час простою обладнання.

Інструментальною основою таких змін є впровадження технологій Інтернету речей (IoT), машинного навчання, обробки великих масивів даних (Big Data) та

хмарних обчислень. Завдяки цим рішенням стало можливим здійснювати безперервний моніторинг технічного стану машин в режимі реального часу. Вібраційна діагностика, термографія, контроль струмів та інших параметрів роботи дозволяють системам технічного моніторингу не лише фіксувати факт виходу з ладу, але й аналізувати динаміку зміни стану обладнання для подальшого прогнозування потенційних несправностей. Як показує практика, використання таких систем дозволяє суттєво знизити загальні витрати на ТО, оптимізувати запаси запасних частин та покращити загальну ефективність виробництва (Overall Equipment Effectiveness, OEE).

У зв’язку з цим, ринок програмного забезпечення для автоматизації ТО активно розвивається та пропонує широкий спектр рішень – від великих корпоративних платформ (таких як IBM Maximo, SAP EAM) до адаптивних хмарних сервісів (наприклад, Fiix, UpKeep, Limble CMMS), що орієнтовані на середній і малий бізнес. Кожен із цих інструментів має свої особливості: деякі пропонують глибоку інтеграцію з ERP-системами підприємства, інші – акцентують на зручності мобільного доступу та можливості швидкого створення заявки на ремонт безпосередньо з робочого місця.

Одним з хмарних сервісів є Fiix (рис. 1.3). Сервіс використовують багато відомих компаній, а саме Liberty Oilfield Services, Magna International, Siemens, і навіть Coca-Cola. Система дозволяє створювати заявки на роботи за кілька натискань, усуваючи неефективність і невизначеність у процесі виконання завдань. Портал дає змогу користувачам подавати необмежену кількість заявок, сортувати та відстежувати їх, а також налаштовувати власні форми й визначати пріоритети. Обслуговування планується за датою, часом, лічильником, подією або станом із підтримкою вкладених ППР і багатопозиційних нарядів. До кожного завдання додаються SOP, списки, фото, рекомендовані запчастини, що спрощує виконання. Дані про виконання, витрати, дати й завдання після перевірок доступні в кілька кліків. Аналітика дозволяє швидко виявляти проблемні роботи, затримки, збої та інші порушення [8].

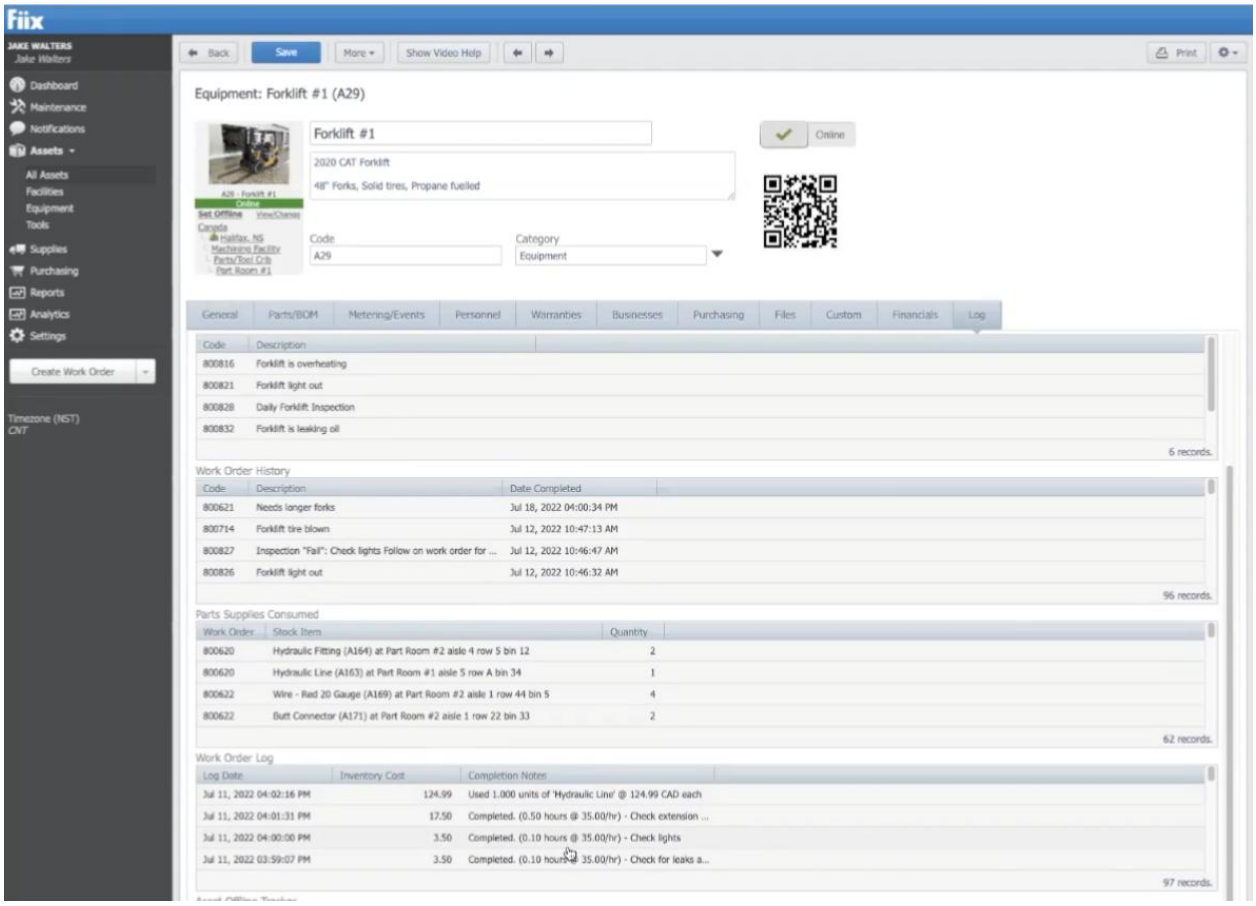


Рисунок 1.3 – Вигляд веб застосунку Fiix [8]

Вся інформація про обладнання зберігається в одному місці, дозволяючи забезпечувати його ефективну й безпечну експлуатацію. Обладнання легко знайти за ієрархією або QR-кодом. Інформація про стан, історію ремонту, SOP, коди несправностей використовується для створення завдань. Лічильники дозволяють миттєво запускати роботи, а дашборд сповіщає про підвищене техобслуговування. Система дозволяє обраховувати витрати для одного пристрою, групи або всього обладнання.

Інвентаризація повністю під контролем: надходять сповіщення про нестачу, формуються заявки на закупівлю, централізовано зберігаються дані постачальників, застосовується FIFO. Проводяться циклічні перевірки, перегляд запасів, формування звітів за витратами й історією запчастин. Прогнозування підказує, які запчастини, у якій кількості та коли необхідно замовити.

Система дозволяє створювати дашборди з реальними даними, переглядати KPI, фільтрувати за проектами, об’єктами, часом, користувачами. Журнали обслуговування зберігають всі зміни, що полегшує аудит. Формуються понад 100 шаблонів звітів або створюються власні, які можна надсилати поштою. Дані візуалізуються на одному екрані, з можливістю фільтрування.

Система інтегрується з будь-яким ПЗ без потреби у програмуванні. Інформація про закупівлі та використання деталей передається між ERP та CMMS, дані з виробничих систем використовуються для запуску техобслуговування. Через відкритий API можна створити власні інтеграції.

Fiix CMMS доступна на мобільних пристроях. Можна створювати, виконувати, відстежувати заявки, працювати онлайн або офлайн, сканувати QR-коди, використовувати голосовий ввід, переглядати списки і керівництва – усе з телефону.

Наступним хмарним сервісом є UpKeep (рис. 1.4). Тисячі менеджерів уже відмовились від паперових документів, Excel і застарілих програм, обравши інтуїтивний мобільний застосунок UpKeep. Він повертає технічним спеціалістам контроль над процесами та комунікацією. Можна зробити фото несправності, намалювати позначки, прикріпити чек-лист і призначити завдання. Весь прогрес буде видно в одному місці [9].

Із мобільним застосунком потрібна інформація завжди доступна тоді, коли вона необхідна. UpKeep отримав високу оцінку серед програм для техобслуговування за версіями Capterra та Gartner. Його можна безкоштовно використовувати зі створенням необмеженої кількості замовлень.

Миттєве оновлення та призначення завдань на ходу дозволяє значно зекономити на витратах часу та праці. Запити подаються просто – із пріоритетами, фото та призначенням виконавця. Домашня сторінка показує прострочені, пріоритетні, збережені замовлення. Доступне додавання термінів виконання, виконавців, розташування, файлів. Штрих-коди можна сканувати камерою та прив’язувати до заявок. Підтримується розклад повторюваних завдань. Виконується як профілактичне, так і аварійне обслуговування. Поля можна

налаштовувати або приховувати. Доступне введення показників часу для критичних інспекцій. Коментування заявок можливе звідусіль.

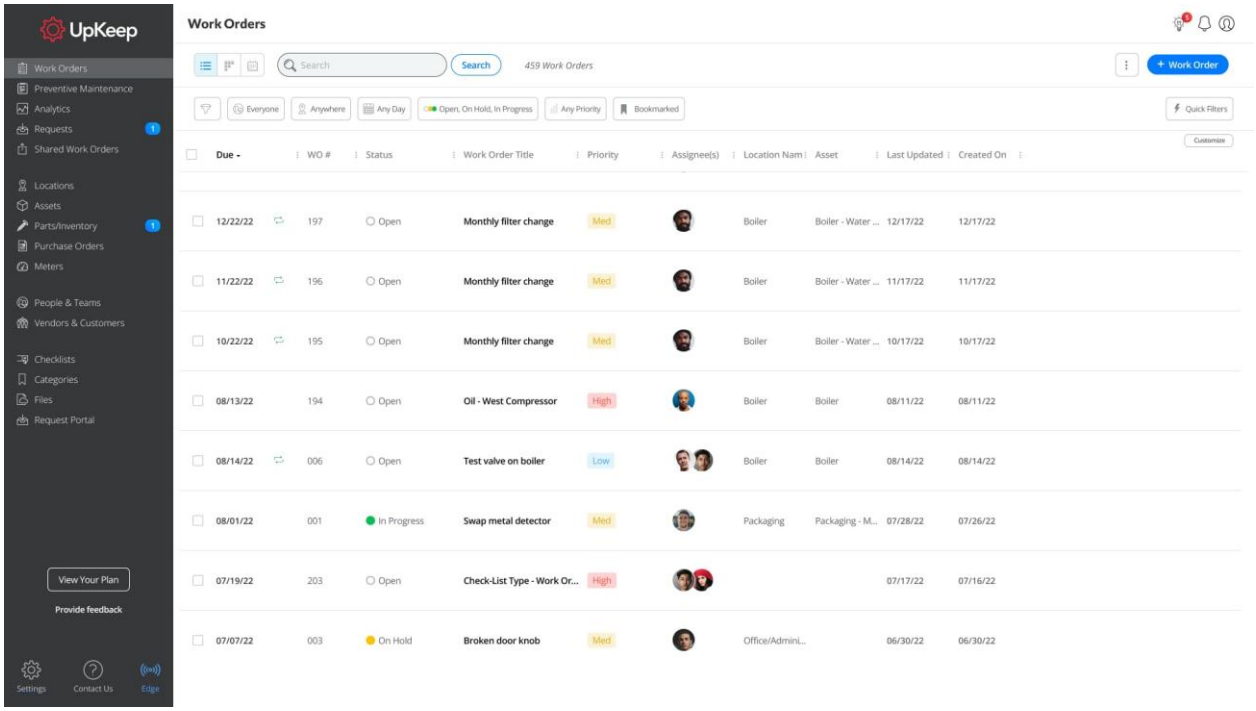


Рисунок 1.4 – Вигляд веб застосунку UpKeep [9]

Відстеження та організація історії обслуговування дає змогу подовжити строк служби обладнання й скоротити витрати на простої. Заявки фільтруються за термінами, статусами, пріоритетами, розташуванням. Обладнання шукається за назвою або штрих-кодом. Також встановлюються статуси для обліку простоїв. У сервісі видно всі попередні заявки, час виконання, пов’язані витрати та запчастини. Також можна прикріплювати інструкції та гарантії для прийняття рішення про ремонт чи заміну.

Командна комунікація в одному місці зменшує кількість дзвінків і перерв. Можна оновлювати статуси в реальному часі, налаштовувати email і push-сповіщення для відстеження прогресу.

UpKeep Mobile і Web працюють разом: зміни в застосунку синхронізуються з ПК. Обслуговування можливе навіть без мережі – дані синхронізуються після повернення онлайн.

Безкоштовна реєстрація і необмежена кількість заявок. Створення замовлень без обмежень доступне кожному. Можна перейти на Starter, Professional або Business Plus для доступу до додаткових функцій і користувачів.

Дивлячись на великі платформи перше що приходить на думку це IBM Maximo (рис. 1.5). IBM Maximo це платформа управління активами, яка поєднує корпоративне управління активами з технічним обслуговуванням. Вона дозволяє компаніям відстежувати повний життєвий цикл активів, включаючи засоби зв'язку, інфраструктуру, транспорт, виробниче та програмне забезпечення [10].

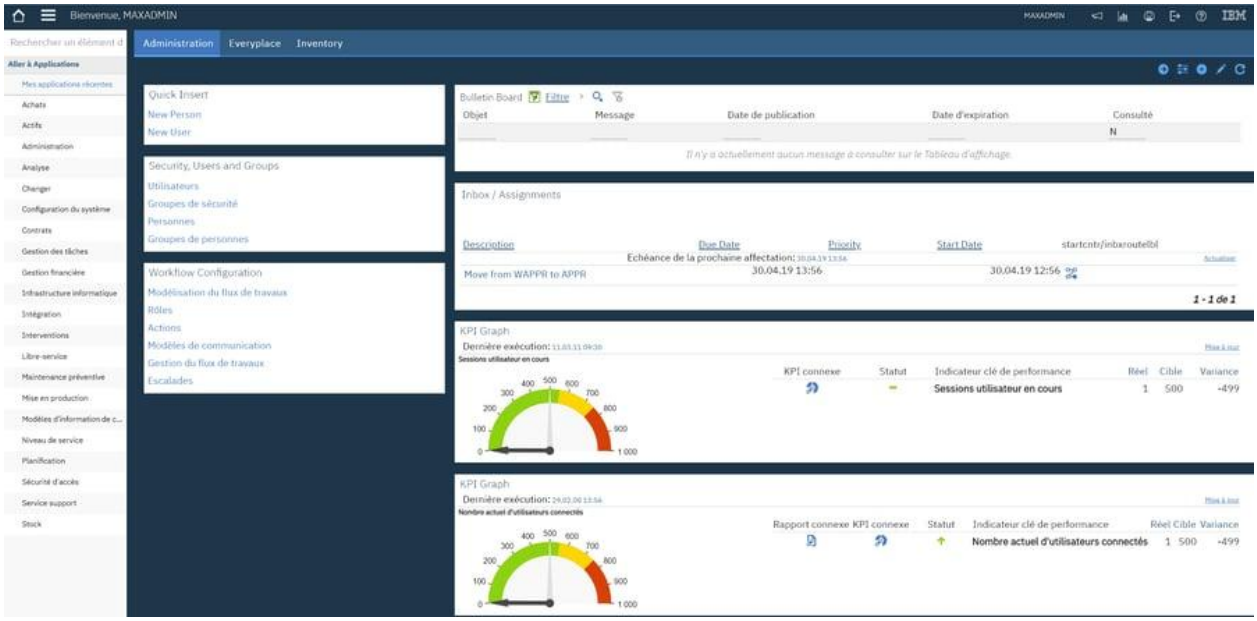


Рисунок 1.5 – Вигляд корпоративної платформи IBM Maximo [11]

Єдина система забезпечує централізований контроль, видимість процесів і мінімізацію простоїв.

Платформа має повний набір інструментів: управління запасами, прогнозне обслуговування, наряди на роботи й аналітику. Доступна з будь-якого пристрою, розгортається в хмарі або локально, підтримує всі типи активів у різних місцях.

До складу Maximo входить базовий модуль IBM Asset Management, що підтримує регулярне відстеження активів, дозволяє класифікувати їх за умовами, управляти матеріалами через профілактичне обслуговування та бачити статуси замовлень. Система дозволяє створювати, змінювати, друкувати, планувати

роботи, автоматизувати запити за наявністю ресурсів, керувати графіками цін, переглядати виставлені рахунки, аналізувати угоди й витрати. Модуль запасів збирає дані з різних джерел для реального перегляду стану активів, надає інформацію про постачальників, автоматизує запити на закупівлю, дозволяє формувати замовлення та оновлювати контрольні точки запасів.

Maximo підтримує створення робочих процесів, шаблонів звітів, графіки техобслуговування, що підвищують ефективність реагування на збої. Система підтримує повну інтеграцію з IoT, приймає дані з сенсорів, людей і пристроїв, має мобільний додаток, дозволяє будувати робочі процеси й сповіщення. Звітність підтримує понад 150 типів звітів, аналіз даних, кольорове кодування KPI, моніторинг збоїв через визначені метрики. У форматі сервісу Maximo зменшує навантаження на компанію завдяки хмарному управлінню, забезпечує високий рівень безпеки, сертифікацію, шифрування та багатофакторний захист. Оновлення та патчі відбуваються постійно, зменшуючи ризики, покращуючи функціональність і закриваючи вразливості.

1.3 Порівняльний аналіз існуючих систем автоматизації обслуговування обладнання

1.3.1 Загальні характеристики та функціональні можливості систем

Коли мова заходить про автоматизацію обслуговування обладнання, найперше, на що звертаєш увагу – це функціональність, зручність і те, наскільки система вписується у реальні умови підприємства. На сьогодні серед найбільш розповсюджених і справді робочих рішень варто згадати IBM Maximo, Fiix і UpKeep. Кожна з них має свої особливості, і щоб зрозуміти, яка з них краще підходить, треба хоча б загально подивитись, що вони пропонують.

IBM Maximo створена для великих компаній, де обладнання багато, процеси складні, і все треба тримати під контролем. Тут все побудовано на модулях, тобто система гнучко підлаштовується під конкретні задачі підприємства. Вона дозволяє не просто створювати заявки на ремонт, а реально відслідковувати стан

обладнання, працювати з профілактичними роботами, інтегруватись з IoT, бачити аналітику і звіти. Причому можна розгорнути її як на своїх серверах, так і в хмарі, все залежить від політики компанії.

Fiix в першу чергу її зручно запускати тим підприємствам, які ще не мали справи з подібними системами. Вона працює повністю в хмарі, тому не треба морочитись з інфраструктурою. Все налаштовується швидко, є мобільний доступ, тобто працівники можуть створювати чи закривати заявки прямо з телефону. Сама система підтримує планування обслуговування, роботу із запасами, автоматичне формування задач і звіти. Із Fiix зручно починати цифрову трансформацію в обслуговуванні обладнання.

Хмарна система UpKeep, зроблена з акцентом саме на мобільність і доступність для персоналу. Якщо у вас працівники постійно в русі, у цехах, і не мають змоги сидіти за комп'ютером – ця система ідеально підходить. Усі основні дії виконуються з мобільного додатку. Це дуже зручно. Також тут є все необхідне: заявки, інвентар, графіки, звіти, аналітика. І що важливо – дуже зрозумілий інтерфейс. Навіть ті, хто далекий від IT, розберуться досить швидко. Вона також добре підходить для середніх і навіть невеликих підприємств.

Ці системи різні за рівнем складності, але кожна має чітку орієнтацію на своїх користувачів. Maximo – для тих, кому потрібно максимально контролювати складні процеси на великому підприємстві. Fiix – для тих, хто хоче почати автоматизацію просто і ефективно. UpKeep – для тих, хто цінує мобільність, швидкість і мінімалізм у роботі. І тут вже не йдеться про те, яка система краща, а яка саме підходить під конкретні умови.

1.3.2 Порівняння переваг та обмежень

Порівнюючи системи автоматизації обслуговування обладнання, завжди виникає питання – яка з них справді краща? Але тут немає простої відповіді. Усе залежить від того, які саме задачі потрібно вирішувати, який бюджет, які масштаби підприємства і наскільки воно готове до цифрових змін.

У IBM Maximo найбільша перевага гнучкість і потужність. Це ідеальний приклад системи корпоративного рівня. Вона дозволяє повністю охопити життєвий цикл обладнання, від установки і введення в експлуатацію до списання. Дуже добре реалізоване планування технічного обслуговування, автоматичне створення робіт, розподіл ресурсів, контроль матеріалів, інтеграція з ERP-системами, модулями енергоменеджменту, IoT-пристроями, аналітикою. Для великих підприємств, де є сотні одиниць обладнання, різні підрозділи та складну структуру IBM Maximo є ідеальним рішенням. Але разом з тим, у нього є свої мінуси. По-перше, це дуже складна система. Її не просто впровадити. Потрібно готувати команду, витрачати час на навчання, адаптувати процеси. По-друге, вона дорога. І саме впровадження, і підтримка та обслуговування. Впровадження цієї системи займає багато часу та засобів. Також варто зауважити, що для невеликого або середнього підприємства Maximo може виявитися занадто важким і перевантаженим.

Fiix, навпаки, простіший і доступніший. Це один із найкращих варіантів для тих, хто тільки починає автоматизувати технічне обслуговування. Головний плюс – це швидкий старт. Не потрібно купувати сервери, все працює в хмарі. Інтерфейс застосунка є зрозумілий а сам застосунок можна легко масштабувати. Впровадження застосунку можна починати з малого та додавати функціонал поступово. Але, звісно, у Fiix є свої обмеження. По-перше, функціональність менша, ніж у Maximo. Якщо підприємство має складні процеси, специфічні вимоги, багато нестандартного обладнання то може бути складно. По-друге, інтеграція з іншими системами є, але вона не така гнучка. Іноді треба докладати зусиль, щоби все з’єднати. Також, хоча система працює в хмарі, не всі підприємства хочуть передавати свої дані назовні – особливо ті, хто має вимоги до безпеки.

UpKeep – це ще один хмарний сервіс, який відомий простотою і мобільністю. Тут усе орієнтовано на зручність для працівника. З мобільного додатку можна створити заявку, зробити фото поломки, додати опис, пріоритет і одразу передати інформацію в систему. Це дуже економить час, особливо якщо обладнання розкидане по різних зонах або територіях. Звітність у сервісі на доволі високому рівні а саме можна подивитися, що було зроблено, які були витрати, які проблеми

повторюються. Сервіс також використовує нагадування і автоматичні повідомлення. Але з іншого боку, якщо потрібно реалізувати щось складніше, наприклад, інтеграцію з SAP або побудувати логіку з декількох рівнів затвердження, тут вже можуть бути труднощі. UpKeep більше заточений під малий і середній бізнес. Але варто враховувати, що не всі завдання зручно робити з телефону, особливо якщо йдеться про глибоку аналітику чи складне планування.

Якщо порівнювати всі три системи між собою, то виходить приблизно так. Maximo – це найпотужніша платформа, але вона вимагає великих ресурсів. Fiix – це золота середина: достатньо функціональна, але при цьому доступна і зрозуміла. UpKeep – дуже легка у використанні, ідеальна для тих, хто хоче простий інструмент для щоденної роботи. Ще важливо враховувати, як швидко можна навчити персонал, яка підтримка надається, чи є локалізація, і яка політика зберігання даних. В реальності немає ідеальної системи. Є та, яка краще підходить саме під твої умови, можливості і задачі. І саме на це потрібно звертати увагу, коли обираєш між цими варіантами.

1.3.3 Аналіз доцільності впровадження на виробничому підприємстві

Коли доходить справа до вибору, що ж саме впроваджувати на реальному виробничому підприємстві, то тут вже не просто порівнюєш функції чи інтерфейс. Тут дивишся глибше – наскільки система впишеться у наявні процеси, чи не стане вона тягарем, скільки часу піде на впровадження, хто буде з нею працювати і чи взагалі воно себе окупить. Бо впровадження будь-якої автоматизації – це не про красиву картинку в презентації, а про живі процеси, людей, обладнання, витрати і ризики.

Якщо говорити про IBM Maximo, то ця система дійсно потужна і підходить саме для великих виробничих підприємств, де обсяги значні, обладнання багато, і потрібна детальна координація обслуговування на різних рівнях. Наприклад, на підприємствах енергетики, хімічної промисловості, металургії або в нафтогазовому секторі – там Maximo показує себе на повну. Але якщо спробувати поставити її на завод середнього масштабу, де ще немає стабільних регламентів, де частину

процесів все ще тримають «у голові» або на папері, то така система буде просто перевантаженою. Її впровадження – це великий проект з командою фахівців, інтеграцією, міграцією даних, навчанням. Тобто сама система чудова, але її впровадження доцільне лише тоді, коли підприємство справді готове до такого рівня цифровізації і має для цього ресурси – і людські, і фінансові.

Fiix виглядає як набагато реалістичніший варіант для більшості вітчизняних виробничих підприємств, особливо тих, які зараз перебувають у стані переходу від паперової документації або Excel до чогось серйознішого. Вона працює в хмарі, не вимагає складної IT-інфраструктури, легко масштабується, а головне – швидко приносить результат.

Підприємство може почати з базових функцій: облік обладнання, планування обслуговування, створення заявок – а потім розвивати систему далі. Доцільність впровадження Fiix висока саме для тих виробництв, де вже усвідомили, що без цифрового управління технічним обслуговуванням далі працювати складно, але при цьому не мають змоги витрачати роки й мільйони на впровадження. Вона дозволяє швидко зробити «цифровий стрибок» і поступово налагодити культуру обслуговування.

UpKeep найкраще підходить там, де обслуговування обладнання досить просте, але дуже залежить від швидкості реакції і зручності в користуванні. Це можуть бути підприємства з мобільними технічними командами, які працюють у цехах, на складах, в ангарах, де не завжди є доступ до комп’ютера.

Наприклад, харчова промисловість, легка промисловість, логістичні компанії. Там, де важливо, щоб технік одразу ж на місці міг зробити фото, створити заявку, подивитися історію обслуговування і рухатись далі. Перевага тут у швидкості впровадження і простоті – реально, система починає працювати буквально за тиждень-два.

Але разом з тим вона обмежена в можливостях. Якщо з часом підприємство почне ускладнювати свої процеси, з’явиться потреба в глибшій аналітиці або інтеграціях, то може виявитися, що UpKeep не покриває всього необхідного.

У підсумку, якщо ми говоримо саме про виробниче підприємство, де є чітка структура обладнання, визначені регламенти обслуговування, бажання впровадити нормальне планування, контроль і аналітику – то найдоцільніше впроваджувати Fiix.

Вона добре балансує між можливостями і доступністю. Maximo має сенс лише тоді, коли підприємство вже має досвід роботи з подібними системами, чітко розуміє свої потреби і готове до серйозного впровадження. UpKeep більше підходить як перший крок для тих, хто хоче швидко оцифрувати обслуговування, але не планує глибоку інтеграцію або складну логіку.

Тож коли обираєш систему для реального виробництва – важливо не те, яка з них найсучасніша чи модна, а яка з них дасть результат у конкретних умовах, з конкретними людьми, процесами і обмеженнями.

1.4. Визначення вимог до автоматизованої системи для приладобудівного виробництва

1.4.1 Аналіз особливостей виробничого процесу приладобудування

Щоб правильно визначити вимоги до автоматизованої системи, спочатку потрібно добре розібратися, чим саме відрізняється приладобудівне виробництво від інших галузей. Із першого погляду може здатися, що це просто ще один вид машинобудування, але на практиці тут є чимало нюансів, які суттєво впливають на організацію всього процесу – від планування до обслуговування обладнання.

Почнемо з того, що приладобудування має справу з високоточними, часто складними у виконанні виробами, де точність до десятих і навіть сотих міліметра не просто бажана, а критично необхідна. Через це і процеси виготовлення, і контроль якості потребують особливої уваги. Наприклад, під час обробки корпусів для приладів або виготовлення мікромеханічних вузлів часто використовуються високоточні верстати з числовим програмним керуванням, які вимагають стабільного технічного обслуговування. Одне найменше відхилення у роботі верстату і вся партія деталей непридатна [12].

Окрім цього, характерною рисою є багатономенклатурність і дрібносерійність. У приладобудуванні дуже часто виготовляється невелика кількість виробів різної конструкції. Це означає, що виробничі маршрути постійно змінюються, склад комплектуючих варіюється, а підготовка виробництва потребує багато часу. У таких умовах важливо мати автоматизовану систему, яка швидко адаптується до змін: оперативно формує завдання, відстежує переміщення компонентів, веде контроль залишків.

Ще один важливий аспект – тісна інтеграція конструкторського, технологічного та виробничого відділів. У приладобудівному підприємстві неможливо побудувати ефективний процес без постійного обміну інформацією між цими підрозділами. Наприклад, при зміні креслень чи технічних вимог потрібно автоматично оновити маршрутні карти, норми часу, списки матеріалів. Усе це повинно відображатися в системі в режимі реального часу, щоби не виникало розриву між проектом і його фактичним виготовленням.

Також не варто забувати про суворі вимоги до контролю якості. В приладобудуванні часто застосовуються декілька етапів перевірок – вхідний контроль матеріалів, проміжний контроль у процесі виробництва і фінальна перевірка з використанням спеціалізованих вимірювальних стендів. Це означає, що система має підтримувати повний простежуваний ланцюг – від моменту закупівлі сировини до здачі готового виробу. Причому всі дані мають зберігатися і бути доступними для аналізу, аудиту чи сертифікації.

Ще одним важливим фактором є висока вартість помилки. Через складність приладів і високу ціну компонентів, кожна виробнича помилка може коштувати дорого. Тому вкрай необхідно, щоб система допомагала не тільки реєструвати поломки або відхилення, а й аналізувати причини, запобігати повторенню проблем, автоматично формувати плани обслуговування на основі статистики відмов.

Окремо варто згадати й про вимоги до кадрового складу. На таких виробництвах працюють висококваліфіковані фахівці, які очікують, що система автоматизації не буде ускладнювати їм роботу, а навпаки – спростить доступ до документації, полегшить заповнення звітів, надасть швидкий доступ до історії

ремонтів чи змін. Тобто інтерфейс має бути інтуїтивно зрозумілим, логіка – прозорою, а можливості – достатніми для підтримки професійної діяльності.

І нарешті, не можна оминути питання безпеки. Багато приладів можуть використовуватись у медичній, оборонній чи енергетичній сферах, де існують жорсткі вимоги до зберігання, передачі та захисту інформації. Це ставить перед системою автоматизації ще один рівень складності – вона має відповідати нормативам безпеки, бути захищеною від несанкціонованого доступу, мати розмежування прав користувачів та систему резервного копіювання.

1.4.2 Формування функціональних і технічних вимог до автоматизованої системи для технічного обслуговування обладнання

Після детального аналізу виробничого процесу у сфері приладобудування логічним кроком стає формування конкретних функціональних і технічних вимог до автоматизованої системи, яка б не просто підтримувала роботу, а реально підвищувала ефективність технічного обслуговування обладнання. Такі вимоги мають враховувати реальні умови, в яких працює підприємство, і одночасно гарантувати безперебійний, безпечний та прозорий процес технічної підтримки всієї наявної техніки.

Перш за все, фундаментом такої системи має стати надійна система аутентифікації. Вхід до системи має бути дозволений виключно авторизованим користувачам, причому – з чітким розмежуванням ролей. Повинен бути актор "працівник", який має доступ лише до створення заявок, а також "ремонтник", який працює з заявками у межах свого або кількох цехів. Такий підхід дозволяє не лише обмежити доступ до зайвої інформації, а й забезпечує більшу відповідальність за дії в системі.

Наступна важлива функція – це можливість ручного розподілу завдань. Жодна автоматизація не може повністю замінити живий контроль з боку майстра або керівника служби, особливо в складних випадках. Тому система повинна дозволяти диспетчеру або відповідальному фахівцю самостійно призначати заявки

конкретним ремонтникам, враховуючи їхню завантаженість, кваліфікацію чи навіть специфіку несправності.

Не менш важливо – забезпечити простий і зручний процес створення заявок на ремонт. Така можливість має бути доступна виключно для працівників підприємства, які щодня мають справу з обладнанням і першими помічають відхилення в його роботі. Саме вони повинні мати змогу оперативно подати заявку, вказати основну суть проблеми, додати фото або відео, що значно полегшить діагностику на ранньому етапі.

Кожна заявка повинна мати чітко фіксований статус – від моменту створення і до її закриття. Система має підтримувати динамічне оновлення статусу: нова, в роботі, виконано, відкладено тощо. Це дає змогу в режимі реального часу бачити, на якому етапі перебуває кожна робота, уникати повторів і мати чітке уявлення про навантаження кожного з учасників процесу.

Для підвищення організованості критично важливо реалізувати систему сповіщень. Йдеться про автоматичні повідомлення, які надходитимуть виконавцям при появі нових заявок, зміні їх статусу або у разі настання строків виконання профілактичних заходів. Такі сповіщення допомагають уникати людського фактору та не втрачати важливі завдання у загальному потоці інформації.

Ще одна обов’язкова складова – це журнал виконаних робіт. Він виконує подвійну функцію: з одного боку, це архів для історичного аналізу, з іншого – доказ виконаної роботи. Такий журнал повинен зберігати дані про всі операції: що саме було зроблено, ким, коли, на якому обладнанні та з яким результатом. Це корисно як для подальших рішень щодо модернізації техніки, так і для перевірок або аудиту.

Дуже суттєвою є можливість фіксації несправностей із додаванням опису і медіаматеріалів. Фото чи відео поломки дозволяють ремонтнику одразу зрозуміти масштаб проблеми, краще підготуватись до виїзду на місце, взяти з собою потрібні інструменти або деталі. Це заощаджує час і зменшує кількість невдалих виїздів.

У структурі системи обов’язково має бути передбачено повноцінний каталог обладнання. Це база даних з описами всіх одиниць техніки, що експлуатуються на підприємстві, включаючи технічні характеристики, інвентарні номери,

місцезнаходження, типові несправності, періодичність обслуговування тощо. Кожне обладнання має бути чітко прив’язане до цеху або виробничої ділянки.

Для забезпечення надійної роботи обладнання система повинна також підтримувати планові технічні обслуговування. Йдеться про профілактику – перевірку, змазування, калібрування, заміну зношених деталей до того, як станеться серйозний збій. Система має самостійно нагадувати про наближення таких робіт і формувати для цього відповідні заявки.

Не менш важливою є функція формування звітів. Це дозволяє керівникам бачити реальну картину стану техніки, кількість виконаних заявок, середній час ремонту, частоту поломок і багато іншого. За допомогою аналітики можна своєчасно виявляти вузькі місця у виробництві, прогнозувати витрати та планувати закупівлі запчастин.

Також велике значення має модуль управління складом запчастин. Система повинна вести облік наявності основних комплектуючих, попереджати про їх критичний залишок і, за потреби, автоматично формувати запит на закупівлю. Без цього будь-яка зупинка може розтягнутись лише через те, що під рукою немає потрібної деталі.

І, нарешті, одна з ключових особливостей – можливість розмежування обладнання за цехами або категоріями. Це робить навігацію у системі зручною, дозволяє чітко бачити відповідальних за кожну ділянку техніки, прискорює пошук потрібного обладнання та сприяє впорядкуванню всієї структури обліку.

2 ВИБІР ТА ОБҐРУНТУВАННЯ ТЕХНІЧНИХ РІШЕНЬ ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ

2.1 Аналіз та вибір інструментальних засобів реалізації системи

2.1.1 Обґрунтування вибору мови програмування та середовища розробки

При виборі мови програмування погляд одразу впав на C#, тому що це сучасна мова програмування, яка має вихідний код. Також ця мова є кросплатформною та об'єктно-орієнтованою і вважається однією з п’яти найкращих мов програмування на GitHub. Якщо, наприклад, програміст має досвід роботи з JavaScript або Java, то йому буде неважко програмувати і на C#.

C# має безпеку, чіткі типи, універсальні шаблони, зіставлення зі зразком, асинхронність. Також C# дозволяє з'єднуватися з багатьма іншими мовами програмування та з ERP-системами. C# дозволяє працювати з багатьма базами даних, які використовуються на різних підприємствах [13].

За допомогою програмного забезпечення Visual Studio можна створювати форми візуалізації, а саме інтерфейс за допомогою Windows Forms. Ця технологія використовується для створення інтерфейсів програмного забезпечення, використовуючи вже встановлені блоки або додаючи нові пакети чи створюючи власні.

Можна створювати складні графічні додатки, які легко оновлювати й зручно використовувати як автономно, так і в мережі. Додатки на Windows Forms можуть отримувати доступ до локального обладнання та файлової системи комп’ютера. Також у Windows Forms реалізована технологія читання й запису даних із файлової системи.

У Windows Forms є елементи управління, тобто це окремі елементи інтерфейсу, призначені для введення або відображення даних. Є багато готових елементів керування, які можна використовувати для створення простих застосунків: текстові поля, кнопки, списки, перемикачі, навіть веб-сторінки. Якщо

вони не підходять, можна встановити нові елементи з бібліотек NuGet або створити свої за допомогою класу UserControl. Коли користувач взаємодіє з формою то створюється подія, яка обробляється з відповідним кодом [14].

Створення інтерфейсу можливе шляхом перетягування елементів на форму. Також зручно змінювати колір елементів, встановлювати маски для введення в текстові поля без коду, лише через налаштування.

Для покращення інтерфейсу я обрав бібліотеку GunaUI, яка використовує стандартні елементи, що були в Visual Studio, але розширює їхні можливості налаштувань. Це дозволяє створювати сучасний інтерфейс.

Для роботи з базами даних я обрав MySQL і візуальний інструмент MySQL Workbench (рис. 2.1). MySQL – це реляційна база даних, яка працює з таблицями, що пов’язані зовнішніми та внутрішніми ключами.

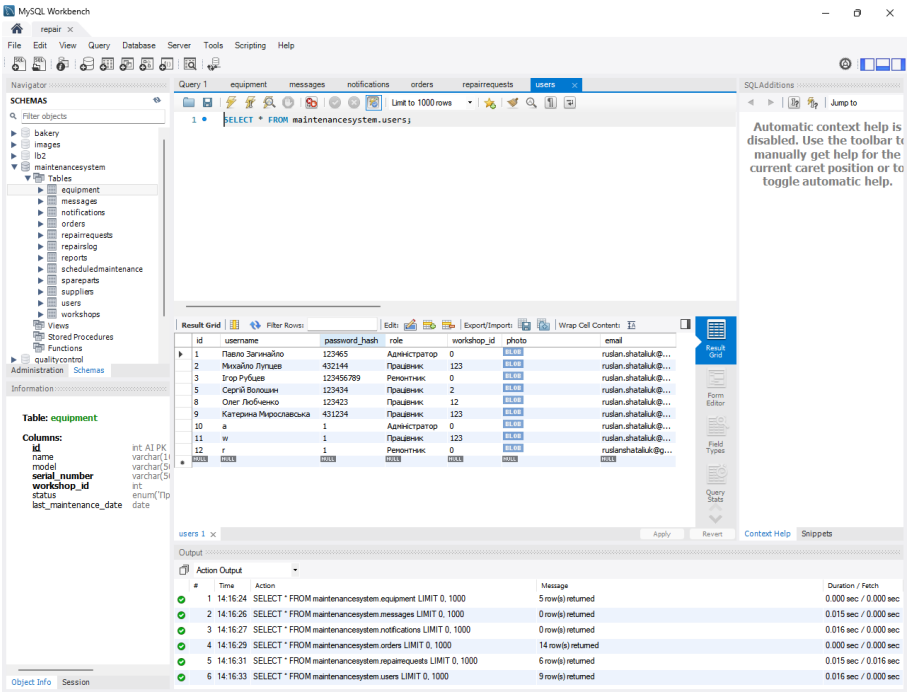


Рисунок 2.1 – MySQL Workbench

MySQL дуже продуктивна: підтримує багато моделей серверів, включно з кластерними, і має високу швидкість обробки аналітики. Вона відповідає промисловим стандартам. Система доступу гарантує безпеку облікових записів, підтримується шифрування паролів та перевірка доступу [15].

Працює за архітектурою клієнт-сервер – клієнт підключається через мережу, надсилає запит, сервер обробляє та повертає відповідь. Для роботи з MySQL я використовую Workbench – уніфікований візуальний інструмент для архітекторів, розробників та адміністраторів. Він дозволяє моделювати БД, писати SQL-запити, налаштовувати сервери, робити резервне копіювання [16].

Workbench доступний для Windows, Linux і macOS. MySQL дозволяє проектувати, моделювати, генерувати й керувати складними базами даних. Він підтримує складні архітектурні рішення та прискорює розробку. Також підтримується міграція даних з Microsoft SQL Server, Access, Sybase ASE, PostgreSQL тощо.

2.1.2 Порівняльний аналіз апаратних платформ та вибір технічних засобів

При виборі апаратної платформи в рамках реалізації системи автоматизації для технічного обслуговування обладнання приладобудівного підприємства було проведено порівняльний аналіз доступних апаратних платформ для реалізації додатка на базі C# з використанням WindowsForms. Розглянуто три основні варіанти: Raspberry Pi, комп'ютерні системи на базі Windows (моноблоки та ноутбуки) та аналогічні системи на базі операційної системи Linux.

Аналізуючи Raspberry Pi було виявлено обмеження для реалізації поставленої задачі. Незважаючи на доступну вартість та компактні розміри, платформа має обмеження, пов'язані з ARM-архітектурою, яка ускладнює повноцінне функціонування .NET Framework. Застосовуючи Raspberry Pi, для запуску системи автоматизації скоріше за все продуктивність графічного інтерфейсу та обробка подій буде незадовільною. Обмежений обсяг оперативної пам'яті (від 1 до 8 ГБ залежно від моделі) та порівняно низька обчислювальна потужність процесора створюють значні перешкоди для комфортної роботи з WindowsForms [17].

Комп'ютери на базі Windows найкраще сумісні з C# та WindowsForms. Нативна підтримка .NET Framework забезпечує безпроблемну інтеграцію всіх компонентів розроблюваного додатка. Навіть системи середнього цінового

діапазону (з процесорами Intel Core i5 або AMD Ryzen 5) скоріше за все будуть забезпечувати достатню продуктивність як для процесу розробки, так і для кінцевого використання програмного продукту. Важливою перевагою є також розвинута система для розробників з наявністю Visual Studio – інтегрованого середовища розробки, оптимізованого саме для C# та WindowsForms.

При аналізі комп'ютерів на базі Linux було визначено,що скоріше за все будуть проблеми з сумісністю та стабільністю роботи інтерфейсу користувача. Хоча комп'ютери на базі Linux аналогічної конфігурації як і на базі Windows забезпечують подібну обчислювальну потужність, але розробка додатків з використанням WindowsForms на цій платформі стикається з серйозними перешкодами. Середовище Linux має принципові відмінності в роботі з графічними інтерфейсами, що призводить до помітних затримок у відображенні візуальних компонентів та проблем з функціонуванням елементів керування [18].

Варто зазначити, що системи розробки для C# у Linux-середовищі менш розвинена порівняно з Windows. Це виявляється у обмеженій документації, присвяченій специфічним аспектам розробки WindowsForms-додатків для Linux, а також у нестачі готових рішень та бібліотек для подолання технічних проблем, які виникають під час такої розробки.

Зробимо табличку для зручного порівняння систем (табл. 2.1).

Таблиця 2.1 – Порівняння приведених систем

Критерій порівняння	Raspberry Pi	Комп'ютери на базі Windows	Комп'ютери на базі Linux
Сумісність з C#/Windows Forms	Обмежена через ARM-архітектуру; ускладнена підтримка .NET Framework	Повна нативна сумісність; підтримка всіх компонентів	Обмежена; проблеми з WindowsForms через архітектурні відмінності Linux

Продовження таблиці 2.1

Критерій порівняння	Raspberry Pi	Комп'ютери на базі Windows	Комп'ютери на базі Linux
Продуктив-ність	Низька; обмежена ОЗП (1–8 ГБ) та слабкий процесор	Висока при середній конфігурації (Intel i5 / Ryzen 5, 16 ГБ ОЗП, SSD)	Висока на аналогічному «залізі», але знижена ефективність GUI
Графічний інтерфейс	Повільна обробка подій, низька якість відображення	Комфортна, стабільна робота інтерфейсу	Часті затримки, проблеми з елементами керування
Середовище розробки	Не оптимізоване; потребує обходів	Visual Studio – повна інтеграція та підтримка	Менш розвинене; нестача бібліотек і документації
Зручність розробки	Низька; ускладнене налагодження і запуск	Висока; швидкий запуск і розгортання	Складна; потребує адаптації коду, висока трудомісткість
Вартість обладнання	Низька	Середня	Середня

Тож для розробки та подальшого функціонування додатка ідеально підходить комп'ютерна система на базі Windows.

Підходящу конфігурацію визначено: процесор не нижче Intel Core i5 10-го покоління або AMD Ryzen 5 4000-серії, 16 ГБ оперативної пам'яті та SSD-накопичувач об'ємом від 256 ГБ. Такі технічні характеристики забезпечують збалансоване співвідношення продуктивності та вартості, а також гарантують комфортну роботу з додатком.

2.2 Розробка функціональної декомпозиції та ієрархічної структури автоматизованої системи

2.2.1 Розробка функціональної моделі у нотації IDEF0

Щоб краще зрозуміти, як працює система автоматизації ремонту обладнання, важливо мати чітке уявлення про всі основні функції, які вона виконує, та про те, як ці функції пов’язані між собою. У цьому нам може допомогти діаграма IDEF0.

Діаграма IDEF0 – це спосіб зобразити, які саме дії (функції) виконує система, що на них впливає, які дані в них входять, що вони створюють на виході, та які ресурси потрібні для виконання. Діаграму IDEF0 можна поділити на 2 частини контекстну діаграму та функціональну модель [19].

Контекстну діаграму можна представити прямокутником із вхідними і вихідними значеннями. Вхід і вихід контекстної діаграми розподіляються не на два, а на чотири сторони прямокутника. Цілі цих сторін такі:

– ліва сторона відповідає системному входу (input), значенню які надходять у систему та обробляється системою у вихідне значення;

– верхня сторона відповідає входу керування(control), а саме різним діям керування, процедурам, стратегіям поведінки, командам, документам, що регламентують виконання робіт і т. д. Ці дії ніяк не змінюються, вони використовуються тільки для контролю;

– права частина відповідає виходу (output) системи, тобто результатам перетворення вхідних значень, відходи тощо;

– нижня сторона відповідає механізмам(mechanism), тобто ресурсам, з використанням яких вхідні значення перетворюються у вихідні.

Для кращого розуміння того, як саме працює наша система, ми побудуємо контекстну діаграму для головної функції а саме «Ремонту обладнання». Вона показує найзагальніше уявлення про весь процес, а вже потім, за потреби, її можна буде деталізувати в діаграмах декомпозиції для кожної окремої функції.

Тож побудуємо контекстну діаграму (рис. 2.2).

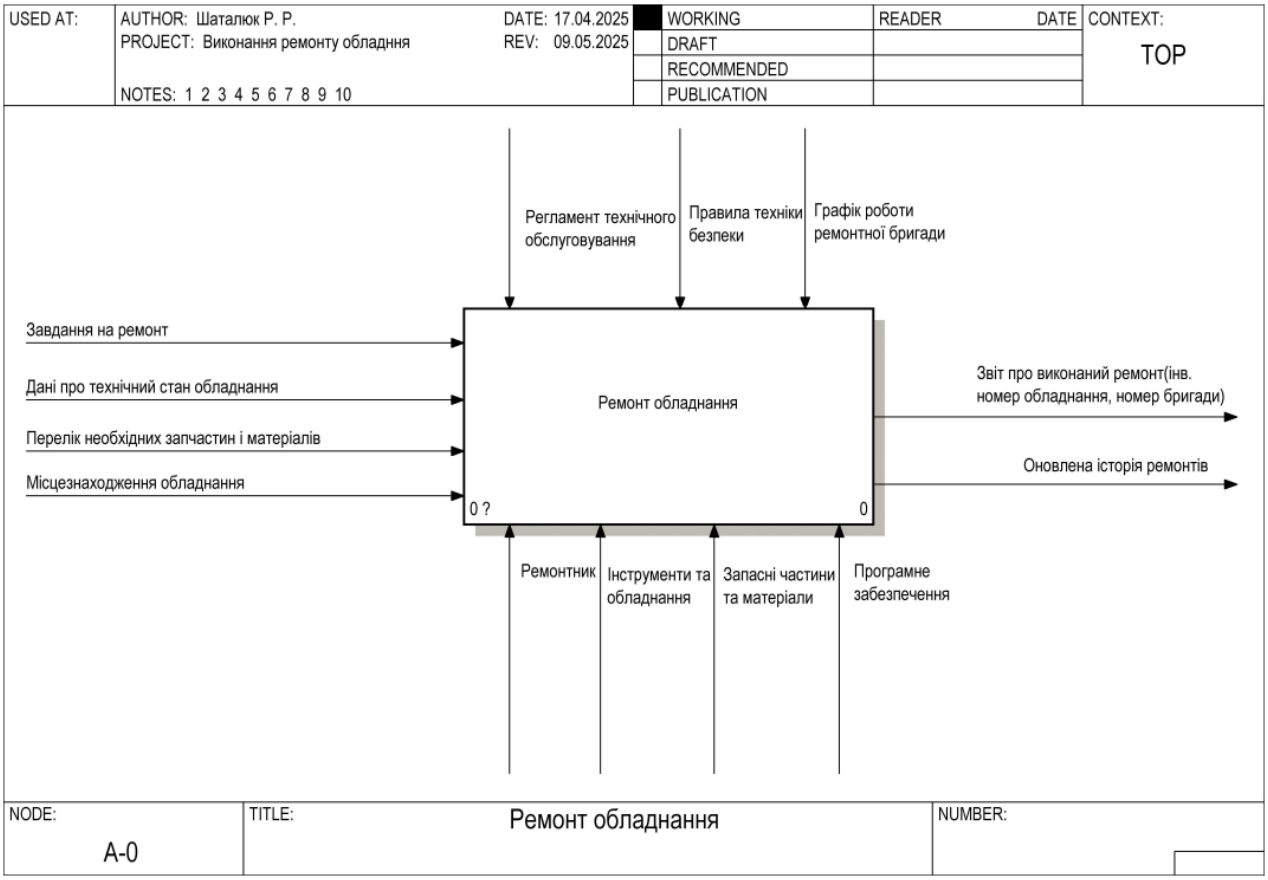


Рисунок 2.2 – Контекстна діаграма на тему «Ремонт обладнання»

Основною функцією на цій діаграмі є "Ремонт обладнання", що охоплює всі етапи – від отримання завдання до формування звіту про виконані роботи.

Розпишемо значення всіх сторін для кращого розуміння виконання функції:

а) вхідні дані (Inputs):

1) завдання на ремонт – інформація про необхідність виконання ремонтних робіт;

2) дані про технічний стан обладнання – відомості, що характеризують поточний стан техніки, яка потребує обслуговування;

3) перелік необхідних запчастин і матеріалів – інформація про ресурси, які потрібні для виконання ремонту;

4) місцезнаходження обладнання – координати або опис розташування техніки, що потребує ремонту.

б) механізми (Mechanisms):

1) ремонтна бригада – персонал, який виконує ремонт;

2) інструменти та обладнання – технічні засоби, які використовуються під час ремонту;

3) запасні частини та матеріали – ресурси, необхідні для усунення несправностей;

4) програмне забезпечення – цифрові інструменти для планування, обліку та контролю процесу ремонту.

в) керування (Controls):

1) регламент технічного обслуговування – встановлені вимоги та стандарти виконання ремонту;

2) правила техніки безпеки – нормативні документи, що регулюють безпечне виконання робіт;

3) графік роботи ремонтної бригади – розклад, що визначає час і черговість виконання завдань.

г) виходи (Outputs):

1) звіт про виконаний ремонт – документ, що містить результати проведених робіт: інвентарний номер обладнання, залучені ресурси, номер ремонтника тощо;

2) оновлена історія ремонтів – доповнення до бази даних про попередні ремонти, що дозволяє відстежувати стан техніки в динаміці.

Стосовно функціональної моделі, то вона є подальшим уточненням (декомпозицією) контекстної діаграми. Спочатку загальна функція системи (мета, призначення, основне завдання) поділяється на кілька окремих функцій (завдань, процесів, цілей). Таких функцій радиться обирати від 2 до 6. Ці функції (іноді їх називають роботами) (activity) відображаються на окремому аркуші декомпозиції як функціональні блоки. Кожен функціональний блок (робота), що представляє собою окрему функцію (роботу, ціль чи завдання), зображається прямокутником. Сторони прямокутників робіт (функціональних блоків) мають те саме призначення, що й розглянуті вище сторони контекстної діаграми.

Побудуємо функціональну модель для нашої функції (рис. 2.3).

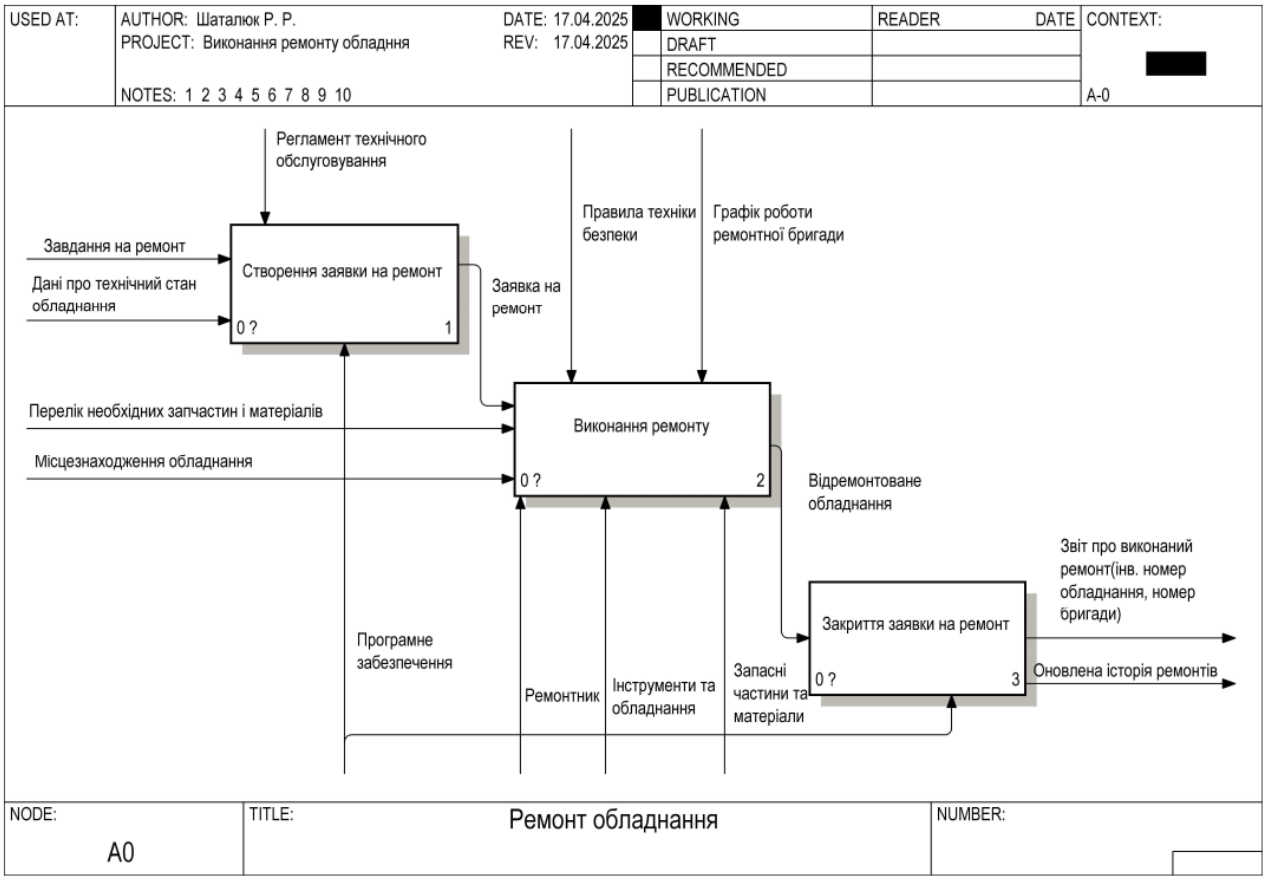


Рисунок 2.3 – Функціональна модель на тему «Ремонт обладнання»

На цій діаграмі показано, як загалом відбувається процес ремонту обладнання на підприємстві. Усе розбито на три основні блоки:

– створення заявки на ремонт. При виявленні несправності формується заявка на ремонт;

– виконання ремонту. На цьому етапі вже працює ремонтник, який, отримавши заявку, приступає до ремонту;

– закриття заявки на ремонт. При закінченні ремонту підбиваються підсумки та оформляється звіт.

2.2.2 Побудова діаграм декомпозицій для блоків функціональної моделі

Для кращого розуміння, як саме відбувається процес створення заявки на ремонт, розглянемо його детальніше. Побудуємо діаграму декомпозиції для блока A1 (рис. 2.4).

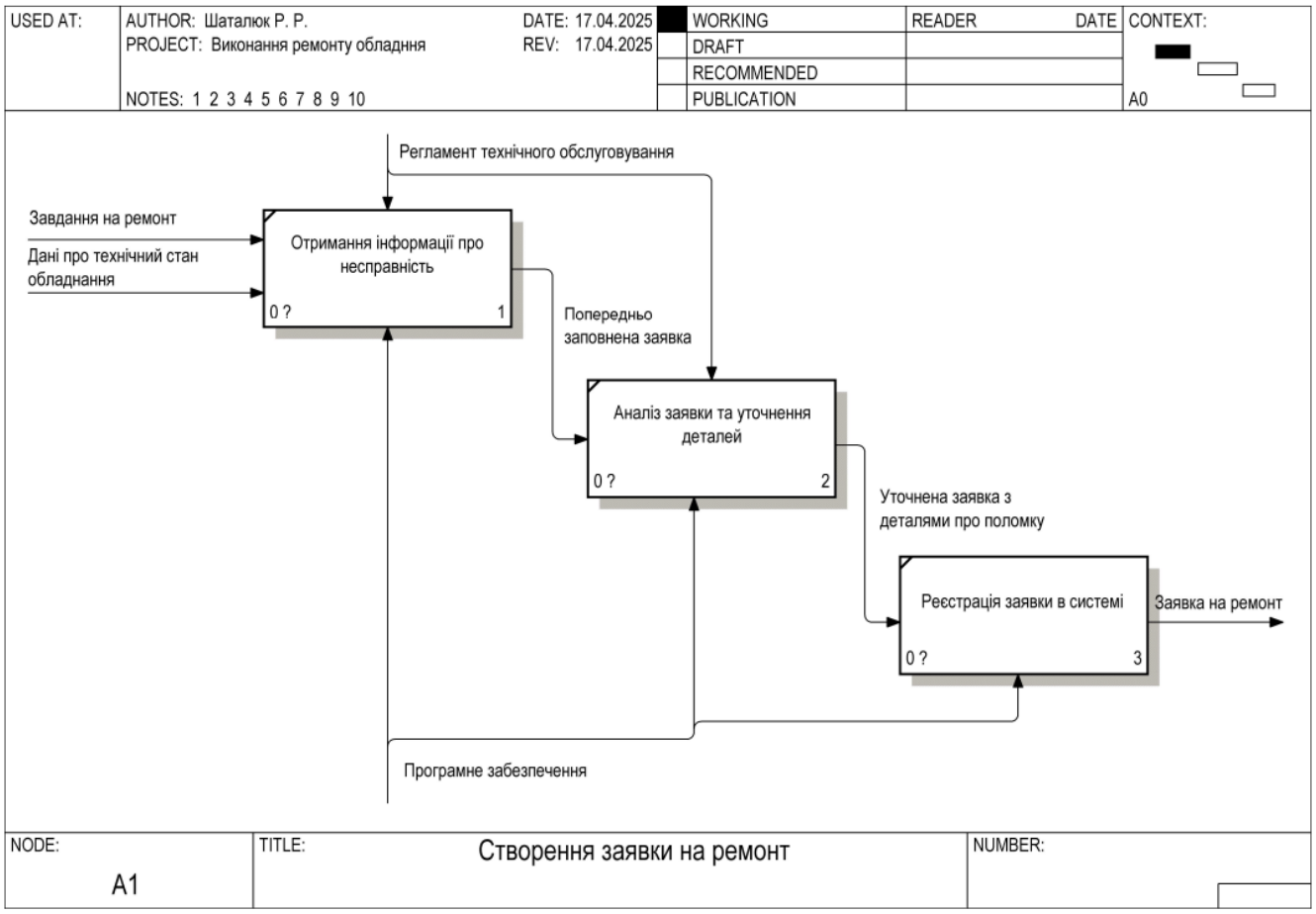


Рисунок 2.4 – Декомпозиція блоку A1

На діаграмі декомпозиції блоку A1 показано, як відбувається створення заявки на ремонт обладнання. Це ще не сам процес ремонту, а лише підготовчий етап, коли потрібно зібрати всю необхідну інформацію про поломку.

Спочатку працівник збирає інформацію про несправність тобто, що саме не працює, де знаходиться обладнання, фото несправності, який у нього технічний стан тощо.

Після цього формується попередньо заповнена заявка, яка далі аналізується на коректність введення даних. Якщо дані введено коректно, та заповнені всі поля та заявка реєструється у системі.

Далі побудуємо діаграму декомпозиції для блоку A2 з назвою виконання ремонту (рис. 2.5).

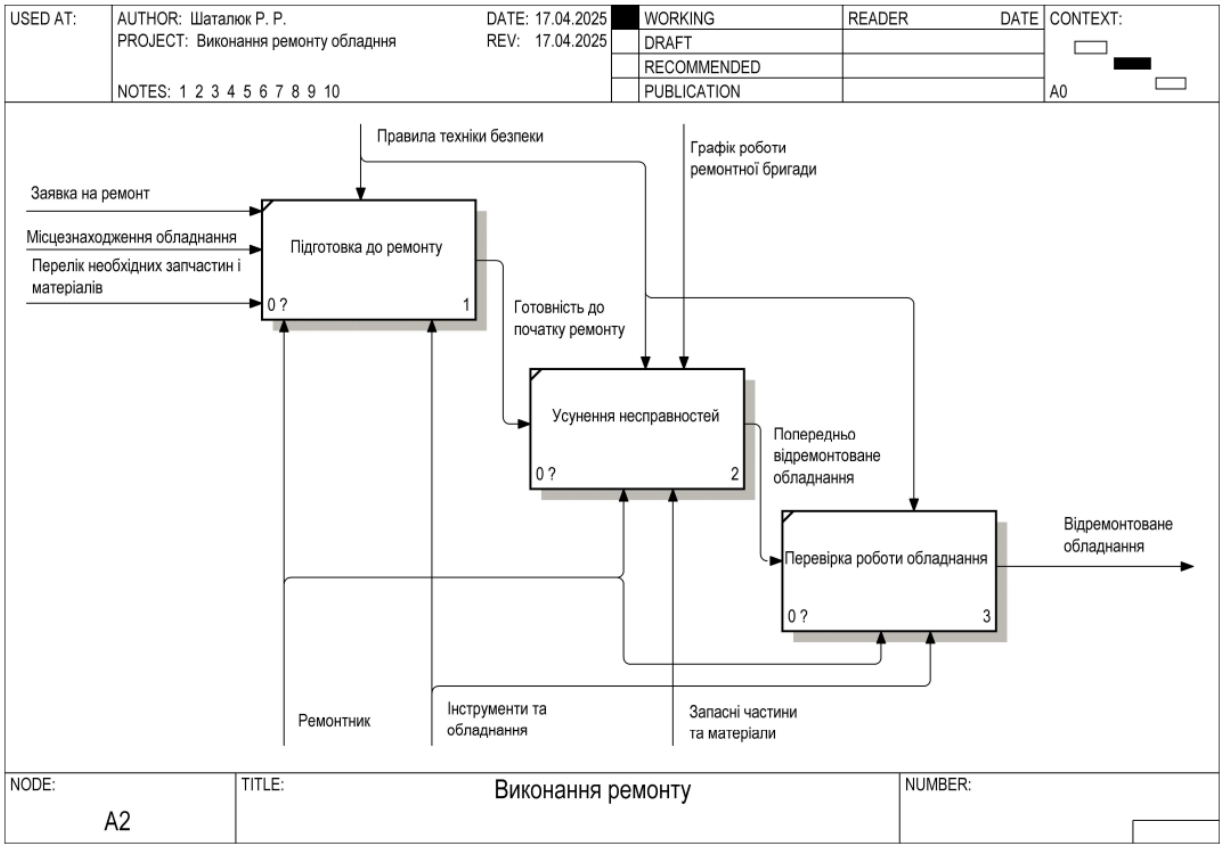


Рисунок 2.5 – Декомпозиція блоку A2

Процес ремонту починається коли у систему надходить заявка, де вказується місцезнаходження обладнання та попередньо заповнений перелік необхідних запчастин і матеріалів які знадобляться у ремонті. Використовуючи цю інформації ремонтник готує інструменти та обладнання необхідні для ремонту устаткування, перевіряється наявність усіх потрібних ресурсів. Якщо необхідних ресурсів для ремонту немає, то ремонтник може створити відповідну накладну на замовлення запчастини у системі.

При завершенні підготовчих робіт ремонтник приступає до усунення несправностей та коли несправність усунено, обладнання переходить до етапу перевірки. Перевіряється, чи правильно функціонує система після ремонту, чи всі дефекти усунено. І лише після успішного завершення цієї перевірки обладнання вважається повністю відремонтованим і готовим до подальшої експлуатації.

І нарешті побудуємо діаграму декомпозиції для блоку A3 з назвою закриття заявки на ремонт (рис. 2.6).

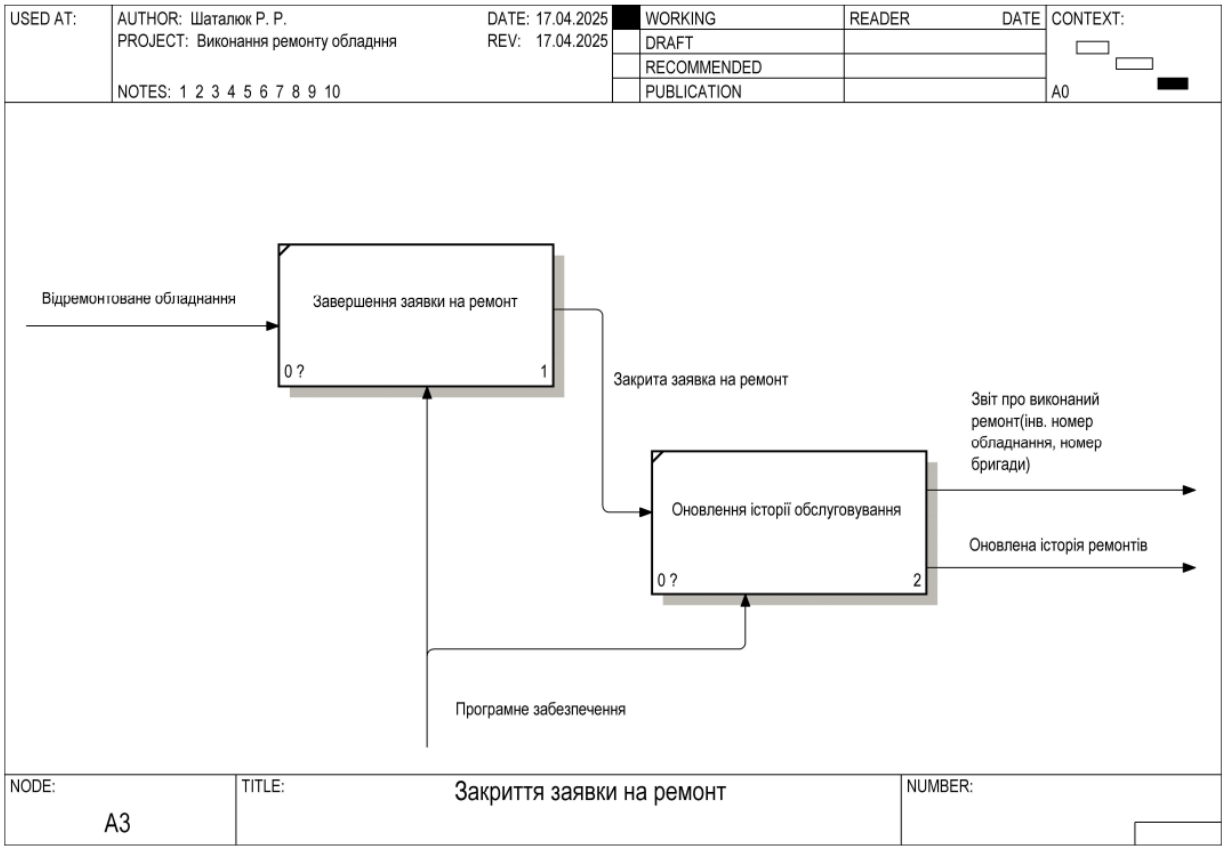


Рисунок 2.6 – Декомпозиція блоку А3

Діаграма показує етапи закриття заявки на ремонт після того, як обладнання вже відремонтоване.

На цьому етапі спочатку відбувається завершення заявки на ремонт тобто фіксується, що роботи виконані, і заявка більше неактуальна. Після того як заявка закрита, відбувається оновлення історії обслуговування. Завдяки цьому потім можна легко переглянути всю історію ремонтів.

2.3 Інженерні та техніко-економічні розрахунки

Розрахунок техніко-економічних показників (ТЕП) – стандартні та обов'язкові обчислення під час проєктування СА.

Критеріями керування системою є техніко-економічні показники (наприклад, собівартість кінцевого продукту, продуктивність технологічного об'єкта управління (ТОУ) при стандартній якості продукту) або технологічні показники

(наприклад, параметри технологічного процесу; параметри якості кінцевого продукту) [19].

До ТЕП процесу входять величини, що комплексно описують ТОУ у конкретний момент або за певний проміжок часу: виробництво основних та супутніх продуктів; витрати усіх типів сировини, палива, електроенергії, пари, повітря, води, допоміжних матеріалів тощо; питомі витрати цих потоків на 1 т основного товарного продукту, що виробляється; продуктивність ТОУ за сировиною та основним продуктом; технологічна собівартість 1 т основного товарного продукту і т.п.

Розпочнемо з розрахунку перед виробничих витрат запроектованої конструкції.

Щоб визначити перед виробничі витрати запроектованої системи, задаємо трудові витрати Т, виміряні у людино-днях.

Для ефективного використання часу, призначеного для будь-якого трудового процесу на виробництві, потрібно знати, наскільки швидко впорається з цією роботою одна людина. Зрозумівши, як оперативно співробітник може вирішити конкретне завдання, можна суттєво поліпшити показники ефективності та збільшити продуктивність.

Обчислимо кількість людино-годин для працівників на повному робочому дні.

При п'ятиденному робочому тижні та восьмигодинному робочому дні розрахунок буде такий [20]:

$$21 \text{ тиждень } \cdot 8 \text{ годин} = 168 \text{ людино годин.}$$

Тоді трудові витрати Т дорівнюють 21 людино-днів.

Розрахунок витрат наведено у табл. 2.2. Для визначення основної заробітної плати розробників СА використовуємо середньоденну заробітну плату:

$$З_{ср.тн.} = \frac{З_{міс}}{Д}, \quad (2.1)$$

де $З_{міс}$ – середня заробітна плата розробника за місяць;

Д – кількість робочих днів в місяці, від 20 днів до 21 дня.

Розрахуємо середньоденну заробітну плату:

$$З_{ср.тн.} = \frac{15000}{21} = 714.29 \text{ грн.}$$

Таблиця 2.2 – Розрахунок перед виробничих витрат

Статті витрат	Формула	Розрахунок (грн.)
Основна заробітна плата розробників	$З_о = T \cdot З_{ср.тн}$	$21 \cdot 714.29 = 15000$
Додаткова заробітна плата розробників	$З_д = (0.1 \div 0.2) \cdot З_о$	$0.2 \cdot 15000 = 3000$
Вартість оренди комп'ютерів (виділеного сервера)	$З_{ок} = С_{ок} \cdot Т_{ок}$	1 568
Загальногосподарські витрати (заробітна плата керівництва фірми, вартість ліцензії тощо)	$З_{зг} = 0.6 \cdot (З_о + З_{д)}$	$0.6 \cdot 18000 = 10800.$
Відрахування на соціальне страхування	$З_{сс} = 0.014 \cdot (З_о + З_{д)}$	$0.014 \cdot 18000 = 252.$
Відрахування на соціальне страхування на випадок безробіття	$З_{сб} = 0.016 \cdot (З_о + З_{д)}$	$0.016 \cdot 18000 = 288.$
Відрахування до пенсійного фонду	$З_{пф} = 0.332 \cdot (З_о + З_{д)}$	$0.332 \cdot 18000 = 5976.$

Продовження таблиці 2.2

Статті витрат	Формула	Розрахунок (грн.)
Відрахування на соціальне страхування від нещасних випадків	$З_{сс} = 0.013 \cdot (З_о + З_{д)}$	$0.013 \cdot 18000 = = 234.$
Комунальний податок	$П_к = 0.18 \cdot З_о$	$0.18 \cdot 15000 = 2700.$
Разом	$К_п$	39818

Розрахунок річних експлуатаційних витрат:

$$С_р = З_{оп} + З_{дп} + В_{сз} + В_{зг} + П_к + В_р, \quad (2.2)$$

де $З_{оп}$ – основна заробітна плата обслуговуючого персоналу за рік;

$З_{дп}$ – додаткова заробітна плата обслуговуючого персоналу за рік;

$В_{сз}$ – відрахування на соціальні заходи;

$В_{зг}$ – загальногосподарські витрати;

$П_к$ – комунальний податок;

$В_р$ – витрати на поточний ремонт.

Для обчислення основної заробітної плати обслуговуючого персоналу протягом року використовуємо табл. 2.3.

Таблиця 2.3 – Розрахунок річної основної заробітної плати обслуговуючого персоналу

Посада	Чисельність персоналу	Основна заробітна плата протягом року, грн.
Адміністратор	1	15000
Працівник	1	17000
Ремонтник	1	17000
Разом	3	49000

Додаткова заробітна плата обслуговуючого персоналу за рік становить (20-40) % від основної заробітної плати обслуговуючого персоналу.

$$З_{дп} = (0.1 \div 0.2) \cdot З_{оп} = 0.1 \cdot 49000 = 4900 \text{ грн.} \quad (2.3)$$

Відрахування на соціальні заходи включають:

– до пенсійного фонду, (33.2% від основної та додаткової заробітної плати);

– на соціальне страхування, (1.4% від основної та додаткової заробітної плати);

– на соціальне страхування на випадок безробіття, (1.6% від основної та додаткової заробітної плати);

– на соціальне страхування від нещасних випадків та профзахворювань (1.4% від основної та додаткової заробітної плати).

Таким чином, відрахування на соціальні заходи обчислюються за формулою:

$$В_{сз} = \frac{37.6}{100} \cdot (З_{оп} + З_{дп}), \quad (2.4)$$

$$В_{сз} = \frac{37.6}{100} \cdot (49000 + 4900) = 0.376 \cdot 53900 = 20266.4 \text{ грн.}$$

Загальногосподарські витрати визначаються таким чином:

$$В_{зг} = 0.7 \cdot (З_{оп} + З_{дп}), \quad (2.5)$$

$$В_{зг} = 0.7 \cdot (49000 + 4900) = 0.7 \cdot 53900 = 37730 \text{ грн.}$$

Комунальний податок за рік обчислюється за формулою:

$$П_{к} = 0.18 \cdot З_{оп} \cdot (Е_1 + \cdots + Е_м), \quad (2.6)$$

де $E_1 \ldots E_m$ – загальна чисельність обслуговуючого персоналу.

$$П_к = 0.18 \cdot 2700 \cdot 3 = 1458 \text{ грн.}$$

Результати розрахунків місячних експлуатаційних витрат за базовим та новим варіантами вносимо до табл. 2.4.

Таблиця 2.4 – Розрахунок місячних експлуатаційних витрат

Статті витрат	Значення
Основна заробітна плата обслуговуючого персоналу	49000
Додаткова заробітна плата обслуговуючого персоналу	4900
Відрахування на соціальні заходи	20266.4
Загальногосподарські витрати	37730
Комунальний податок	1458
Разом	113354.4

2.4 Застосування елементів теорії автоматичного управління в системі

2.4.1 Визначення передавальної функції замкненої системи з урахуванням зворотного зв'язку

Систему автоматизації технічного обслуговування обладнання можна розглядати як систему автоматичного регулювання, у якій вхідними параметрами є обсяг запитів на обслуговування, а вихідними – виконані ремонтні дії або усунуті несправності. З метою аналізу ефективності роботи системи використаємо апарат лінійних динамічних систем.

Припустимо, що протягом робочого дня на систему надходить у середньому 3 заявки на технічне обслуговування на день. Середній час реагування оператора (або системи) до моменту запуску процесу виконання заявки складає 15 хвилин, а середній час виконання однієї заявки 120 хвилин.

Таким чином, загальний цикл обслуговування однієї заявки:

$$T_{\text{ц}} = T_{\text{р}} + T_{\text{в}}, \tag{2.7}$$

де $T_{\text{ц}}$ – загальний час циклу обслуговування;

$T_{\text{р}}$ – час реакції на створену заявку;

$T_{\text{в}}$ – час виконання заявки при наявності необхідних запчастин.

Підставимо значення у формулу (2.7):

$$T_{\text{ц}} = 15 + 120 = 135 \text{ хв} = 2.25 \text{ год.}$$

У рамках побудови математичної моделі системи обслуговування обладнання приймається припущення, що її поведінка відповідає аперіодичній ланці першого порядку. Це обґрунтовано тим, що в системі немає коливальних процесів або перевищень вихідного сигналу, які притаманні системам з коливальними властивостями. Заявка надходить, обробляється із певною інерцією, і після завершення система повертається у вихідний стан без реверсивних впливів чи накопичення помилок.

Крім того, система має сталу затримку у реагуванні, яка є сумою часу очікування та часу виконання заявки. Така динаміка характерна саме для аперіодичних систем. Це дозволяє адекватно змоделювати її поведінку за допомогою передавальної функції виду:

$$G(s) = \frac{K}{Ts + 1}, \tag{2.8}$$

де T – стала часу, що відповідає загальній тривалості виконання однієї заявки;

K – коефіцієнт підсилення, що характеризує максимальну продуктивність системи.

Припускаємо, що при максимальному навантаженні система обробляє 5 заявок за 10-годинний робочий день. Знайдемо коефіцієнт підсилення з визначеними значеннями:

$$K = \frac{5}{10} = 0.5 \text{ заявок/год.}$$

Для нашої системи передавальна функція має наступний вигляд:

$$G(s) = \frac{0.5}{2.25s + 1}.$$

Для аналізу стійкості системи технічного обслуговування обладнання, ми маємо визначити передавальну функцію замкненої системи з урахуванням зворотного зв’язку. Тож визначимо передавальну функцію замкненої системи, враховуючи одиничний зворотний зв'язок. Згідно з теорією автоматичного управління, передавальна функція замкненої системи має вигляд:

$$W(s) = \frac{G(s)}{1 + G(s)}, \tag{2.9}$$

де *G(s)* – передавальна функція відкритої частини системи (тобто без урахування реакції на зворотний зв’язок).

Підставимо вираз для *G(s)*:

$$W(s) = \frac{\frac{0.5}{2.25s + 1}}{1 + \frac{0.5}{2.25s + 1}} = \frac{\frac{0.5}{2.25s + 1}}{\frac{2.25s + 1}{2.25s + 1} + \frac{0.5}{2.25s + 1}} = \frac{\frac{0.5}{2.25s + 1}}{\frac{2.25s + 1 + 0.5}{2.25s + 1}} =$$

$$= \frac{0.5}{2.25s + 1 + 0.5} = \frac{0.5}{2.25s + 1.5}.$$

Отримана передавальна функція W(s) описує поведінку замкненої системи, де враховано зворотне коригування на основі поточної продуктивності та затримок у обробці заявок.

2.4.2 Аналіз стійкості системи за допомогою алгебраїчних та частотних критеріїв

Для нашої системи передавальна функція має такий вигляд:

$$W(s) = \frac{0.5}{2.25s + 1.5}.$$

Щоб система вважалась стійкою за кореневим критерієм, потрібно й достатньо, щоб усі корені її характеристичного рівняння мали від'ємні дійсні частини, тобто Re λ < 0 [21].

Таких коренів називають «лівими», адже на комплексній площині вони лежать ліворуч від уявної осі. Відповідно, корені з додатними дійсними частинами називають «правими». Для аналізу нашої системи характеристичне рівняння передавальної функції набуває такого вигляду:

$$2.25s + 1.5 = 0.$$

Визначимо корені для нашого характеристичного рівняння:

$$s = -\frac{1.5}{2.25} = -0.6.$$

За кореневим критерієм можна визначити, що система є стійкою.

Згідно з критерієм Гурвіца, система автоматичного керування вважається стійкою лише за тієї умови, що всі визначники матриці Гурвіца мають додатні значення $\Delta_i > 0$, i = $\overline{1,n}$ при $a_0 > 0$. [21]

Нехай характеристичне рівняння нашої системи має вигляд:

$$2.25s + 1.5 = 0.$$

Визначимо коефіцієнти нашого рівняння:

$$a_0 = 2.25; a_1 = 1.5.$$

У випадку системи першого порядку перевірка стійкості за критерієм Гурвіца зводиться лише до оцінки знаків її коефіцієнтів, адже їх усього два. У цьому випадку визначник Гурвіца Δ_1 фактично дорівнює коефіцієнту a_1:

$$\Delta_1 = 2.25.$$

За критерієм Гурвіца система є стійкою оскільки $\Delta_1 > 0$.

Критерій Михайлова стверджує, що система автоматичного керування буде стійкою тоді й лише тоді, коли її амплітудно-фазова характеристика(так звана крива Михайлова) при зростанні частоти ω від 0 до нескінченності, починаючись на додатній частині дійсної осі (за умови $a_0 > 0$), послідовно проходить n квадрантів комплексної площини проти годинникової стрілки та не перетинає початку координат [21].

Запишемо нашу передавальну функцію:

$$D(s) = 2.25s + 1.5.$$

Підставимо $s=j\omega$ та отримаємо вираз для побудови кривої Михайлова:

$$D(j\omega) = 2.25j\omega + 1.5.$$

Визначимо уявну та дійсну частину виразу:

$$Re(\omega) = 1.5,$$

$$Im(\omega) = 2.25\omega.$$

Для побудови кривої Михайлова розрахуємо значення дійсної та уявної частотних функцій при зміні частоти від 0 до ∞ та представимо розраховані дані у вигляді табл. 2.5.

Таблиця 2.5 – Розрахункові дані

ω	$Re(\omega)$	$Im(\omega)$
0	1,5	0
1	1,5	2,25
∞	1,5	$+\infty$

Побудуємо криву Михайлова за допомогою Matlab рис. 2.7.

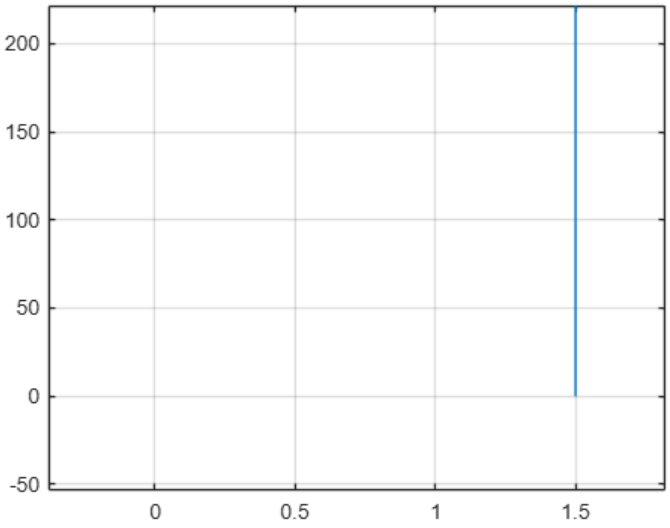


Рисунок 2.7 – Крива Михайлова

Оскільки крива Михайлова залишається в першій чверті, а порядок системи дорівнює одному, то умова критерію Михайлова виконується. Отже, система є стійкою за критерієм Михайлова.

Відповідно до критеріїв Найквіста, для забезпечення стійкості замкнутої системи необхідною й достатньою умовою є те, щоб амплітудно-фазова характеристика (АФЧХ) розімкнутої системи при зростанні частоти від 0 до нескінченності охоплювала точку з координатами (–1; j0) проти годинникової стрілки l/2 разів, де l кількість коренів характеристичного рівняння розімкнутої системи, що мають додатні дійсні частини [21].

Наведемо корені, які ми отримали при застосуванні алгебраїчного критерію:

$$s = -0.6.$$

Підставимо *s=jω* для передавальної функції яка має вигляд:

$$W(s) = \frac{0.5}{2.25s + 1.5},$$

$$W(j\omega) = \frac{0.5}{2.25j\omega + 1.5} = \frac{0.5}{2.25j\omega + 1.5} * \frac{2.25j\omega - 1.5}{2.25j\omega - 1.5} =$$
$$= \frac{0.5 * (2.25j\omega - 1.5)}{(2.25j\omega)^2 - 1.5^2} = \frac{1.13j\omega - 0.75}{-5.1\omega^2 - 2.25}.$$

Виділимо реальну та уявну частину:

$$Re(\omega) = \frac{-0.75}{-5.1\omega^2 - 2.25},$$

$$Im(\omega) = \frac{1.13\omega}{-5.1\omega^2 - 2.25}.$$

Для побудови амплітудно-фазочастотної характеристики розрахуємо значення дійсної та уявної частотних функцій при зміні частоти від 0 до ∞ та представимо розраховані дані у вигляді табл. 2.6.

Таблиця 2.6 – Розрахункові дані

ω	$Re(\omega)$	$Im(\omega)$
0	0,33	0
1	0,10	0,15
∞	0	0

Визначимо скільки разів АФЧХ повинна охоплювати точку (-1, j_0):

$$l = 0 => \frac{0}{2} = 0.$$

Побудуємо амплітудно-фазочастотну характеристику (АФЧХ) для розімкнутої системи за допомогою Matlab рис. 2.8.

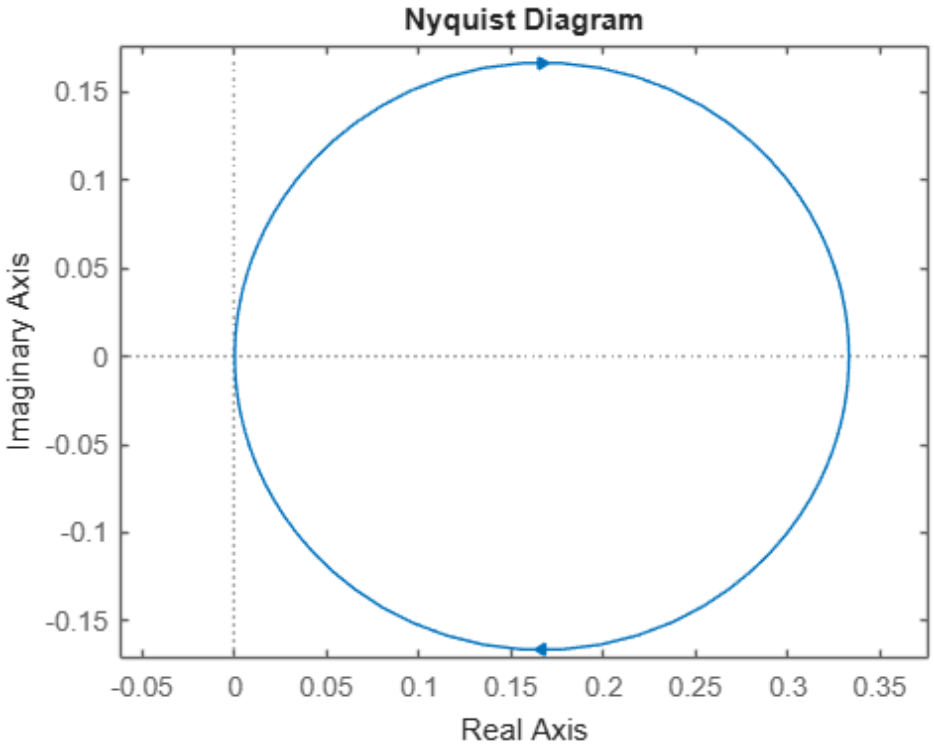


Рисунок 2.8 – Амплітудно-фазочастотна характеристика (АФЧХ)

За критерієм Найквіста амплітудна-фазочастотна характеристика охоплює точку (-1; j_0) 0 разів, тому можна визначити, що система є стійкою.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ ОБЛАДНАННЯ

3.1 Визначення функцій програмного забезпечення

Перш ніж розпочинати безпосередню розробку програмного забезпечення, необхідно визначити його основні функціональні можливості, що випливають із потреб користувачів, які будуть взаємодіяти із системою. Для нашого програмного забезпечення, яке має автоматизувати технічне обслуговування на підприємстві, можна визначити 3 актора, а саме: адміністратор, робітник та ремонтник.

Для актора адміністратора, який має повний доступ до даних у застосунку, управляє інформацією щодо верстатів, працівників та постачальників, а також контролює заявки, що були створені у застосунку, були визначені наступні функціональні можливості:

– вхід у систему. Здійснення авторизації користувача на основі облікових даних із перевіркою прав доступу до адміністративного функціоналу;

– вихід із системи. Завершення сеансу роботи адміністратора з подальшим автоматичним переходом до вікна входу до застосунку;

– редагування даних профілю. Можливість змінювати власну контактну інформацію, пароль, ім’я користувача та фото;

– перехід по вкладкам застосунку. Перехід по вкладкам застосунку які доступні лише адміністратору за допомогою навігаційного меню, та перегляд таблиць, що зображені на вкладках;

– керування інформацією про обладнання (верстатів). Створення нових записів про обладнання з внесенням технічних характеристик, інвентарних номерів, місця розташування, дати останнього технічного обслуговування тощо, а також оновлення (редагування) наявних записів для актуалізації інформації;

– видалення обладнання (верстатів). Видалення застарілої або некоректної інформації про обладнання з бази даних;

– пошук за параметрами в таблиці верстатів. Реалізація фільтрації даних за заданими критеріями (назва, модель, серійний номер або номер цеха);

– керування інформацією про працівників. Створення облікових записів співробітників із зазначенням ПІБ, посади, підрозділу, контактних даних, а також оновлення персональних або посадових даних у разі змін;

– видалення інформації про працівників. Видалення облікових записів співробітників, які більше не працюють на підприємстві або не потребують доступу до системи;

– пошук за параметрами в таблиці працівників. Фільтрація записів за критеріями, як-от ім’я, роль або цех роботи;

– керування інформацією про постачальників. Створення записів компаній-постачальників із зазначенням назви, контактних осіб, асортименту товарів, а також оновлення інформації про постачальників у разі потреби;

– видалення інформації про постачальників. Видалення записів про постачальників, з якими підприємство більше не співпрацює;

– пошук за параметрами в таблиці постачальників. Здійснення пошуку постачальників за різними критеріями;

– перегляд журналу ремонтів. Доступ до повного списку створених заявок на ремонт або технічне обслуговування;

– перегляд детальної інформації за обраною заявкою. Отримання повного опису проблеми, даних про заявника, статус виконання, призначеного ремонтника, вкладених зображень тощо;

– пошук за параметрами в журналі ремонтів. Пошуку даних за конкретними критеріями, зокрема назва верстату, ім’я робітника тощо;

– створення звіту по кожній таблиці. Автоматизоване формування звітів у табличному форматі на основі даних таблиць (верстатів, журналу ремонтів, працівників або постачальників) для подальшого аналізу або архівації;

– зміна стану заявки. Можливість адміністратора вручну змінити статус заявки у разі помилкового завершення заявки або потреби доопрацювання заявки.

Функціональні можливості зображені у вигляді діаграми використання можна побачити на рис. 3.1.

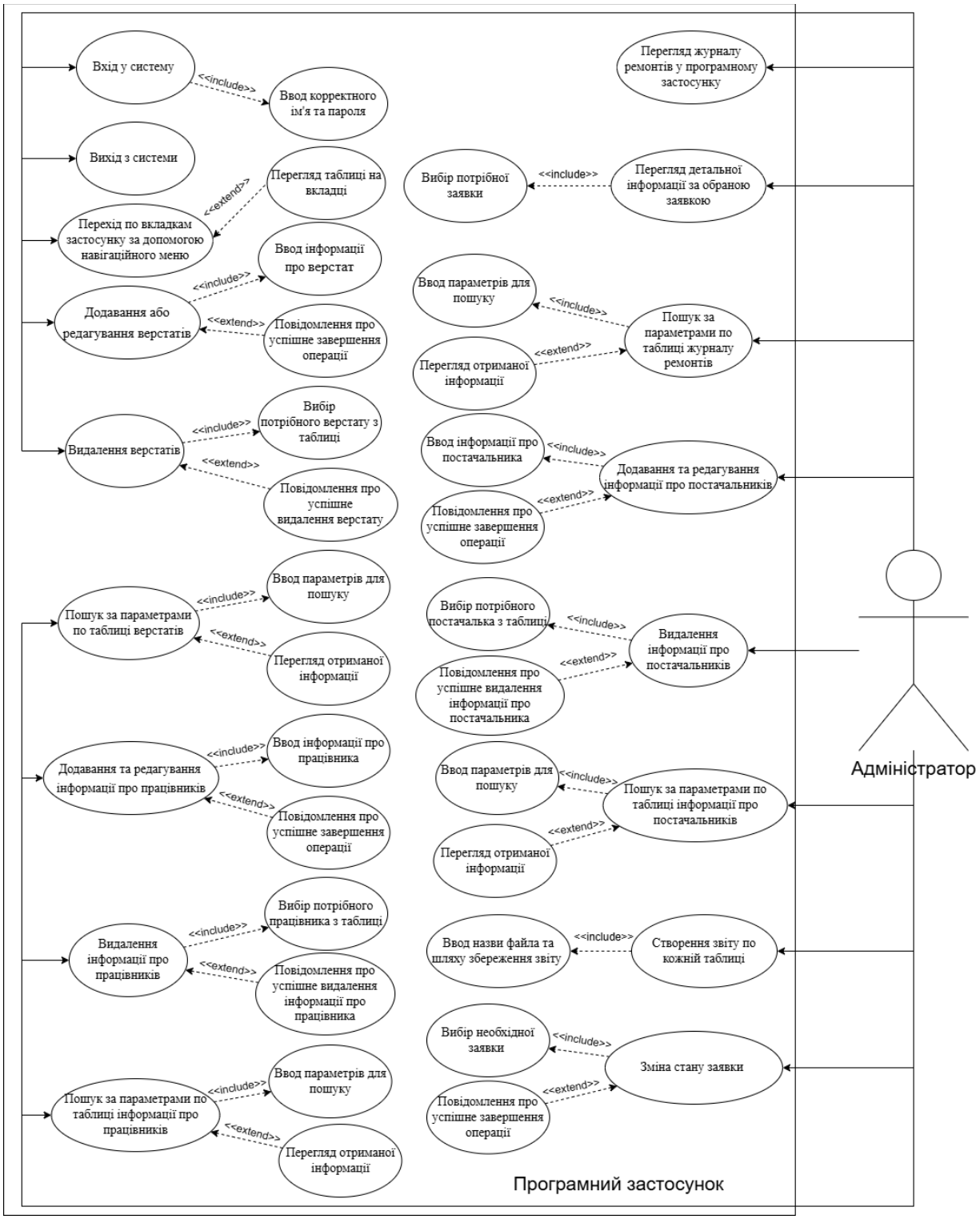


Рисунок 3.1 – Діаграма використання застосунку для актора адміністратор

Робітник є актором, який створює заявки на ремонт верстатів у разі необхідності, тому має дещо обмежений функціонал порівняно з адміністратором. Функціональні можливості робітника з поясненням виглядають наступним чином:

– вхід у систему. Авторизація користувача на основі облікових даних із перевіркою прав доступу до функціоналу робітника;

– вихід із системи. Завершення роботи з застосунком з подальшим автоматичним переходом до вікна входу до застосунку;

– редагування даних профілю. Можливість змінювати власну контактну інформацію, пароль, ім’я користувача та фото;

– перехід по вкладкам застосунку. Перехід по вкладкам застосунку які доступні лише робітнику за допомогою навігаційного меню, та перегляд таблиць, що зображені на вкладках;

– створення заявки на ремонт верстату. Функція формування нової заявки з вказанням опису проблеми, вибору конкретного верстату та фото несправності для початку процесу ремонту;

– перегляд статусу заявок. Можливість отримувати актуальну інформацію про поточний стан створених робітником заявок (наприклад, “нова”, “виконується” або “завершена”) для контролю ходу виконання;

– видалення заявки. Можливість робітника вручну видаляти заявку при необхідності (наприклад, якщо заявка має статус “нової ” або “відхиленої”) у разі помилкового створення заявки або не коректного написання опису до заявки;

– перегляд інформації про створену заявку. Надання детальної інформації щодо конкретної заявки, включаючи опис проблеми, дату створення, призначених виконавців тощо;

– пошук по таблиці поданих заявок за параметрами. Реалізація фільтрації даних за заданими критеріями (назва верстату, ім’я ремонтника, статус заявки, номер цеха тощо);

– створення звіту по таблиці створених заявок. Автоматизоване формування звітів у табличному форматі на основі даних створених заявок для подальшого аналізу, архівації або звітності.

Функціональні можливості зображені у вигляді діаграми використання для актора робітника можна побачити на рис. 3.2.

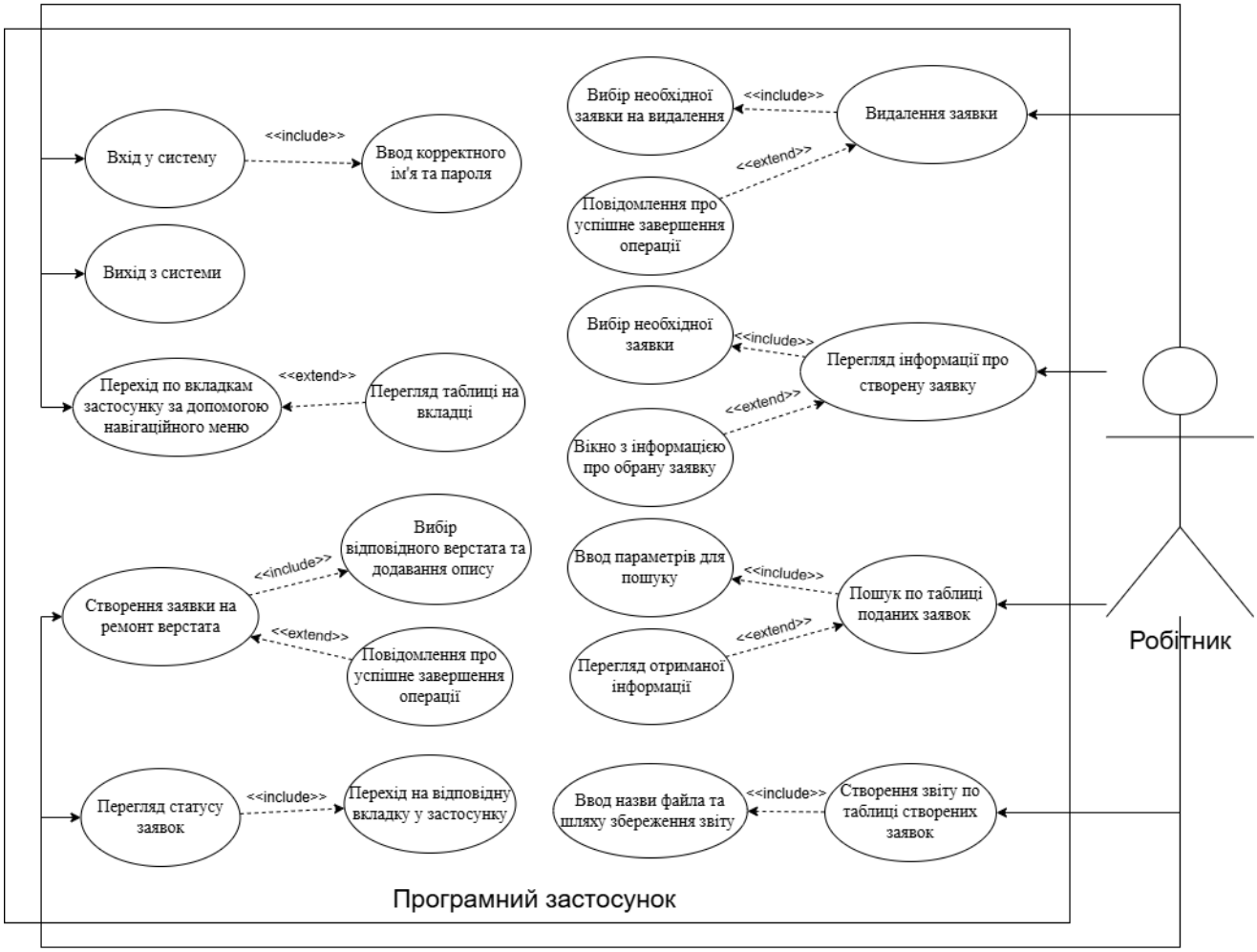


Рисунок 3.2 – Діаграма використання застосунку для актора робітник

Актор ремонтник є основним актором програмного застосунку та має такі функціональні можливості:

– вхід у систему. Авторизація користувача на основі облікових даних;

– вихід із системи. Завершення роботи з застосунком з подальшим автоматичним переходом до вікна входу до застосунку;

– редагування даних профілю. Можливість змінювати власну контактну інформацію, пароль, ім’я користувача та фото;

– перехід по вкладкам застосунку. Перехід по вкладкам застосунку які доступні лише робітнику за допомогою навігаційного меню;

– перегляд повідомлень про необхідність ремонту верстатів або технічного обслуговування;

– перегляд інформації про заявки. Детальний перегляд даних заявок;

– прийняття у роботу заявки або технічного обслуговування. Можливість підтвердити відповідальність за виконання робіт по конкретній заявці;

– додавання нових деталей до заявки;

– зміна стану заявки;

– додавання або редагування запчастини. Можливість вносити нові записи про запчастини або коригувати існуючі дані у відповідній таблиці;

– видалення запчастини. Видалення даних про запчастини з бази даних;

– пошук за параметрами по таблиці запчастин. Пошук інформації про запчастини за різними критеріями;

– створення заявки на замовлення нових запчастин. Формування запиту на закупівлю необхідних деталей для ремонту або технічного обслуговування;

– завершення заявок на замовлення запчастин. Позначення заявок як виконаних після отримання необхідних деталей;

– пошук по таблиці заявок замовлення запчастин. Фільтрація заявок за параметрами для швидкого знаходження потрібної інформації;

– пошук за параметрами по таблиці наявних верстатів. Здійснення пошуку та фільтрації верстатів за різними критеріями;

– створення заявки на технічне обслуговування. Ініціювання запиту на проведення робіт з технічного обслуговування верстату;

– зміна дати технічного обслуговування. Можливість коригувати дату ТО;

– створення звіту по кожній таблиці. Автоматизоване формування звітів у табличному форматі на основі таблиць (запчастин або замовлених запчастин).

Функціональні можливості зображені у вигляді діаграми використання для актора ремонтника можна побачити на рис. 3.3.

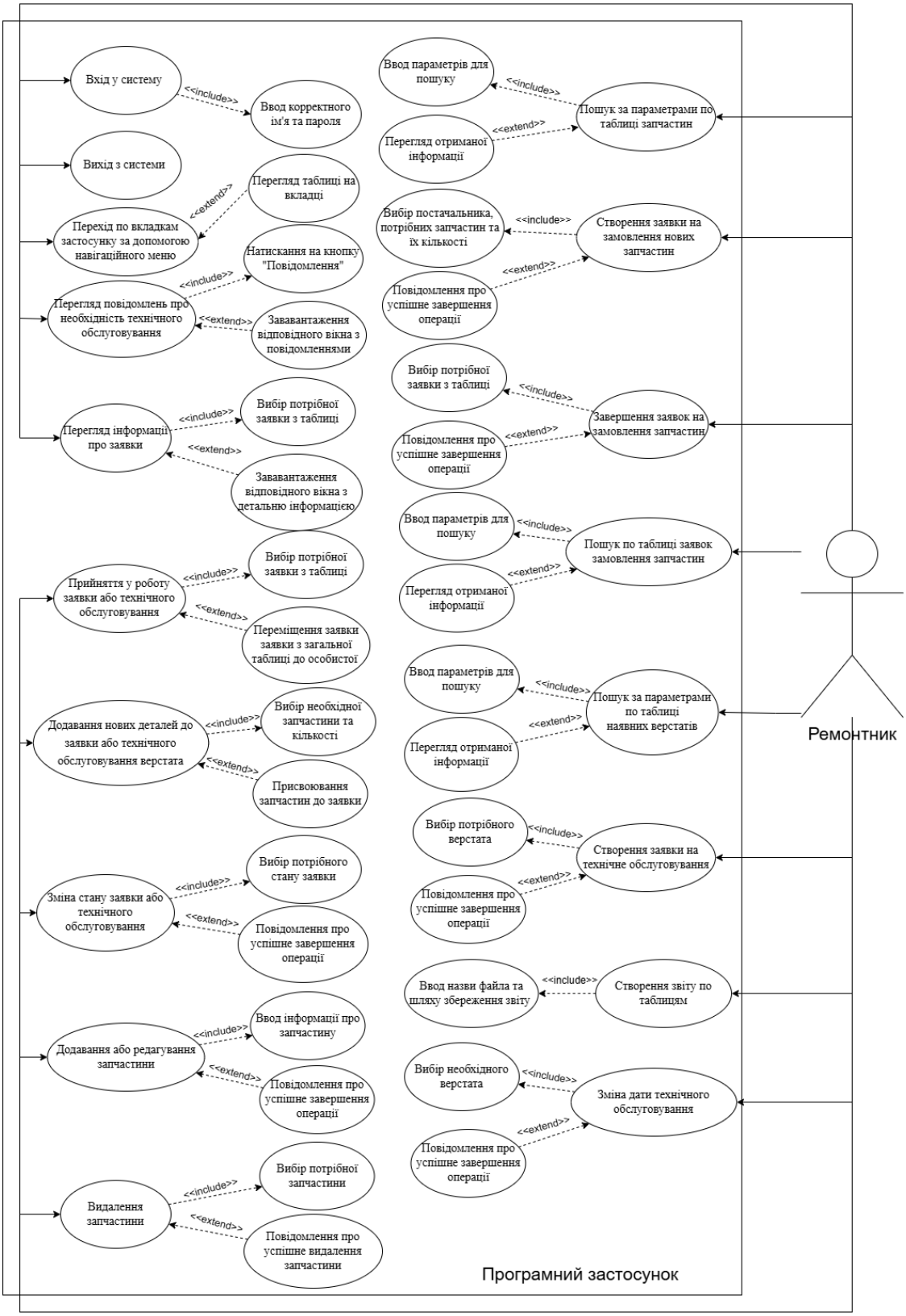


Рисунок 3.3 – Діаграма використання застосунку для актора ремонтник

3.2 Розробка бази даних для програмного застосунку

Визначивши функціональні можливості для кожного актора, ми створимо таблиці для бази даних, що є основою для програмного застосунку, у вигляді створення таблиці для користувачів, майстерень, машин, заявок на ремонт, повідомлень, постачальників, деталей та замовлень. Кожна таблиця бази даних відповідає окремому поняттю предметної області, а взаємозв'язки між ними реалізуються через зовнішні ключі, що забезпечують цілісність даних. Таблиці нашого програмного застосунку повинні містити поля відповідно до табл. 3.1 – 3.8.

Таблиця 3.1 – Таблиця «Користувачі»

Поле	Тип	Обмеження	Назва
id	INT	PRIMARY KEY, AUTO_INCREMENT	Ідентифікатор
username	VARCHAR(50)	NOT NULL, UNIQUE	Логін
password_hash	VARCHAR(255)	NOT NULL	Хеш паролю
role	ENUM	NOT NULL	Роль
workshop_id	INT	NULL	Цех
photo	LONGBLOB	NULL	Фото
email	VARCHAR(100)	NULL	Електронна пошта

Таблиця 3.2 – Таблиця «Верстати»

Поле	Тип	Обмеження	Назва
id	INT	PRIMARY KEY, AUTO_INCREMENT	Ідентифікатор
name	VARCHAR(100)	NOT NULL	Назва
model	VARCHAR(50)	NOT NULL	Модель

Продовження таблиці 3.2

Поле	Тип	Обмеження	Назва
serial_number	VARCHAR(50)	UNIQUE	Серійний номер
workshop_id	INT	FOREIGN KEY → workshops.id	Цех
status	ENUM	NOT NULL, DEFAULT 'Працює'	Статус
last_maintenance_date	DATE	NULL	Дата останнього ТО

Таблиця 3.3 – Таблиця «Цехи»

Поле	Тип	Обмеження	Назва
id	INT	PRIMARY KEY, AUTO_INCREMENT	Ідентифікатор
name	VARCHAR(100)	NOT NULL	Назва

Таблиця 3.4 – Таблиця «Заявки на ремонт»

Поле	Тип	Обмеження	Назва
id	INT	PRIMARY KEY, AUTO_INCREMENT	Ідентифікатор заявки
equipment_id	INT	FOREIGN KEY → equipment.id	Ідентифікатор обладнання
worker_id	INT	FOREIGN KEY → users.id	Автор заявки
workshop_id	INT	FOREIGN KEY → workshops.id	Цех
description	TEXT	NOT NULL	Опис проблеми

Продовження таблиці 3.4

Поле	Тип	Обмеження	Назва
photo_url	LONGBLOB	NULL	Фото
status	ENUM	NULL, DEFAULT NULL	Статус заявки
assigned_repairman_id	INT	FOREIGN KEY → users.id	Призначений ремонтник
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Дата створення
parts	VARCHAR (255)	NULL	Необхідні запчастини

Таблиця 3.5 – Таблиця «Сповіщення»

Поле	Тип	Обмеження	Назва
id	INT	PRIMARY KEY, AUTO_INCREMENT	Ідентифікатор
role	ENUM	NOT NULL	Роль адресата
message	VARCHAR (255)	NOT NULL	Повідомлення
date_created	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата створення
is_read	TINYINT(1)	DEFAULT 0	Прочитане/непрочитане

Таблиця 3.6 – Таблиця «Постачальники»

Поле	Тип	Обмеження	Назва
id	INT	PRIMARY KEY, AUTO_INCREMENT	Ідентифікатор
name	VARCHAR(255)	NOT NULL	Назва постачальника

Продовження таблиці 3.6

Поле	Тип	Обмеження	Назва
contact_person	VARCHAR(255)	NULL	Контактна особа
phone	VARCHAR(20)	NULL	Телефон
email	VARCHAR(100)	NULL	Електронна пошта
address	TEXT	NULL	Адреса

Таблиця 3.7 – Таблиця «Запчастини»

Поле	Тип	Обмеження	Назва
id	INT	PRIMARY KEY, AUTO_INCREMENT	Ідентифікатор
name	VARCHAR(100)	NOT NULL	Назва
quantity	INT	NOT NULL	Кількість
min_stock_level	INT	NOT NULL, DEFAULT 1	Мінімальний запас
auto_order	INT	DEFAULT 0	Автозамовлення

Таблиця 3.8 – Таблиця «Замовлення»

Поле	Тип	Обмеження	Назва
id	INT	PRIMARY KEY, AUTO_INCREMENT	Ідентифікатор
sparepart_id	INT	FOREIGN KEY → spareparts.id	Запчастина
quantity	INT	NOT NULL	Кількість
order_date	DATE	NOT NULL	Дата замовлення
supplier_id	INT	FOREIGN KEY → suppliers.id, ON DELETE SET NULL	Постачальник

Продовження таблиці 3.8

Поле	Тип	Обмеження	Назва
ordered_by	INT	FOREIGN KEY → users.id, ON DELETE SET NULL	Хто замовив
status	ENUM	DEFAULT 'Очікується'	Статус

Визначивши структуру бази даних, написали код для створення бази даних(Додаток Б) та створили EER діаграму яку можна побачити на рис. 3.4.

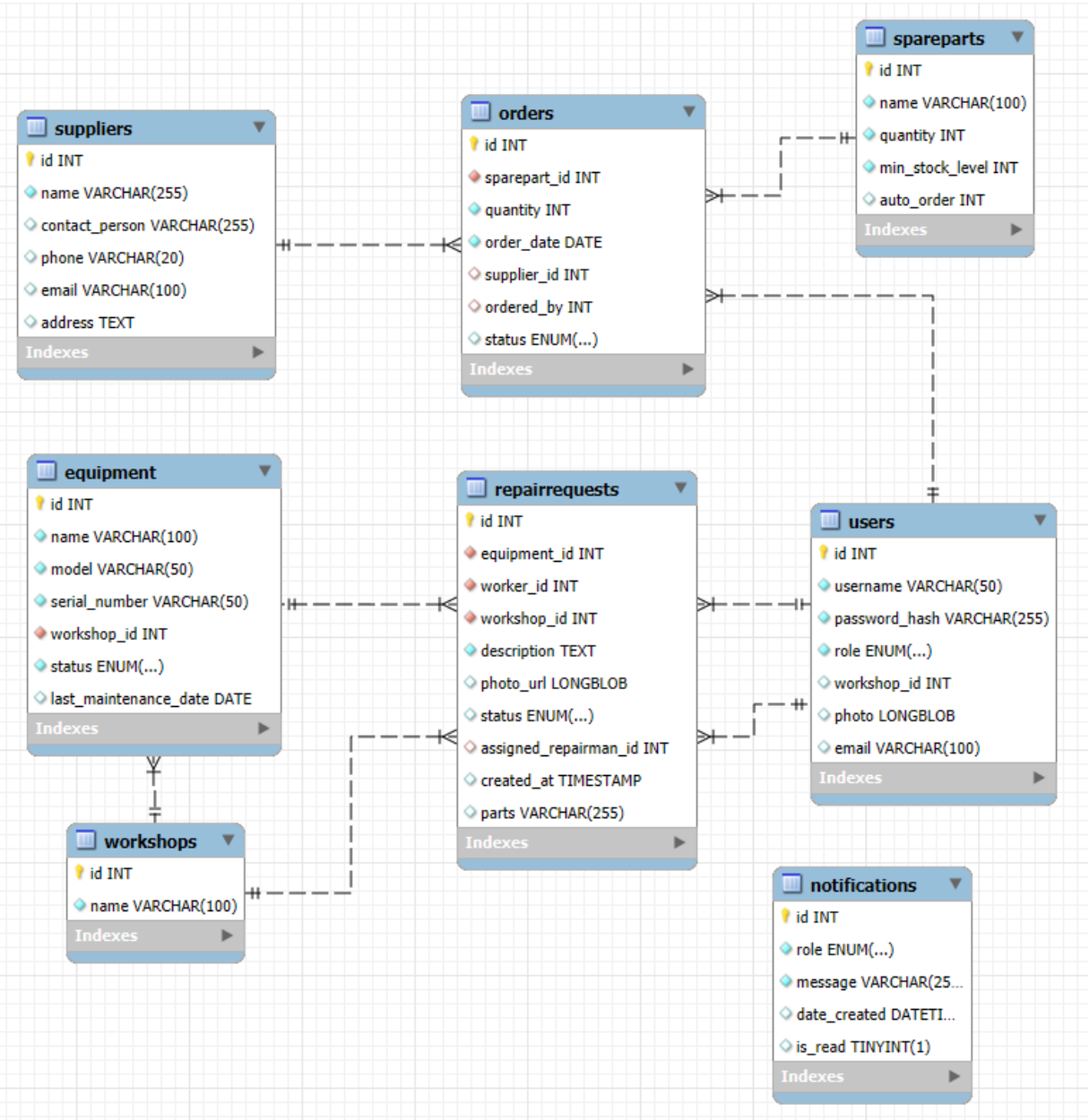


Рисунок 3.4 – EER діаграма створеної бази даних

3.3 Створення програмного застосунку

Перед створенням програмного коду при використанні WinForms потрібно спочатку створити форми з наповненням, а потім написати функціонал, зв’язавши його з формами. Розглянемо екранні форми для ремонтника (головного актора застосунку).

Спочатку було створено вікно входу з полями для введення імені та пароля, а також кнопками для входу до застосунку або виходу з нього. Вікно входу до застосунку зображене на рис. 3.5.

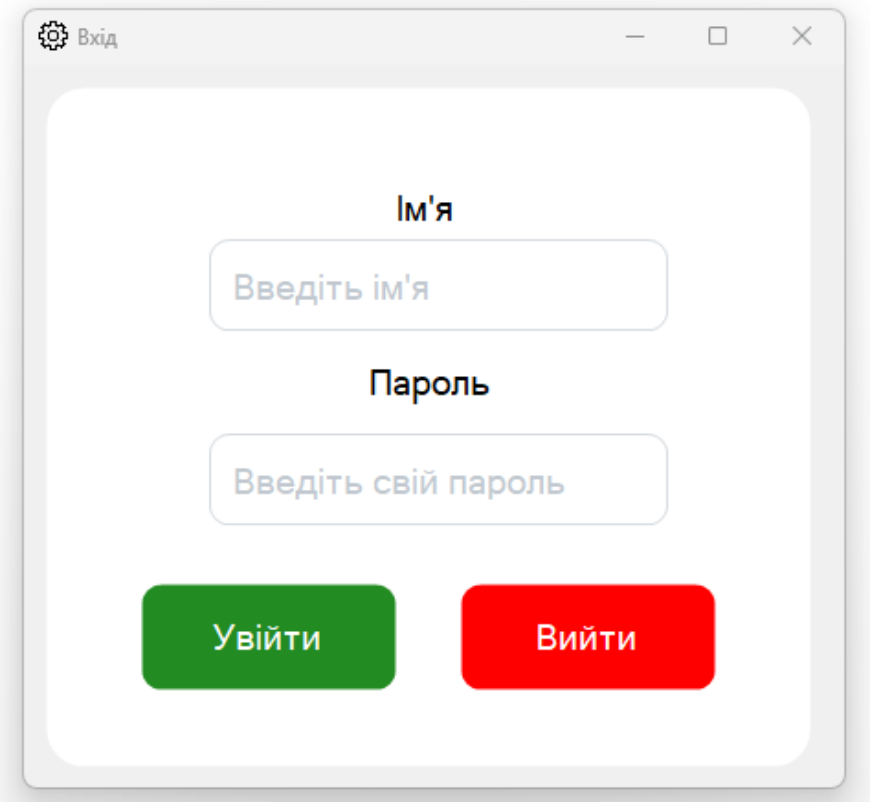


Рисунок 3.5 – Вікно входу до застосунку

Під час входу до застосунку ремонтник вводить ім’я та пароль, а при натисканні кнопки входу застосунок перевіряє, чи існує такий користувач у базі даних, і перенаправляє його до форми заявок згідно з кодом, що наведено на рис. 3.6.

```
// SQL-запит для перевірки користувача в базі
string query = "SELECT id, role FROM users WHERE username = '" + username + "' AND password_hash = '" + password + "'";
DataSet ds = fn.getData(query);

// Перевірка, чи знайдений користувач
if (ds.Tables[0].Rows.Count == 1)
{
    int userId = Convert.ToInt32(ds.Tables[0].Rows[0]["id"]);
    string role = ds.Tables[0].Rows[0]["role"].ToString();

    this.Hide();

    // Визначаємо, яка форма повинна бути відкритою залежно від ролі
    if (role == "Адміністратор")
    {
        Admin_equip adminForm = new Admin_equip(userId);
        adminForm.Show();
    }
    else if (role == "Працівник")
    {
        Worker_workshops workerForm = new Worker_workshops(userId);
        workerForm.Show();
    }
    else if (role == "Ремонтник")
    {
        Repairman_request repairForm = new Repairman_request(userId);
        repairForm.Show();
    }
    else
    {
        MessageBox.Show("Невідома роль користувача.");
        this.Show();
    }
}
else
{
    MessageBox.Show("Невірне ім'я користувача або пароль.");
}
```

Рисунок 3.6 – Програмний код входу до застосунку

Після входу користувача до застосунку автоматично завантажується вікно заявок, яке зображене на рис. 3.7.

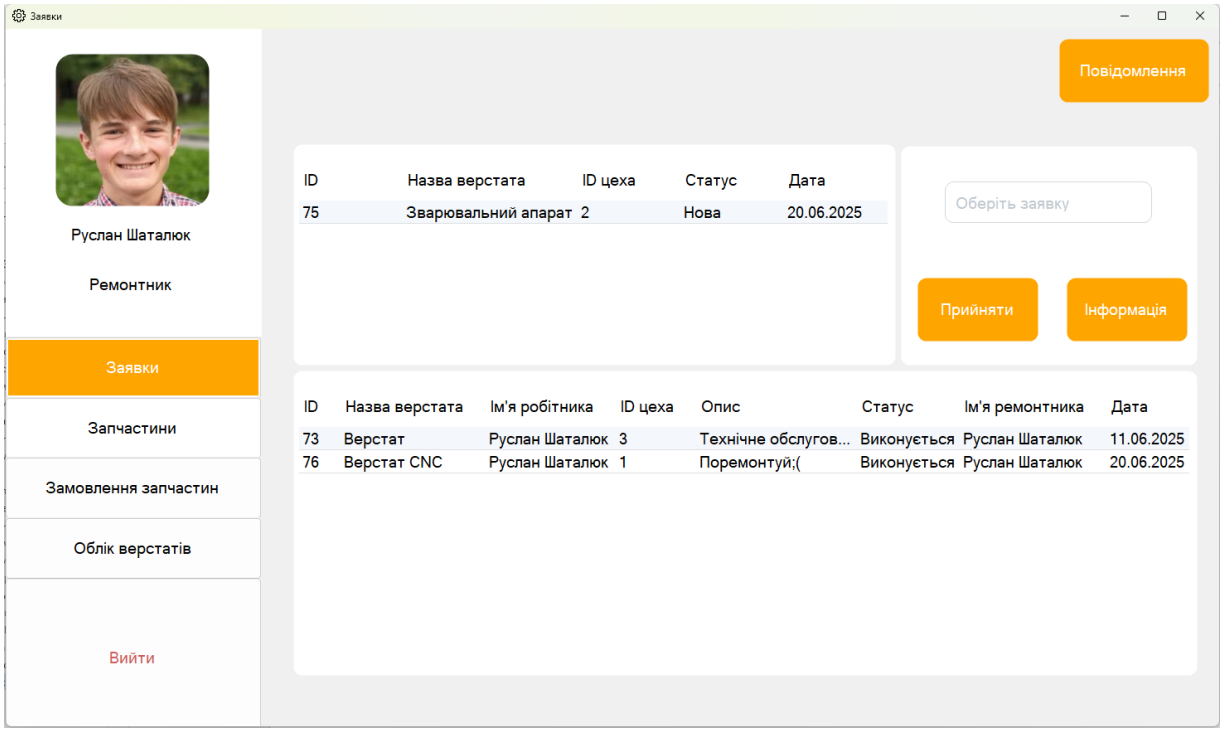


Рисунок 3.7 – Вікно «Заявки»

Перед завантаженням форми та її відображенням користувачу, її необхідно наповнити даними, а саме інформацією про користувача та відомостями про заявки. Програмний код, що виконує ці дії, наведено на рис. 3.8.

```
private void Repairman_request_Load(object sender, EventArgs e)
{
    loaddata();
    // Отримуємо ID з id_admin
    string userId = id_rep.Text.Trim();

    // SQL-запит через fn.getData
    string query = "SELECT username, role, photo FROM users WHERE id = '" + userId + "'";
    DataSet ds = fn.getData(query);

    // Якщо є такий користувач
    if (ds.Tables[0].Rows.Count == 1)
    {
        string name = ds.Tables[0].Rows[0]["username"].ToString();
        string role = ds.Tables[0].Rows[0]["role"].ToString();

        label1.Text = name;
        label2.Text = role;

        // Перевіряємо, чи є фото (photo не NULL)
        if (ds.Tables[0].Rows[0]["photo"] != DBNull.Value)
        {
            byte[] imageBytes = (byte[])ds.Tables[0].Rows[0]["photo"];
            MemoryStream ms = new MemoryStream(imageBytes);
            Picture_Rep.Image = Image.FromStream(ms);

            // Встановлюємо розтягування
            Picture_Rep.SizeMode = PictureBoxSizeMode.StretchImage;
        }
        else
        {
            Picture_Rep.Image = null; // або заглушка
        }
    }
    else
    {
        label1.Text = "Невідомий";
        label2.Text = "Немає ролі";
        Picture_Rep.Image = null;
    }
}

public void loaddata()
{
    query_new = $@"SELECT rr.id AS 'ID', e.name AS 'Назва верстата', rr.workshop_id AS 'ID цеха',
    rr.status AS 'Статус', rr.created_at AS 'Дата' FROM repairrequests rr JOIN equipment e
    ON rr.equipment_id = e.id WHERE rr.status IN ('Нова');";
    DataSet ds = fn.getData(query_new);
    dataGridView1.DataSource = ds.Tables[0];

    query_accept = $@"SELECT rr.id AS 'ID', e.name AS 'Назва верстата', u1.username AS 'Ім''я робітника',
    rr.workshop_id AS 'ID цеха', rr.description AS 'Опис', rr.status AS 'Статус', u2.username AS 'Ім''я ремонтника',
    rr.created_at AS 'Дата' FROM repairrequests rr JOIN equipment e ON rr.equipment_id = e.id LEFT JOIN users u1
    ON rr.worker_id = u1.id LEFT JOIN users u2 ON rr.assigned_repairman_id = u2.id WHERE rr.status = 'Виконується'
    AND rr.assigned_repairman_id = " + id_rep.Text +";";
    DataSet ds_a = fn.getData(query_accept);
    dataGridView2.DataSource = ds_a.Tables[0];

    foreach (DataGridViewColumn column in dataGridView1.Columns)
    {
        column.AutoSizeMode = DataGridViewAutoSizeColumnMode.ColumnHeader;
        dataGridView1.AutoResizeColumn(column.Index);
        int headerWidth = column.Width;

        column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
        column.MinimumWidth = headerWidth;
    }

    // Розтягуємо колонку "Опис" на всю ширину
    if (dataGridView2.Columns.Contains("Опис"))
    {
        dataGridView2.Columns["Опис"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
    }

    // Скорочуємо текст в колонці "Опис" до 20 символів (17 + "...")
    foreach (DataGridViewRow row in dataGridView2.Rows)
    {
        if (!row.IsNewRow)
        {
            var cell = row.Cells["Опис"];
            if (cell?.Value != null)
            {
                string fullText = cell.Value.ToString();
                if (fullText.Length > 20)
                {
                    cell.Value = fullText.Substring(0, 17) + "...";
                }
                cell.ToolTipText = fullText; // показує повний текст при наведенні
            }
        }
    }
}
```

Рисунок 3.8 – Код для наповнення форми даними

Щоб переглянути повідомлення, користувач повинен натиснути кнопку «Повідомлення», яка відкриває додаткове вікно (рис. 3.9) з відображенням усіх наявних повідомлень щодо ремонту або технічного обслуговування. Програмний код, що відповідає за відображення повідомлень, наведено на рис. 3.10.

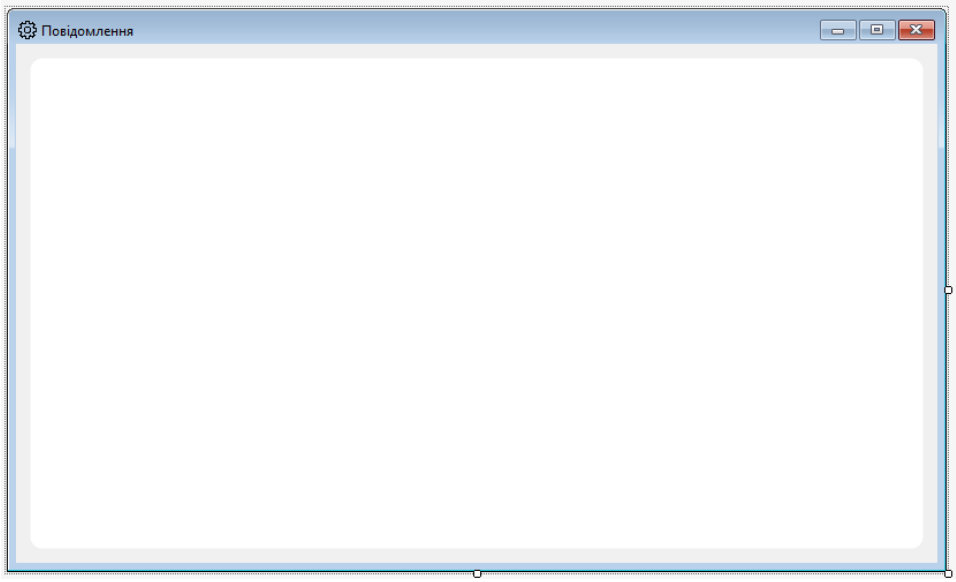


Рисунок 3.9 – Вікно повідомлень

```
Ссылок: 1
private void Notifications_Load(object sender, EventArgs e)
{
    // Очистка DataGridView
    not_grid.Rows.Clear();
    not_grid.Columns.Clear();
    not_grid.DefaultCellStyle.WrapMode = DataGridViewTriState.True;
    not_grid.AllowUserToAddRows = false;
    not_grid.RowHeadersVisible = false;
    not_grid.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

    // Додати колонки
    not_grid.Columns.Add("Icon", ""); // Іконка
    not_grid.Columns.Add("Type", "Тип події");
    not_grid.Columns.Add("Message", "Повідомлення");

    not_grid.Columns[0].Width = 50;
    not_grid.Columns[1].Width = 200;
    not_grid.Columns[2].Width = 500;

    // Завантажити повідомлення
    FilterAndAddNewRequests();
    FilterAndAddMaintenanceRequests();

    // Очистити з бази
    string deleteQuery = "DELETE FROM notifications";
    fn.not(deleteQuery);
}
```

Рисунок 3.10 – Код, що виконується після завантаження вікна

Як видно з коду, у ньому реалізовано дві окремі функції: для завантаження повідомлень про ремонт та технічне обслуговування. Програмну реалізацію однієї з цих функцій наведено на рис. 3.11.

```
Ссылок: 1
private void FilterAndAddMaintenanceRequests()
{
    string query = "SELECT e.name, e.last_maintenance_date, e.workshop_id FROM equipment e " +
                   "WHERE e.last_maintenance_date <= CURDATE()";

    DataSet dsMaintenance = fn.getData(query);

    if (dsMaintenance.Tables[0].Rows.Count > 0)
    {
        foreach (DataRow row in dsMaintenance.Tables[0].Rows)
        {
            string equipmentName = row["name"].ToString();
            string workshopId = row["workshop_id"].ToString();

            string icon = "";
            string title = "Технічне обслуговування";
            string message = $"Верстат ''{equipmentName}'' в цеху №{workshopId} потребує обслуговування.";
            string fullMessage = $"{icon}\n{title}\n{message}";

            // Додати в базу
            string insertQuery = $"INSERT INTO notifications (role, message) VALUES ('Ремонтник', '{fullMessage}')";
            fn.not(insertQuery);

            // Додати в DataGridView
            not_grid.Rows.Add(icon, title, message);
        }
    }
}
```

Рисунок 3.11 – Програмна реалізація функції завантаження повідомлень на ремонт

Окрім кнопки «Повідомлення», на формі, зображеній на рис. 3.7, також розміщена кнопка «Прийняти», яка дає змогу приймати заявки як на «Технічне обслуговування», так і на «Ремонт обладнання». Програмну реалізацію цієї функції наведено на рис. 3.12.

```
Ссылок: 1
private void button2_Click(object sender, EventArgs e)
{
    //Перевірка: чи обрано заявку
    if (string.IsNullOrWhiteSpace(id_req.Text) || id_req.Text == "Оберіть заявку")
    {
        MessageBox.Show("Будь ласка, оберіть заявку зі списку.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    //Перевірка: чи заявка вже Виконується
    string checkStatusQuery = "SELECT status FROM repairrequests WHERE id = " + id_req.Text;
    DataSet dsStatus = fn.getData(checkStatusQuery);

    if (dsStatus.Tables[0].Rows.Count > 0)
    {
        string currentStatus = dsStatus.Tables[0].Rows[0]["status"].ToString();

        if (currentStatus == "Виконується")
        {
            MessageBox.Show("Цей верстат вже перебуває у стані ремонту.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Information);
            return;
        }
    }
    else
    {
        MessageBox.Show("Заявку не знайдено.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    //Оновлення статусу заявки
    query_accept = "UPDATE repairrequests SET assigned_repairman_id = " + id_rep.Text + ", status = 'Виконується' WHERE id = " + id_req.Text;
    fn.setData(query_accept, "Верстат успішно змінено");
    loaddata();
}
```

Рисунок 3.12 – Програмна реалізація прийняття заявок

На формі, зображеній на рис. 3.7, також розташована кнопка «Інформація», при натисканні на яку відкривається вікно з детальною інформацією про заявку. Це вікно показано на рис. 3.13.

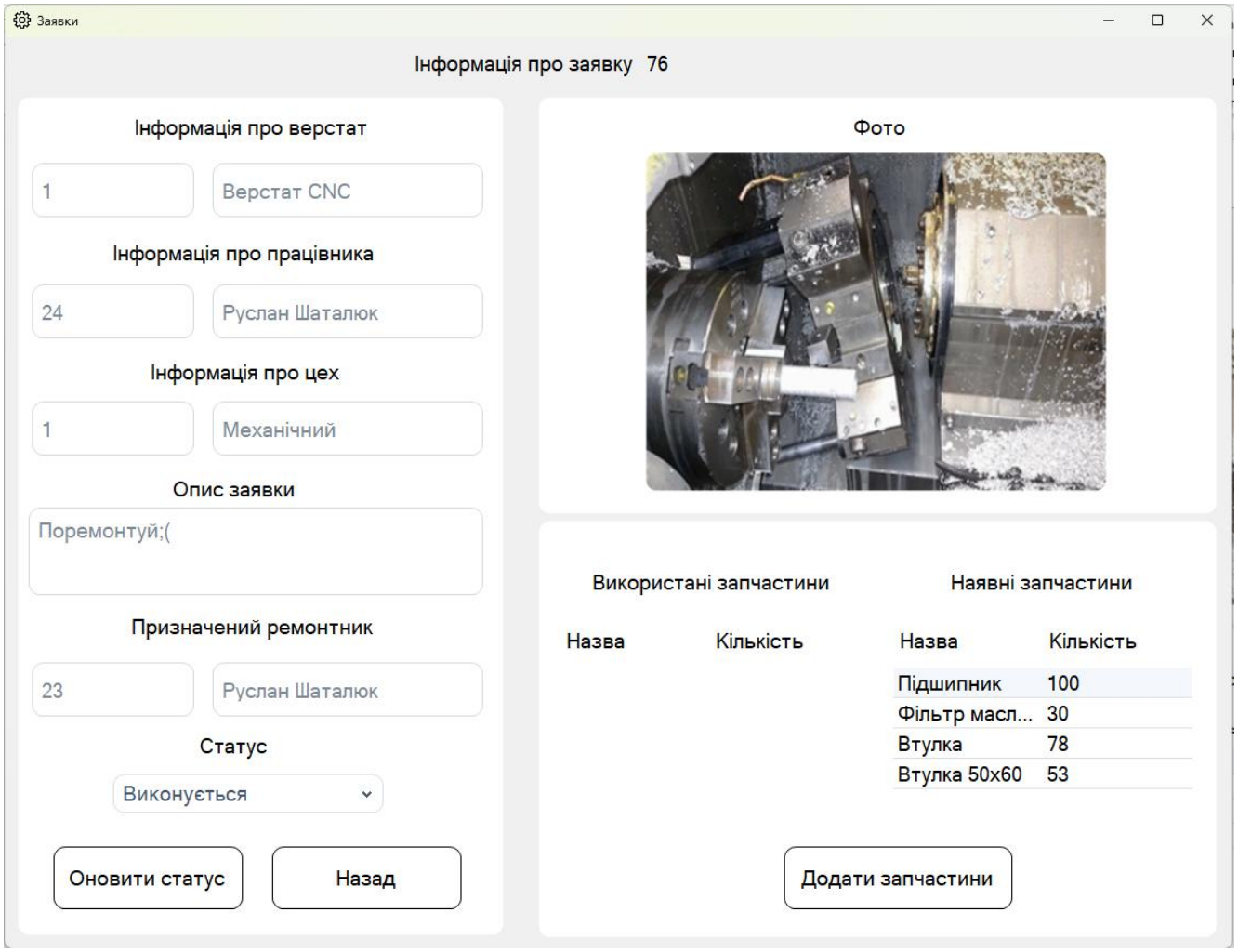


Рисунок 3.13 – Вікно детальної інформації про заявку

Щоб відкрити це вікно, користувачу потрібно вибрати заявку з таблиці заявок і натиснути кнопку «Інформація». У вікні з деталями заявки, якщо її статус «Виконується», ремонтники можуть за потреби додати використані при роботі компоненти. Для цього ремонтник вибирає необхідні компоненти з таблиці «Наявні компоненти» та вказує їх кількість у таблиці «Використані компоненти». Збереження компонентів за заявкою здійснюється за допомогою кнопки «Додати компоненти». Програмна реалізація цієї функції наведена на рис. 3.14.

```
// Розрахунок змін
Dictionary<string, int> partsToDecrease = new Dictionary<string, int>();
Dictionary<string, int> partsToIncrease = new Dictionary<string, int>();

foreach (var item in newParts)
{
    int oldQuantity = oldParts.ContainsKey(item.Key) ? oldParts[item.Key] : 0;
    int difference = item.Value - oldQuantity;

    if (difference > 0)
    {
        partsToDecrease[item.Key] = difference;
    }
    else if (difference < 0)
    {
        partsToIncrease[item.Key] = Math.Abs(difference);
    }
}

// Зменшення кількості на складі з перевіркою мінімального рівня
foreach (var item in partsToDecrease)
{
    string checkQuery = $"SELECT quantity, min_stock_level FROM SpareParts WHERE name = '{item.Key}'";
    DataSet checkDs = fn.getData(checkQuery);

    if (checkDs.Tables[0].Rows.Count > 0)
    {
        int currentQty = Convert.ToInt32(checkDs.Tables[0].Rows[0]["quantity"]);
        int minQty = Convert.ToInt32(checkDs.Tables[0].Rows[0]["min_stock_level"]);
        int newQty = currentQty - item.Value;

        if (newQty < 0)
        {
            MessageBox.Show($"Недостатньо {item.Key} на складі!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }
        else if (newQty < minQty)
        {
            MessageBox.Show($"Мінімальна залишкова кількість {item.Key} має бути {minQty}, після зменшення буде лише {newQty}. " +
                $"Будь ласка змініть кількість!", "Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        string updateQtyQuery = $"UPDATE SpareParts SET quantity = {newQty} WHERE name = '{item.Key}'";
        fn.setData(updateQtyQuery, "Кількість оновлено");
    }
}

// Повернення частин на склад
foreach (var item in partsToIncrease)
{
    string checkQuery = $"SELECT quantity FROM SpareParts WHERE name = '{item.Key}'";
    DataSet checkDs = fn.getData(checkQuery);

    if (checkDs.Tables[0].Rows.Count > 0)
    {
        int currentQty = Convert.ToInt32(checkDs.Tables[0].Rows[0]["quantity"]);
        int newQty = currentQty + item.Value;

        string updateQtyQuery = $"UPDATE SpareParts SET quantity = {newQty} WHERE name = '{item.Key}'";
        fn.setData(updateQtyQuery, "Кількість оновлено");
    }
}

// Оновлюємо поле parts у заявці
string partsString = string.Join(", ", newParts.Select(x => $"{x.Key} {x.Value}"));
string updateRequestQuery = $"UPDATE RepairRequests SET parts = '{partsString}' WHERE id = '{requestId}'";
fn.setData(updateRequestQuery, "Заявку оновлено успішно!");
```

Рисунок 3.14 – Програмна реалізація додавання компонентів

У вікні інформації також передбачена можливість змінити статус заявки. Для цього ремонтнику слід обрати потрібний статус із випадаючого меню та натиснути кнопку «Оновити статус». Програмна реалізація цієї операції наведена на рис. 3.15.

```
// Оновлюємо статус заявки
string query = "UPDATE repairrequests SET status = '" + status_equip.Text + "' WHERE id = " + id_request.Text;
fn.setData(query, "Заявку успішно змінено");

string query2 = "UPDATE equipment SET status =  'Працює' WHERE id = '" + id_equip.Text + "';";
fn.getData(query2);

// Якщо опис заявки містить "технічне обслуговування" та статус = "Завершена" — оновити дату ТО
if (describe.Text.Trim().ToLower() == "технічне обслуговування" && status_equip.Text == "Завершена")
{
    // Зміщення last_maintenance_date на 7 днів вперед
    string updateMaintenanceDateQuery = @"
    UPDATE equipment
    SET last_maintenance_date = CURDATE() + INTERVAL 7 DAY
    WHERE id = '" + id_equip.Text + "';";

    fn.getData(updateMaintenanceDateQuery);
}

// Якщо статус "Завершена" або "Відхилена", закриваємо поточну вкладку та відкриваємо іншу
if (status_equip.Text == "Завершена" || status_equip.Text == "Відхилена")
{
    this.Hide();
    Repairman_request regiseform = new Repairman_request(Convert.ToInt32(id_rep.Text));
    regiseform.Show();
}
```

Рисунок 3.15 – Програмна реалізація зміни статусу заявки

Щоб повернутися до вікна заявок, ремонтник повинен натиснути кнопку «Назад». За допомогою навігаційного меню, розташованого зліва у програмному застосунку, ремонтник може перейти до вікна «Запчастини», натиснувши відповідну кнопку. Вікно «Запчастини» наведено на рис. 3.16.

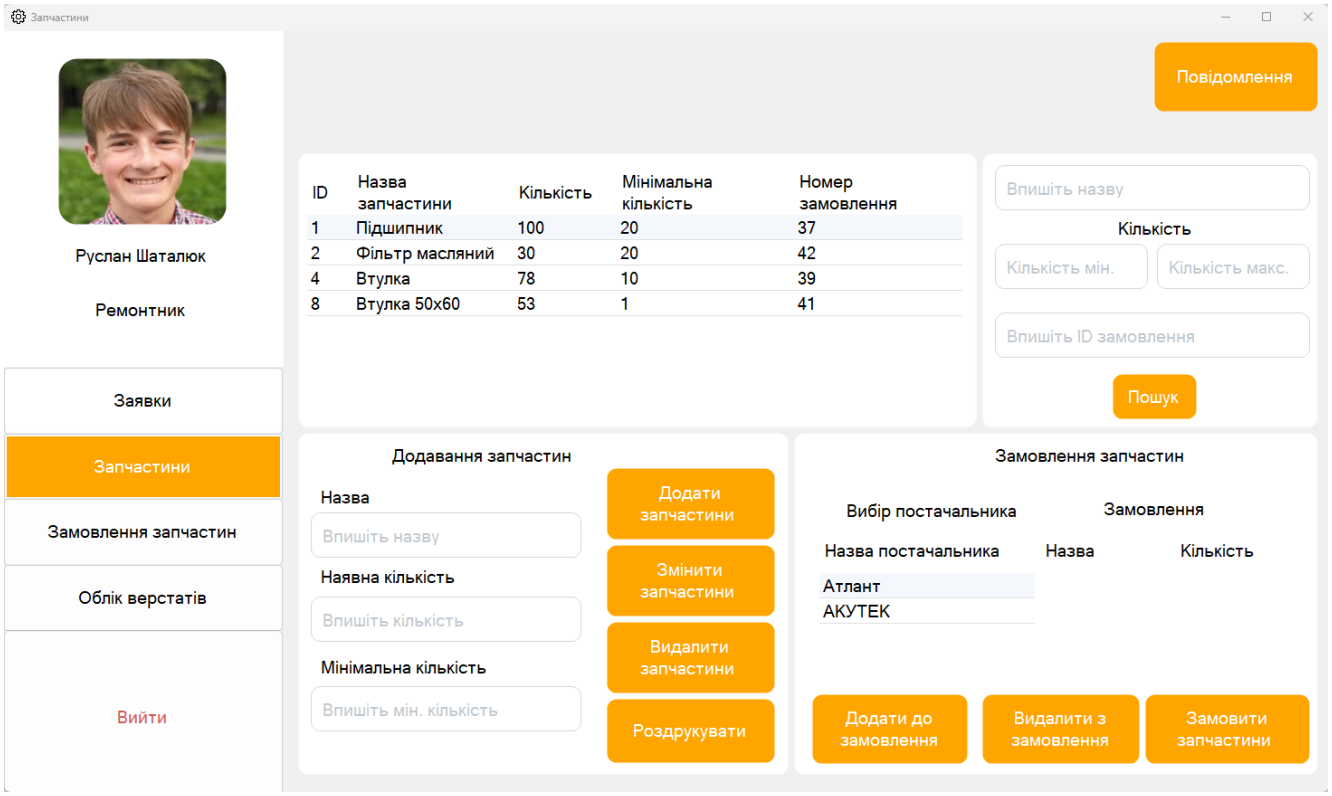


Рисунок 3.16 – Вікно «Запчастини»

У вікні «Запчастини» ремонтник має можливість додавати, змінювати або видаляти запчастини. Також тут можна виконувати пошук по таблиці з запчастинами та створювати нові замовлення. Для додавання запчастин ремонтнику потрібно заповнити відповідні поля у вікні та натиснути кнопку «Додати запчастини». Програмну реалізацію цієї операції наведено на рис. 3.17.

```
Ссылок: 1
private void add_1_Click(object sender, EventArgs e)
{
    // Перевірка на порожнє ім'я
    if (string.IsNullOrWhiteSpace(id_equip.Text))
    {
        MessageBox.Show("Назва деталі не може бути порожньою", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Перевірка на спеціальні символи в назві
    if (!System.Text.RegularExpressions.Regex.IsMatch(id_equip.Text, @"^[a-zA-Za-яА-ЯіІїЇєЄ0-9\s\-_]+$"))
    {
        MessageBox.Show("Назва деталі містить недопустимі символи. Дозволені лише літери, цифри, пробіли, тире та підкреслення.",
                        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Перевірка на кількість (quantity)
    if (!int.TryParse(ID_workshop.Text, out int quantity) || quantity < 0)
    {
        MessageBox.Show("Кількість повинна бути числом 0 або більше", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Перевірка на мінімальний рівень (min_stock_level)
    if (!int.TryParse(richTextBox4.Text, out int minStockLevel) || minStockLevel < 0)
    {
        MessageBox.Show("Мінімальний рівень повинен бути числом 0 або більше", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Перевірка, чи така деталь вже існує
    string checkQuery = "SELECT COUNT(*) FROM spareparts WHERE name = '" + id_equip.Text + "'";
    DataSet dsCheck = fn.getData(checkQuery);
    int count = Convert.ToInt32(dsCheck.Tables[0].Rows[0][0]);

    if (count > 0)
    {
        MessageBox.Show("Деталь з такою назвою вже існує у базі даних.", "Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Додаємо запис
    string query = "INSERT INTO spareparts (name, quantity, min_stock_level) " +
                   "VALUES ('" + id_equip.Text + "', '" + quantity + "', '" + minStockLevel + "')";
    fn.setData(query, "Деталь успішно додана");

    // Оновлення таблиці
    loaddata_parts();
}
```

Рисунок 3.17 – Програмна реалізація додавання запчастини

Ремонтник також може змінювати або видаляти запчастини. Для редагування необхідно вибрати потрібну запчастину з таблиці, внести зміни у відповідні поля (які використовувалися при додаванні) та натиснути кнопку «Змінити запчастину». Програмна реалізація цієї функції наведена на рис. 3.18.

```
private void button4_Click(object sender, EventArgs e)
{
    // Перевірка на спеціальні символи в назві
    if (!System.Text.RegularExpressions.Regex.IsMatch(id_equip.Text, @"^[a-zA-Za-яА-ЯіІїЇєЄ0-9\s\-_]+$"))
    {
        MessageBox.Show("Назва деталі містить недопустимі символи. Дозволені лише літери, цифри, пробіли, тире та підкреслення.",
                        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Перевірка кількості
    if (!int.TryParse(ID_workshop.Text, out int quantity) || quantity < 0)
    {
        MessageBox.Show("Кількість повинна бути цілим числом 0 або більше.",
                        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Перевірка мінімального запасу
    if (!int.TryParse(richTextBox4.Text, out int minStock) || minStock < 0)
    {
        MessageBox.Show("Мінімальний залишок повинен бути цілим числом 0 або більше.",
                        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Перевірка ID запчастини
    if (!int.TryParse(richTextBox5.Text, out int partId))
    {
        MessageBox.Show("Невірний ID запчастини.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Формуємо запит
    string query = "UPDATE spareparts SET name = '" + id_equip.Text + "', quantity = '" + quantity + "', " +
                   "min_stock_level = '" + minStock + "' WHERE id = " + partId;

    // Виконання
    fn.setData(query, "Запчастину успішно змінено");
    loaddata_parts();
    this.Hide();
    Repairman_parts regiseform = new Repairman_parts(Convert.ToInt32(id_rep.Text));
    regiseform.Show();

}
```

Рисунок 3.18 – Програмна реалізація зміни запчастини

Для видалення запчастини ремонтнику достатньо вибрати необхідну позицію у списку та натиснути кнопку «Видалити запчастину». Процес видалення здійснюється за допомогою коду, показаного на рис. 3.19.

```
Ссылок: 1
private void button6_Click(object sender, EventArgs e)
{
    // Перевірка ID запчастини
    if (!int.TryParse(richTextBox5.Text, out int partId))
    {
        MessageBox.Show("Будь ласка, виберіть запчастину для видалення.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    string query = "DELETE FROM spareparts WHERE id = " + richTextBox5.Text + ";";
    fn.setData(query, "Запчастину успішно видалено");
    loaddata_parts();
}
```

Рисунок 3.19 – Програмна реалізація видалення запчастини

Як вже було зазначено, ремонтнику доступна можливість створення замовлення на запчастини у вкладці «Запчастини». Для цього він повинен вибрати потрібні запчастини з таблиці та натиснути кнопку «Додати до замовлення». Потім, змінивши кількість і обравши відповідного постачальника, потрібно натиснути кнопку «Замовити запчастини». Якщо до списку потрапила зайва запчастина, її можна видалити, натиснувши кнопку «Видалити з замовлення». Після натискання кнопки «Замовити запчастини» програма автоматично надсилає менеджеру постачальника лист із переліком необхідних запчастин. Код, що реалізує цей процес, наведено на рис. 3.20.

```
// Тепер — створення замовлень
foreach (DataGridViewRow row in Req.Rows)
{
    if (row.IsNewRow) continue;

    string partName = row.Cells[0].Value?.ToString();
    string quantityText = row.Cells[1].Value?.ToString();

    if (string.IsNullOrWhiteSpace(partName)) continue;

    int quantity = int.Parse(quantityText);

    // Отримуємо ID запчастини
    query = $"SELECT id FROM spareparts WHERE name = '{partName}'";
    DataSet dsPart = fn.getData(query);

    if (dsPart.Tables[0].Rows.Count == 0)
    {
        MessageBox.Show($"Запчастину '{partName}' не знайдено в базі.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        continue;
    }

    int sparepart_id = Convert.ToInt32(dsPart.Tables[0].Rows[0]["id"]);

    // Перевірка на дублювання замовлення
    query = $"SELECT id FROM orders WHERE sparepart_id = {sparepart_id} AND status = 'Очікується'";
    DataSet dsCheck = fn.getData(query);

    if (dsCheck.Tables[0].Rows.Count > 0)
    {
        MessageBox.Show($"Запчастина '{partName}' вже замовлена і очікується.", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);
        continue;
    }

    // Вставка нового замовлення
    query = "INSERT INTO orders (sparepart_id, quantity, order_date, supplier_id, ordered_by, status) " +
            $"VALUES ('{sparepart_id}', '{quantity}', '{order_date}', '{supplier_id}', {id_rep.Text}, 'Очікується')";
    fn.setData(query, $"Запчастина '{partName}' успішно замовлена");

    // Отримання ID нового замовлення
    query = "SELECT LAST_INSERT_ID()";
    DataSet dsNewOrder = fn.getData(query);
    int order_id = Convert.ToInt32(dsNewOrder.Tables[0].Rows[0][0]);

    // Оновлення поля auto_order
    query = $"UPDATE spareparts SET auto_order = {order_id} WHERE id = {sparepart_id}";
    fn.setData(query, $"Номер замовлення оновлено для '{partName}'");

    successfulOrders++;
}
```

Рисунок 3.20 – Реалізація замовлення запчастин

У вікні «Запчастини» (рис. 3.16) також передбачена можливість пошуку по таблиці запчастин. Для цього необхідно заповнити відповідні поля пошуку та натиснути кнопку «Пошук». Програмну реалізацію цієї функції наведено на рис. 3.21.

```
private void guna2Button1_Click(object sender, EventArgs e)
{
    string partName = guna2TextBox1.Text.Trim();       // Назва запчастини
    string minQuantity = guna2TextBox2.Text.Trim();    // Мінімальна кількість
    string maxQuantity = guna2TextBox3.Text.Trim();    // Максимальна кількість
    string orderNumber = guna2TextBox4.Text.Trim();    // Номер замовлення

    List<string> conditions = new List<string>();

    // Додавання умов пошуку, якщо параметри не порожні
    if (!string.IsNullOrEmpty(partName))
        conditions.Add($"name LIKE '%{partName}%'");

    int minQty = 0;
    if (!string.IsNullOrEmpty(minQuantity) && int.TryParse(minQuantity, out minQty))
    {
        conditions.Add($"quantity >= {minQty}");
    }
    else if (!string.IsNullOrEmpty(minQuantity))
    {
        MessageBox.Show("Мінімальна кількість має бути числом!");
        return;
    }

    int maxQty = 0;
    if (!string.IsNullOrEmpty(maxQuantity) && int.TryParse(maxQuantity, out maxQty))
    {
        // Перевірка, що максимальна кількість не менша мінімальної
        if (maxQty < minQty)
        {
            MessageBox.Show("Максимальна кількість не може бути меншою за мінімальну!");
            return;
        }
        conditions.Add($"quantity <= {maxQty}");
    }
    else if (!string.IsNullOrEmpty(maxQuantity))
    {
        MessageBox.Show("Максимальна кількість має бути числом!");
        return;
    }

    if (!string.IsNullOrEmpty(orderNumber))
        conditions.Add($"auto_order LIKE '%{orderNumber}%'");

    string query = @"SELECT id AS 'ID',
                name AS 'Назва запчастини',
                quantity AS 'Кількість',
                min_stock_level AS 'Мінімальна кількість',
                auto_order AS 'Номер замовлення'
        FROM spareparts";

    // Якщо є умови пошуку, додаємо їх до запиту
    if (conditions.Count > 0)
    {
        query += " WHERE " + string.Join(" AND ", conditions);
    }

    // Виконання запиту та заповнення DataGridView
    DataSet ds = fn.getData(query);
    dataGridView1.DataSource = ds.Tables[0];
}
```

Рисунок 3.21 – Реалізація пошуку по таблиці запчастин

Ремонтник має змогу переглянути свої замовлення перейшовши до вікна «Замовлення запчастин» у навігаційному меню. Вікно «Замовлення запчастин» можна побачити на рис. 3.22.

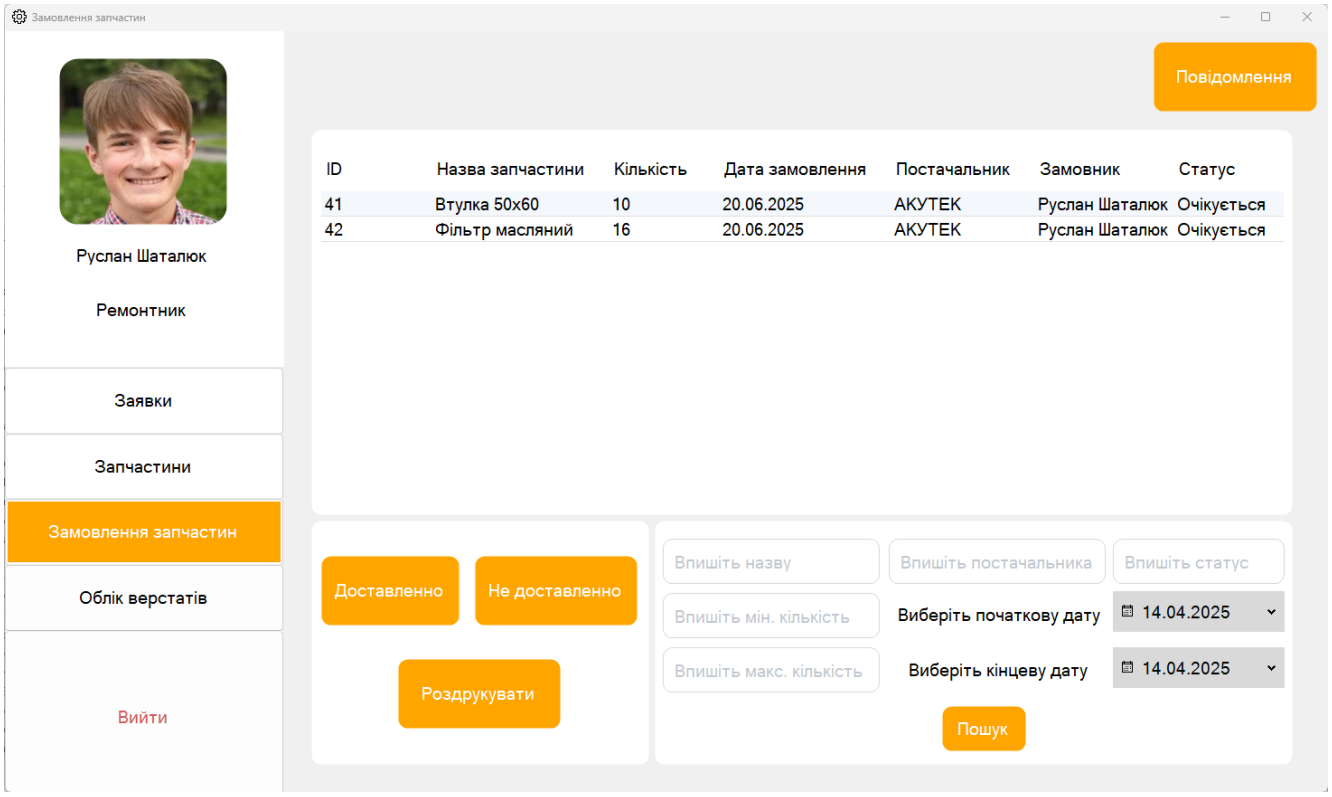


Рисунок 3.22 – Вікно «Замовлення запчастин»

У вікні «Замовлення запчастин» ремонтник може відмітити, чи було замовлення доставлено, або вказати, що воно не доставлене, наприклад, якщо у постачальника відсутні потрібні запчастини. Код для реалізації додавання запчастин наведено на рис. 3.23.

Крім того, ремонтник має можливість здійснювати пошук по таблиці замовлень за різними критеріями: за назвою запчастини, за кількістю запчастин від мінімальної до максимальної, за назвою постачальника, за статусом замовлення та за проміжком часу створення замовлення. Для цього потрібно заповнити відповідні поля пошуку і натиснути кнопку «Пошук». Програмна реалізація пошуку виконується за схожим алгоритмом, який показано на рис. 3.21.

```
// Показуємо вікно підтвердження
DialogResult result = MessageBox.Show("Чи точно доставлено запчастину?", "Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)
{
    // Отримуємо інформацію про запчастину і кількість з таблиці orders
    string query = "SELECT sparepart_id, quantity FROM orders WHERE id = " + orderId;
    DataSet ds = fn.getData(query);

    if (ds.Tables[0].Rows.Count > 0)
    {
        int sparepartId = Convert.ToInt32(ds.Tables[0].Rows[0]["sparepart_id"]);
        int orderedQuantity = Convert.ToInt32(ds.Tables[0].Rows[0]["quantity"]);

        // Отримуємо поточну кількість запчастини
        string quantityQuery = "SELECT quantity FROM spareparts WHERE id = " + sparepartId;
        DataSet quantityDs = fn.getData(quantityQuery);

        if (quantityDs.Tables[0].Rows.Count > 0)
        {
            int currentQuantity = Convert.ToInt32(quantityDs.Tables[0].Rows[0]["quantity"]);
            int newQuantity = currentQuantity + orderedQuantity;

            // Оновлюємо кількість запчастин
            string updateQuantityQuery = "UPDATE spareparts SET quantity = " + newQuantity + " WHERE id = " + sparepartId;
            fn.setData(updateQuantityQuery, "Кількість запчастин успішно оновлена");

            // Оновлюємо статус замовлення на "Доставлено"
            string updateStatusQuery = "UPDATE orders SET status = 'Доставлено' WHERE id = " + orderId;
            fn.setData(updateStatusQuery, "Статус замовлення оновлений");

            loaddata();
        }
        else
        {
            MessageBox.Show("Не знайдено запчастину для цього замовлення.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    else
    {
        MessageBox.Show("Не знайдено дані для цього замовлення.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Рисунок 3.23 – Код реалізації додавання запчастин після успішної доставки

Останнім вікном, доступним для актора ремонтника, є вікно «Облік верстатів», яке наведено на рис. 3.24.

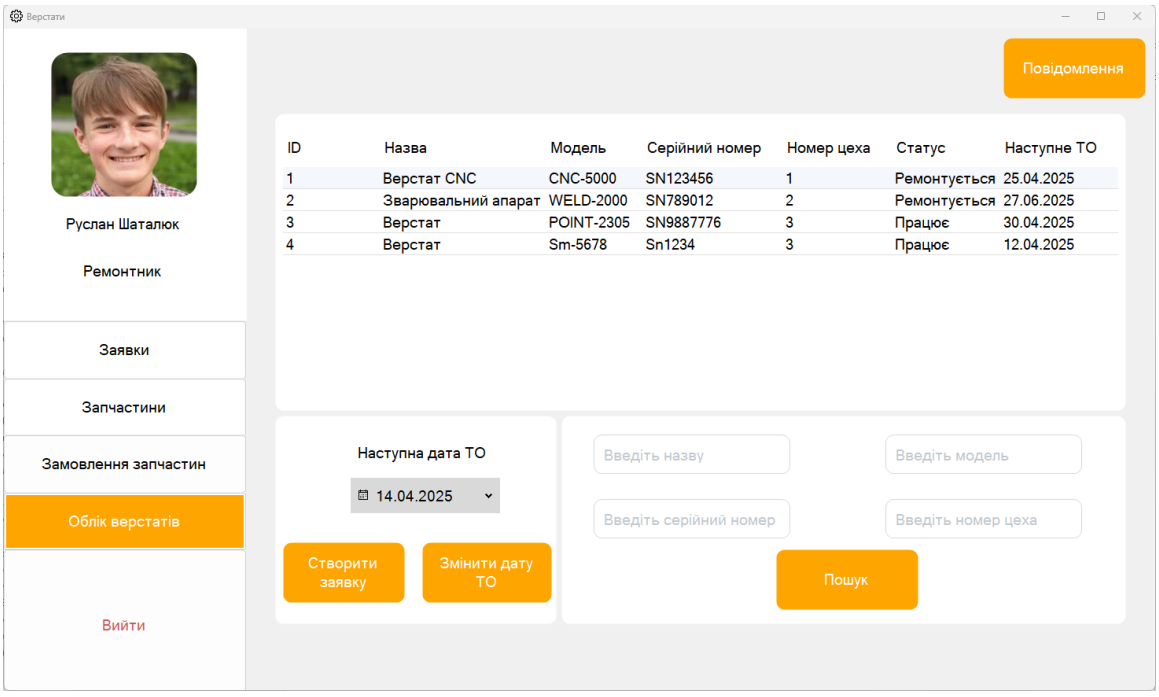


Рисунок 3.24 – Вікно «Облік верстатів»

У цьому вікні ремонтник може перевірити дату останнього технічного обслуговування, а за потреби продовжити її або створити заявку на технічне обслуговування. Програмний код для створення такої заявки наведено на рис. 3.25.

```
// Перевірка на наявність активної заявки (нової або ремонту)
string checkQuery = $@"
SELECT COUNT(*) FROM repairrequests
WHERE equipment_id = {id_req.Text}
  AND status IN ('Нова', 'Виконується')";
DataSet checkData = fn.getData(checkQuery);
if (Convert.ToInt32(checkData.Tables[0].Rows[0][0]) > 0)
{
    MessageBox.Show("Заявка для цього обладнання вже існує та ще не завершена.",
                    "Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return;
}

// Створення порожнього зображення (нейтрального значення)
byte[] imageData = new byte[0];

// Вставка заявки на технічне обслуговування
string insertQuery = $@"
INSERT INTO repairrequests
(equipment_id, worker_id, workshop_id, description, photo_url, status, assigned_repairman_id, created_at)
VALUES
({id_req.Text}, {id_rep.Text}, {id_work.Text}, 'Технічне обслуговування', @imagedata, 'Нова', 1, '{formattedDate}')";
fn.setDataWithImage(insertQuery, imageData);
MessageBox.Show("Заявка на технічне обслуговування успішно додана.", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);
// Оновлення статусу обладнання
string updateEquipment = $"UPDATE equipment SET status = 'Ремонтується' WHERE id = {id_req.Text}";
fn.getData(updateEquipment);
```

Рисунок 3.25 – Код для створення заявки на технічне обслуговування

Пошук у вкладці реалізовано за аналогією з іншими вікнами застосунку. Він базується на однаковому принципі: користувач вводить необхідні параметри або критерії у відповідні поля пошуку, після чого натискає кнопку «Пошук». Система обробляє запит, виконує фільтрацію даних у таблиці за заданими параметрами та відображає результати.

Слід підкреслити, що робота програмного застосунку була б неможливою без належного зв’язку з базою даних та функцій для відправлення запитів. Усі функції, що відповідають за надсилання запитів до бази, були реалізовані в окремому класі, код якого наведено у додатку В.

3.4 Охорона праці

У процесі проєктування та впровадження системи автоматизації технічного обслуговування обладнання приладобудівного підприємства особлива увага

приділяється питанням охорони праці. Це зумовлено як підвищеними ризиками травматизму, пов’язаними з обслуговуванням механізмів і електрообладнання, так і вимогами чинного законодавства України у сфері безпеки праці. Відповідно до Закону України «Про охорону праці» № 2694-XII від 14.10.1992 року (в редакції від 2024 року), роботодавець зобов’язаний забезпечити створення на кожному робочому місці таких умов праці, які відповідають вимогам безпеки, гігієни праці та виробничого середовища.

У галузі технічного обслуговування систем автоматизації на підприємстві працівник може наражатися на вплив цілого спектра небезпечних і шкідливих виробничих чинників. До таких належать: ураження електричним струмом, травмування через відкриті або рухомі частини обладнання, перевантаження опорно-рухового апарату та надмірний шум [22]. Класифікацію основних ризиків, їх джерела та рекомендовані методи захисту можна побачити на табл. 3.9.

Таблиця 3.9 – Основні небезпечні та шкідливі фактори при технічному обслуговуванні автоматизованих систем

№	Фактор ризику	Джерело	Заходи безпеки
1	Ураження електричним струмом	Електрощити, кабелі, роз'єми	Захисне заземлення, ПЗВ, інструктаж
2	Механічне травмування	Відкриті рухомі елементи	Екранування, автоматичне блокування
3	Надмірне фізичне навантаження	Ручне перенесення інструментів	Використання візків, ергономіка
4	Пожежонебезпечна ситуація	Замикання, перегрів	Сигналізація, вогнегасники, вентиляція
5	Вплив шуму	Промислові приводи	Використання ЗІЗ слуху, звукоізоляція

Згідно з Правилами охорони праці під час експлуатації електроустановок споживачів (НПАОП 40.1-1.21-98), при виконанні робіт з устаткуванням, що перебуває під напругою, допускаються лише особи, які мають відповідну групу з електробезпеки (не нижче II для обслуговування до 1000 В). Заборонено працювати з електрообладнанням без засобів індивідуального захисту (ЗІЗ), серед яких: діелектричні рукавиці, окуляри, ізолювальні інструменти, спецвзуття та головні убори. Найчастіше використовувані засоби захисту систематизовано на табл. 3.10.

Таблиця 3.10 – Засоби індивідуального захисту для працівників, що обслуговують обладнання

Засіб індивідуального захисту	Призначення
Діелектричні рукавиці	Захист від ураження електричним струмом
Захисні окуляри	Захист очей при роботі з пилом чи рідинами
Каска	Захист голови від механічних ушкоджень
Спецвзуття з антиковзною підошвою	Захист ніг, стійкість на слизьких поверхнях
Навушники	Захист від шуму понад 85 дБ

Крім того, велике значення має правильна організація робочого місця. Відповідно до ДСТУ EN ISO 6385:2005 та Санітарних норм ДСН 3.3.6.042-99, робоче місце має бути освітленим не менше ніж на рівні 300 лк, температура повітря має утримуватись у межах 18–25°C, а площа робочої зони на одного працівника має становити не менше 4,5 м². Усі відеотермінали мають розташовуватись на відстані не ближче ніж 50 см від очей користувача. Для зменшення втомлюваності персоналу передбачено чергування роботи за комп’ютером з активними паузами.

Особливе значення має пожежна безпека, що регламентується Правилами пожежної безпеки в Україні (НАПБ А.01.001-2014). У приміщеннях, де

здійснюється обслуговування обладнання або функціонування систем автоматизації, мають бути вогнегасники не рідше ніж один на кожні 20 м² площі, евакуаційні виходи мають бути вільними, а також має бути візуально доступний план евакуації персоналу.

Усі працівники, які мають доступ до технічного обладнання, повинні пройти навчання і періодичні інструктажі з охорони праці відповідно до Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці (наказ Мінсоцполітики №15 від 26.01.2005). У вступний інструктаж входить ознайомлення з виробничими ризиками, протипожежними заходами та алгоритмом дій у разі аварії [22]. Повторні інструктажі проводяться кожні 6 місяців, а позапланові – у разі зміни умов праці, запуску нового обладнання або нещасного випадку.

Варто також зазначити, що згідно з Законом України «Про охорону навколишнього природного середовища» №1264-XII, діяльність у сфері технічного обслуговування має передбачати мінімізацію впливу на довкілля. Система автоматизації, що розробляється, дозволяє зменшити втручання людини в технологічні процеси, знижує споживання енергії та сприяє зменшенню техногенного навантаження.

Підсумовуючи, можна сказати, що впровадження сучасної системи автоматизації на приладобудівному підприємстві, з урахуванням усіх вимог охорони праці, дозволяє не лише оптимізувати виробничі процеси, але й значно підвищити безпеку персоналу. Всі передбачені заходи відповідають чинному законодавству України та спрямовані на збереження життя і здоров’я працівників.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи створили повноцінний настільний застосунок, який дійсно спрощує роботу з технічним обслуговуванням обладнання. У програмі реалізовано окремий функціонал для трьох ролей – адміністратора, працівника і ремонтника. Кожен бачить лише те, що йому потрібно, тому інтерфейс не перевантажений і легко зрозумілий. Дані зберігаються в надійній базі, що дозволяє ефективно керувати заявками, верстатами та процесом ремонту без плутанини й втрат.

У першому розділі роботи ми проаналізували, як виглядають схожі системи, і чому саме ми обрали C# разом з Visual Studio, .NET Framework і Windows Forms. Для зберігання даних використали MySQL, а структуру бази зручно спроєктували через MySQL Workbench, де одразу побудували EER-діаграму для 3 розділу.

Другий розділ був більше про технічну частину тут проводився аналіз автоматичної системи управління. Розрахували Tау, побудувавши передавальну функцію. Щоб переконатися, що система стабільна, використали кілька методів визначення стійності: Михайлова, Найквіста, Гурвіца та кореневий. Для повноти картини також побудували амплітудно-частотну характеристику, яка допомогла краще зрозуміти поведінку системи.

У третьому розділі розписали, хто й що може робити в системі. Адміністратор керує працівниками, обладнанням, постачальниками й заявками. Працівник створює заявки, додає до них фото, переглядає статус. Ремонтник найбільш завантажений, бо саме він працює із заявками, додає запчастини, оформлює замовлення і контролює весь процес ремонту. Для кожної ролі створили Use Case-діаграми, які добре показують, як користувачі взаємодіють із системою.

Також детально розробили структуру бази даних описавши всі таблиці (users, repairrequests, equipment, spareparts тощо), прописали поля, типи даних, зв’язки. Завдяки цьому база вийшла логічною та зручною в роботі.

Особливу увагу приділили роботі ремонтника, бо це найбільш “живий” і насичений функціоналом користувач. Його частина системи дозволяє бачити заявки, змінювати їх статус, додавати запчастини, формувати замовлення тощо. Цей модуль вийшов чи не найважливішим у всьому застосунку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП «УкрНДНЦ». 2016. 30 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд.: І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2022. – 55 с.

3. Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2025) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2025. – Вип. 1. – 262с.

4. Невлюдов І. Ш. Виробничі процеси та обладнання об'єктів автоматизації: Підручник для студентів вищих навчальних закладів / І. Ш. Невлюдов та інш. Кривий Ріг: Криворізький коледж НАУ, 2017 р. – 444 с.

5. ДСТУ 2861-94 Надійність техніки. Аналіз надійності. Основні положення/ РОЗРОБНИКИ: В.Л. Стрельніков, д-р техн. наук (керівник розробки), І.З. Аронов, канд. техн. наук, В.П Коньков, канд. техн. наук, О.В. Федухін, канд. техн. наук.

6. Total Productive Maintenance [Електронний ресурс] : Режим доступу: https://www.it.ua/knowledge-base/technology-innovation/total-productive-maintenance-tpm. (дата звернення: 30.04.2025).

7. What is predictive maintenance (PdM)? [Електронний ресурс] : Режим доступу: https://fiixsoftware.com/maintenance-strategies/predictive-maintenance/#:~:text=Predictive%20maintenance%20(PdM)%20uses%20data,opens%20in%20new%20tab)%20costs. (дата звернення: 03.04.2025).

8. Fiix | #1 CMMS Software, AI-Powered Work Orders & Much More [Електронний ресурс] : Режим доступу: https://fiixsoftware.com/cmms/cmms-

software/. (дата звернення: 05.05.2025).

9. UpKeep Work Order Maintenance [Електронний ресурс] : Режим доступу: https://apps.apple.com/us/app/upkeep-work-order-maintenance/id921799415. (дата звернення: 08.05.2025).

10. The complete guide to IBM Maximo - Sedin [Електронний ресурс] : Режим доступу: https://sedintechnologies.com/blogs/what-is-maximo/#:~:text=IBM%20Maximo%20is%20a%20cloud,create%20new%20work%20order%20hierarchies. (дата звернення: 17.05.2025).

11. What is IBM Maximo Software? Definition & Guide [Електронний ресурс] : Режим доступу: https://na.talan.com/blog/what-is-ibm-maximo-software-definition-guide. (дата звернення: 20.05.2025).

12. Технологія приладобудування: навч. посіб. для студентів напряму підгот. 6.051003 «Приладобудування», 7.090902 «Наукові, аналітичні та екологічні прилади та системи» / В.В. Шевченко, О.В. Осадчий, М.О. Симута ; за ред. В.О. Румбешти ; НТУУ «КПІ». – Київ: НТУУ «КПІ», 2010. – 128 с..

13. C#: що це за мова та де її використовують [Електронний ресурс] : Режим доступу: https://robotdreams.cc/uk/blog/284-s-chto-eto-za-yazyk-i-gde-ego-ispolzuyut. (дата звернення: 24.05.2025).

14. Посібник із настільних програм (Windows Forms .NET) [Електронний ресурс] : Режим доступу: https://hub.kyivstar.ua/articles/shho-take-microsoft-forms-ta-yak-stvoryty-opytuvannya (дата звернення: 26.05.2025).

15. Що таке MySQL? [Електронний ресурс] : Режим доступу: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-mysql/. (дата звернення: 26.05.2025).

16. MySQL Workbench [Електронний ресурс] : Режим доступу: https://www.mysql.com/products/workbench/. (дата звернення: 26.05.2025).

17. Що таке Raspberry Pi? [Електронний ресурс] : Режим доступу: https://blog.avislab.com/raspberry-pi-install/. (дата звернення: 26.05.2025).

18. Огляд операційної системи Linux (клієнтської) [Електронний ресурс] : Режим доступу: https://www.dell.com/support/kbdoc/uk-ua/000141447 (дата звернення: 28.05.2025).

19. Методологія IDEF0 [Електронний ресурс] : Режим доступу: https://stud.com.ua/87184/ekonomika/metodologiya_idef0. (дата звернення: 29.05.2025).

20. Техніко-економічне обґрунтування інженерних рішень в інтелектуальному виробництві: підручник / І. Ш. Невлюдов. Кривий Ріг: Чернявський Д. О., 2024. 388 с.

21. Методичні вказівки до лабораторних робіт з дисципліни «Теорія автоматичного управління» для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» освітня програма Автоматизація та комп’ютерно-інтегровані технології [Електронне видання]/Упоряд.: О.В. Токарєва.–Харків:ХНУРЕ, 2022. – 86 с.

22. 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.