Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему : «Розроблення інформаційного та програмного забезпечення для управління технічним обслуговуванням і ремонтами на виробництві»

Виконав: студент групи ІПЗ20-2
Спеціальність 121 «Інженерія програмного забезпечення»
Маліновський Антон Едуардович
(прізвище та ініціали)
Керівник к.ф.-м.н., доц. Рудянова Т.М.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
Доцент кафедри кібербезпеки та інформаційних технологій
(посада)
к.т.н., доцент Прокопович- Ткаченко Д.І.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Маліновський А. Е. Розроблення інформаційного та програмного забезпечення для управління технічним обслуговуванням і ремонтами на виробництві.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

У сучасних виробничих умовах, ефективний контроль та планування ремонтів обладнання є ключовими аспектами успішної експлуатації підприємства. Дана робота спрямована на розробку системи контролю проведення ремонтів обладнання з метою підвищення ефективності виробничих процесів та зниження витрат на обслуговування.

Першим кроком у роботі є аналіз поточного стану системи контролю ремонтів та визначення проблемних моментів. На основі цього аналізу формулюються вимоги до нової системи, з урахуванням потреб користувачів та сучасних технологічних можливостей.

Далі виконується проектування та розробка системи контролю ремонтів, що включає розробку функціональності, дизайн користувацького інтерфейсу та програмну реалізацію. Після цього система піддається тестуванню та оптимізації.

Очікується, що реалізація цієї системи дозволить оптимізувати процес планування та виконання ремонтів обладнання, підвищить ефективність виробничих процесів та зменшить витрати на обслуговування. Таким чином, дана робота сприятиме підвищенню конкурентоспроможності та стабільності виробничого підприємства.

Ключові слова: .NET Framework, C#, мова програмування, MSSQLServer, база даних, ADO.NET.

ABSTRACT

Malinovskyi A. E. Development of information and software for managing maintenance and repairs in production.

Qualification work for a bachelor's degree in specialty 121 «Software Engineering». – University of Customs and Finance, Dnipro, 2024.

In modern production conditions, effective control and planning of equipment repairs are key aspects of successful enterprise operation. This work is aimed at developing a control system for equipment repairs in order to increase the efficiency of production processes and reduce maintenance costs.

The first step in the work is the analysis of the current state of the repair control system and the identification of problematic points. Based on this analysis, the requirements for the new system are formulated, taking into account the needs of users and modern technological capabilities.

Next, design and development of the repair control system is performed, which includes functionality development, user interface design, and software implementation. After that, the system is tested and optimized.

It is expected that the implementation of this system will allow to optimize the process of planning and execution of equipment repairs, increase the efficiency of production processes and reduce maintenance costs. Thus, this work will contribute to increasing the competitiveness and stability of the production enterprise.

Keywords: .NET Framework, C#, programming language, MSSQLServer, database, ADO.NET.

ЗМІСТ

ВСТУП

В умовах сучасного виробництва ефективний контроль та планування проведення ремонтів обладнання відіграють вирішальну роль у забезпеченні безперебійності виробничих процесів та збереженні оптимального рівня продуктивності. Підприємства зіштовхуються з необхідністю вдосконалення систем управління ремонтами та обслуговуванням обладнання з метою забезпечення максимальної ефективності та безперебійності виробництва.

Ця робота присвячена розробці системи контролю проведення ремонтів обладнання на виробничому підприємстві. Розробка такої системи передбачає створення інструменту, що дозволить ефективно керувати процесом планування, виконання та моніторингу ремонтних робіт, забезпечуючи при цьому максимальний рівень ефективності та безпеки.

Управління ремонтами обладнання є однією з ключових складових успішної діяльності будь-якого виробничого підприємства. Недостатня увага до цього аспекту може призвести до значних втрат продуктивності, збільшення ризику аварійного стану обладнання та недосягнення виробничих показників.

Однією з основних проблем управління ремонтами є несистематичний підхід до планування та проведення ремонтних робіт. Частість ремонтів, їх обсяги та терміни проведення можуть бути неоптимальними, що призводить до збільшення витрат часу та ресурсів на обслуговування обладнання.

Також, недостатній контроль за якістю та ефективністю ремонтних робіт може призвести до недоліків у виконанні та подальших проблем у роботі обладнання. Відсутність систематичного моніторингу та аналізу даних про ремонти ускладнює прийняття управлінських рішень щодо оптимізації процесів обслуговування.

Отже, розробка системи контролю проведення ремонтів обладнання є актуальною задачею, що спрямована на забезпечення ефективного управління ремонтами, підвищення безпеки праці та оптимізацію виробничих процесів.

Метою даної роботи є розробка системи контролю проведення ремонтів обладнання на виробничому підприємстві. Для досягнення цієї мети необхідно вирішити наступні завдання:

1) Аналіз потреб та вимог користувачів – провести детальний аналіз вимог та очікувань користувачів щодо системи контролю ремонтів обладнання.

2) Вивчення сучасних технологій – ознайомитися з сучасними технологіями та інструментами, які можуть бути використані для розробки ефективної системи управління ремонтами.

3) Розробка бази даних – створити базу даних, яка буде зберігати інформацію про стан обладнання, проведені ремонтні роботи та інші необхідні дані.

4) Створення користувацького інтерфейсу – розробити зручний та інтуїтивно зрозумілий інтерфейс, що дозволить користувачам ефективно взаємодіяти з системою.

5) Реалізація функціональності системи – провести програмування та реалізацію всієї необхідної функціональності системи контролю ремонтів обладнання.

6) Тестування та впровадження – провести тестування розробленої системи на реальних даних та впровадити її на виробничому підприємстві.

Виконання цих завдань дозволить створити ефективну систему контролю проведення ремонтів обладнання, яка відповідатиме потребам та вимогам виробничого процесу.

Основними аспектами мети є:

1) Підвищення ефективності виробничого процесу: розроблена система повинна допомагати у плануванні та контролі ремонтних робіт, зменшуючи час простою обладнання та оптимізуючи витрати на обслуговування.

2) Забезпечення безпеки праці та надійності обладнання: система контролю ремонтів має сприяти у попередженні аварій та забезпеченні безпечних умов роботи для працівників.

3) Оптимізація використання ресурсів: завдяки системі контролю ремонтів можна зменшити непланові витрати на ремонт обладнання, раціонально використовуючи ресурси та матеріали.

4) Використання сучасних технологій: розроблена система має використовувати сучасні технології та інструменти для забезпечення ефективного управління та моніторингу ремонтів.

5) Підвищення конкурентоспроможності підприємства: шляхом впровадження ефективної системи контролю ремонтів підприємство може підвищити свою конкурентоспроможність, забезпечуючи стабільність та надійність виробничого процесу.

На сучасному етапі розвитку інформаційних технологій і промисловості, розробка систем контролю ремонтів обладнання стає актуальною задачею для багатьох виробничих підприємств.

Взаємозв'язок даного завдання з іншими роботами може проявляється у інтеграції з ERP-системами. Системи контролю ремонтів можуть бути інтегровані з існуючими системами планування ресурсів підприємства (ERP), що дозволяє автоматизувати процеси управління обладнанням та ремонтними роботами.

Сучасні системи контролю ремонтів можуть використовувати методи аналізу даних та штучного інтелекту для прогнозування потреб у ремонті, виявлення аномалій та оптимізації планування робіт.

Очікувані результати у роботі – створення програмного забезпечення, яке дозволить ефективно планувати, відстежувати та аналізувати ремонтні роботи на виробничому підприємстві.

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з найменувань, додатків.

Структура роботи. Обсяг кваліфікаційної роботи складається з 110 сторінок, 24 рисунків та 13 таблиць.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд існуючого програмного забезпечення для управління технічним обслуговуванням і ремонтами на виробництві

Існують багато програмних систем для контролю ремонтних робіт, які допомагають ефективно керувати технічним обслуговуванням, ремонтом та іншими пов'язаними процесами.

1) Dude Solutions (Asset Essentials) – це комплексне хмарне рішення для управління активами та технічним обслуговуванням (CMMS – Computerized Maintenance Management System). Воно допомагає організаціям ефективно керувати своїми активами, планувати та контролювати ремонтні роботи, а також оптимізувати технічне обслуговування.

Інтерфейс та графічна форма наведено на рис. 1.1, 1.2.

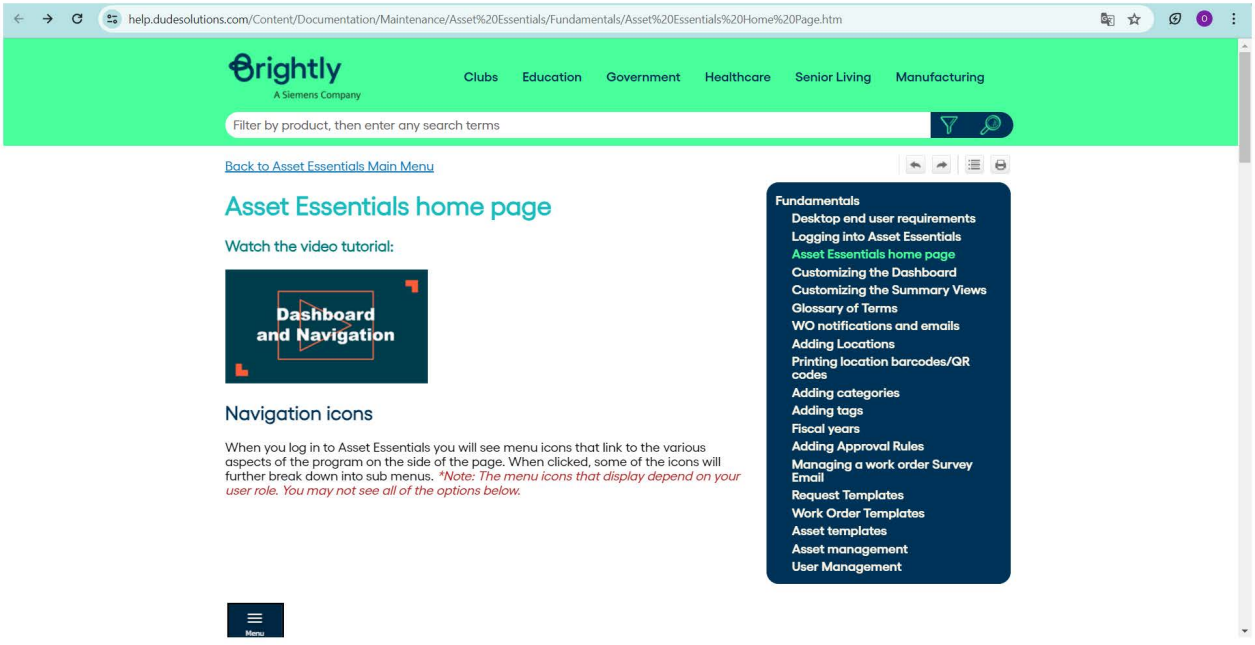

Рисунок 1.1 – Інтерфейс системи Dude Solutions (Asset Essentials)

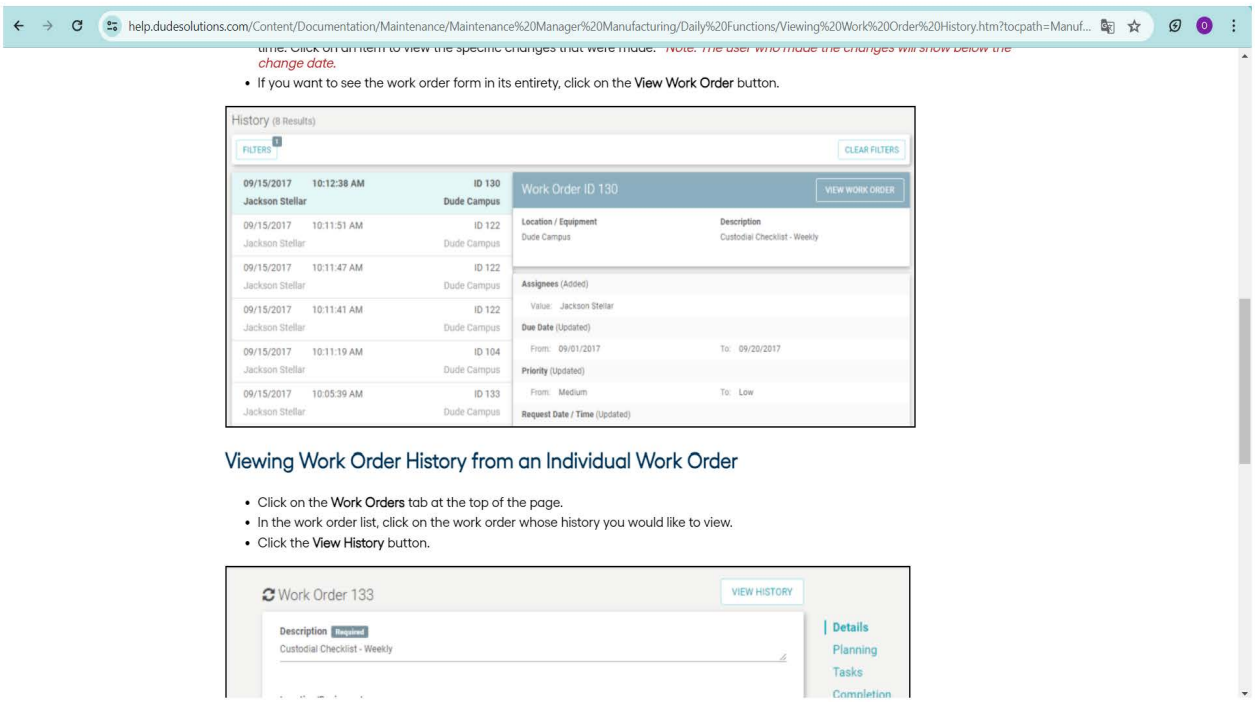

Рисунок 1.2 – Графічна форма системи Dude Solutions (Asset Essentials)

Основні характеристики та функції Asset Essentials: управління робочими замовленнями; планування технічного обслуговування: управління запасами; управління активами; звітність та аналітика; мобільний доступ; інтеграція з іншими системами; користувацькі налаштування.

Переваги використання Asset Essentials:

– Автоматизація процесів технічного обслуговування та ремонту дозволяє зменшити час простою обладнання.

– Хмарне рішення та мобільний доступ забезпечують швидкий і зручний спосіб керування робочими замовленнями в будь-який час і в будь-якому місці.

– Ефективне управління запасами та планування обслуговування дозволяє знизити витрати на закупівлю запасних частин та ремонт.

– Аналітика та звітність допомагають у прийнятті рішень на основі даних.

– Інтуїтивно зрозумілий інтерфейс та прості у налаштуванні робочі процеси роблять систему зручною у використанні навіть для некваліфікованих користувачів.

2) Hippo CMMS – це хмарне рішення для управління технічним обслуговуванням, призначене для оптимізації процесів технічного обслуговування та ремонту обладнання. Відома своєю простотою у використанні, Hippo CMMS допомагає організаціям покращити ефективність роботи, зменшити час простою обладнання та знизити витрати на технічне обслуговування.

Одна із графічних форм системи Hippo CMMS наведена на рис. 1.3.

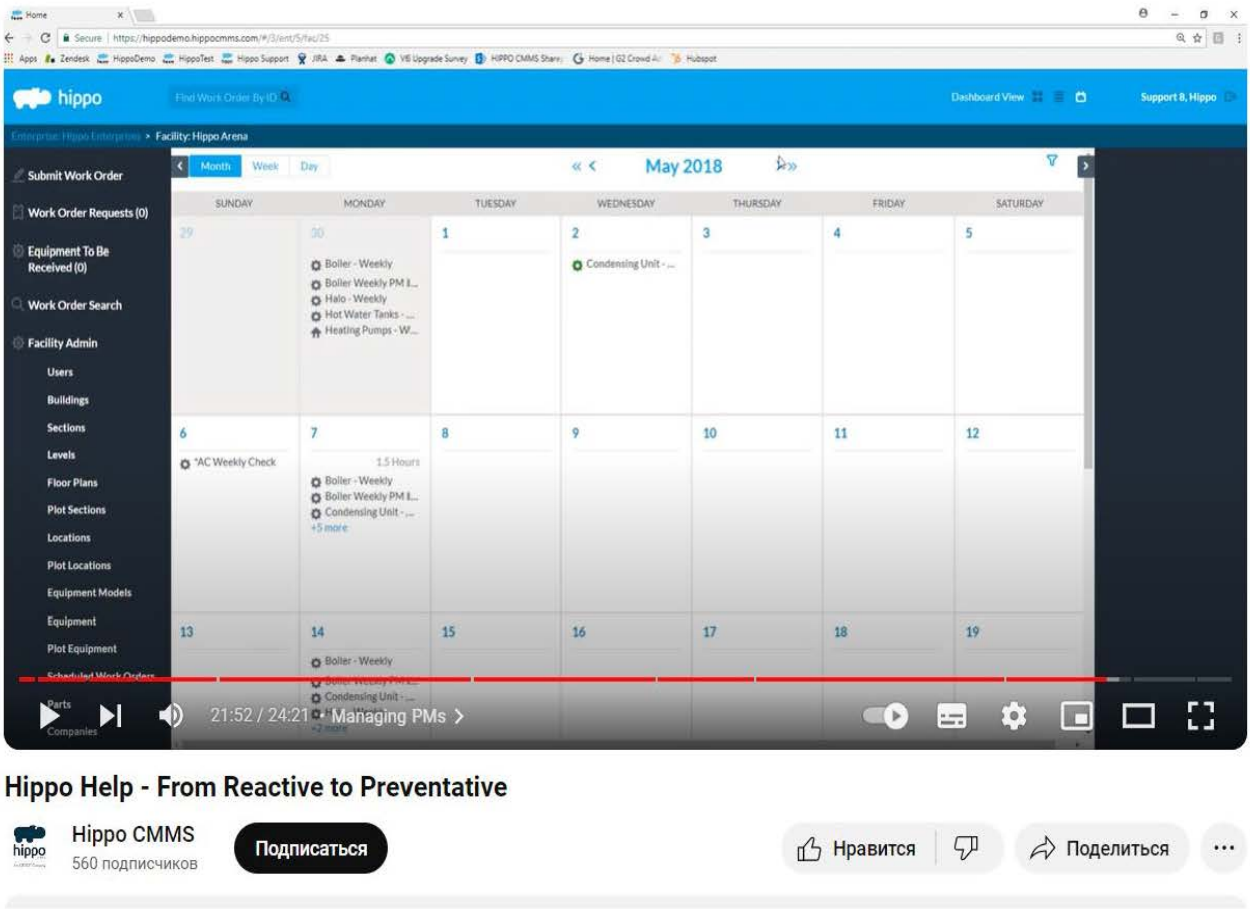

Рисунок 1.3 – Графічна форма системи Hippo CMMS

Основні характеристики та функції Hippo CMMS: управління робочими; управління запасами; управління активами; звітність та аналітика; мобільний доступ; інтеграція з іншими системами; користувацькі налаштування.

Переваги використання Hippo CMMS:

– Інтуїтивно зрозумілий інтерфейс, який не потребує великого навчання

для користувачів.

– Автоматизація процесів технічного обслуговування та ремонту дозволяє зменшити час простою обладнання.

– Ефективне управління запасами та планування обслуговування дозволяє знизити витрати на закупівлю запасних частин та ремонт.

– Дозволяє персоналу створювати та оновлювати робочі замовлення у будь-який час і в будь-якому місці, підвищуючи продуктивність.

– Забезпечує детальні звіти та аналітику для прийняття обґрунтованих рішень на основі даних.

3) UpKeep – це сучасне мобільне та веборієнтоване рішення для управління технічним обслуговуванням, яке допомагає організаціям ефективно керувати ремонтними роботами та технічним обслуговуванням. UpKeep відзначається своєю зручністю використання та широкими функціональними можливостями.

Інтерфейс та графічні форми наведено на рис. 1.4–1.6.

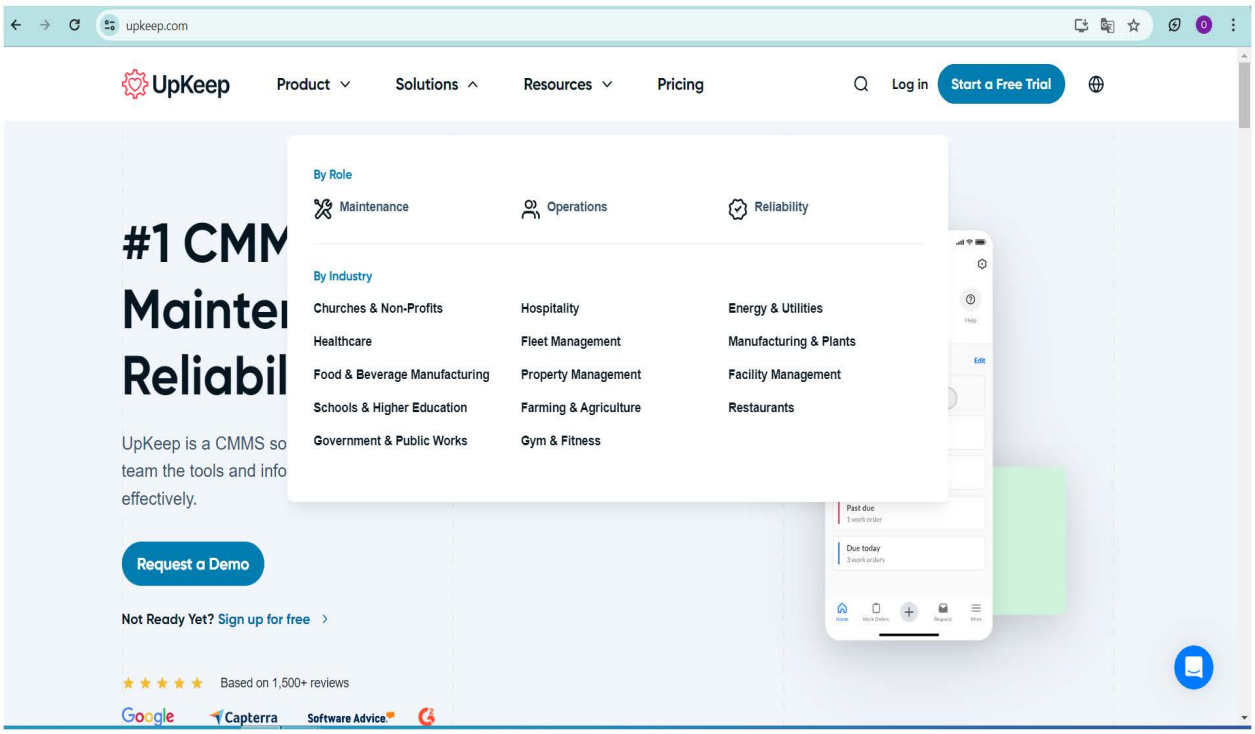

Рисунок 1.4 – Інтерфейс системи UpKeep

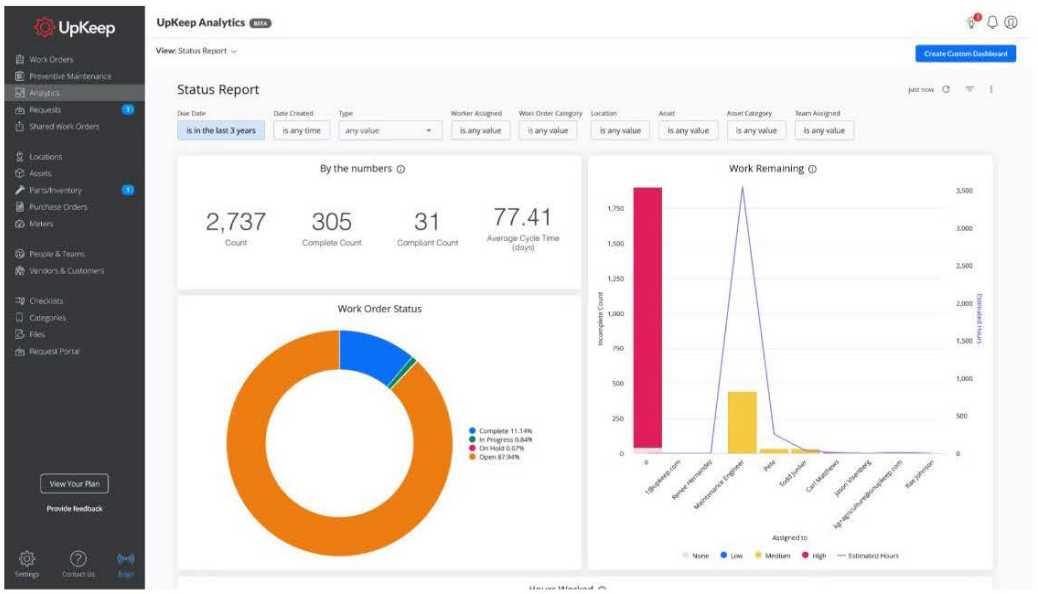

Рисунок 1.5 – Графічна форма_1 системи UpKeep

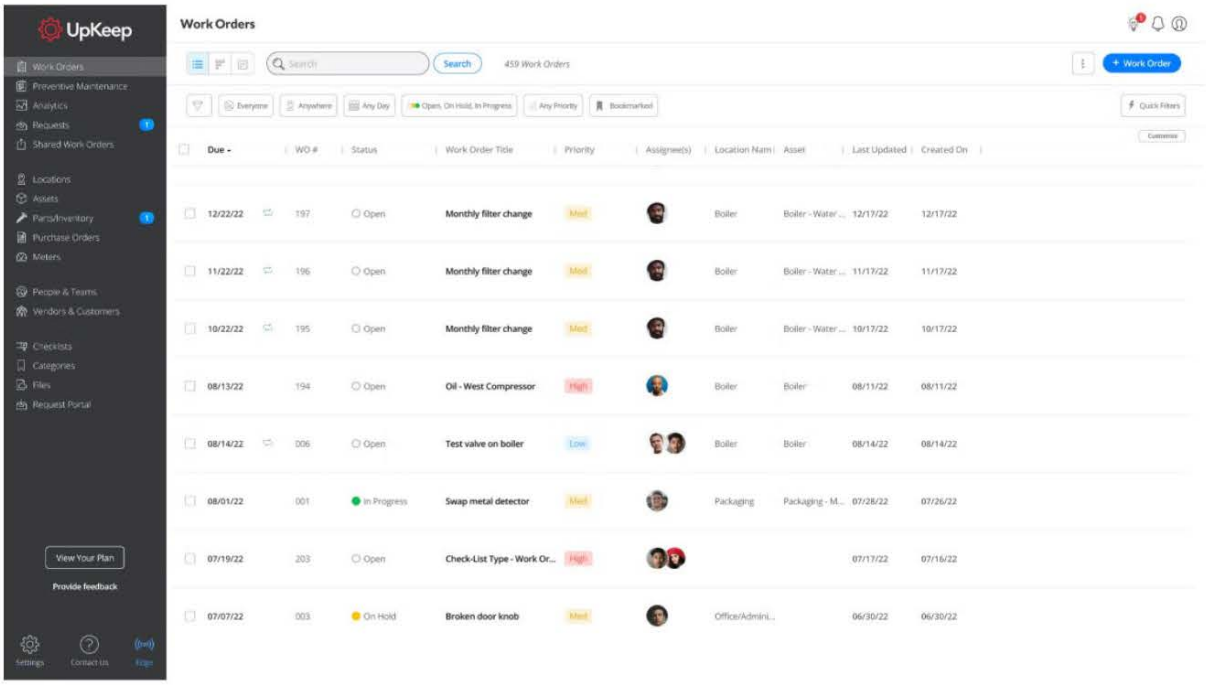

Рисунок 1.6 – Графічна форма_2 системи UpKeep

Опис основних характеристик та функцій UpKeep.

– Управління робочими замовленнями: створення, призначення та

моніторинг робочих замовлень. можливість додавати фото, файли та коментарі до замовлень. відстеження стану виконання робіт у режимі реального часу.

– Планування технічного обслуговування: автоматичне створення робочих замовлень на основі календарних графіків або на основі показників; настроювання регулярних завдань для різних видів обладнання; відправка нагадувань і сповіщень про майбутні роботи.

– Управління запасами: відстеження запасів, їх місцезнаходження та рівнів; автоматичне створення замовлень на закупівлю при досягненні мінімальних рівнів запасів; управління постачальниками та закупівельними замовленнями.

– Управління активами: реєстрація та відстеження всіх активів організації; докладна історія обслуговування та ремонту для кожного активу; відстеження амортизації та вартості володіння активами.

– Звітність та аналітика: генерація звітів для аналізу ефективності технічного обслуговування; настроювані звіти та панелі моніторингу для візуалізації даних; аналітика для прийняття обґрунтованих рішень щодо технічного обслуговування.

– Мобільний доступ: повний доступ до системи з мобільних пристроїв (ios та android); можливість створення та оновлення робочих замовлень у режимі реального часу; підтримка роботи в офлайн-режимі з синхронізацією даних при підключенні до мережі.

– Інтеграція з іншими системами: інтеграція з різними erp-системами, iot-пристроями та іншими корпоративними рішеннями; використання api для обміну даними між системами.

– Користувацькі налаштування: інтуїтивно зрозумілий інтерфейс з гнучкими налаштуваннями; можливість налаштування робочих процесів відповідно до специфічних вимог організації.

4) Maintenance Connection – це веборієнтоване рішення для управління технічним обслуговуванням, яке надає комплексні інструменти для

планування, організації та контролю ремонтних робіт.

Графічна форма системи Maintenance Connection представлена на рис. 1.7.

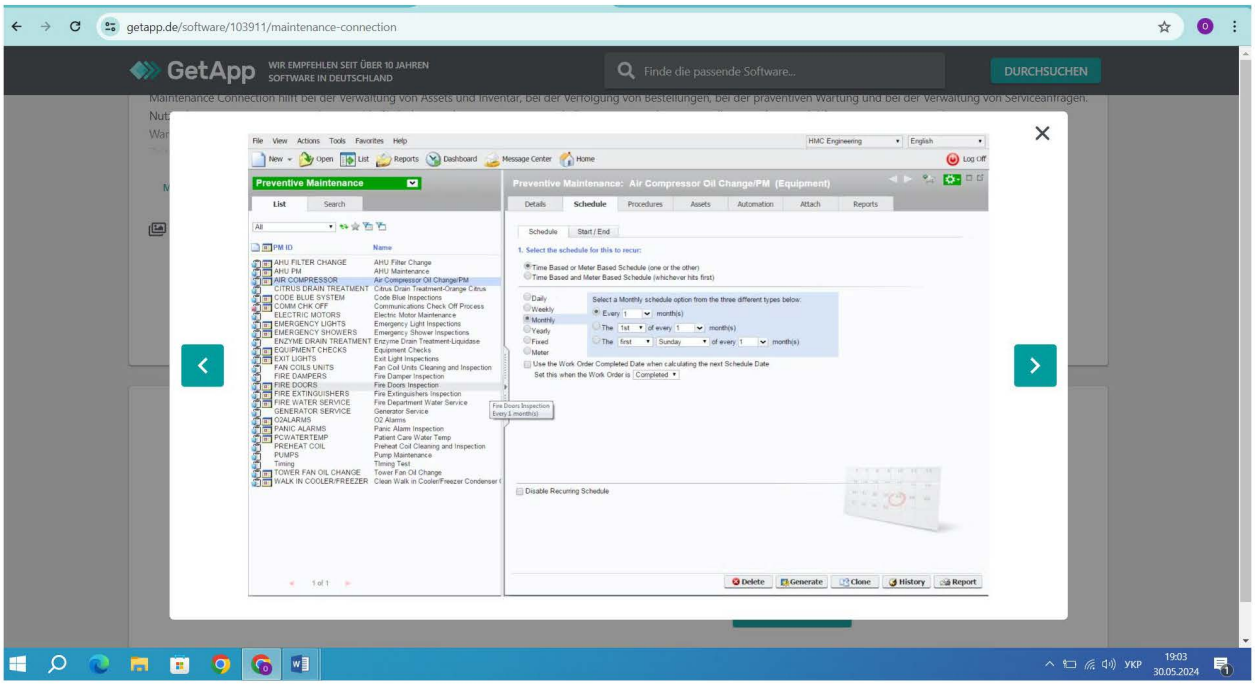

Рисунок 1.7 – Графічна форма системи Maintenance Connection

Переваги використання Maintenance Connection:

– Автоматизація процесів технічного обслуговування та ремонту дозволяє зменшити час простою обладнання.

– Центральне управління всіма аспектами технічного обслуговування допомагає оптимізувати робочі процеси.

– Ефективне управління запасами та планування обслуговування дозволяє знизити витрати на закупівлю запасних частин та ремонт.

– Аналітика та звітність допомагають у прийнятті рішень на основі даних.

Для порівняння систем управління технічним обслуговуванням Maintenance Connection, UpKeep, Dude Solutions (Asset Essentials) та Hippo CMMS, розглянемо їх основні характеристики, переваги та недоліки, що зведені у таблицю 1.1.

Таблиця 1.1

Порівняння найвідоміших систем управління

Характеристика	Maintenance Connection	UpKeep	Dude Solutions (Asset Essentials)	Hippo CMMS
Управління робочими замовленнями	Так	Так	Так	Так
Планування технічного обслуговування	Так	Так	Так	Так
Управління запасами	Так	Так	Так	Так
Управління активами	Так	Так	Так	Так
Звітність та аналітика	Розширена	Обмежена	Розширена	Обмежена
Мобільний доступ	Так	Сильний	Так	Сильний
Інтеграція з іншими системами	Сильна	Обмежена	Сильна	Обмежена
Користувацькі налаштування	Так	Так	Так	Так
Інтуїтивність інтерфейсу	Середня	Висока	Середня	Висока
Вартість	Висока	Помірна	Висока	Низька
Підходить для великих підприємств	Так	Обмежено	Так	Обмежено

1.2 Аналіз предметної області

Аналіз предметної області є важливим етапом у розробці програмного забезпечення.

Основні етапи аналізу предметної області

1) Визначення мети та завдань аналізу

– Визначення цілей та очікуваних результатів аналізу.

– Постановка завдань, які потрібно вирішити в процесі аналізу.

2) Збір вимог

– Ідентифікація зацікавлених сторін (користувачів, замовників, адміністраторів тощо).

– Проведення інтерв'ю, опитувань, воркшопів для збору вимог.

– Аналіз існуючої документації, нормативних актів, стандартів.

3) Опис бізнес-процесів

– Визначення та документування основних бізнес-процесів.

– Використання методів моделювання бізнес-процесів.

4) Аналіз даних

– Визначення ключових об’єктів та сутностей предметної області.

– Моделювання даних за допомогою діаграм сутність-зв’язок (ER-діаграм).

– Визначення атрибутів сутностей та зв’язків між ними.

5) Формулювання вимог до функціональності

– Визначення функціональних вимог до системи.

– Опис функціональних сценаріїв використання системи.

6) Аналіз не функціональних вимог

7) Моделювання системи

– Розробка концептуальних моделей системи.

– Використання методів і інструментів моделювання (UML, IDEF0, IDEF3 тощо).

8) Аналіз ризиків

9) Валідація та верифікація вимог

10) Документування результатів аналізу

– Створення звітів, технічної документації та специфікацій.

– Візуалізація результатів аналізу у вигляді діаграм, схем, таблиць.

Інструментом та методом аналізу є, наприклад, IDEF (Integration DEFinition). IDEF – використання стандартів IDEF0 для функціонального моделювання та IDEF3 для моделювання процесів.

В якості засобу розробки функціональної моделі предметної області обрана методологія IDEF3.

Методологія IDEF3 (Integration DEFinition for Process Description Capture) є частиною сімейства стандартів IDEF і призначена для моделювання та опису бізнес-процесів і робочих процесів у вигляді діаграм. Вона допомагає зрозуміти, як процеси виконуються в поточному стані, і як вони повинні виконуватися в майбутньому.

Основні аспекти методології IDEF3

1) Мета та призначення:

– Мета: документування та аналіз бізнес-процесів для забезпечення їхнього розуміння та покращення.

– Призначення використовується для опису послідовності дій, взаємодій між процесами та потоків даних.

2) Основні компоненти:

– Об’єкти процесів (Process Objects) включають завдання, дії, події та рішення.

– Послідовності процесів (Process Flow) відображають логічний порядок виконання завдань та дій.

– Об’єкти посилань (Referent Objects: включають умови, засоби та об'єкти, що впливають на процес.

3) Основні типи діаграм:

– Схеми процесів (Process Schematic) відображають основні етапи та їх послідовність.

– Діаграми об’єктів посилань (Object Schematic) показують взаємодію між об’єктами, що впливають на процес.

4) Графічні елементи:

– Квадрати (Boxes) відображають дії чи завдання.

– Стрілки (Arrows) показують напрямок і послідовність дій.

– Ромбоподібні блоки (Diamond-shaped Blocks) позначають рішення або розгалуження процесу.

Переваги використання IDEF3:

– забезпечує візуальне представлення процесів, що полегшує їх

розуміння;

– допомагає формалізувати та стандартизувати опис процесів;

– забезпечує основу для аналізу і вдосконалення бізнес-процесів;

– сприяє кращій комунікації між членами команди, замовниками та іншими зацікавленими сторонами.

1.3 Висновки до першого розділу. Постановка завдань дослідження

Мета цієї роботи полягає в розробці інформаційного та програмного забезпечення для ефективного управління технічним обслуговуванням і ремонтами на виробництві. Це включає створення системи, яка автоматизує процеси планування, виконання та контролю технічного обслуговування і ремонтів, зменшуючи час простою обладнання і підвищуючи загальну ефективність виробничих операцій.

Постановка задачі.

Для досягнення мети роботи необхідно вирішити наступні завдання:

1) Аналіз предметної області:

– Дослідження поточних бізнес-процесів, пов'язаних з технічним обслуговуванням і ремонтами обладнання на виробництві.

– Виявлення основних проблем і вузьких місць в існуючій системі управління технічним обслуговуванням.

– Ідентифікація вимог до нової системи з боку користувачів і зацікавлених сторін.

2) Визначення вимог до системи:

– Формулювання функціональних вимог (подача заявок на ремонт, планування технічного обслуговування, моніторинг стану обладнання тощо).

– Формулювання нефункціональних вимог (продуктивність, безпека, зручність використання, масштабованість).

3) Проектування системи:

– Розробка функціональної моделі системи з використанням відповідних методологій моделювання (наприклад, IDEF3, UML).

– Створення детальних діаграм, які відображають основні процеси і потоки даних.

4) Проектування бази даних:

– Розробка логічної моделі бази даних для зберігання інформації про обладнання, запити на обслуговування, графіки технічного обслуговування і ремонти.

– Проектування фізичної моделі бази даних на обраній платформі (наприклад, MSSQLServer).

5) Розробка програмного забезпечення:

– Створення інтерфейсної частини системи, яка забезпечує зручний і інтуїтивно зрозумілий доступ до функцій управління технічним обслуговуванням і ремонтами.

– Реалізація основних функціональних модулів, таких як модулі подачі заявок, планування технічного обслуговування, моніторингу виконання робіт та звітності.

6) Тестування системи:

– Проведення функціонального тестування для перевірки відповідності системи вимогам.

– Виконання нефункціонального тестування для оцінки продуктивності, безпеки та зручності використання системи.

Виконання цих завдань дозволить створити систему управління технічним обслуговуванням і ремонтами, що сприятиме підвищенню надійності та ефективності виробничого обладнання, а також оптимізації процесів управління технічним обслуговуванням

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Вибір програмних засобів для реалізації проекту

Для розробки програмного забезпеченя можуть бути використані наступні технології.

1) .NET Framework спрощує розробку, надає широкий набір інструментів та бібліотек, які дозволяють розробникам ефективно створювати та підтримувати програмне забезпечення різної складності.

2) C# надає розробникам можливість швидко та ефективно створювати високоякісні додатки для різних платформ та сфер застосування

3) MSSQLServer є одним з найпопулярніших рішень у сфері управління базами даних.

4) ADO.NET для реалізації механізму доступу до даних бази даних.

2.2. .NET Framework

.NET Framework – це платформа для розробки та виконання програмного забезпечення, створена компанією Microsoft. Вона забезпечує розробникам інструменти та бібліотеки для створення різноманітних додатків: від десктопних програм до вебсервісів. .NET Framework спрощує розробку, надаючи багатий набір функцій для роботи з графікою, базами даних, мережевими протоколами та іншими ресурсами.

Розглянемо основні компоненти .NET Framework.

1) Common Language Runtime (CLR):

– Віртуальна машина виконання коду: CLR виконує керований код і надає послуги, такі як управління пам’яттю, безпека, управління винятками та інші.

– Garbage Collection: автоматично звільняє пам’ять, яка більше не використовується додатком.

2) .NET Framework Class Library (FCL): бібліотека класів: велика колекція багаторазово використовуваних типів, таких як класи, інтерфейси та структури, які надають функціональність для загальних програмних завдань (робота з файлами, введення-виведення, обробка даних, мережеві операції тощо).

3) Асінхронне програмування: підтримка асинхронного програмування для написання неблокуючих операцій вводу-виводу та інших асинхронних завдань.

4) ADO.NET: технологія для доступу до даних, яка дозволяє працювати з різними джерелами даних, такими як бази даних SQL, XML-файли тощо.

5) ASP.NET: фреймворк для створення вебдодатків та сервісів, включаючи вебAPI та MVC.

6) Windows Forms: бібліотека для створення графічних інтерфейсів користувача (GUI) для Windows-додатків.

7) Windows Presentation Foundation (WPF): бібліотека для побудови сучасних настільних додатків з використанням декларативного підходу до проектування інтерфейсів.

8) Windows Communication Foundation (WCF): фреймворк для побудови сервіс-орієнтованих додатків (SOA) і роботи з вебсервісами.

.NET Framework підтримує кілька мов програмування, таких як C#, VB.NET, F# та інші.

CLR забезпечує різні рівні безпеки для виконуваного коду, включаючи валідацію та перевірку коду (Code Access Security).

.NET Framework є потужною і універсальною платформою для розробки додатків на Windows. Він надає широкий набір інструментів та бібліотек, які дозволяють розробникам ефективно створювати та підтримувати програмне забезпечення різної складності.

BindingSource – це клас в .NET Framework, який забезпечує функціональність для зв'язування джерела даних з елементами управління у Windows Forms. Він полегшує роботу з даними, дозволяючи легко

здійснювати такі операції, як навігація, фільтрація, сортування та редагування даних.

Основні функції BindingSource

1) Зв’язування даних:

– BindingSource використовується для зв’язування даних з візуальними елементами управління, такими як DataGridView, TextBox, ComboBox тощо.

– Він абстрагує джерело даних, дозволяючи вам легко змінювати джерело без необхідності змінювати код, який взаємодіє з елементами управління.

2) Навігація: забезпечує функції для навігації по записах у джерелі даних.

3) Редагування даних: підтримує додавання, редагування та видалення записів у зв'язаному джерелі даних.

4) Фільтрація та сортування: Можливість фільтрувати та сортувати дані за допомогою властивостей Filter і Sort.

5) Синхронізація даних: Автоматично синхронізує дані між джерелом даних і візуальними елементами управління.

Переваги використання BindingSource:

– Зв’язування даних з елементами управління стає простішим та більш інтуїтивним.

– Легко змінювати джерело даних без необхідності змінювати код зв'язування.

– Підтримка навігації, редагування, фільтрації та сортування даних прямо з класу BindingSource.

BindingSource є потужним інструментом для роботи з даними у додатках Windows Forms, надаючи розробникам зручний спосіб управління та маніпулювання даними.

2.3 C#

C# (C Sharp) – це об’єктно-орієнтована мова програмування, розроблена компанією Microsoft як частина платформи .NET. Вона була створена для забезпечення простоти, надійності та високої продуктивності при розробці програмного забезпечення. Її дизайн та архітектура спиралися на вже існуючі мови програмування, такі як C++ і Java, щоб зробити мову знайомою для розробників. З тих пір C# постійно розвивається, і кожна нова версія додає нові можливості та покращення [1–5].

Основна перевага C# полягає у тому, що вона базується на об'єктно-орієнтованому підході, що спрощує розробку програм та забезпечує стабільність коду. Також важливою особливістю є те, що додатки, написані на C#, можуть бути запущені на різних платформах, які підтримують платформу .NET Framework.

Розробники використовують C# для створення широкого спектру додатків – від невеликих ігор та додатків для керування даними до великих корпоративних систем. Багатофункціональна бібліотека класів .NET Framework, яка інтегрується з C#, надає доступ до різноманітних інструментів для роботи з даними, мережами, графікою та іншими аспектами програмування.

Мова програмування C# може бути використана для створення різноманітних компонентів та модулів:

1) Інтерфейс користувача: Ви можете використовувати C# для розробки десктопних або вебінтерфейсів для взаємодії з системою. Це може включати створення форм для введення та відображення даних про ремонтне обладнання, звітів про виконані ремонтні роботи, а також інтерфейсів для адміністрування системи.

2) Бізнес-логіка та обробка даних: Ви можете використовувати C# для реалізації бізнес-логіки вашої системи, такої як обробка запитів користувачів, перевірка даних на валідність, виконання розрахунків для планування

ремонтних робіт тощо.

3) Взаємодія з базою даних: C# може бути використаний для підключення до бази даних MSSQLServer за допомогою технології ADO.NET та виконання запитів до бази даних для отримання та збереження інформації про ремонтне обладнання та проведені ремонтні роботи.

4) Розробка звітів та аналітики: Ви можете використовувати C# для створення скриптів та компонентів, що аналізують дані про проведені ремонтні роботи та генерують звіти або візуалізації для аналізу ефективності робіт та планування майбутніх заходів.

5) Інтеграція з іншими системами: C# може бути використаний для реалізації інтерфейсів для взаємодії з іншими системами, такими як системи обліку обладнання, системи планування ресурсів підприємства (ERP), системи моніторингу та діагностики тощо.

Загалом, C# є потужним інструментом для розробки програмного забезпечення, який надає розробникам можливість швидко та ефективно створювати високоякісні додатки для різних платформ та сфер застосування.

2.4 MSSQLServer

Microsoft SQL Server (MSSQL Server) – це реляційна система управління базами даних (СУБД), розроблена компанією Microsoft. MSSQL Server є одним з найпопулярніших рішень у сфері управління базами даних та широко використовується у бізнес-середовищах для зберігання, керування та обробки великих обсягів даних [6–10].

Основні характеристики та функції MSSQL Server включають:

1) підтримує реляційну модель даних, де дані зберігаються у вигляді таблиць, що містять рядки та стовпці. Це дозволяє ефективно організовувати та керувати даними у базі даних.

2) використовує стандартну мову запитів SQL (Structured Query Language) для взаємодії з базою даних. SQL дозволяє виконувати різноманітні

операції з даними, такі як вибірка, вставка, оновлення та видалення.

3) підтримує транзакції, що забезпечує консистентність даних у базі даних. Це дозволяє виконувати групу операцій як єдину атомарну операцію, яка виконується повністю або не виконується зовсім.

4) має вбудовані механізми забезпечення безпеки, такі як аутентифікація та авторизація користувачів, шифрування даних, аудит дій користувачів та інші.

5) має велику кількість функцій, що включають в себе індексацію, оптимізацію запитів, реплікацію даних, аналітичні можливості та інші. Він також відомий своєю високою продуктивністю та надійністю.

6) підтримує різноманітні типи додаткових компонентів та розширень, такі як Analysis Services, Reporting Services, Integration Services та інші, що розширюють його можливості та дозволяють виконувати різноманітні завдання.

SQL (Structured Query Language) – це стандартна мова запитів, яка використовується для взаємодії з реляційними базами даних. Вона надає зручний та потужний спосіб виконання різноманітних операцій з даними, таких як вибірка, вставка, оновлення та видалення даних, а також для управління структурою бази даних.

Основні елементи мови SQL включають:

1) Запит SELECT: використовується для вибірки даних з бази даних. Може бути використаний для вибору всіх рядків з таблиці або вибору певних стовпців чи рядків за певною умовою.

2) Оператори INSERT, UPDATE та DELETE: використовуються для додавання нових записів у таблицю (INSERT), оновлення існуючих записів (UPDATE) та видалення записів з таблиці (DELETE).

3) Клаузи WHERE, GROUP BY та HAVING: використовуються для фільтрації, групування та агрегації даних. Наприклад, клауза WHERE дозволяє встановлювати умови для вибірки певних рядків, клауза GROUP BY дозволяє групувати дані за певними полями, а клауза HAVING застосовується для

фільтрації результатів групування.

4) Функції агрегування: SQL має вбудовані функції для обчислення агрегатних значень, таких як SUM, AVG, COUNT, MAX і MIN. Ці функції використовуються для обчислення суми, середнього значення, кількості рядків, максимального та мінімального значення в результатах запиту.

5) Підзапити: SQL дозволяє вкладати один запит в інший, що дозволяє виконувати складніші запити або отримувати дані з одного запиту на основі результатів іншого.

6) Сортування та обмеження результатів: SQL дозволяє сортувати результати запиту за певними полями за допомогою ORDER BY, а також обмежувати кількість виведених рядків за допомогою LIMIT (або TOP в деяких СУБД).

Мова SQL є потужним інструментом для роботи з базами даних і використовується в різних сферах розробки програмного забезпечення.

Загалом, Microsoft SQL Server є потужним інструментом для управління базами даних, який відповідає потребам різних типів бізнесів та додатків, які вимагають надійного та ефективного зберігання та обробки даних.

2.5 ADO.NET

ADO.NET (ActiveX Data Objects for .NET) – це набір класів, які входять до складу платформи .NET і забезпечують доступ до даних, збережених в різних джерелах, таких як реляційні бази даних, XML документи та інші джерела даних. ADO.NET надає функціональність для підключення до баз даних, виконання команд та запитів, а також для роботи з результатами цих запитів.

Основні компоненти ADO.NET [11–15]

1) Data Providers, що вкдючає:

– SqlConnection: використовується для підключення до SQL Server.

– SqlCommand: використовується для виконання SQL-команд.

– SqlDataReader: застосовується для читання даних з бази даних в режимі тільки вперед (forward-only).

– SqlDataAdapter: використовується для заповнення DataSet та синхронізації змін з базою даних.

2) DataSet:

– DataSet: Набір даних, що містить одну або кілька таблиць даних (DataTable), а також відносини між ними (DataRelation). DataSet є дисконнектним представленням даних, що дозволяє працювати з даними без постійного підключення до джерела даних.

– DataTable: Представляє таблицю даних в пам'яті, містить колекцію рядків (DataRow) та стовпців (DataColumn).

– DataRelation: Представляє зв’язки між таблицями в DataSet.

Переваги ADO.NET:

– Підтримка роботи з різними джерелами даних, такими як реляційні бази даних, XML-документи та інші.

– Можливість працювати з даними в дисконнектному режимі, що знижує навантаження на базу даних.

– Підтримка параметризованих запитів, що запобігає SQL-ін'єкціям.

– Можливість обробки великих обсягів даних за рахунок використання дисконнектного доступу та потокового читання даних.

TableAdapter– це компонент, який входить в склад технології ADO.NET і дозволяє зв'язати дані з бази даних з об'єктами даних у програмі. В основному, TableAdapter використовується для виконання запитів до бази даних, отримання результатів запитів та збереження змін у базі даних.

Основні функції TableAdapter:

1) Виконання запитів до бази даних: TableAdapter може виконувати різноманітні запити до бази даних, такі як SELECT, INSERT, UPDATE або DELETE. Він генерує відповідні SQL-запити і виконує їх у базі даних.

2) Передача даних між базою даних і об'єктами даних: TableAdapter отримує дані з бази даних та передає їх об'єктам даних у програмі. Це може

бути DataSet, DataTable або інший об'єкт, що представляє дані у відповідному форматі.

3) Автоматичне збереження змін у базі даних: Після того як дані були змінені у DataSet або DataTable, TableAdapter може автоматично зберегти ці зміни у базі даних за допомогою відповідних SQL-запитів.

4) Генерація коду для доступу до даних: Під час додавання TableAdapter до проекту, Visual Studio може згенерувати код для доступу до даних, що дозволяє швидко та легко отримувати доступ до даних у програмі.

Загалом, TableAdapter дозволяє програмістам легко та ефективно працювати з базами даних у програмах, що використовують технологію ADO.NET. Він дозволяє розробникам виконувати запити до бази даних, отримувати та зберігати дані, не затримуючись на написанні складних SQL-запитів або коду для доступу до даних.

ADO.NET є потужним інструментом для роботи з даними в додатках на платформі .NET, забезпечуючи багатий набір функцій для підключення до баз даних, виконання команд і роботи з результатами запитів.

2.6 Висновки до другого розділу

Використання .NET Framework, мови програмування C#, MSSQLServer та ADO.NET у розробці системи контролю проведення ремонтів обладнання на виробничому підприємстві є важливими кроками для забезпечення успішної та ефективної розробки програмного забезпечення.

.NET Framework спрощує розробку, надаючи широкий набір інструментів та бібліотек, що дозволяє розробникам ефективно створювати та підтримувати програмне забезпечення різної складності.

C# надає можливість швидко та ефективно створювати високоякісні додатки для різних платформ та сфер застосування, що відповідає нашим потребам у розробці системи контролю ремонтів.

MSSQLServer, як одне з найпопулярніших рішень у сфері управління

базами даних, забезпечить надійне та ефективне зберігання та управління даними нашої системи.

ADO.NET, у свою чергу, дозволить нам ефективно реалізувати механізм доступу до даних бази даних, забезпечуючи швидку та безпечну обробку даних у нашій системі.

Усі ці технології в поєднанні забезпечать розробку ефективної, надійної та функціональної системи контролю проведення ремонтів обладнання, яка відповідатиме потребам і очікуванням користувачів.

РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ КОНТРОЛЮ ПРОВЕДЕННЯ РЕМОНТІВ ОБЛАДНАННЯ

3.1 Діаграми потоку даних системи контролю проведення ремонтів обладнання

Діаграми потоку даних (DFD, Data Flow Diagrams) є стандартною графічною нотацією для моделювання інформаційних систем. Вони використовуються для представлення процесів, що обробляють дані, і потоків даних між цими процесами. DFD допомагають зрозуміти, як дані переміщуються всередині системи, і є важливим інструментом для аналізу і проектування інформаційних систем.

Основними компонентами DFD є процеси, потоки даних, сховища даних та зовнішні сутності.

Процеси (Processes) відображають операції або функції, що обробляють дані. На діаграмі позначаються прямокутниками із заокругленими кутами. Кожен процес повинен мати назву, яка описує його функцію.

Потоки даних (Data Flows) вказують на переміщення даних між процесами, сховищами даних та зовнішніми сутностями. На діаграмі позначаються стрілками, які вказують напрямок потоку даних. Кожен потік даних повинен мати назву, що описує тип даних, які переміщуються.

Сховища даних (Data Stores) відображають місця зберігання даних. На діаграмі позначаються двома паралельними лініями або відкритим прямокутником. Кожне сховище даних повинне мати назву, що описує тип збережених даних.

Зовнішні сутності (External Entities) відображають зовнішні системи або користувачів, що взаємодіють з моделлю. На діаграмі позначаються прямокутниками. Кожна зовнішня сутність повинна мати назву, що описує її роль.

Рівні деталізації DFD:

1) Контекстна діаграма (Context Diagram) – найвищий рівень деталізації, відображає всю систему як один процес, показує взаємодію системи з зовнішніми сутностями.

2) Діаграми нульового рівня (Level 0 DFD) розбивають контекстну діаграму на основні підпроцеси, показують основні функції системи та взаємозв'язки між ними.

3) Діаграми першого рівня (Level 1 DFD) і далі деталізують окремі підпроцеси з діаграм нульового рівня; показують більш детальний потік даних та внутрішню структуру процесів.

Для системи контролю проведення ремонтів обладнання контекстна діаграма (рис. 3.1):

1) Процес: Система контролю проведення ремонтів обладнання.

2) Потоки даних: замовлення на ремонт або заточення інструментів / приладів, Акт приймання-здачі відремонтованих, реконструйованих та модернізованих об’єктів, звіти про виконані роботи – контроль ремонтних робіт.

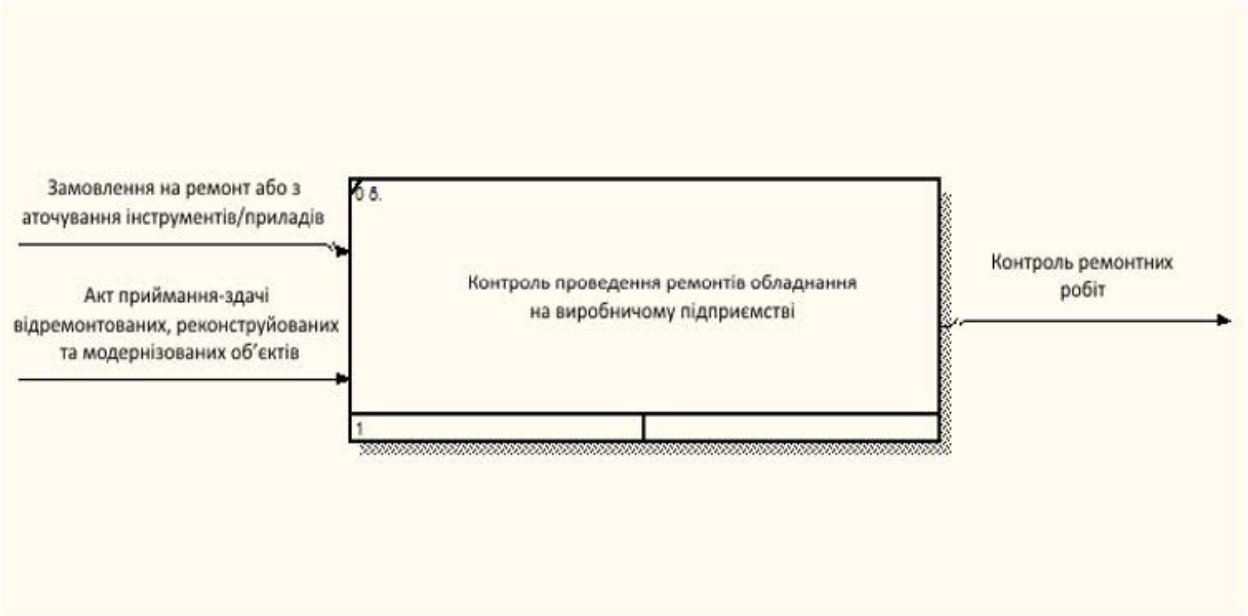

Рисунок 3.1 – Context Diagram системи контролю проведення ремонтів обладнання

Діаграма декомпозиції першого рівня предметної області системи контролю проведення ремонтів обладнання показана на рис. 3.2.

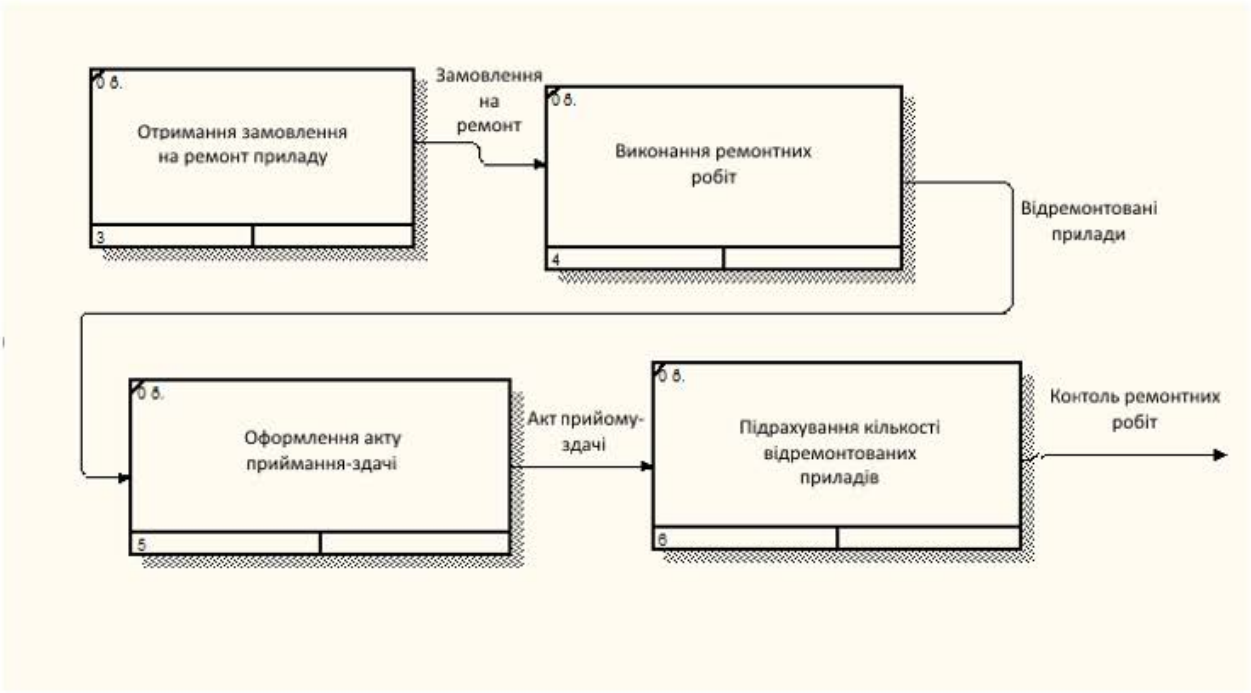

Рисунок 3.2 – Діаграма декомпозиції першого рівня предметної області системи контролю проведення ремонтів обладнання

DFD є потужним інструментом для візуалізації та аналізу потоків даних у системі, що допомагає зрозуміти, як інформація переміщується між різними частинами системи та які процеси її обробляють. Це дозволяє виявити потенційні проблеми та покращення у проектуванні інформаційних систем.

3.2 Функціональна і об’єктна модель системи контролю проведення ремонтів обладнання

Статичними об’єктами предметної області є співробітники, посади, відділи, позначки (табл. 3.1 – 3.3).

Таблиця 3.1

Специфікація сутностей системи контролю проведення ремонтів обладнання

№	Назва сутності	Описання сутності
1	Організаційна одиниця підприємства	Статична сутність, яка описує цеха підприємства або інші організаційні одиниці.
2	Персони	Статична сутність, яка описує персон, що працюють на підприємстві.
3	Обладнання	Статична сутність, яка описує обладнання, що є в цехах.
4	Види робіт	Статична сутність, яка описує які види робіт виконуються над обладнанням.
5	Замовлення на ремонт або заточування інструментів/приладів	Документ предметної області, в якому фіксується обладнання що замовляється на ремонтні роботи та кількість обладнання, що віддали на ремонт.
6	Акт приймання-здачі відремонтованих, реконструйованих та модернізованих об’єктів	Документ предметної області, в якому фіксується яке обладнання було повернуто з ремонтних робіт та кількість цього обладнання.

Таблиця 3.2

Специфікація атрибутів сутностей

№	Назва сутності	Назва атрибуту	Опис атрибуту
1	Організаційна одиниця підприємства	Назва	Символьний тип даних. Задає назву організаційної одиниці підприємства.
2	Персони	ПІБ персони	Символьний тип даних. Задає ПІБ персони.
		Номер паспорту	Символьний тип даних. Задає номер паспорту персони.
		Організаційна одиниця підприємства	Посилання на об’єкт «Організаційна одиниця підприємства»
3	Обладнання	Назва	Символьний тип даних. Задає назву обладнання

		Номенклатурний номер	Символьний тип даних. Задає номенклатурний номер.
		Інвентарний номер	Символьний тип даних. Задає інвентарний номер.
		Заводський номер	Символьний тип даних. Задає заводський номер.
4	Види робіт	Назва виду робіт	Символьний тип даних. Задає назву виду робіт
5	Замовлення на ремонт або заточування інструментів/приладів	Номер	Чисельний тип даних. Задає номер документу.
		Дата	Тип даних – дата. Задає дату створення документу.
		Строк виконання, днів	Чисельний тип даних. Задає строк виконання у днях
		Організаційна одиниця замовника	Посилання на об’єкт «Організаційна одиниця підприємства»
		Замовник, що створив документ	Посилання на об’єкт «Персона»
		Організаційна одиниця виконавця	Посилання на об’єкт «Організаційна одиниця підприємства»
		Виконавець , що прийняв замовлення	Посилання на об’єкт «Персона»
		Обладнання	Посилання на об’єкт «Обладнання»
		Кількість	Чисельний тип даних. Задає кількість обладнання що здали на ремонт.
6	Акт приймання-	Номер	Символьний тип даних.

	здачі відремонтованих, реконструйованих та модернізованих об'єктів		Задає назву відділу
		Дата	Тип даних – дата. Задає дату створення документу.
		Дата з	Тип даних – дата. Задає дату початку робіт.
		Дата по	Тип даних – дата. Задає дату кінця робіт.
		Особа, що повернула обладнання з робіт	Посилання на об'єкт «Персона»
		Особа, що прийняла обладнання з робіт	Посилання на об'єкт «Персона»
		Обладнання	Посилання на об'єкт «Обладнання»
		Вид робіт	Посилання на об'єкт «Види робіт»
		Кількість	Чисельний тип даних. Задає кількість обладнання що повернено з ремонту.

Таблиця 3.3

Зв'язки між сутностями

Номер зв'язку	Назва 1 сутності	Назва 2 сутності	Множинність зв'язку
1	Персони	Організаційна одиниця підприємства	1:1
2	Обладнання	Організаційна одиниця підприємства	1:1
3	Замовлення на ремонт або заточування інструментів/приладів	Персони	1:N
4	Замовлення на ремонт або заточування інструментів/приладів	Організаційна одиниця замовника	1:N
5	Замовлення на ремонт або заточування інструментів/приладів	Обладнання	1:N
6	Акт приймання-здачі	Персони	1:N

	відремонтованих, реконструйованих та модернізованих об’єктів		
7	Акт приймання-здачі відремонтованих, реконструйованих та модернізованих об’єктів	Обладнання	1:N
8	Акт приймання-здачі відремонтованих, реконструйованих та модернізованих об’єктів	Види робіт	1:N

Діаграма «сутність-зв’язок» (ERD, Entity-Relationship Diagram) є популярним інструментом для моделювання даних, який використовується для представлення структури бази даних. ERD показує сутності (об’єкти) у системі, атрибути цих сутностей та зв’язки між ними. Це допомагає зрозуміти, як дані зберігаються та взаємодіють у системі.

Основні компоненти ERD:

Сутності (Entities) – основні об’єкти, які існують незалежно і мають атрибути, що їх описують. На діаграмі позначаються прямокутниками. Кожна сутність повинна мати унікальне ім'я.

Атрибути (Attributes) – властивості або характеристики сутностей. На діаграмі позначаються еліпсами і з’єднуються лініями з відповідними сутностями. Один із атрибутів, який ідентифікує сутність унікально, називається ключовим атрибутом (Primary Key) і зазвичай підкреслюється.

Зв’язки (Relationships) відображають асоціації між сутностями. На діаграмі позначаються ромбами або прямими лініями, що з'єднують сутності. Зв’язки можуть мати атрибути, які описують додаткові характеристики асоціації.

Кратність (Cardinality) вказує на кількість сутностей, що можуть бути пов'язані однією сутністю іншого типу (один до одного, один до багатьох, багато до багатьох).

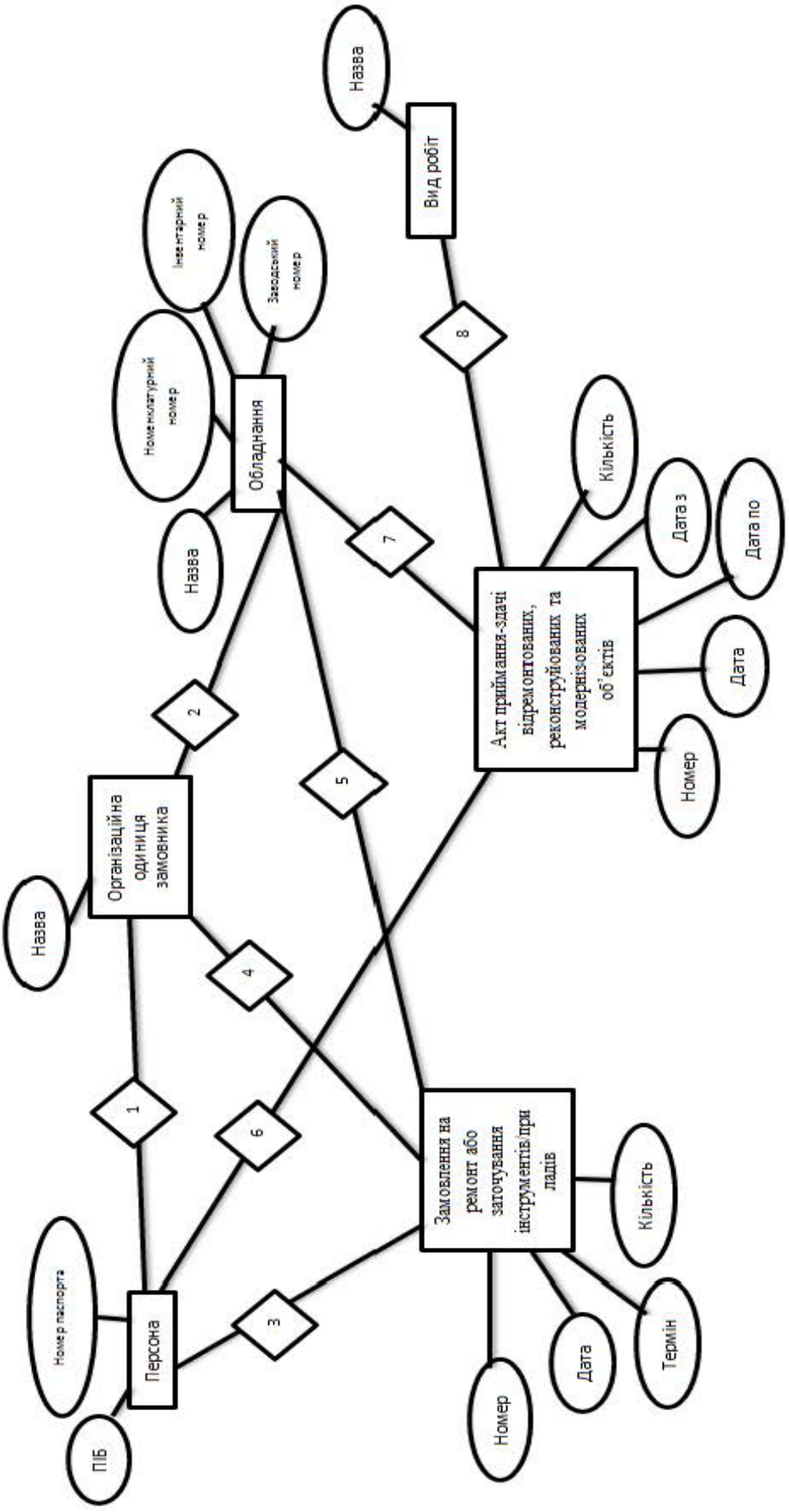

Рисунок 3.1 – Діаграма «сутність-зв'язок» системи контролю проведення ремонтів обладнання

3.3 Організація меню

Будь-яка автоматизована інформаційна система складається з кількох типів інформації, зокрема нормативно-довідникової, вхідної та вихідної.

1) Нормативно-довідникова інформація включає дані, які використовуються для регламентації та стандартизації процесів у системі. Ця інформація зазвичай є статичною і рідко змінюється. Вона може містити:

– Стандарти та класифікації, що використовуються для кодування даних (наприклад, коди ремонтних робіт, коди типів обладнання).

– Довідники – переліки стандартних значень (наприклад, список типів обладнання, список запчастин, довідник постачальників).

– Нормативи – встановлені норми та стандарти, що регулюють процеси (наприклад, норми часу на виконання ремонтних робіт, нормативи споживання матеріалів).

2) Вхідна інформація включає дані, які вводяться в систему для подальшої обробки. Ці дані можуть бути отримані від користувачів системи або зовнішніх джерел. Приклади вхідної інформації:

Заявки на ремонт: Інформація про потребу в проведенні ремонту (опис проблеми, дата подання заявки, пріоритет).

Дані про стан обладнання: Результати інспекцій та перевірок (технічний стан, зауваження, рекомендації).

Дані про використані матеріали та ресурси: Інформація про використані запчастини, витратні матеріали та залучені ресурси.

3) Вихідна інформація включає дані, які система генерує в результаті обробки вхідної інформації. Ця інформація зазвичай використовується для прийняття рішень, звітності та контролю. Приклади вихідної інформації:

– звіти про виконані роботи.

– графіки технічного обслуговування та ремонтів.

– аналіз витрат.

– оцінка ефективності.

Система контролю проведення ремонтів обладнання повинна містити нормативно-довідникову інформацію (рис. 3.2):

1) персони;

2) організаційна одиниця замовника;

3) види робіт;

4) обладнання.

До складу вхідної інформації системи необхідно віднести наступні документи предметної області:

1) Замовлення на ремонт або заточування інструментів/приладів;

2) Акт приймання-здачі відремонтованих, реконструйованих та модернізованих об'єктів.

Рисунок 3.2 – Організація меню система контролю проведення ремонтів обладнання

3.4 Розробка бази даних

Розробка бази даних (БД) складається з двох основних етапів: розробки логічної та фізичної моделі. Ці два етапи є ключовими для створення ефективної та функціональної бази даних.

Логічна модель бази даних є концептуальним проектом, який визначає

структуру даних без урахування фізичних аспектів їх зберігання. Вона зосереджується на організації даних, відносинах між ними та правилах, які визначають їх цілісність

Нормативно-довідникова інформація АІС

Довідник «Організаційна одиниця підприємства»

Таблиця 3.4

Логічна структура таблиці БД «Організаційна одиниця підприємства» (Departments)

Код організаційної одиниці	Назва
ID_Department	Dep_Nam
I3	C100

Довідник «Персони»

Таблиця 3.5

Логічна структура таблиці бази даних «Персони» (Persons)

Код персони	Код організаційної одиниці	Прізвище	Ім'я	По батькові	Номер паспорта
ID_Person	ID_Department	Surname	Name	Patronimyc	Passport_nom
I4	I3	C50	C50	C50	C8

Довідник «Види робіт»

Таблиця 3.5

Логічна структура таблиці бази даних «Види робіт» (Work_Types)

Код виду робіт	Назва
ID_work_type	wrk_tp
I2	C50

Довідник «Обладнання»

Фізична модель бази даних визначає конкретні способи зберігання даних у системі управління базами даних (СУБД). Вона зосереджується на оптимізації продуктивності, використанні ресурсів та забезпеченні надійності

і безпеки даних (рис. 3.3).

Таблиця 3.6

Логічна структура таблиці бази даних «Обладнання» (Equipment)

Код обладнання	Код організаційної одиниці	Номенклатурний номер	Інвентарний номер	Заводський номер	Назва
ID_Equipment	ID_Departmen t	Nomenc_nom	Invent_nom	Factory_ nom	Eq_name
I4	I3	C11	C6	C11	C70

Вхідна інформація

Таблиця 3.7

Логічна структура таблиці бази даних «Замовлення на ремонт або заточування інструментів/приладів» (Order_rem):

Код замовлення	Організац. одиниця замовника	Організац. одиниця виконавець	Замовник, що створив документ	Виконавець, що прийняв замовлення	Номер
ID_Order_ rem	ID_DepOrd	ID_DepP erf	ID_PersonG iv	ID_PersonR et	No m
I4	I3	I3	I4	I4	I4
Дата	Строк виконання, днів				
Date_	Term				
D8	I3				

Таблиця 3.8

Логічна структура таблиці бази даних «Обладнання у замовленні» (Equipment_in_Order):

Код обладнання у замовленні	Код замовлення	Код обладнання	Кількість
ID_Ord_Eq	ID_Order_rem	ID_Equipment	Amount
I4	I4	I4	I2

Таблиця 3.9

Логічна структура таблиці бази даних «Акт приймання-здачі відремонтованих, реконструйованих та модернізованих об'єктів» (Act_return):

Код акту	Код замовлення	Особа, що повернула обладнання з робіт	Особа, що прийняла обладнання з робіт	Номер	Дата
ID_Act	ID_Order_rem	ID_PersonGive	ID_PersonRet	Nomer	Date_
I4	I4	I4	I4	I4	D8
Дата з	Дата по				
Date_From	Date_To				
D8	D8				

Таблиця 3.10

Логічна структура таблиці бази даних «Обладнання в акті» (Equipment_in_Act)

Код обладнання в акті	Код акту	Код обладнання	Код виду робіт	Кількість
ID_Equipment_in_Act	ID_Act	ID_Equipment	ID_work_type	Amount
I4	I4	I4	I2	I3

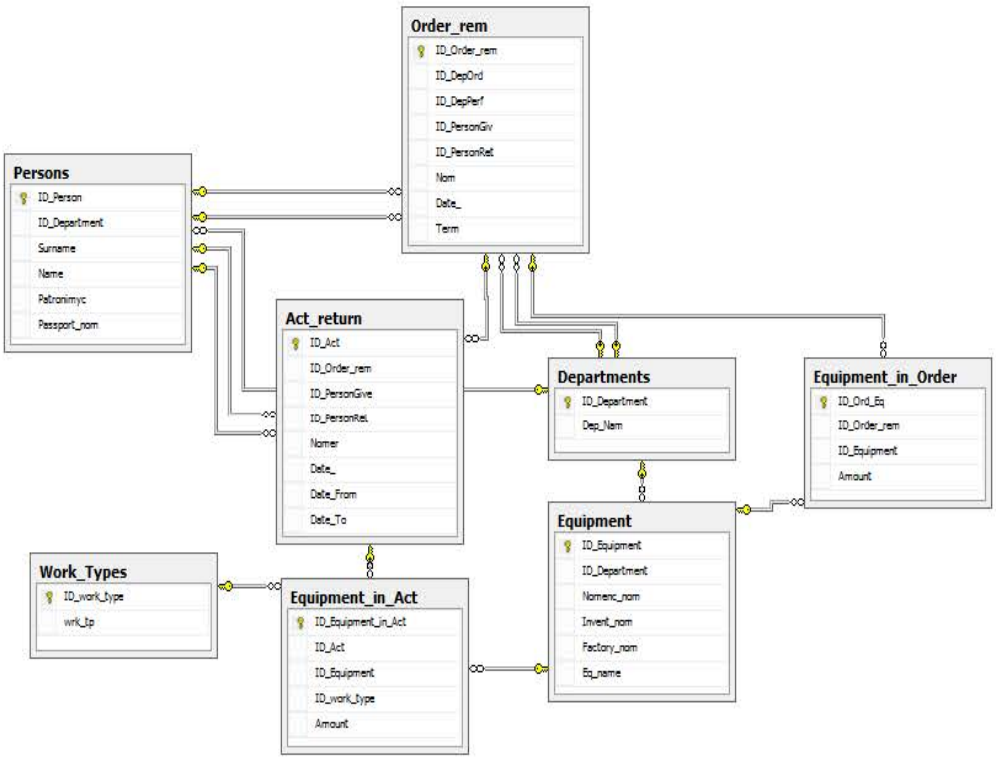

Рисунок 3.3 – Фізична модель системи контролю проведення ремонтів обладнання

3.5 Запити для аналізу даних (Data Analysis Queries)

SQL-запити вихідної інформації – це команди, написані на мові структурованих запитів (SQL), які використовуються для отримання даних із бази даних на основі певних умов або критеріїв. Ці запити зазвичай використовуються для генерації звітів, аналізу даних, моніторингу стану системи та прийняття рішень.

1) Представлення «Контроль виконання ремонту».

SQL-запит для формування вибірки даних на основі представлення «Контроль виконання ремонту»

```
SELECT Equipment.Eq_name, SUM(Equipment_in_Order.Amount) AS NEED, SUM(Equipment_in_Act.Amount) AS DONE, SUM(Equipment_in_Order.Amount - Equipment_in_Act.Amount) AS DOLG
FROM Act_return INNER JOIN
Equipment_in_Act ON Act_return.ID_Act = Equipment_in_Act.ID_Act INNER JOIN Equipment ON Equipment_in_Act.ID_Equipment = Equipment.ID_Equipment INNER JOIN Equipment_in_Order ON Equipment.ID_Equipment = Equipment_in_Order.ID_Equipment INNER JOIN
Order_rem ON Act_return.ID_Order_rem = Order_rem.ID_Order_rem AND Equipment_in_Order.ID_Order_rem = Order_rem.ID_Order_rem
GROUP BY Equipment.Eq_name
```

2) Представлення «Кількість відремонтованого обладнання».

SQL-запит для формування вибірки даних на основі представлення «Кількість відремонтованого обладнання»

```
SELECT Equipment.Eq_name, SUM(Equipment_in_Act.Amount) AS DONE
FROM Act_return INNER JOIN
Equipment_in_Act ON Act_return.ID_Act = Equipment_in_Act.ID_Act INNER JOIN
```

```
                Equipment ON Equipment_in_Act.ID_Equipment =
Equipment.ID_Equipment
    WHERE (Act_return.Date_ BETWEEN @FROM AND @TO)
    GROUP BY Equipment.Eq_name
```

3.6 Користувацький інтерфейс (User Interface, UI)

Інтерфейсна частина – це компонент системи, що забезпечує взаємодію між користувачами і функціональністю системи. Вона включає всі елементи, з якими користувачі взаємодіють безпосередньо, і служить для представлення даних та управління системою в зручній для користувача формі.

Основні елементи інтерфейсної частини АІС:

1) Графічний інтерфейс користувача (GUI):

– Візуальні елементи, такі як кнопки, меню, форми, діалогові вікна.

– Відображення даних у вигляді таблиць, графіків, діаграм.

– Забезпечення зворотного зв'язку для користувача (наприклад, повідомлення про помилки, підтвердження дій).

2) Форми введення даних:

– Елементи для введення даних, такі як текстові поля, випадаючі списки, перемикачі.

– Валідація введених даних для забезпечення їхньої коректності.

3) Меню навігації:

– Структуровані списки для переміщення між різними розділами та функціями системи.

– Може включати головне меню, бокові панелі навігації, контекстні меню.

4) Інформаційні панелі (Dashboards):

– Зведені екрани, які відображають ключові показники, статуси та звіти.

– Використовуються для моніторингу та аналізу даних в режимі реального часу.

5) Звіти та результати запитів:

– Відображення результатів SQL-запитів у зрозумілому вигляді.

– Можливість експорту звітів у різні формати (PDF, Excel).

6) Контрольні панелі та налаштування:

– Інструменти для конфігурації системи, управління користувачами, налаштування прав доступу.

– Панелі адміністрування для моніторингу системи та управління її роботою.

В рамках предметної області визначено 4 типи користувачів із різним рівнем доступу до даних.

1) Адміністратор (А) – особа з найвищим рівнем доступу до даних, якій надано можливість додавати, видаляти, редагувати та шукати дані з БД.

2) Оператор довідників (О) – особа з середнім рівнем доступу до даних, якій надано можливість здійснювати додавання, редагування та виалення довідників БД.

3) Оператор документів (Д) – особа з середнім рівнем доступу до даних, якій надано можливість здійснювати додавання, редагування та виалення документів БД.

4) Оператор звітів (З) – особа з низьким рівнем доступу до даних, якій надано лише можливість вибірок данних по кількості відремонтованого обладнання.

Операції з даними наведено в табл. 3.11.

Таблиця 3.11

Визначення прав доступу до даних різних користувачів

Програмна назва сторінки	Опис функціонального призначення сторінки	Доступ для користувачів			
		А	О	Д	З
Form1.cs	Авторизація	+	+	+	+
DepartmentsForm.cs	Перегляд	+	+	-	-

	Редагування	+	+	-	-
	Видалення записів	+	+	-	-
	Пошук	+	+	-	-
PersonsForm.cs	Перегляд	+	+	-	-
	Редагування	+	+	-	-
	Видалення	+	+	-	-
	Пошук	+	+	-	-
EquipmentForm.cs	Перегляд	+	+	-	-
	Редагування	+	+	-	-
	Видалення	+	+	-	-
	Пошук	+	+	-	-
WorkTypesForm.cs	Перегляд	+	+	-	-
	Редагування	+	+	-	-
	Видалення	+	+	-	-
CreateOrderForm.cs	Перегляд	+	-	+	-
	Додавання	+	-	+	-
	Редагування	+	-	+	-
	Пошук	+	-	+	-
OrderForm.cs	Перегляд	+	-	+	-
	Додавання	+	-	+	-
	Редагування	+	-	+	-
	Пошук	+	-	+	-
CreateActForm.cs	Перегляд	+	-	+	-
	Додавання	+	-	+	-
	Редагування	+	-	+	-
	Пошук	+	-	+	-
ActForm.cs	Перегляд	+	-	+	-
	Додавання	+	-	+	-
	Редагування	+	-	+	-
	Пошук	+	-	+	-
ControlForm.cs	Перегляд	+	-	-	+
	Пошук	+	-	-	+
ReportCreatingForm.cs	Пошук	+	-	-	+
	Перегляд	+	-	-	+
ReportForm.cs	Пошук	+	-	-	+

3.6 Графічні форми

Екранні форми в автоматизованій інформаційній системі виконують важливі функції, які сприяють ефективному введенню, обробці та відображенню інформації. Вони забезпечують зручний і інтуїтивно зрозумілий спосіб взаємодії користувача із системою. Ось кілька ключових причин, для чого використовуються екранні форми:

Основні функції екранних форм

– введення даних;

– валідація даних;

– відображення інформації;

– навігація по системі;

– формування та подання запитів.

Дизайн інтерфейсу для вихідних даних (Output Data Interface Design) показаний в таблиці 3.12.

Таблиця 3.12

Дизайн інтерфейсу для вихідних даних (Output Data Interface Design)

Графічна форма	Опис
Form1.cs	Графічна форма авторизації користувача
Form1.cs	Головна форма АІС
DepartmentsForm.cs	Графічна форма довідника «Організаційні одиниці підприємства»
PersonsForm.cs	Графічна форма довідника «Персони»
EquipmentForm.cs	Графічна форма довідника «Обладнання»
WorkTypesForm.cs	Графічна форма довідника «Види робіт»
CreateOrderForm.cs	Графічна форма створення документу «Замовлення на ремонт або заточування інструментів/приладів»
OrderForm.cs	Графічна форма пошуку, перегляду та графічного відображення документу «Замовлення на ремонт або заточування інструментів/приладів»
CreateActForm.cs	Графічна форма створення документу «Акт приймання-здачі відремонтованих, реконструйованих та модернізованих об’єктів»
ActForm.cs	Графічна форма пошуку, перегляду та графічного відображення документу «Акт приймання-здачі відремонтованих, реконструйованих та модернізованих об’єктів».
ControlForm.cs	Графічна форма перегляду контролю за виконанням

	ремонтних робіт.
ReportCreatingForm.cs	Графічна форма створення звіту по кількості відремонтованого обладнання.
ReportForm.cs	Графічна форма звіту по кількості відремонтованого обладнання.

Графічні форми нормативно-довідникової інформації представлено на рис. 3.3 – рис. 3.8. Графічна форма вихідної інформації представлена на рис. 3.9 – 4.13. Для додавання, оновлення, видалення та пошуку даних із складових частин довідникової або вхідної інформації користувачам необхідно вибрати відповідний пункт головного меню системи (рис. 3.2).

Рисунок 3.4 – Головна графічна форма

Графічні форми довідників одиниць підприємства, персон, обладнання та видів робіт представлено на рис. 3.5 – рис. 3.8. Діалог між користувачем та системою має аналогічну схему в рамках кожної графічної форми і складається з наступних дій:

1) для додавання нового значення до бази даних необхідно ввести у текстові поля, розташовані на екранних формах відповідні значення та натиснути кнопку «Додати» внесені дані будуть відображення в табличній

частині форми.

2) для видалення значення з бази даних необхідно вибрати в таблиці відповідний запис та натиснути кнопку «Видалити», в результаті користувачеві видається повідомлення про підтвердження операції видалення.

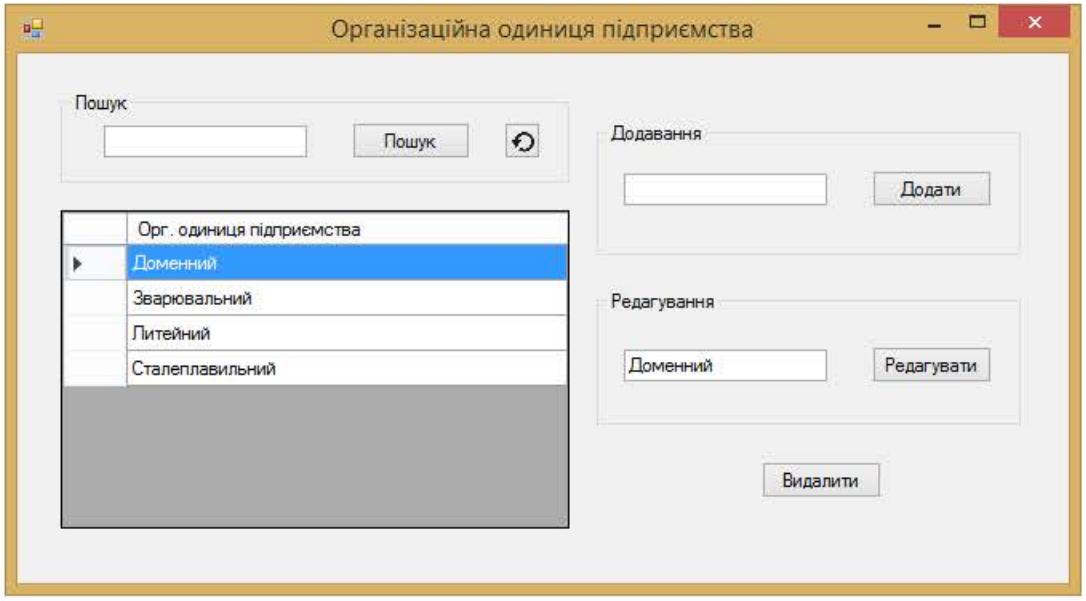

Рисунок 3.5 – Графічна форма довідника «Організаційні одиниці підприємства»

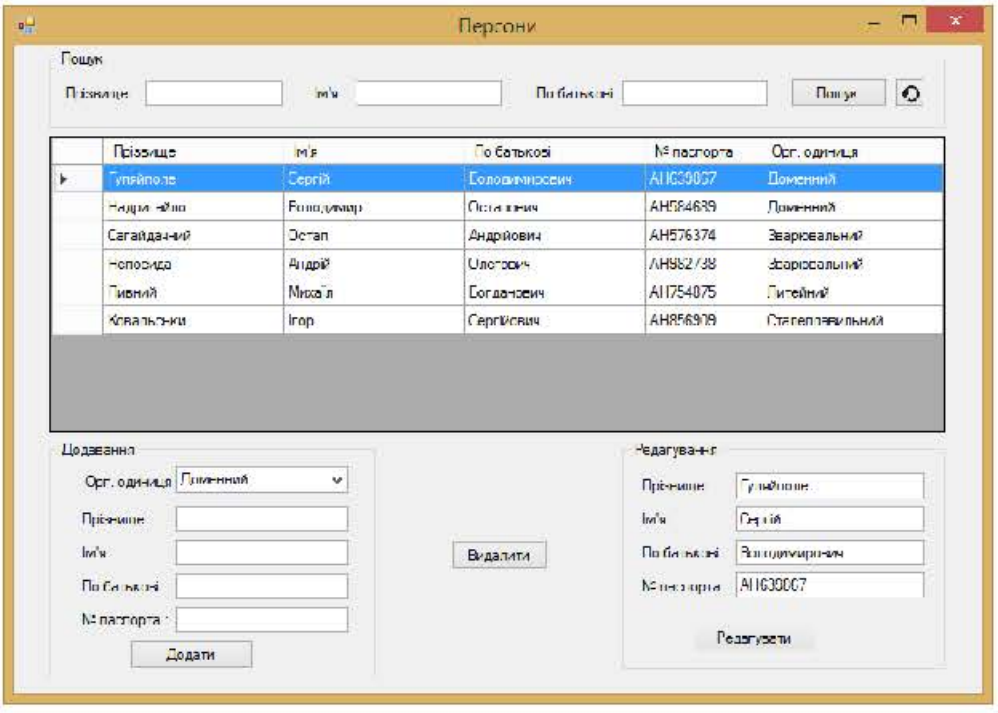

Рисунок 3.6 – Графічна форма довідника «Персони»

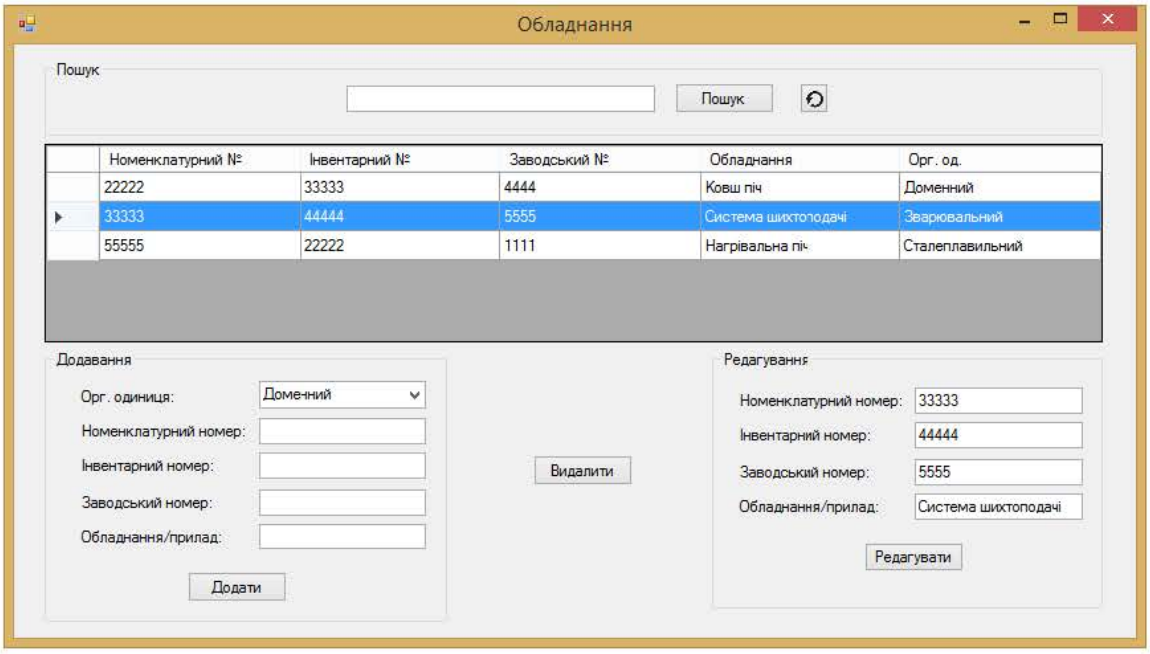

Рисунок 3.7 – Графічна форма довідника «Обладнання»

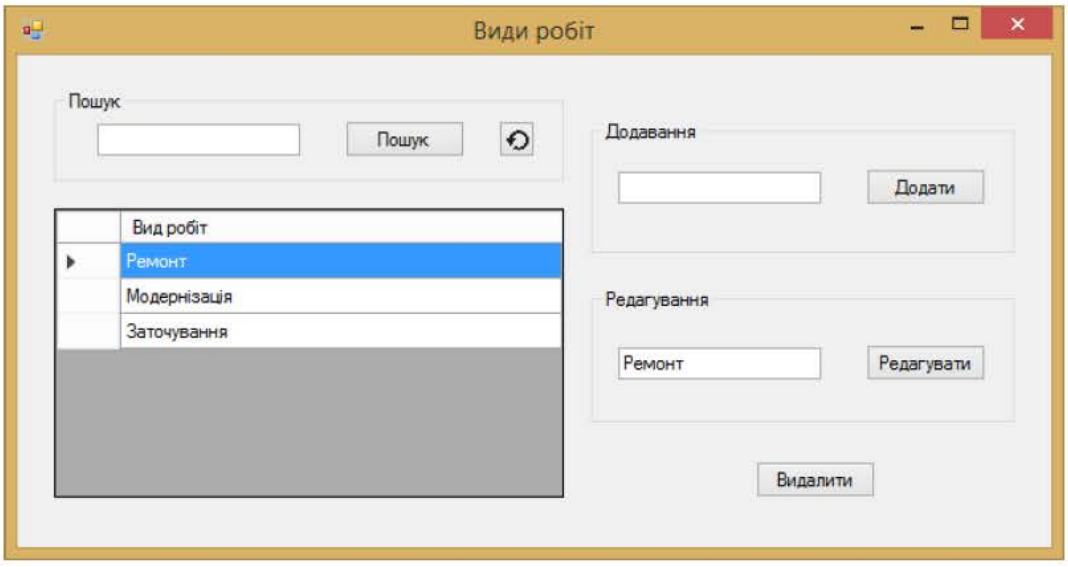

Рисунок 3.8 – Графічна форма довідника «Види робіт»

Для пошуку запису необхідно задати відповідний параметр запиту до БД та натиснути кнопку «Пошук», в результаті в тій самій таблиці буде відображено результат пошуку.

Графічну форму вхідної інформації системи створено з урахуванням вимог уніфікації та стандартизації. Діалог між користувачем та системою має

аналогічну схему в рамках кожної графічної форми.

1) Для видалення документів, необхідно у відповідній табличній частині графічної форми обрати відповідний документ, що необхідно видалити, та натиснути кнопку «Видалити».

2) Для здійснення пошуку документу за датою його створення із всіх збережених в базі даних користувачеві необхідно вибрати необхідну дату або номер, або обидва і натиснути кнопку «Пошук».

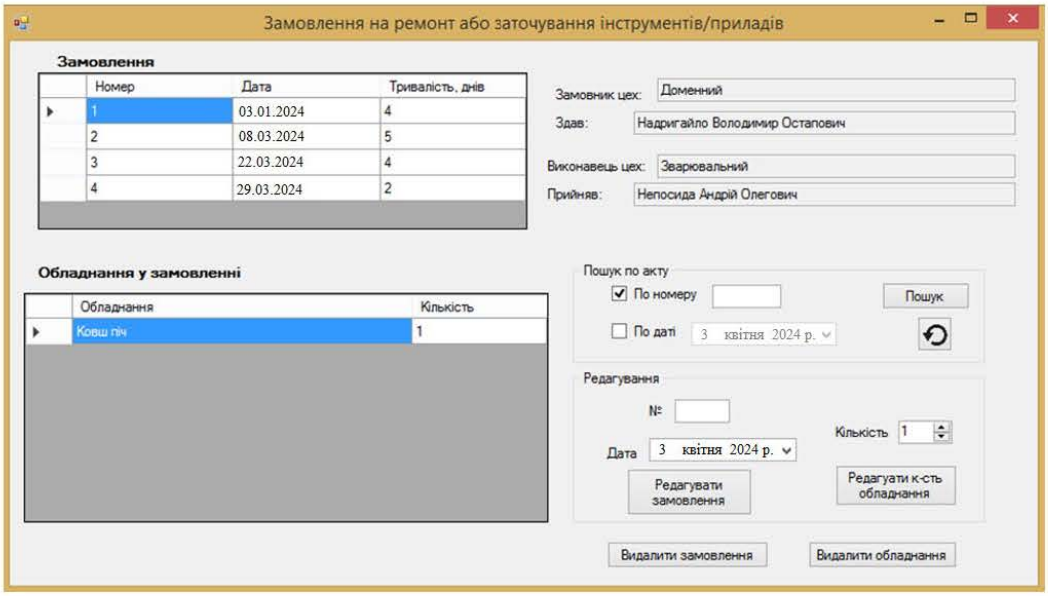

Рисунок 3.9 – Графічна форма документу «Замовлення на ремонт або заточування інструментів/приладів»

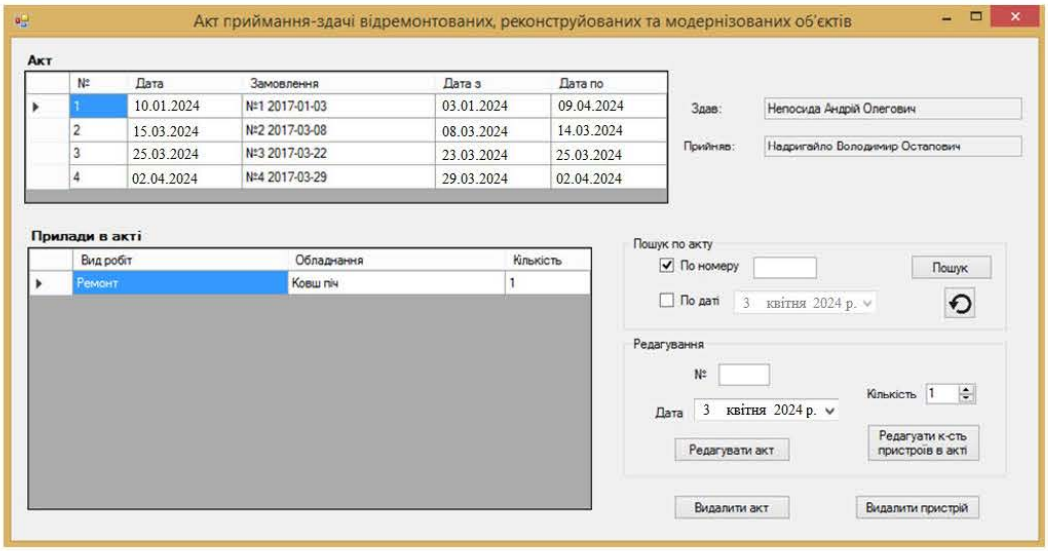

Рисунок 3.10 – Графічна форма «Акту приймання-здачі відремонтованих, реконструйованиї та модернізованих об'єктів»

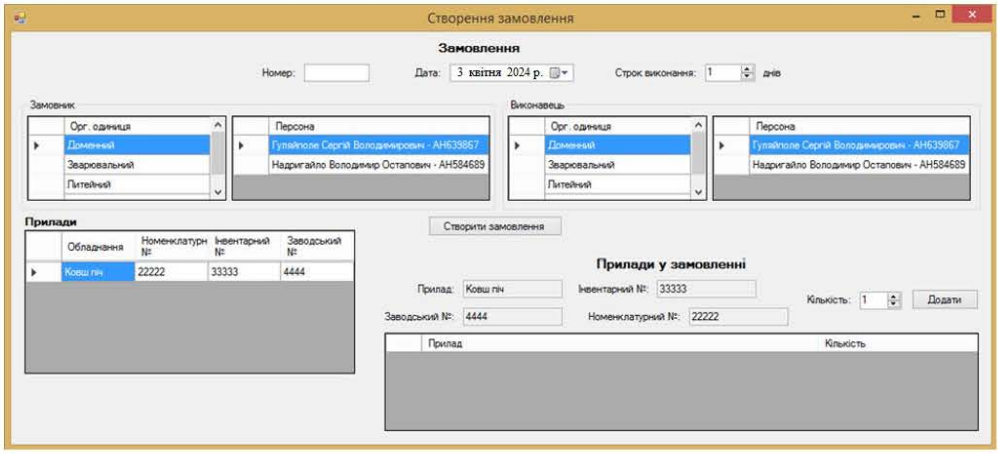

Рисунок 3.11 – Графічна форма документу «Створення замовлення»

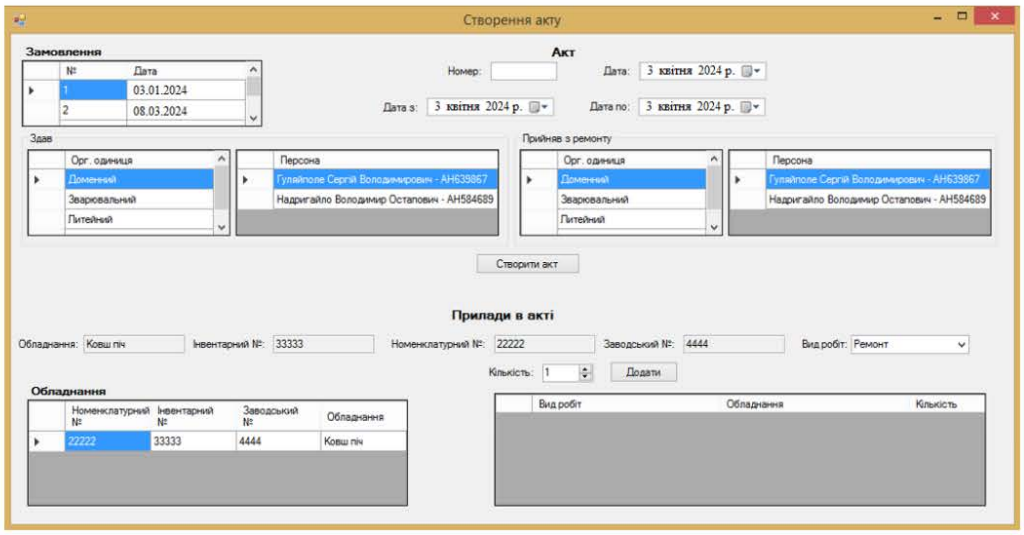

Рисунок 3.12 – Графічна форма документу «Створення акту»

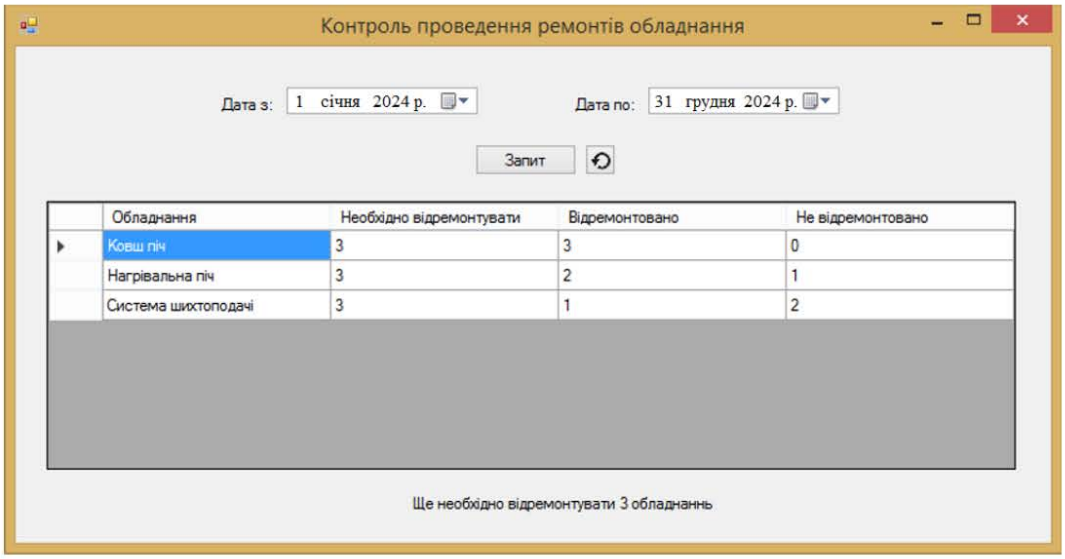

Рисунок 3.13 – Графічна форма «Контроль проведення ремонтів обладнання»

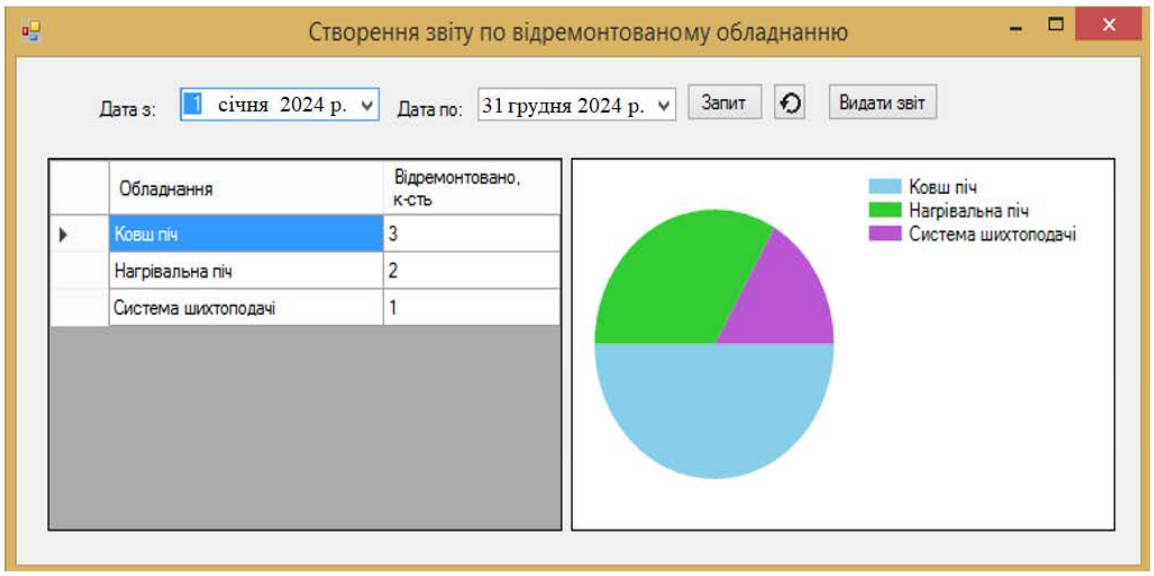

Рисунок 3.14 – Графічна форма «Створеннння звіту по відремонтованому обладнанню»

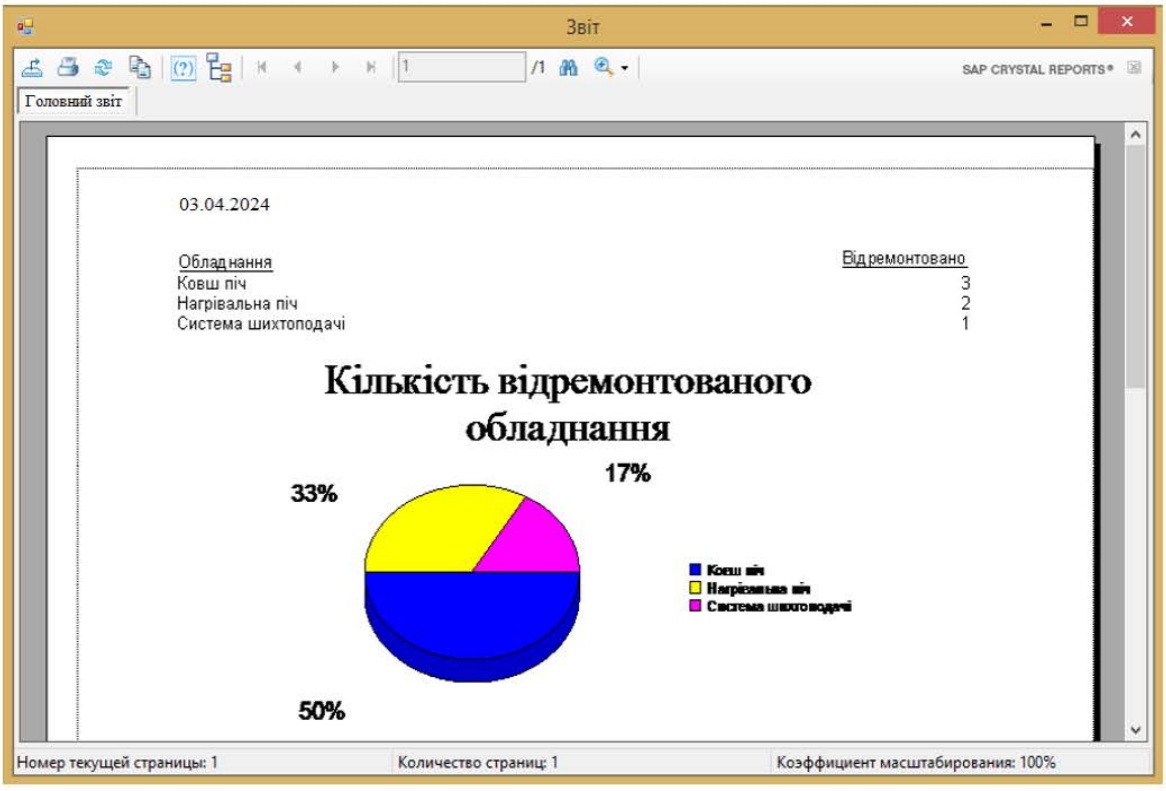

Рисунок 3.15 – Графічна форма «Звіту по відремонтованому обладнанню»

3.7 Висновки до третього розділу

У розділі «Розробка системи контролю проведення ремонтів

обладнання» було проведено комплексну розробку системи, яка забезпечить ефективний та надійний контроль за проведенням ремонтів обладнання на виробничому підприємстві. Розроблені компоненти системи включають в себе:

1) Діаграми потоку даних: Ці діаграми надають візуальне представлення потоку і обробки даних у системі, допомагаючи зрозуміти та оптимізувати процеси обробки і передачі інформації.

2) Функціональна і об'єктна модель: Моделі системи дозволяють чітко визначити функціональні можливості системи та структуру об'єктів, що використовуються у процесі контролю ремонтів обладнання.

3) Організація меню: Правильно організоване меню спрощує навігацію користувачів у системі та дозволяє їм швидко знаходити потрібні функції.

4) Розробка бази даних: База даних системи забезпечує надійне зберігання та організацію даних про обладнання та ремонтні роботи.

5) Запити для аналізу даних: Розроблені запити дозволяють аналізувати дані про ремонтне обладнання та ефективно планувати майбутні ремонтні роботи.

6) Користувацький інтерфейс та графічні форми: Інтерфейс системи надає зручний та інтуїтивно зрозумілий спосіб взаємодії користувачів з системою, а графічні форми допомагають у візуалізації даних та результатів аналізу.

У цілому, розроблена система контролю проведення ремонтів обладнання відповідає вимогам та очікуванням користувачів, забезпечуючи зручну та ефективну роботу з даними та процесами управління ремонтами.

ВИСНОВКИ

Мета цієї роботи полягає в розробці інформаційного та програмного забезпечення для ефективного управління технічним обслуговуванням і ремонтами на виробництві. Це включає створення системи, яка автоматизує процеси планування, виконання та контролю технічного обслуговування і ремонтів, зменшуючи час простою обладнання і підвищуючи загальну ефективність виробничих операцій.

Для досягнення поставленої мети роботи було вирішено наступні завдання:

1) Проведено аналіз предметної області:

– Досліджено поточні бізнес-процеси, пов'язані з технічним обслуговуванням і ремонтами обладнання на виробництві.

– Виявлено основні проблеми і вузькі місця в існуючій системі управління технічним обслуговуванням.

– Ідентифіковані вимоги до нової системи з боку користувачів і зацікавлених сторін.

2) Визначено вимоги до системи:

– функціональні вимоги.

– нефункціональні вимоги.

3) Виконано проектування системи:

– Розроблено функціональну модель системи з використанням відповідних методологій моделювання (IDEF3).

– Створено діаграми, які відображають основні процеси і потоки даних.

4) Проектування бази даних:

– Розроблено логічну модель бази даних для зберігання інформації про обладнання, запити на обслуговування, графіки технічного обслуговування і ремонти.

– Спроектувано фізичну модель бази даних на обраній платформі (MSSQLServer).

5) Розроблено програмного забезпечення:

– Створено інтерфейсну частину системи, яка забезпечує зручний і інтуїтивно зрозумілий доступ до функцій управління технічним обслуговуванням і ремонтами.

– Реалізовано основні функціональних модулі.

5) Виконано тестування системи:

– Проведено функціональне тестування для перевірки відповідності системи вимогам.

– Виконано нефункціональне тестування для оцінки продуктивності, безпеки та зручності використання системи.

Об'єктом дослідження є сама система контролю, яка включає в себе різноманітні компоненти, такі як база даних, користувацький інтерфейс, логіка бізнес-процесів тощо.

Предметом дослідження є розробка та впровадження цієї системи з метою забезпечення ефективного контролю за проведенням ремонтів обладнання на підприємстві.

Актуальність даної роботи полягає в тому, що ефективний контроль проведення ремонтів обладнання є критично важливим для безперебійного функціонування виробничих підприємств. Неплановані зупинки обладнання через неправильно проведені ремонтні роботи можуть призвести до великих втрат у виробництві та фінансових витрат.

Отже, основні аспекти актуальності даної роботи:

1) Ефективність виробництва: виробничі підприємства потребують надійних систем контролю ремонтів обладнання для підтримки оптимального режиму роботи та запобігання втратам продуктивності.

2) Безпека працівників: неналежно проведені ремонтні роботи можуть створити небезпечні умови для працівників. Система контролю ремонтів допомагає забезпечити дотримання норм безпеки та запобігти нещасним випадкам на виробництві.

3) Економічна вигода: ефективний контроль проведення ремонтів

дозволяє економити час та ресурси, зменшуючи кількість непланованих зупинок обладнання та витрат на надмірні ремонтні роботи.

4) Сучасні технології: з розвитком інформаційних технологій та впровадженням систем управління ресурсами, необхідно мати сучасні та ефективні системи контролю ремонтів, які відповідають сучасним стандартам та потребам виробництва.

Отже, розробка системи контролю проведення ремонтів обладнання є актуальною задачею, яка сприятиме підвищенню ефективності виробництва, безпеці працівників та економічній вигоді для підприємства.

Після завершення цієї роботи можна зробити наступні висновки.

1) У результаті дослідження та розробки була створена система, яка забезпечує ефективний контроль за процесом проведення ремонтів обладнання на виробничому підприємстві.

2) Для реалізації системи були використані сучасні технології, такі як .NET Framework, C#, MSSQLServer та ADO.NET, що дозволило створити функціональний та надійний продукт.

3) Впровадження системи контролю ремонтів дозволить підвищити ефективність виробництва шляхом оптимізації процесів управління ремонтами та запобігання непланованим зупинкам обладнання.

4) Система допоможе забезпечити безпечні умови праці для співробітників шляхом контролю якості та правильності проведення ремонтних робіт.

5) Впровадження системи контролю ремонтів також призведе до економічних вигод, зокрема, зменшення витрат на непланові ремонти та підвищення тривалості служби обладнання.

У цілому, дана робота дозволила створити продукт, який відповідає сучасним вимогам виробництва та забезпечить підприємство необхідними інструментами для успішного контролю ремонтних робіт.

Виконання цих завдань дозволило створити систему управління технічним обслуговуванням і ремонтами, що сприятиме підвищенню

надійності та ефективності виробничого обладнання, а також оптимізації процесів управління технічним обслуговуванням.

Кваліфікаційна робота виконана у відповідності до стандарту спеціальності 121 – «Інженерія програмного забезпечення» і демонструє володіння такими компетентностями як:

- Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;

- Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

- Здатність розробляти архітектури, модулі та компоненти програмних систем;

- Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням та стандартами;

- Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу;

- Володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних;

- Здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення;

- Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення;

- Здатність до алгоритмічного та логічного мислення тощо.

Серед результатів навчання, визначених стандартом, кваліфікаційна робота реалізує наступні:

- Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки;

- Вміти розробляти людино-машинний інтерфейс;

- Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення;

- Проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування;

- Вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання;

- Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення;

- Знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних;

- Мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення;

- Знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних;

- Вміти документувати та презентувати результати розробки програмного забезпечення тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Joseph Albahari, Ben Albahari. C# 9.0 in a Nutshell: The Definitive Reference, 2021.

2. Mark J. Price. C# 10 and .NET 6 – Modern Cross-Platform Development, 2021.

3. Andrew Troelsen, Philip Japikse. Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming, 2020.

4. Christian Nagel. Professional C# and .NET - 2021 Edition, 2021.

5. Erik Lippert, Jon Skeet. C# in Depth, 4th Edition, 2019.

6. Itzik Ben-Gan. T-SQL Fundamentals, 4th Edition, 2023.

7. Kalen Delaney. SQL Server 2019 Internals, 2020.

8. Dusan Petkovic. Microsoft SQL Server 2019: A Beginner's Guide, 7th Edition, 2020.

9. Bob Ward. SQL Server 2019 Revealed, 2019.

10. William Durkin, Mikael Wedham, and Rie Irish. SQL Server 2019 Administration Inside Out, 2020.

11. Glenn Johnson. Professional ADO.NET 4.5 with Entity Framework 6, 2019.

12. Bipin Joshi. Beginning Database Programming with C#, 2020.

13. Philip Japikse, Kevin Grossnicklaus, Ben Dewey. Building Web Applications with .NET Core 2.1 and JavaScript: Leveraging Modern JavaScript Frameworks, 2019.

14. Adam Freeman. Pro C# 8 with .NET Core and ADO.NET, 2019.

15. John Paul Mueller, Bill Sempf, Chuck Sphar. C# 8.0 All-in-One For Dum.

ДОДАТОК А

Лістинг створення бази даних

```
create database EquipmentRepair2
create table Departments(ID_Department int primary key identity(1,1),Dep_Nam
nvarchar(100))
create table Persons (ID_Person int primary key identity(1,1), ID_Department int
default 1 foreign key references Departments(ID_Department) ON DELETE SET
DEFAULT, Surname nvarchar(50), Name nvarchar(50), Patronimyc nvarchar(50),
Passport_nom nvarchar(8))
create table Work_Types (ID_work_type int primary key identity(1,1), wrk_tp
nvarchar(70))
create table Equipment(ID_Equipment int primary key identity(1,1),
ID_Department int default 1 foreign key references Departments(ID_Department)
ON DELETE SET DEFAULT, Nomenc_nom nvarchar(11), Invent_nom
nvarchar(6), Factory_nom nvarchar(11), Eq_name nvarchar(70))
create table Order_rem (ID_Order_rem int primary key identity(1,1), ID_DepOrd
int foreign key references Departments(ID_Department), ID_DepPerf int foreign
key references Departments(ID_Department), ID_PersonGiv int foreign key
references Persons(ID_Person), ID_PersonRet int foreign key references
Persons(ID_Person), Nom int, Date_ date, Term int)
create table Equipment_in_Order (ID_Ord_Eq int primary key identity(1,1),
ID_Order_rem int default 1 foreign key references Order_rem(ID_Order_rem) ON
DELETE SET DEFAULT, ID_Equipment int default 1 foreign key references
Equipment(ID_Equipment) ON DELETE SET DEFAULT, Amount int)
create table Act_return (ID_Act int primary key identity(1,1), ID_Order_rem int
default 1 foreign key references Order_rem(ID_Order_rem) ON DELETE SET
DEFAULT, ID_PersonGive int foreign key references Persons(ID_Person),
ID_PersonRet int foreign key references Persons(ID_Person), Nomer int, Date_
date, Date_From date, Date_To date)
create table Equipment_in_Act(ID_Equipment_in_Act int primary key
identity(1,1), ID_Act int foreign key references Act_return(ID_Act),
ID_Equipment int default 1 foreign key references Equipment(ID_Equipment) ON
DELETE SET DEFAULT, ID_work_type int default 1 foreign key references
Work_Types(ID_work_type) ON DELETE SET DEFAULT, Amount int)
```

ДОДАТОК Б

Лістинг створення програмного забезпечення

Form1.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace EquipmentRepair2
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

   private void Form1_Load(object sender, EventArgs e)
      {
         pictureBox1.Visible = false;
      }

      private void DepartmentMenuItem_Click(object sender, EventArgs e)
      {
         DepartmentsForm DepForm = new DepartmentsForm();
         DepForm.Show();
      }

      private void PersonsMenuItem_Click(object sender, EventArgs e)
      {
         PersonsForm RerForm = new PersonsForm();
         RerForm.Show();
      }

      private void EquipmentMenuItem_Click(object sender, EventArgs e)
      {
```

```
    EquipmentForm EqForm = new EquipmentForm();
    EqForm.Show();
}

private void WorkTypesMenuItem_Click(object sender, EventArgs e)
{
    WorkTypesForm WorkForm = new WorkTypesForm();
    WorkForm.Show();
}

private void CreateOrderMenuItem_Click(object sender, EventArgs e)
{
    CreateOrderForm CrOrdForm = new CreateOrderForm();
    CrOrdForm.Show();
}

private void OrderMenuItem_Click(object sender, EventArgs e)
{
    OrderForm OrdForm = new OrderForm();
    OrdForm.Show();
}

private void CreateActMenuItem_Click(object sender, EventArgs e)
{
    CreateActForm CreActForm = new CreateActForm();
    CreActForm.Show();
}

private void ActMenuItem_Click(object sender, EventArgs e)
{
    ActForm ActFrm = new ActForm();
    ActFrm.Show();
}

private void ControlMenuItem_Click(object sender, EventArgs e)
{
    ControlForm CtrlForm = new ControlForm();
    CtrlForm.Show();
}

private void CreateRepornMenuItem_Click(object sender, EventArgs e)
{
    ReportCreatingForm RepCrForm = new ReportCreatingForm();
    RepCrForm.Show();
}
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                string connectionString = "Data Source=idea-pc; User ID=" +
textBox1.Text + "; Password=" + textBox2.Text + ";Initial
Catalog=EquipmentRepair2";

                SqlConnectionStringBuilder builder =
                    new SqlConnectionStringBuilder(connectionString);

                builder.IntegratedSecurity = false;


                using (SqlConnection connection = new
SqlConnection(builder.ConnectionString))
                {
                    connection.Open();
                    pictureBox1.Visible = true;
                    panel1.Visible = false;
                    menuStrip1.Enabled = true;
                    UserManager.MyLogin = textBox1.Text;
            if (UserManager.MyLogin == "GodAdmin"){
довідникиToolStripMenuItem.Enabled = true;
вхіднаІнформаціяToolStripMenuItem.Enabled = true;
вихіднаІнформаціяToolStripMenuItem.Enabled = true; }
        if (UserManager.MyLogin == "CatalogMaster"){
довідникиToolStripMenuItem.Enabled = true;
вхіднаІнформаціяToolStripMenuItem.Enabled = false;
вихіднаІнформаціяToolStripMenuItem.Enabled = false;
                    }
                    if (UserManager.MyLogin == "DocumMaster")
                    {
                        довідникиToolStripMenuItem.Enabled = false;
                        вхіднаІнформаціяToolStripMenuItem.Enabled = true;
                        вихіднаІнформаціяToolStripMenuItem.Enabled = false;
                    }
                    if (UserManager.MyLogin == "PoorReader")
                    {
                        довідникиToolStripMenuItem.Enabled = false;
вхіднаІнформаціяToolStripMenuItem.Enabled = false;
вихіднаІнформаціяToolStripMenuItem.Enabled = true;
                    }
                    connection.Close();
```

```
            }   }
        catch (Exception ex) {
            MessageBox.Show(ex.Message);
        } }}}
```

DepartmentsForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace EquipmentRepair2
{
   public partial class DepartmentsForm : Form
   {
      public DepartmentsForm()
      {
         InitializeComponent();
      }

      private void DepartmentsForm_Load(object sender, EventArgs e)
      {
this.departmentsTableAdapter.SelectDepartments(this.departmentsDataSet.Depart
ments);
      }

      private void buttonAdd_Click(object sender, EventArgs e)
      {
         if (textAdd.Text != "")
         {
            departmentsTableAdapter.AddDepartment(textAdd.Text);
            departmentsTableAdapter.Update(departmentsDataSet.Departments);

departmentsTableAdapter.SelectByLastDepartment(departmentsDataSet.Departme
nts);
            textAdd.Clear();
         }
         else
         {
            MessageBox.Show("Поле для заповнення не повинно бути
```

```
пустим!");
            }
        }

        private void buttonEdit_Click(object sender, EventArgs e)
        {
            string old2 = dataGridView1.CurrentRow.Cells[1].Value.ToString();

            if (dataGridView1.Rows.Count != 0)
            {
                if (textEdit.Text != "" && dataGridView1.Rows.Count != 0)
                {
                    DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете виконати редагування?", "Редагування", MessageBoxButtons.YesNo);
                    if (dialogResult == DialogResult.Yes)
                    {
                        int position;
                        position = this.departmentsBindingSource.Position;
                        departmentsTableAdapter.UpdateDepartment(textEdit.Text,
(int)dataGridView1.CurrentRow.Cells[0].Value,
(int)dataGridView1.CurrentRow.Cells[0].Value);

departmentsTableAdapter.Update(departmentsDataSet.Departments);

departmentsTableAdapter.SelectDepartments(departmentsDataSet.Departments);
                        Refresh();
                        this.departmentsBindingSource.Position = position;
                    }
                    else
                    {
                        textEdit.Text = old2;
                    }
                }
                else
                {
                    textEdit.Text = old2;
                    MessageBox.Show("Усі поля для редагування не повинні бути
пустими!");
                }
            }
            else
            {
                MessageBox.Show("Немає елемента для редагування!");
            }
        }
```

```
        private void buttonDelete_Click(object sender, EventArgs e)
        {
            if (dataGridView1.Rows.Count != 0)
            {
                DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете видалити цех :\"" + dataGridView1.CurrentRow.Cells[1].Value.ToString()
+ "\"?", "Підтвердження", MessageBoxButtons.YesNo);
                if (dialogResult == DialogResult.Yes)
                {
                    int position;
                    position = this.departmentsBindingSource.Position;

                    ChangeThisDepartmentInDocument(position);

                    departmentsDataSet.Departments.Rows[position].Delete();
                    departmentsTableAdapter.Update(departmentsDataSet.Departments);
                    departmentsDataSet.Departments.AcceptChanges();
                }
            }
            else
            {
                MessageBox.Show("Немає записів для видалення!");
            }
        }

        public void ChangeThisDepartmentInDocument(int key)
        {
            string command4 = "update Order_rem set ID_DepOrd = 1 where
ID_DepOrd = @Pa4";
            string command5 = "update Order_rem set ID_DepPerf = 1 where
ID_DepPerf = @Pa5";

            sqlConnection1.Open();
            SqlCommand com_Sql4 = new SqlCommand(); SqlCommand com_Sql5 =
new SqlCommand();
            com_Sql4.Connection = sqlConnection1; com_Sql5.Connection =
sqlConnection1;
            com_Sql4.CommandType = CommandType.Text;
com_Sql5.CommandType = CommandType.Text;
            com_Sql4.CommandText = command4; com_Sql5.CommandText =
command5;
            com_Sql4.Parameters.AddWithValue("@Pa4", key);
            com_Sql5.Parameters.AddWithValue("@Pa5", key);
            com_Sql4.ExecuteNonQuery(); com_Sql5.ExecuteNonQuery();
```

```
            sqlConnection1.Close();
        }

        private void buttonSearch_Click(object sender, EventArgs e)
        {

departmentsTableAdapter.SearchDepartment(departmentsDataSet.Departments,
"%" + textSearch.Text + "%");
        }

        private void buttonRewind_Click(object sender, EventArgs e)
        {

this.departmentsTableAdapter.SelectDepartments(this.departmentsDataSet.Depart
ments);
        }

        private void textAdd_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!Char.IsLetter(e.KeyChar) && e.KeyChar != (char)Keys.Back &&
(e.KeyChar != '\'') && (e.KeyChar != '\"') && (e.KeyChar != '.') && (e.KeyChar
!= ',') && (e.KeyChar != '-') && (e.KeyChar != ' '))
            {
                e.Handled = true;
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textEdit.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        }
    }
}
```

PersonsForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Text.RegularExpressions;
```

```
using System.Data.SqlClient;

namespace EquipmentRepair2
{
   public partial class PersonsForm : Form
   {
      public PersonsForm()
      {
         InitializeComponent();
      }

      private void PersonsForm_Load(object sender, EventArgs e)
      {
         this.personsTableAdapter.SelectPersons(this.personsDataSet.Persons);
FillMyDepartments();
      }

      public void FillMyDepartments()
      {
         string command = "SELECT ID_Department, Dep_Nam FROM
Departments WHERE ID_Department <> 1";
         System.Data.SqlClient.SqlDataAdapter da = new
SqlDataAdapter(command, sqlConnection1);

         DataSet ds = new DataSet();
         sqlConnection1.Open();
         da.Fill(ds);
         sqlConnection1.Close();

         comboBoxDepartment.DataSource = ds.Tables[0];
         comboBoxDepartment.DisplayMember = "Dep_Nam";
         comboBoxDepartment.ValueMember = "ID_Department";
      }

      private void ButtonAdd_Click(object sender, EventArgs e)
      {
         Regex pattern = new Regex("[A-Z]{2}[0-9]{6}");

         if (textNameAdd.Text != "" && textSurAdd.Text != "" &&
textPadreAdd.Text != "" && textPasportAdd.Text != "")
         {
            if (!pattern.IsMatch(textPasportAdd.Text))
            {
               MessageBox.Show("Неправильний формат! \nПриклад:
AH123456");
```

```
            textPasportAdd.Clear();
        }
        else
        {

personsTableAdapter.AddPerson(Convert.ToInt32(comboBoxDepartment.Selected
Value), textSurAdd.Text, textNameAdd.Text, textPadreAdd.Text,
textPasportAdd.Text);
            personsTableAdapter.Update(personsDataSet.Persons);
            personsTableAdapter.SelectLastPerson(personsDataSet.Persons);
            textNameAdd.Clear();
            textSurAdd.Clear();
            textPadreAdd.Clear();
            textPasportAdd.Clear();
        }
      }
      else
      {
        MessageBox.Show("Усі поля для заповнення не повинні бути
пустими!");
      }
    }

    private void buttonUpdate_Click(object sender, EventArgs e)
    {
      Regex pattern = new Regex("[A-Z]{2}[0-9]{6}");
      string old = dataGridView1.CurrentRow.Cells[2].Value.ToString();
      string old2 = dataGridView1.CurrentRow.Cells[3].Value.ToString();
      string old3 = dataGridView1.CurrentRow.Cells[4].Value.ToString();
      string old4 = dataGridView1.CurrentRow.Cells[5].Value.ToString();

      if (dataGridView1.Rows.Count != 0)
      {
        if (textSurUp.Text != "" && textNamUp.Text != "" &&
textPadreUp.Text != "" && textPassUp.Text != "" &&
dataGridView1.Rows.Count != 0)
        {
          if (!pattern.IsMatch(textPasportAdd.Text))
          {
            MessageBox.Show("Неправильний формат! \nПриклад:
AH123456");
            textSurUp.Text = old;
            textNamUp.Text = old2;
            textPadreUp.Text = old3;
            textPassUp.Text = old4;
```

```
                }
                else
                {
                    DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете виконати редагування?", "Редагування", MessageBoxButtons.YesNo);
                    if (dialogResult == DialogResult.Yes)
                    {
                        int position;
                        position = this.personsBindingSource.Position;
                        personsTableAdapter.UpdatePerson(textSurUp.Text,
textNamUp.Text, textPadreUp.Text, textPassUp.Text,
(int)dataGridView1.CurrentRow.Cells[0].Value,
(int)dataGridView1.CurrentRow.Cells[0].Value);
                        personsTableAdapter.Update(personsDataSet.Persons);
                        personsTableAdapter.SelectPersons(personsDataSet.Persons);
                        Refresh();
                        this.personsBindingSource.Position = position;
                    }
                    else
                    {
                        textSurUp.Text = old;
                        textNamUp.Text = old2;
                        textPadreUp.Text = old3;
                        textPassUp.Text = old4;
                    }
                }
            }
            else
            {
                textSurUp.Text = old;
                textNamUp.Text = old2;
                textPadreUp.Text = old3;
                textPassUp.Text = old4;
                MessageBox.Show("Усі поля для редагування не повинні бути
пустими!");
            }
        }
        else
        {
            MessageBox.Show("Немає елемента для редагування!");
        }
    }

    private void buttonDelete_Click(object sender, EventArgs e)
    {
```

```
        if (dataGridView1.Rows.Count != 0)
        {
            if (MessageBox.Show("Ви впевнені, що хочете видалити цю
персону?", "Підтвердження",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
            {

ChangeThisPersonInDocuments((int)dataGridView1.CurrentRow.Cells[0].Value);


personsTableAdapter.DeletePerson((int)dataGridView1.CurrentRow.Cells[0].Valu
e);
                personsTableAdapter.Update(personsDataSet.Persons);
                personsTableAdapter.SelectPersons(personsDataSet.Persons);

            }
        }
        else
        {
            MessageBox.Show("Немає записів для видалення!");
        }
    }

    public void ChangeThisPersonInDocuments(int key)
    {
        string command2 = "update Act_return set ID_PersonGive = 1 where
ID_PersonGive = @Pa2";
        string command3 = "update Act_return set ID_PersonRet = 1 where
ID_PersonRet = @Pa3";
        string command6 = "update Order_rem set ID_PersonGiv = 1 where
ID_PersonGiv = @Pa6";
        string command7 = "update Order_rem set ID_PersonRet = 1 where
ID_PersonRet = @Pa7";

        sqlConnection1.Open();
        SqlCommand com_Sql2 = new SqlCommand();  SqlCommand com_Sql3 =
new SqlCommand();
        SqlCommand com_Sql6 = new SqlCommand();  SqlCommand com_Sql7 =
new SqlCommand();

        com_Sql2.Connection = sqlConnection1; com_Sql3.Connection =
sqlConnection1;
        com_Sql6.Connection = sqlConnection1; com_Sql7.Connection =
sqlConnection1;
```

```
            com_Sql2.CommandType = CommandType.Text;
com_Sql3.CommandType = CommandType.Text;
            com_Sql6.CommandType = CommandType.Text;
com_Sql7.CommandType = CommandType.Text;

            com_Sql2.CommandText = command2;  com_Sql3.CommandText =
command3;
            com_Sql6.CommandText = command6;  com_Sql7.CommandText =
command7;

            com_Sql2.Parameters.AddWithValue("@Pa2", key);
            com_Sql3.Parameters.AddWithValue("@Pa3", key);
            com_Sql6.Parameters.AddWithValue("@Pa6", key);
            com_Sql7.Parameters.AddWithValue("@Pa7", key);

            com_Sql2.ExecuteNonQuery(); com_Sql3.ExecuteNonQuery();
            com_Sql6.ExecuteNonQuery(); com_Sql7.ExecuteNonQuery();

            sqlConnection1.Close();
        }

        private void buttonSearch_Click(object sender, EventArgs e)
        {
            personsTableAdapter.SearchPersons(personsDataSet.Persons, "%" +
textSearchSur.Text + "%", "%" + textSearchName.Text + "%", "%" +
textSearchPadre.Text + "%");
        }

        private void buttonRewind_Click(object sender, EventArgs e)
        {
            this.personsTableAdapter.SelectPersons(this.personsDataSet.Persons);
        }

        private void textSurAdd_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!Char.IsLetter(e.KeyChar) && e.KeyChar != (char)Keys.Back &&
(e.KeyChar != '\''))
            {
                e.Handled = true;
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
```

```
        {
            textSurUp.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();
            textNamUp.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();
            textPadreUp.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString();
            textPassUp.Text = dataGridView1.CurrentRow.Cells[5].Value.ToString();
        }
    }
}
```

WorkTypesForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EquipmentRepair2
{
    public partial class WorkTypesForm : Form
    {
        public WorkTypesForm()
        {
            InitializeComponent();
        }

        private void WorkTypesForm_Load(object sender, EventArgs e)
        {

this.work_TypesTableAdapter.SelectWork(this.workDataSet.Work_Types);

            if (dataGridView1.Rows.Count != 0)
            {
                textEdit.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
            }
        }

        private void buttonAdd_Click(object sender, EventArgs e)
        {
            if (textAdd.Text != "")
```

```
            {
                work_TypesTableAdapter.AddWork(textAdd.Text);
                work_TypesTableAdapter.Update(workDataSet.Work_Types);
                work_TypesTableAdapter.SelectLastWork(workDataSet.Work_Types);
                textAdd.Clear();
            }
            else
            {
                MessageBox.Show("Поле для заповнення не повинно бути
пустим!");
            }
        }

        private void buttonEdit_Click(object sender, EventArgs e)
        {
            string old2 = dataGridView1.CurrentRow.Cells[1].Value.ToString();

            if (dataGridView1.Rows.Count != 0)
            {
                if (textEdit.Text != "" && dataGridView1.Rows.Count != 0)
                {
                    DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете виконати редагування?", "Редагування", MessageBoxButtons.YesNo);
                    if (dialogResult == DialogResult.Yes)
                    {
                        int position;
                        position = this.workTypesBindingSource.Position;
                        work_TypesTableAdapter.UpdateWork(textEdit.Text,
(int)dataGridView1.CurrentRow.Cells[0].Value,
(int)dataGridView1.CurrentRow.Cells[0].Value);
                        work_TypesTableAdapter.Update(workDataSet.Work_Types);
                        work_TypesTableAdapter.SelectWork(workDataSet.Work_Types);
                        Refresh();
                        this.workTypesBindingSource.Position = position;
                    }
                    else
                    {
                        textEdit.Text = old2;
                    }
                }
                else
                {
                    textEdit.Text = old2;
                    MessageBox.Show("Усі поля для редагування не повинні бути
пустими!");
```

```
            }
        }
        else
        {
            MessageBox.Show("Немає елемента для редагування!");
        }
    }

    private void buttonDelete_Click(object sender, EventArgs e)
    {
        if (dataGridView1.Rows.Count != 0)
        {
            DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете видалити вид роботи :\"" +
dataGridView1.CurrentRow.Cells[1].Value.ToString() + "\"?", "Підтвердження",
MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                int position;
                position = this.workTypesBindingSource.Position;
                workDataSet.Work_Types.Rows[position].Delete();
                work_TypesTableAdapter.Update(workDataSet.Work_Types);
                workDataSet.Work_Types.AcceptChanges();
            }
        }
        else
        {
            MessageBox.Show("Немає записів для видалення!");
        }
    }

    private void buttonSearch_Click(object sender, EventArgs e)
    {
        work_TypesTableAdapter.SearchWork(workDataSet.Work_Types,
textSearch.Text);
    }

    private void buttonRewind_Click(object sender, EventArgs e)
    {

this.work_TypesTableAdapter.SelectWork(this.workDataSet.Work_Types);
    }

    private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
```

```
        {
            textEdit.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        }

        private void textAdd_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!Char.IsLetter(e.KeyChar) && e.KeyChar != (char)Keys.Back &&
(e.KeyChar != '\'') && (e.KeyChar != '\"') && (e.KeyChar != '.') && (e.KeyChar
!= ',') && (e.KeyChar != '-') && (e.KeyChar != ' '))
            {
                e.Handled = true;
            }
        }
    }
}
```

EquipmentForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Text.RegularExpressions;
using System.Data.SqlClient;

namespace EquipmentRepair2
{
    public partial class EquipmentForm : Form
    {
        public EquipmentForm()
        {
            InitializeComponent();
        }

        private void EquipmentForm_Load(object sender, EventArgs e)
        {
            // TODO: дана строка коду дозволяє завантажити дані в таблицю
"equipmentDataSet.Equipment". При необхідності вона може бути переміщена
або видалена.

this.equipmentTableAdapter.SelectEquipment(this.equipmentDataSet.Equipment);
```

```
            FillMyDepartments();
        }

        public void FillMyDepartments()
        {
            string command = "SELECT ID_Department, Dep_Nam FROM
Departments WHERE ID_Department <> 1";
            System.Data.SqlClient.SqlDataAdapter da = new
SqlDataAdapter(command, sqlConnection1);

            DataSet ds = new DataSet();
            sqlConnection1.Open();
            da.Fill(ds);
            sqlConnection1.Close();

            comboBoxDepartment.DataSource = ds.Tables[0];
            comboBoxDepartment.DisplayMember = "Dep_Nam";
            comboBoxDepartment.ValueMember = "ID_Department";
        }

        private void buttonAdd_Click(object sender, EventArgs e)
        {
            if (comboBoxDepartment.Items.Count != 0)
            {
                if (textAddNomenk.Text != "" && textAddInvent.Text != "" &&
textAddFactor.Text != "" && textAddEquip.Text != "")
                {

equipmentTableAdapter.AddEquipment(Convert.ToInt32(comboBoxDepartment.S
electedValue) ,textAddNomenk.Text, textAddInvent.Text, textAddFactor.Text,
textAddEquip.Text);
                    equipmentTableAdapter.Update(equipmentDataSet.Equipment);

equipmentTableAdapter.SelectLastEquipment(equipmentDataSet.Equipment);
                    textAddNomenk.Clear();
                    textAddInvent.Clear();
                    textAddFactor.Clear();
                    textAddEquip.Clear();
                }
                else
                {
                    MessageBox.Show("Усі поля для заповнення не повинні бути
пустими!");
                }
            }
```

```
            else
            {
                MessageBox.Show("Не має цехів!");
            }
        }

        private void buttonEdit_Click(object sender, EventArgs e)
        {
            string old = dataGridView1.CurrentRow.Cells[2].Value.ToString();
            string old2 = dataGridView1.CurrentRow.Cells[3].Value.ToString();
            string old3 = dataGridView1.CurrentRow.Cells[4].Value.ToString();
            string old4 = dataGridView1.CurrentRow.Cells[5].Value.ToString();

            if (dataGridView1.Rows.Count != 0)
            {
                if (textEditNomenc.Text != "" && textEditInvent.Text != "" &&
textEditFactor.Text != "" && textEditEquip.Text != "" &&
dataGridView1.Rows.Count != 0)
                {
                    DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете виконати редагування?", "Редагування", MessageBoxButtons.YesNo);
                    if (dialogResult == DialogResult.Yes)
                    {
                        int position;
                        position = this.equipmentBindingSource.Position;
                        equipmentTableAdapter.UpdateEquipment(textEditNomenc.Text,
textEditInvent.Text, textEditFactor.Text, textEditEquip.Text,
(int)dataGridView1.CurrentRow.Cells[0].Value,
(int)dataGridView1.CurrentRow.Cells[0].Value);
                        equipmentTableAdapter.Update(equipmentDataSet.Equipment);

equipmentTableAdapter.SelectEquipment(equipmentDataSet.Equipment);
                        Refresh();
                        this.equipmentBindingSource.Position = position;
                    }
                    else
                    {
                        textEditNomenc.Text = old;
                        textEditInvent.Text = old2;
                        textEditFactor.Text = old3;
                        textEditEquip.Text = old4;
                    }
                }
                else
                {
```

```
                textEditNomenc.Text = old;
                textEditInvent.Text = old2;
                textEditFactor.Text = old3;
                textEditEquip.Text = old4;
                MessageBox.Show("Усі поля для редагування не повинні бути
пустими!");
            }
        }
        else
        {
            MessageBox.Show("Немає елемента для редагування!");
        }
    }

    private void buttonDelete_Click(object sender, EventArgs e)
    {
        if (dataGridView1.Rows.Count != 0)
        {
            if (MessageBox.Show("Ви впевнені, що хочете пидалити це
обладнання?", "Підтвердження",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
            {
                int position;
                position = this.equipmentBindingSource.Position;
                equipmentDataSet.Equipment.Rows[position].Delete();
                equipmentTableAdapter.Update(equipmentDataSet.Equipment);
                equipmentDataSet.Equipment.AcceptChanges();

            }
        }
    }

    private void buttonSearch_Click(object sender, EventArgs e)
    {
        equipmentTableAdapter.SearchEquipment(equipmentDataSet.Equipment,
textSearch.Text);
    }

    private void buttonRewind_Click(object sender, EventArgs e)
    {

this.equipmentTableAdapter.SelectEquipment(this.equipmentDataSet.Equipment);
    }
```

```
        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textEditNomenc.Text =
dataGridView1.CurrentRow.Cells[2].Value.ToString();
            textEditInvent.Text =
dataGridView1.CurrentRow.Cells[3].Value.ToString();
            textEditFactor.Text =
dataGridView1.CurrentRow.Cells[4].Value.ToString();
            textEditEquip.Text =
dataGridView1.CurrentRow.Cells[5].Value.ToString();
        }

        private void textAddNomenk_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!Char.IsLetter(e.KeyChar) && !char.IsDigit(e.KeyChar) &&
(e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
            }
        }

        private void textAddEquip_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!Char.IsLetter(e.KeyChar) && e.KeyChar != (char)Keys.Back &&
(e.KeyChar != ' '))
            {
                e.Handled = true;
            }}}}
```

CreateOrderForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Text.RegularExpressions;
using System.Data.SqlClient;


namespace EquipmentRepair2
{
```

```
public partial class CreateOrderForm : Form
{
    public int kol;
    public CreateOrderForm()
    {
        InitializeComponent();
        kol = 0;
    }

    private void CreateOrderForm_Load(object sender, EventArgs e)
    {

this.departmentsTableAdapter.SelectOrdererDepartment(this.ordererDataSet.Depar
tments);
        personsTableAdapter.SelectOrdererPerson(ordererDataSet.Persons);

equipmentTableAdapter.SelectOrdererEquipment(ordererDataSet.Equipment);

        //this.order_remTableAdapter.Fill(this.ordersDataSet.Order_rem);

this.departmentsTableAdapter1.SelectPerformerDepartment(this.performerDataSet
.Departments);

this.personsTableAdapter1.SelectPerformerPerson(this.performerDataSet.Persons);

        fillEquipmentTextboxes();
    }

    private void dataGridEquipment_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        fillEquipmentTextboxes();
    }

    public void fillEquipmentTextboxes()
    {
        if (dataGridEquipment.Rows.Count != 0)
        {
            textBox3.Text =
dataGridEquipment.CurrentRow.Cells[2].Value.ToString();
            textBox4.Text =
dataGridEquipment.CurrentRow.Cells[4].Value.ToString();
            textBox5.Text =
dataGridEquipment.CurrentRow.Cells[5].Value.ToString();
            textBox6.Text =
```

```
dataGridEquipment.CurrentRow.Cells[3].Value.ToString();
            }
            else
            {
                textBox3.Clear(); textBox4.Clear(); textBox5.Clear(); textBox6.Clear();
            }
        }

        private void textNom_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsDigit(e.KeyChar) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
            }
        }

        private void buttonCreateOrder_Click(object sender, EventArgs e)
        {
            kol = 0;


            if(dataGridPerformerDepartment.Rows.Count != 0 &&
dataGridPerformerPerson.Rows.Count != 0 &&
                dataGridOrderDepartment.Rows.Count != 0 &&
dataGridOrderPerson.Rows.Count != 0)
            {
                if (textNom.Text != "")
                {
                    sqlConnection1.Open();
                    SqlCommand com_Sql = new SqlCommand();
                    string sql = "SELECT COUNT(*) FROM Order_rem WHERE (Nom
= @No) AND (YEAR(Date_) = YEAR(@DA))";
                    com_Sql.Connection = sqlConnection1;
                    com_Sql.CommandType = CommandType.Text;
                    com_Sql.CommandText = sql;
                    com_Sql.Parameters.AddWithValue("@No", textNom.Text);
                    com_Sql.Parameters.AddWithValue("@DA",
dateTimePicker1.Value.ToString());
                    int res = ((int)com_Sql.ExecuteScalar());
                    sqlConnection1.Close();

                    if (res == 0)
                    {

order_remTableAdapter.AddOrder((int)dataGridOrderDepartment.CurrentRow.Cel
```

```
ls[0].Value, (int)dataGridPerformerDepartment.CurrentRow.Cells[0].Value,

(int)dataGridOrderPerson.CurrentRow.Cells[0].Value,
(int)dataGridPerformerPerson.CurrentRow.Cells[0].Value,
                                        Convert.ToInt32(textNom.Text),
dateTimePicker1.Value.ToString("dd.MM.yyyy"),
Convert.ToInt32(numericUpDown1.Value));
                    order_remTableAdapter.Update(ordersDataSet.Order_rem);
                    order_remTableAdapter.Fill(ordersDataSet.Order_rem);

                    MessageBox.Show("Замовлення створено!");


equipment_in_OrderTableAdapter.SelectEquipmentInCurrentOrder(ordersDataSet.
Equipment_in_Order,
                    (int)order_remTableAdapter.MaxOrder());
                }
                else
                {
                    MessageBox.Show("Такий номер замовлення в цьому році вже
є!");
                }
            }
            else
            {
                MessageBox.Show("Заповніть номер замовлення!");
            }
        }
        else
        {
            MessageBox.Show("Недостатньо данних для заповнення!");
        }
    }

    private void buttonAddEquipment_Click(object sender, EventArgs e)
    {
        int KO = 1;
        KO = (int)order_remTableAdapter.CountOrders();

        if (KO == 0)
        {
            MessageBox.Show("Немає замовлень!");
        }
        else
        {
```

```
            if (MessageBox.Show("Додати прилад до замовлення?",
"Підтвердження",
            MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
            {
               kol++;

equipment_in_OrderTableAdapter.AddEquipmentInOrder((int)order_remTableAd
apter.MaxOrder(),
                  (int)dataGridEquipment.CurrentRow.Cells[0].Value,
(int)numericUpDown2.Value);

equipment_in_OrderTableAdapter.Update(ordersDataSet.Equipment_in_Order);

equipment_in_OrderTableAdapter.SelectEquipmentInCurrentOrder(ordersDataSet.
Equipment_in_Order,
                  (int)order_remTableAdapter.MaxOrder());
            }
         }
      }

      private void dataGridOrderDepartment_CellClick(object sender,
DataGridViewCellEventArgs e)
      {
         fillEquipmentTextboxes();
      }
   }
}
```

OrderForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace EquipmentRepair2
{
   public partial class OrderForm : Form
   {
      string DepOrd, DepPerf, PersOrd, PersPerf;
```

```
        public OrderForm()
        {
            InitializeComponent();
            DepOrd = ""; DepPerf = "";
            PersOrd = ""; PersPerf = "";
        }

        private void OrderForm_Load(object sender, EventArgs e)
        {
            this.order_remTableAdapter.SelectOrders(this.ordersDataSet.Order_rem);

this.equipment_in_OrderTableAdapter.SelectEquipmentInOrder(this.ordersDataSe
t.Equipment_in_Order);

            ShowPerformerAndOrderer();
        }

        public void ShowPerformerAndOrderer()
        {
            DepOrd = ""; DepPerf = "";
            PersOrd = ""; PersPerf = "";

            if (dataGridOrder.Rows.Count != 0)
            {
                sqlConnection1.Open();
                SqlCommand comPersOrd = new SqlCommand();
                SqlCommand comPersPerf = new SqlCommand();
                SqlCommand comDepOrd = new SqlCommand();
                SqlCommand comDepPerf = new SqlCommand();
                string PeOrd = "select Surname+' '+Name+' '+Patronimyc as FIO from
Persons where ID_Person = " +
dataGridOrder.CurrentRow.Cells[3].Value.ToString();
                string PePer = "select Surname+' '+Name+' '+Patronimyc as FIO from
Persons where ID_Person = " +
dataGridOrder.CurrentRow.Cells[4].Value.ToString();
                string DeOrd = "select Dep_Nam from Departments where
ID_Department = " + dataGridOrder.CurrentRow.Cells[1].Value.ToString();
                string DePer = "select Dep_Nam from Departments where
ID_Department = " + dataGridOrder.CurrentRow.Cells[2].Value.ToString();

                comPersOrd.Connection = sqlConnection1; comPersPerf.Connection =
sqlConnection1;
                comDepOrd.Connection = sqlConnection1; comDepPerf.Connection =
sqlConnection1;
```

```
            comPersOrd.CommandType = CommandType.Text;
comPersPerf.CommandType = CommandType.Text;
            comDepOrd.CommandType = CommandType.Text;
comDepPerf.CommandType = CommandType.Text;

            comPersOrd.CommandText = PeOrd; comPersPerf.CommandText =
PePer;
            comDepOrd.CommandText = DeOrd; comDepPerf.CommandText =
DePer;

            DepOrd = (string)comDepOrd.ExecuteScalar();
            DepPerf = (string)comDepPerf.ExecuteScalar();
            PersOrd = (string)comPersOrd.ExecuteScalar();
            PersPerf = (string)comPersPerf.ExecuteScalar();

            sqlConnection1.Close();
        }

        textOrdererDepartment.Text = DepOrd;
        textOredererPerson.Text = PersOrd;
        textPerformerDepartment.Text = DepPerf;
        textPerformerPerson.Text = PersPerf;
    }

    private void buttonSearch_Click(object sender, EventArgs e)
    {
        if (checkBoxNom.Checked && checkBoxDate.Checked == false)
        {
            if (textSearchNom.Text != "")
            {
                order_remTableAdapter.SearchByNom(ordersDataSet.Order_rem,
Convert.ToInt32(textSearchNom.Text));

equipment_in_OrderTableAdapter.SelectEquipmentInOrder(this.ordersDataSet.Eq
uipment_in_Order);
            }
            else
            {
                MessageBox.Show("Заповніть поле!");
            }
        }
        if (checkBoxNom.Checked == false && checkBoxDate.Checked)
        {
            order_remTableAdapter.SearchByDate(ordersDataSet.Order_rem,
```

```
dateTimePicker1.Value.ToString("dd.MM.yyyy"));

equipment_in_OrderTableAdapter.SelectEquipmentInOrder(this.ordersDataSet.Eq
uipment_in_Order);
            }
            if (checkBoxNom.Checked && checkBoxDate.Checked)
            {
                if (textSearchNom.Text != "")
                {

order_remTableAdapter.SearchByNomDate(ordersDataSet.Order_rem,
dateTimePicker1.Value.ToString("dd.MM.yyyy"),
Convert.ToInt32(textSearchNom.Text));

equipment_in_OrderTableAdapter.SelectEquipmentInOrder(this.ordersDataSet.Eq
uipment_in_Order);
                }
                else
                {
                    MessageBox.Show("Заповніть поле!");
                }
            }
            if (checkBoxNom.Checked == false && checkBoxDate.Checked == false)
            {
                MessageBox.Show("Оберіть параметри для пошуку!");
            }
        }

        private void ShowReset_Click(object sender, EventArgs e)
        {
            this.order_remTableAdapter.SelectOrders(this.ordersDataSet.Order_rem);

this.equipment_in_OrderTableAdapter.SelectEquipmentInOrder(this.ordersDataSe
t.Equipment_in_Order);

            ShowPerformerAndOrderer();
        }

        private void buttonEditAct_Click(object sender, EventArgs e)
        {
            string old = dataGridOrder.CurrentRow.Cells[5].Value.ToString();
            DateTime OLD =
Convert.ToDateTime(dataGridOrder.CurrentRow.Cells[6].Value);

            if (dataGridOrder.Rows.Count != 0)
```

```
        {
            if (textEditNom.Text != "")
            {
                DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете виконати редагування?", "Редагування", MessageBoxButtons.YesNo);
                if (dialogResult == DialogResult.Yes)
                {
                    sqlConnection1.Open();
                    SqlCommand com_Sql = new SqlCommand();
                    string sql = "SELECT COUNT(*) FROM Order_rem WHERE
(Nom = @No) AND (YEAR(Date_) = YEAR(@DA))";
                    com_Sql.Connection = sqlConnection1;
                    com_Sql.CommandType = CommandType.Text;
                    com_Sql.CommandText = sql;
                    com_Sql.Parameters.AddWithValue("@No", textEditNom.Text);
                    com_Sql.Parameters.AddWithValue("@DA",
dateTimePicker1.Value.ToString());
                    int res = ((int)com_Sql.ExecuteScalar());
                    sqlConnection1.Close();

                    if (res == 0)
                    {
                        int position;
                        position = this.orderremBindingSource.Position;

order_remTableAdapter.UpdateOrder(Convert.ToInt32(textEditNom.Text),
dateTimePickerEdit.Value.ToString("dd.MM.yyyy"),
(int)dataGridOrder.CurrentRow.Cells[0].Value,
(int)dataGridOrder.CurrentRow.Cells[0].Value);
                        order_remTableAdapter.Update(ordersDataSet.Order_rem);
                        order_remTableAdapter.SelectOrders(ordersDataSet.Order_rem);
                        Refresh();
                    }
                    else
                    {
                        MessageBox.Show("Такий номер замовлення в цьому році
вже є!");
                        textEditNom.Text = old;
                        dateTimePickerEdit.Value = OLD;
                    }
                }
                else
                {
                    textEditNom.Text = old;
                    dateTimePickerEdit.Value = OLD;
```

```
            }
        }
        else
        {
            textEditNom.Text = old;
            dateTimePickerEdit.Value = OLD;
            MessageBox.Show("Поле для редагування не повинно бути
пустим!");
        }
    }
    else
    {
        MessageBox.Show("Немає елемента для редагування!");
    }
}

private void buttonEditAmauntInAct_Click(object sender, EventArgs e)
{
    Decimal Old =
Convert.ToDecimal(dataGridView1.CurrentRow.Cells[4].Value.ToString());

    if (dataGridView1.Rows.Count != 0)
    {
        DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете виконати редагування?", "Редагування", MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
            int position;
            position = this.equipmentInOrderOrderBindingSource.Position;

equipment_in_OrderTableAdapter.UpdateEquipmentInOrder((int)numericUpDow
nEdit.Value, (int)dataGridView1.CurrentRow.Cells[0].Value,
(int)dataGridView1.CurrentRow.Cells[0].Value);

equipment_in_OrderTableAdapter.Update(ordersDataSet.Equipment_in_Order);

equipment_in_OrderTableAdapter.SelectEquipmentInOrder(ordersDataSet.Equip
ment_in_Order);
            Refresh();
            this.equipmentInOrderOrderBindingSource.Position = position;
        }
        else
        {
            numericUpDownEdit.Value = Old;
        }
```

```
        }
        else
        {
            MessageBox.Show("Немає елемента для редагування!");
        }
    }

    private void buttonDeleteAct_Click(object sender, EventArgs e)
    {
        if (dataGridOrder.Rows.Count != 0)
        {
            DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете видалити акт?", "Підтвердження", MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                int position;
                position = this.orderremBindingSource.Position;
                ordersDataSet.Order_rem.Rows[position].Delete();
                order_remTableAdapter.Update(ordersDataSet.Order_rem);
                ordersDataSet.Order_rem.AcceptChanges();
            }
        }
        else
        {
            MessageBox.Show("Нема що видалити!");
        }
    }

    private void buttonDeleteEquipment_Click(object sender, EventArgs e)
    {
        if (dataGridView1.Rows.Count != 0)
        {
            DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете видалити цей елемент ?", "Підтвердження",
MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {

equipment_in_OrderTableAdapter.DeleteEqvInOrder(Convert.ToInt32(dataGridVi
ew1.CurrentRow.Cells[0].Value.ToString()));

equipment_in_OrderTableAdapter.Update(ordersDataSet.Equipment_in_Order);

equipment_in_OrderTableAdapter.SelectEquipmentInOrder(ordersDataSet.Equip
ment_in_Order);
```

```
                Refresh();
            }
        }
        else
        {
            MessageBox.Show("Нема що видалити!");
        }
    }

    private void dataGridOrder_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        ShowPerformerAndOrderer();

        textEditNom.Text = dataGridOrder.CurrentRow.Cells[5].Value.ToString();
        dateTimePickerEdit.Value =
Convert.ToDateTime(dataGridOrder.CurrentRow.Cells[6].Value);

        if (dataGridView1.Rows.Count != 0)
        {
            numericUpDownEdit.Enabled = true;
            buttonEditAmauntInAct.Enabled = true;
            numericUpDownEdit.Value =
Convert.ToDecimal(dataGridView1.CurrentRow.Cells[4].Value.ToString());
        }
        else
        {
            numericUpDownEdit.Enabled = false;
            buttonEditAmauntInAct.Enabled = false;
        }
    }

    private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        numericUpDownEdit.Value =
Convert.ToDecimal(dataGridView1.CurrentRow.Cells[4].Value.ToString());
    }

    private void textSearchNom_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsDigit(e.KeyChar) && (e.KeyChar != (char)Keys.Back))
        {
            e.Handled = true;
        }
```

```
        }

        private void checkBoxNom_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBoxNom.Checked)
            {
                textSearchNom.Enabled = true;
            }
            else
            {
                textSearchNom.Enabled = false;
            }
        }

        private void checkBoxDate_CheckedChanged(object sender, EventArgs e) {
            if (checkBoxDate.Checked) {
                dateTimePicker1.Enabled = true;}
            else{dateTimePicker1.Enabled = false;
            }} }}
```

CreateActForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace EquipmentRepair2
{
    public partial class CreateActForm : Form
    {
        public int kol;
        public CreateActForm()
        {
            InitializeComponent();
            kol = 0;
        }

        private void CreateActForm_Load(object sender, EventArgs e)
        {
            // TODO: дана строка коду дозволяє завантажити дані в таблицю
```

```
"ordererDataSet.Equipment". При необхідності вона може бути переміщена
або видалена.

this.equipmentTableAdapter1.SelectOrdererEquipment(this.ordererDataSet.Equip
ment);


        // TODO: дана строка коду дозволяє завантажити дані в таблицю
"actDataSet.Act_return". При необхідності вона може бути переміщена або
видалена.
        this.act_returnTableAdapter.Fill(this.actDataSet.Act_return);

//this.equipmentTableAdapter.SelectEquipment(this.actDataSet.Equipment);
        this.order_remTableAdapter.SelectOrders(this.actDataSet.Order_rem);

this.departmentsTableAdapter1.SelectPerformerDepartment(this.performerDataSet
.Departments);

this.personsTableAdapter1.SelectPerformerPerson(this.performerDataSet.Persons);

this.departmentsTableAdapter.SelectOrdererDepartment(this.ordererDataSet.Depar
tments);

this.personsTableAdapter.SelectOrdererPerson(this.ordererDataSet.Persons);

        FillMyWorks();

        if (dataGridEquip.Rows.Count != 0)
        {
            textBox3.Text = dataGridEquip.CurrentRow.Cells[5].Value.ToString();
            textBox4.Text = dataGridEquip.CurrentRow.Cells[3].Value.ToString();
            textBox5.Text = dataGridEquip.CurrentRow.Cells[4].Value.ToString();
            textBox6.Text = dataGridEquip.CurrentRow.Cells[2].Value.ToString();
        }
    }

    public void FillMyWorks()
    {
        string command = "SELECT ID_work_type, wrk_tp FROM Work_Types
WHERE ID_work_type <> 1";
        System.Data.SqlClient.SqlDataAdapter da = new
SqlDataAdapter(command, sqlConnection1);

        DataSet ds = new DataSet();
        sqlConnection1.Open();
```

```
            da.Fill(ds);
            sqlConnection1.Close();

            comboBoxWork.DataSource = ds.Tables[0];
            comboBoxWork.DisplayMember = "wrk_tp";
            comboBoxWork.ValueMember = "ID_work_type";
        }

        private void buttonCreateAct_Click(object sender, EventArgs e)
        {
            kol = 0;

            if (dataGridEquip.Rows.Count == 0 ||
dataGridPerformerDepartment.Rows.Count == 0 || comboBoxWork.Items.Count
== 0 ||
                dataGridPerformerPerson.Rows.Count == 0 ||
dataGridOrder.Rows.Count == 0 || dataGridOrderDepartment.Rows.Count == 0 ||
                dataGridOrderPerson.Rows.Count == 0)
            {
                MessageBox.Show("Недостатньо данних для заповнення!");
            }
            else
            {
                if (textNom.Text != "")
                {
                    if (dateTimePicker2.Value.ToString("dd.MM.yyyy") ==
dateTimePicker3.Value.ToString("dd.MM.yyyy") ||
                dateTimePicker2.Value < dateTimePicker3.Value)
                    {
                        sqlConnection1.Open();
                        SqlCommand com_Sql = new SqlCommand();
                        string sql = "SELECT COUNT(*) FROM Act_return WHERE
(Nomer = @No) AND (YEAR(Date_) = YEAR(@DA))";
                        com_Sql.Connection = sqlConnection1;
                        com_Sql.CommandType = CommandType.Text;
                        com_Sql.CommandText = sql;
                        com_Sql.Parameters.AddWithValue("@No", textNom.Text);
                        com_Sql.Parameters.AddWithValue("@DA",
dateTimePicker1.Value.ToString());
                        int res = ((int)com_Sql.ExecuteScalar());
                        sqlConnection1.Close();

                        if (res == 0)
                        {
```

```
act_returnTableAdapter.AddAct((int)dataGridOrder.CurrentRow.Cells[0].Value,
(int)dataGridPerformerPerson.CurrentRow.Cells[0].Value,
(int)dataGridOrderPerson.CurrentRow.Cells[0].Value,
                        Convert.ToInt32(textNom.Text),
dateTimePicker1.Value.ToString("dd.MM.yyyy"),
dateTimePicker2.Value.ToString("dd.MM.yyyy"),
dateTimePicker3.Value.ToString("dd.MM.yyyy"));
                    act_returnTableAdapter.Update(actDataSet.Act_return);
                    act_returnTableAdapter.Fill(actDataSet.Act_return);

                    MessageBox.Show("Акт створено!");


equipment_in_ActTableAdapter.SelectEquipmentInCurrentAct(actDataSet.Equip
ment_in_Act, (int)act_returnTableAdapter.MaxAct());
                }
                else
                {
                    MessageBox.Show("Такий номер акту в цьому році вже є!");
                }
            }
            else
            {
                MessageBox.Show("\"Дата з\" повинна бути менша \"дати
по\"");
            }
        }
        else
        {
            MessageBox.Show("Заповніть номер акту!");
        }
    }
}

private void buttonAddEquipment_Click(object sender, EventArgs e)
{
    int KO = 1;
    KO = (int)act_returnTableAdapter.CountAct();

    if (KO == 0)
    {
        MessageBox.Show("Немає замовлень!");
    }
    else
    {
```

```
            if (MessageBox.Show("Додати прилад до акту?", "Підтвердження",
            MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
            {
               kol++;

equipment_in_ActTableAdapter.AddEquipmentInAct((int)act_returnTableAdapter
.MaxAct(), (int)dataGridEquip.CurrentRow.Cells[0].Value,
                     Convert.ToInt32(comboBoxWork.SelectedValue),
(int)numericUpDown1.Value);

equipment_in_ActTableAdapter.Update(actDataSet.Equipment_in_Act);

equipment_in_ActTableAdapter.SelectEquipmentInCurrentAct(actDataSet.Equip
ment_in_Act,
                     (int)act_returnTableAdapter.MaxAct());
            }
         }
      }

      private void dataGridEquip_CellClick(object sender,
DataGridViewCellEventArgs e)
      {
         textBox3.Text = dataGridEquip.CurrentRow.Cells[5].Value.ToString();
         textBox4.Text = dataGridEquip.CurrentRow.Cells[3].Value.ToString();
         textBox5.Text = dataGridEquip.CurrentRow.Cells[4].Value.ToString();
         textBox6.Text = dataGridEquip.CurrentRow.Cells[2].Value.ToString();
      }

      private void textNom_KeyPress(object sender, KeyPressEventArgs e)
      {
         if (!char.IsDigit(e.KeyChar) && (e.KeyChar != (char)Keys.Back))
         {
            e.Handled = true;
         }
      }

      private void dataGridPerformerDepartment_CellClick(object sender,
DataGridViewCellEventArgs e)
      {
         if (dataGridEquip.Rows.Count != 0)
         {
            textBox3.Text = dataGridEquip.CurrentRow.Cells[5].Value.ToString();
            textBox4.Text = dataGridEquip.CurrentRow.Cells[3].Value.ToString();
            textBox5.Text = dataGridEquip.CurrentRow.Cells[4].Value.ToString();
```

```
            textBox6.Text = dataGridEquip.CurrentRow.Cells[2].Value.ToString();
        }
        else
        {
            textBox3.Text = "";
            textBox4.Text = "";
            textBox5.Text = "";
            textBox6.Text = "";
        } } } }
```

ActForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace EquipmentRepair2
{
    public partial class ActForm : Form
    {
        string PersOrd, PersPerf;
        public ActForm()
        {
            InitializeComponent();
            PersOrd = ""; PersPerf = "";
        }

        private void ActForm_Load(object sender, EventArgs e)
        {
            this.act_returnTableAdapter.SelectAct(this.actDataSet.Act_return);

this.equipment_in_ActTableAdapter.SelectEquipmentInAct(this.actDataSet.Equip
ment_in_Act);

            ShowPerformerAndOrderer();
        }

        public void ShowPerformerAndOrderer()
        {
            PersOrd = ""; PersPerf = "";
```

```
            if (dataGridAct.Rows.Count != 0)
            {
                sqlConnection1.Open();
                SqlCommand comPersOrd = new SqlCommand();
                SqlCommand comPersPerf = new SqlCommand();
                string PeOrd = "select Surname+' '+Name+' '+Patronimyc as FIO from
Persons where ID_Person = " +
dataGridAct.CurrentRow.Cells[5].Value.ToString();
                string PePer = "select Surname+' '+Name+' '+Patronimyc as FIO from
Persons where ID_Person = " +
dataGridAct.CurrentRow.Cells[6].Value.ToString();

                comPersOrd.Connection = sqlConnection1; comPersPerf.Connection =
sqlConnection1;

                comPersOrd.CommandType = CommandType.Text;
comPersPerf.CommandType = CommandType.Text;

                comPersOrd.CommandText = PeOrd; comPersPerf.CommandText =
PePer;

                PersOrd = (string)comPersOrd.ExecuteScalar();
                PersPerf = (string)comPersPerf.ExecuteScalar();

                sqlConnection1.Close();
            }

            textOredererPerson.Text = PersOrd;
            textPerformerPerson.Text = PersPerf;
        }

        private void buttonSearch_Click(object sender, EventArgs e)
        {
            if (checkBoxNom.Checked && checkBoxDate.Checked == false)
            {
                if (textSearchNom.Text != "")
                {
                    act_returnTableAdapter.SearchActByNom(actDataSet.Act_return,
Convert.ToInt32(textSearchNom.Text));

equipment_in_ActTableAdapter.SelectEquipmentInAct(actDataSet.Equipment_in_
Act);
                }
                else
```

```
            {
                MessageBox.Show("Заповніть поле!");
            }
        }
        if (checkBoxNom.Checked == false && checkBoxDate.Checked)
        {
            act_returnTableAdapter.SearchActByDate(actDataSet.Act_return,
dateTimePicker1.Value.ToString("dd.MM.yyyy"));

equipment_in_ActTableAdapter.SelectEquipmentInAct(actDataSet.Equipment_in_
Act);
        }
        if (checkBoxNom.Checked && checkBoxDate.Checked)
        {
            if (textSearchNom.Text != "")
            {

act_returnTableAdapter.SearchActByNomDate(actDataSet.Act_return,
Convert.ToInt32(textSearchNom.Text),
dateTimePicker1.Value.ToString("dd.MM.yyyy"));

equipment_in_ActTableAdapter.SelectEquipmentInAct(actDataSet.Equipment_in_
Act);
            }
            else
            {
                MessageBox.Show("Заповніть поле!");
            }
        }
        if (checkBoxNom.Checked == false && checkBoxDate.Checked == false)
        {
            MessageBox.Show("Оберіть параметри для пошуку!");
        }
    }

    private void ShowReset_Click(object sender, EventArgs e)
    {
        this.act_returnTableAdapter.SelectAct(this.actDataSet.Act_return);

this.equipment_in_ActTableAdapter.SelectEquipmentInAct(this.actDataSet.Equip
ment_in_Act);

        ShowPerformerAndOrderer();
    }
```

```
        private void buttonEditAct_Click(object sender, EventArgs e)
        {
            string old = dataGridAct.CurrentRow.Cells[1].Value.ToString();
            DateTime OLD =
Convert.ToDateTime(dataGridAct.CurrentRow.Cells[2].Value);

            if (dataGridAct.Rows.Count != 0)
            {
                if (textEditNom.Text != "")
                {
                    DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете виконати редагування?", "Редагування", MessageBoxButtons.YesNo);
                    if (dialogResult == DialogResult.Yes)
                    {
                        sqlConnection1.Open();
                        SqlCommand com_Sql = new SqlCommand();
                        string sql = "SELECT COUNT(*) FROM Act_return WHERE
(Nomer = @No) AND (YEAR(Date_) = YEAR(@DA))";
                        com_Sql.Connection = sqlConnection1;
                        com_Sql.CommandType = CommandType.Text;
                        com_Sql.CommandText = sql;
                        com_Sql.Parameters.AddWithValue("@No", textEditNom.Text);
                        com_Sql.Parameters.AddWithValue("@DA",
dateTimePicker1.Value.ToString());
                        int res = ((int)com_Sql.ExecuteScalar());
                        sqlConnection1.Close();

                        if (res == 0)
                        {
                            int position;
                            position = this.actreturnBindingSource.Position;

act_returnTableAdapter.UpdateAct(Convert.ToInt32(textEditNom.Text),
dateTimePickerEdit.Value.ToString("dd.MM.yyyy"),
(int)dataGridAct.CurrentRow.Cells[0].Value,
(int)dataGridAct.CurrentRow.Cells[0].Value);
                            act_returnTableAdapter.Update(actDataSet.Act_return);
                            act_returnTableAdapter.SelectAct(actDataSet.Act_return);
                            Refresh();
                        }
                        else
                        {
                            MessageBox.Show("Такий номер чеку в цьому році вже є!");
                            textEditNom.Text = old;
                            dateTimePickerEdit.Value = OLD;
```

```
                }
            }
            else
            {
                textEditNom.Text = old;
                dateTimePickerEdit.Value = OLD;
            }
        }
        else
        {
            textEditNom.Text = old;
            dateTimePickerEdit.Value = OLD;
            MessageBox.Show("Поле для редагування не повинно бути
пустим!");
        }
    }
    else
    {
        MessageBox.Show("Немає елемента для редагування!");
    }
}

private void buttonEditAmauntInAct_Click(object sender, EventArgs e)
{
    Decimal Old =
Convert.ToDecimal(dataGridEqInAct.CurrentRow.Cells[4].Value.ToString());

    if (dataGridEqInAct.Rows.Count != 0)
    {
        DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете виконати редагування?", "Редагування", MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
            int position;
            position = this.equipmentInActActBindingSource.Position;

equipment_in_ActTableAdapter.UpdateEquipmentInAct((int)numericUpDownEdit
.Value, (int)dataGridEqInAct.CurrentRow.Cells[0].Value,
(int)dataGridEqInAct.CurrentRow.Cells[0].Value);

equipment_in_ActTableAdapter.Update(actDataSet.Equipment_in_Act);

equipment_in_ActTableAdapter.SelectEquipmentInAct(actDataSet.Equipment_in_
Act);
            Refresh();
```

```
                this.equipmentInActActBindingSource.Position = position;
            }
            else
            {
                numericUpDownEdit.Value = Old;
            }
        }
        else
        {
            MessageBox.Show("Немає елемента для редагування!");
        }
    }

    private void buttonDeleteAct_Click(object sender, EventArgs e)
    {
        if (dataGridAct.Rows.Count != 0)
        {
            DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете видалити акт?", "Підтвердження", MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                int position;
                position = this.actreturnBindingSource.Position;
                actDataSet.Act_return.Rows[position].Delete();
                act_returnTableAdapter.Update(actDataSet.Act_return);
                actDataSet.Act_return.AcceptChanges();
            }
        }
        else
        {
            MessageBox.Show("Нема що видалити!");
        }
    }

    private void buttonDeleteEquipment_Click(object sender, EventArgs e)
    {
        if (dataGridEqInAct.Rows.Count != 0)
        {
            DialogResult dialogResult = MessageBox.Show("Ви впевнені, що
хочете видалити цей елемент?", "Підтвердження",
MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {

equipment_in_ActTableAdapter.DeleteEquipmentInAct(Convert.ToInt32(dataGrid
```

```
EqInAct.CurrentRow.Cells[0].Value.ToString()));

equipment_in_ActTableAdapter.Update(actDataSet.Equipment_in_Act);

equipment_in_ActTableAdapter.SelectEquipmentInAct(actDataSet.Equipment_in_
Act);
                Refresh();
            }
        }
        else
        {
            MessageBox.Show("Нема що видалити!");
        }
    }

    private void dataGridAct_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        ShowPerformerAndOrderer();

        textEditNom.Text = dataGridAct.CurrentRow.Cells[1].Value.ToString();
        dateTimePickerEdit.Value =
Convert.ToDateTime(dataGridAct.CurrentRow.Cells[2].Value);

        if (dataGridEqInAct.Rows.Count != 0)
        {
            numericUpDownEdit.Enabled = true;
            buttonEditAmauntInAct.Enabled = true;
            numericUpDownEdit.Value =
Convert.ToDecimal(dataGridEqInAct.CurrentRow.Cells[6].Value.ToString());
        }
        else
        {
            numericUpDownEdit.Enabled = false;
            buttonEditAmauntInAct.Enabled = false;
        }
    }

    private void dataGridEqInAct_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        numericUpDownEdit.Value =
Convert.ToDecimal(dataGridEqInAct.CurrentRow.Cells[6].Value.ToString());
    }
```

```
        private void checkBoxNom_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBoxNom.Checked)
            {
                textSearchNom.Enabled = true;
            }
            else
            {
                textSearchNom.Enabled = false;
            }
        }

        private void checkBoxDate_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBoxDate.Checked)
            {
                dateTimePicker1.Enabled = true;
            }
            else
            {
                dateTimePicker1.Enabled = false;
            }
        }

        private void textEditNom_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsDigit(e.KeyChar) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
            }
        }

        private void textSearchNom_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsDigit(e.KeyChar) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
            }
        }
    }
}
```

ControlForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EquipmentRepair2
{
   public partial class ControlForm : Form
   {
      string currentYear, beginYear, endYear;
      public ControlForm()
      {
         InitializeComponent();
      }

      private void ControlForm_Load(object sender, EventArgs e)
      {
         currentYear = DateTime.Now.Year.ToString();
         beginYear = "01.01." + currentYear;
         endYear = "31.12." + currentYear;

         dateTimePicker1.Value = Convert.ToDateTime(beginYear);
         dateTimePicker2.Value = Convert.ToDateTime(endYear);

         act_returnTableAdapter.SearchControlData(controlDataSet.Act_return,
beginYear, endYear);
         CountDolg();
      }

      public void CountDolg()
      {
         int allSum = 0;

         if (dataGridView1.RowCount != 0)
         {
            for (int i = 0; i < dataGridView1.RowCount; i++){
               allSum +=
Convert.ToInt32(dataGridView1.Rows[i].Cells[11].Value.ToString());
            }
         }

         label1.Text = "Ще необхідно відремонтувати " + allSum + "
обладнанн";
```

```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (dateTimePicker1.Value.ToString("dd.MM.yyyy") ==
dateTimePicker2.Value.ToString("dd.MM.yyyy") ||
                dateTimePicker1.Value < dateTimePicker2.Value)
            {
                act_returnTableAdapter.SearchControlData(controlDataSet.Act_return,
dateTimePicker1.Value.ToString("dd.MM.yyyy"),
dateTimePicker2.Value.ToString("dd.MM.yyyy"));
                CountDolg();
            }
            else
            {
                MessageBox.Show("\"Дата з\" повинна бути менша \"дати по\"");
            }
        }

        private void ShowReset_Click(object sender, EventArgs e)
        {
            dateTimePicker1.Value = Convert.ToDateTime(beginYear);
            dateTimePicker2.Value = Convert.ToDateTime(endYear);

            act_returnTableAdapter.SearchControlData(controlDataSet.Act_return,
beginYear, endYear);
            CountDolg();
        }}}
```

ReportCreatingForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EquipmentRepair2
{
    public partial class ReportCreatingForm : Form
    {
        string currentYear, beginYear, endYear;
```

```
        public ReportCreatingForm()
        {
            InitializeComponent();
            MyDataTables.datTab = reportDataSet11.Tables["Equipment"];
        }

        private void ReportCreatingForm_Load(object sender, EventArgs e)
        {
            currentYear = DateTime.Now.Year.ToString();
            beginYear = "01.01." + currentYear;
            endYear = "31.12." + currentYear;

            dateTimePicker1.Value = Convert.ToDateTime(beginYear);
            dateTimePicker2.Value = Convert.ToDateTime(endYear);

            sqlDataAdapter1.SelectCommand.Parameters[0].Value =
dateTimePicker1.MinDate.ToString("dd.MM.yyyy");
            sqlDataAdapter1.SelectCommand.Parameters[1].Value =
dateTimePicker2.MaxDate.ToString("dd.MM.yyyy");
            sqlDataAdapter1.Fill(reportDataSet11.Equipment);
        }

        private void buttonReport_Click(object sender, EventArgs e)
        {
            ReportForm repFor = new ReportForm();
            repFor.Show();
        }

        private void buttonQuerry_Click(object sender, EventArgs e)
        {
            if (dateTimePicker1.Value.ToString("dd.MM.yyyy") ==
dateTimePicker2.Value.ToString("dd.MM.yyyy") ||
                dateTimePicker1.Value < dateTimePicker2.Value)
            {
                reportDataSet11.Clear();
                sqlDataAdapter1.SelectCommand.Parameters[0].Value =
dateTimePicker1.Value.ToString("dd.MM.yyyy");
                sqlDataAdapter1.SelectCommand.Parameters[1].Value =
dateTimePicker2.Value.ToString("dd.MM.yyyy");
                sqlDataAdapter1.Fill(reportDataSet11.Equipment);

                GetDataFromGrid();
            }
            else
            {
```

```
            MessageBox.Show("\"Дата з\" повинна бути менша \"дати по\"");
        }
    }

    public void GetDataFromGrid()
    {
        int[] gridI = new int[dataGridView1.RowCount];
        string[] gridS = new string[dataGridView1.RowCount];

        for (int i = 0; i < dataGridView1.RowCount; i++)
        {
            gridS[i] = dataGridView1.Rows[i].Cells[0].Value.ToString();
            gridI[i] =
Convert.ToInt32(dataGridView1.Rows[i].Cells[1].Value.ToString());
        }

        chart1.Series[0].Points.DataBindXY(gridS, gridI);
    }

    private void buttonRewind_Click(object sender, EventArgs e)
    {
        dateTimePicker1.Value = Convert.ToDateTime(beginYear);
        dateTimePicker2.Value = Convert.ToDateTime(endYear);

        sqlDataAdapter1.SelectCommand.Parameters[0].Value =
dateTimePicker1.MinDate.ToString("dd.MM.yyyy");
        sqlDataAdapter1.SelectCommand.Parameters[1].Value =
dateTimePicker2.MaxDate.ToString("dd.MM.yyyy");
        sqlDataAdapter1.Fill(reportDataSet11.Equipment);

        GetDataFromGrid();
    }
}

public static class MyDataTables
{
    public static DataTable datTab;
}
}
```

ReportForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EquipmentRepair2
{
    public partial class ReportForm : Form
    {
        public ReportForm()
        {
            InitializeComponent();
        }

        private void ReportForm_Load(object sender, EventArgs e)
        {
            CrystalReport1 repa = new CrystalReport1();
            repa.SetDataSource(MyDataTables.datTab);
            crystalReportViewer1.ReportSource = repa;
        }
    }
}
```

UserManager.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace EquipmentRepair2
{
    public static class UserManager
    {
        public static string MyLogin = "";
    }
}
```