Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «Система контролю за рухом деталей на меткомбінаті та оцінка їх залишкового ресурса»
за освітньою програмою **12 Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**

Виконав: студент групи ПЗ1912: /Кирило ЯРОВИЙ/
(підпис)

Керівник: /Валентин РАЗНОСІЛІН/
(підпис)

Нормоконтролер: /Світлана ВОЛКОВА/
(підпис)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент: _____
(підпис)

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note

to Master's Thesis
bachelor

on the topic: «The system for monitoring of the components circulation at a steel plant and estimating their residual life»

according to educational curriculum **12 software engineering**
in the Speciality: **121 software engineering**

Done by the student of the group PZ1912: _____ /Kyrylo YAROVYI/
(підпис)

Scientific Supervisor: _____ /Valentin RAZNOSILIN/
(підпис)

Normative controller: _____ /Svitlana VOLKOVA/
(підпис)

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем
Кафедра: Комп’ютерні інформаційні технології
Рівень вищої освіти: магістр
Освітня програма: Інженерія програмного забезпечення
Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ____КІТ
_____ Вадим ГОРЯЧКІН
_____ 202_ р.

ЗАВДАННЯ

На кваліфікаційну роботу ____Бакалавр_____
студенту Яровому Кирилу Вікторовичу_____

1. Тема дипломної роботи: Система контролю за рухом деталей на меткомбінаті та оцінка їх залишкового ресурса
   Керівник роботи: Разносілін Валентин В'ячеславович
   затверджені наказом 1196 ст від 19.06.2023 року

2. Строк подання студентом роботи 22.06. 2023 року

3. Вихідні дані до дипломної роботи: _____.

4. Зміст пояснювальної записки (перелік питань до розробки): реферат, вступ, аналіз сучасного стану предметної області, проектування, тестування та налагодження, висновки, бібліографічний список.
5. Перелік демонстраційного матеріалу:
   5.1. доповідь;
   5.2. презентація;
   5.3. демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Вступ	14.09.22 – 28.10.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	29.10.22 – 10.03.23	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	11.03.23 – 31.04.23	
4	Постановка задачі, технічне завдання	01.05.23 – 07.05.23	30%
5	Техніко-економічні показники	08.05.23 – 14.05.23	
6	Розробка інструментальних засобів дослідження	15.05.23 – 21.05.23	
7	Виконання досліджень	22.05.23 – 28.05.23	60%
8	Оформлення тез доповідей	29.05.23 – 02.06.23	
9	Оформлення статті у фаховий журнал	03.06.23 – 07.06.23	
10	Оформлення пояснювальної записки	08.06.23 – 11.06.23	
11	Розробка демонстраційних матеріалів	12.06.23 – 18.06.23	100%
12	Подання кваліфікаційної роботи до кафедри	22.06.23	
13	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	28.06.23	

Студент _____ /Кирило ЯРОВИЙ/

Керівник роботи _____ /Валентин РАЗНОСІЛІН/

Зміст

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра 146 с., 23 рис., 32 табл., 5 додатки, 13 джерел.

Об’єктом дослідження є прокатний стан на меткомбінаті. Який виконує роскат металевих слябів у металеві смуги. Складається з деталей різного типу, що собою утворюють передатні механізми від електродвигуна до прокатних валків. Кожна група клітей відрізняються одна від одної технічними характеристиками.

Метою дослідження в контексті даної роботи є побудова зручної та функціональної програми для нагляду за деталями та з можливістю прогнозування їх огляду та замін у робочих клітях, які складають прокатний стан. А також, перегляд параметрів прокатки металевих слябів у смуги, навантаження та швидкість окремих частин прокатного стану та історія їх замін.

Методом дослідження є технічне вивчення побудови прокатного стану: його складових та принципу дії, а також, робота з статистичними даними навантажень, швидкостей та часових проміжків роботи клітей, що виконують прокатку металевих слябів та прокатують їх у смуги.

Пояснювальна записка складається зі вступу, трьох розділів, висновків, бібліографічного списку та п’яти додатків.

Вступ описує суть, мету та актуальність роботи (2 сторінки).

Перший розділ: аналіз сучасного стану предметної області (2 сторінки).

Другий розділ: проектування (49 сторінок).

Третій розділ: опис програмного продукту (12 сторінок).

Додатки: технічне завдання, специфікація, листи затвердження, текст програми, текст скриптів.

Ключові слова: прокатний стан; робоча кліть; металевий сляб; металева смуга; база даних.

ВСТУП

Актуальність роботи.

Металопрокат є дуже розвиненою областю в сучасному світі. Історія цієї галузі вже дуже довга та цікава. Можна з повною впевненістю сказати, що прокатка є нащадком старої технології під назвою ковка металу. На той час така технологія виробу металу була дуже й дуже складна тому, що метал сам по собі дуже важкий, щоб перетворювати на вироби різної форми його потрібно було нагрівати до дуже високої температури, а також, роботу виконувала одна людина. Саме тому, такі вироби були коштовними й не могли використовуватися в більш ширшому обсязі галузей.

Вважається, що ідею отримання прокатного металу вигадав Леонардо да Вінчі, але через брак технології у його роки технологія так і не була повністю закінчена.

Час йшов і в людства з'являлись все більш сучасні технології. Першою появою прокатки було використання парових машин та валків у 20-му сторіччі. Але, якість продукції залишалась ще дуже далекою в порівнянні із сучасністю.

І вже, поява такої технології підштовхнула людство до використання виробів із прокатного стану. І це була чудова ідея адже такі вироби були дуже надійними та міцними, а також, планета Земля має величезні запаси залізної руди із якою і робились вироби.

Все це призвело до неймовірного розвитку галузі, а в сучасному світі всі дані потрібно переводити у електронний формат для швидкої взаємодії з ними. Це все і підштовхнуло подібні фабрики створювати та використовувати програмне забезпечення для більш швидкої роботи.

Мета роботи.

Створення програми для нагляду за станом деталей у прокатному стані. Заміна кожної деталі, перегляд її історії, навантажень та швидкостей і можливість відправки на списання, а саме, у металобрухт вже зламаної деталі. А також, перегляд вже прокатаних металевих смуг та їх характеристик.

Експлуатаційне призначення.

Створене програмне забезпечення призначається для використання на меткомбінаті для відстежування життєвого циклу деталей прокатного стану та оцінка їх залишкового ресурсу. З метою поліпшення процедур технічного огляду та ремонту прокатного стану.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПРЕДМЕТНОЇ ОБЛАСТІ

На даний час існує велика кількість систем управління підприємствами (ERP), які призначаються для керування малими, середніми та великими підприємствами. В залежності від масштабу підприємства система включає в себе ту чи іншу кількість модулів.

Такі системи, зазвичай, побудовані по моделі MVC (англ. Model-view-controller), тобто розділяють модель даних, вигляд (інтерфейс користувача) та модуль керування. Цей шаблон полегшує подальші зміни чи розширення програми, надає можливість неодноразового використання одного з компонентів.

Діяльність різних підприємств має схожі задачі, наприклад, бухгалтерія, складський облік, управління кадрами. Тому, промислові ERP системи включають в себе готові модулі для вирішення перерахованих вище задач, а також, дозволяють налаштувати ці модулі відповідно до потреб конкретного підприємства.

Прикладом розробника подібних систем є SAP SE. Ця компанія створює програмне забезпечення корпоративного формату та виконує його підтримку. Такі продукти виділяються: функціональністю, інтеграцією різних процесів, модульний принцип побудови програми, а саме, ізолювання окремих частин підсистем та можливі зміни протягом експлуатаційного часу продукту.

Іншим прикладом подібного розробника є американська корпорація Oracle, яка є постачальником від серверного обладнання до СУБД та ERP систем. Oracle подібно SAP SE будують готові програмні продукти дотримуючись шаблонів, які потребують окремі підприємства.

Економічна діяльність деяких підприємств може бути настільки складної та унікальною, що це потребує створення окремих програмних продуктів для автоматизації роботи підприємства.

Прикладом такої ситуації є меткомбінат, який випускає широкий спектр продукції і для цього використовує значну кількість обладнання різного типу, яке може поєднуватися у відповідності до різних схем виробництва.

Для автоматизації подібних випадків підприємство замовляє або у стороннього розробника, або в свого технічного відділу пропрієтарне програмне забезпечення з оглядом на свої потреби та обмеження. Така розробка ведеться в тісній взаємодії інженерів конкретної виробничої ділянки (цеху) та команди програмного відділу.

Створення подібного продукту є кропітким процесом з величезною кількістю деталей та різних технологій, тому перед розробкою проекту приділяють величезну кількість часу на проектування та на розуміння кожного технологічного процесу підприємства.

2 ПРОЕКТУВАННЯ

2.1 Зовнішнє проектування

2.1.1 Функціональне призначення

Функціональним призначенням програмного забезпечення є спрощення керуванням прокатним станом на виробництві з розкатки металу. Полегшення розуміння надійності встановлених деталей та заміна їх.

2.1.2 Експлуатаційне призначення

Експлуатаційним призначенням є полегшення роботи персоналу на прокатному стані шляхом використання створеного програмного забезпечення. Зменшення людських помилок та покращення фінансового стану виробництва через більш точного використання ресурсу деталей прокатного стану.

2.1.3 Функціональні вимоги

Створене програмне забезпечення надає можливість:

- Підключення до різних серверів та баз даних;
- Перегляд прокатаних смуг;
- Додавання прокатних валків з вказанням максимального пробігу на склад з вказанням кількості;
- Додавання шпинделя на склад з вказанням кількості;
- Додавання шестеренної кліті на склад з вказанням кількості;
- Додавання муфти на склад з вказанням кількості;
- Додавання редуктора на склад з вказанням кількості;
- Додавання електродвигуна на склад з вказанням кількості;
- Додавання деталі за типом кліті;
- Перегляд встановленого обладнання за групою клітей;
- Перегляд обладнання на складі за групою клітей;
- Перегляд встановленого обладнання за деталлю – валок;
- Перегляд встановленого обладнання за деталлю – шпиндель;
- Перегляд встановленого обладнання за деталлю – шестеренна кліть;

- Перегляд встановленого обладнання за деталлю – муфта;
- Перегляд встановленого обладнання за деталлю – редуктор;
- Перегляд встановленого обладнання за деталлю – електродвигун;
- Перегляд обладнання на складі за деталлю – валок;
- Перегляд обладнання на складі за деталлю – шпиндель;
- Перегляд обладнання на складі за деталлю – шестеренна кліть;
- Перегляд обладнання на складі за деталлю – муфта;
- Перегляд обладнання на складі за деталлю – редуктор;
- Перегляд обладнання на складі за деталлю – електродвигун;
- Заміна деталі «валок»;
- Заміна деталі «шпиндель»;
- Заміна деталі «шестеренна кліть»;
- Заміна деталі «муфта»;
- Заміна деталі «редуктор»;
- Заміна деталі «електродвигун»;
- Перегляд історії замін по кожній деталі;
- Перегляд записів навантажень та швидкостей по кожній деталі;
- Відправка деталі «валок» у смітник;
- Відправка деталі «шпиндель» у смітник;
- Відправка деталі «шестеренна кліть» у смітник;
- Відправка деталі «муфта» у смітник;
- Відправка деталі «редуктор» у смітник;
- Відправка деталі «електродвигун» у смітник;
- Симуляція прокатки металевих слябів.

2.1.4 Вхідні та вихідні дані

Вхідними даними є:

- Введення ім’я серверу;
- Введення назви бази даних;

- Дані прокатаної металевої смуги;
- Деталі додані на склад;
- Вказання максимального пробігу деталі;
- Вибір меню перегляду стану за різним типом кліті;
- Вибір меню перегляду стану за різним типом деталі;
- Вибір деталі для заміни;
- Вибір деталі для відправки у смітник;
- Вибір металевого слябу для прокатки.

Вихідними даними є:

- Виведення ім’я серверу;
- Виведення назви бази даних;
- Виведення прокатаних смуг металу;
- Виведення інформації щодо вибраної деталі;
- Виведення інформації щодо вибраної групи клітей;
- Виведення інформації щодо наявності деталі на складі;
- Виведення інформаційних вказівок для користувача щодо його дій.

2.1.5 Опис зовнішнього інформаційного середовища

Для використання програмного забезпечення необхідно мати Windows 7 – 11. Мати фреймворк .NET для сталого функціонування програми та бази даних Microsoft SQL Server.

Для підключення до бази даних користувач повинен налаштувати конфігурацію. Для програми з перегляду прокатного стану це можна зробити через перше вікно де надається можливість змінювати параметри серверу та бази даних, або через файл конфігурації. Для програми з симуляції прокатки дані для підключення змінюються у файлі конфігурації.

Для роботи програми використовуються Stored Procedure. Ці запити прописані в окремих файлах для кожної з програм. Користувач на свій розсуд може їх модифікувати, якщо така потреба з’явиться.

Всі дії користувача в інтерфейсі мають зворотну дію, тобто сповіщають про статус виконання команди.

Специфікація функціональних вимог виконана у вигляді діаграм прецедентів (рис. 2.1.1 – 2.1.2).

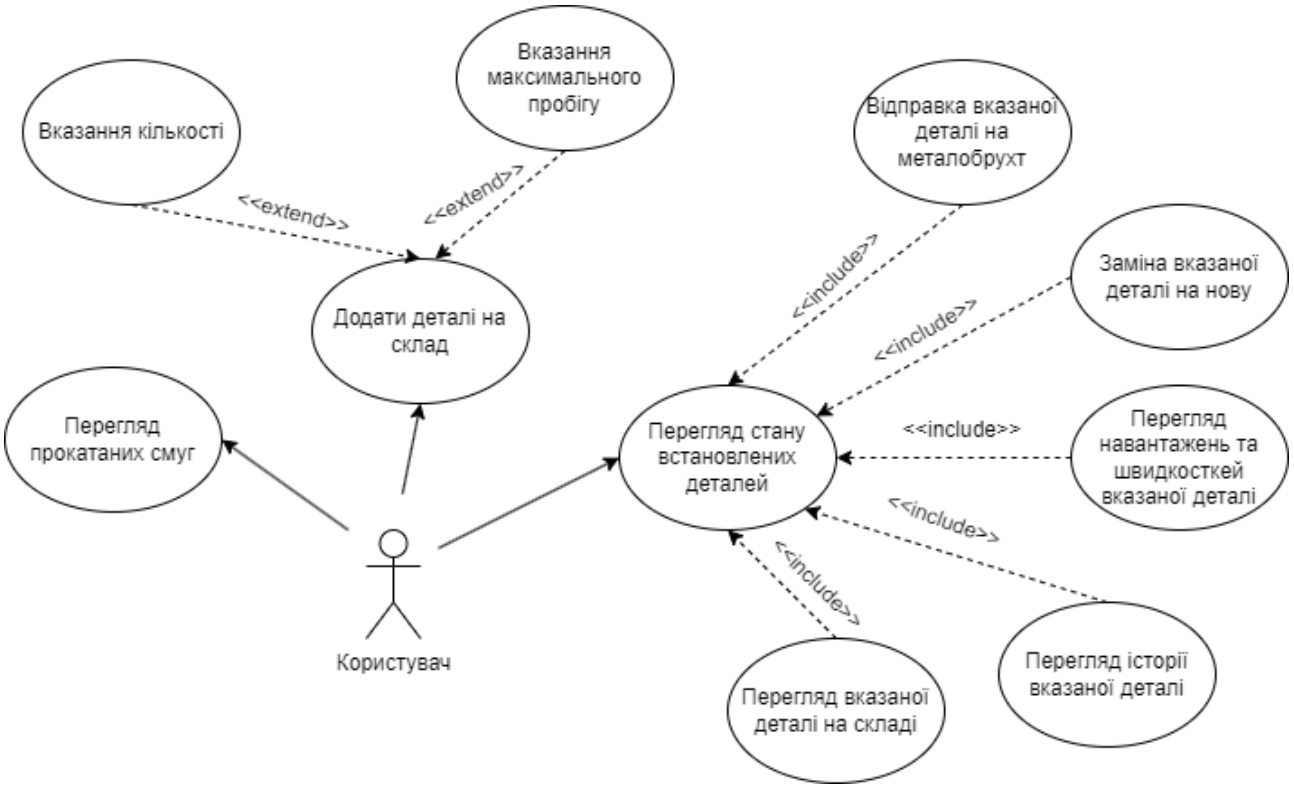


Рисунок 2.1.1 – Діаграма прецедентів програми перегляду деталей прокатного стану

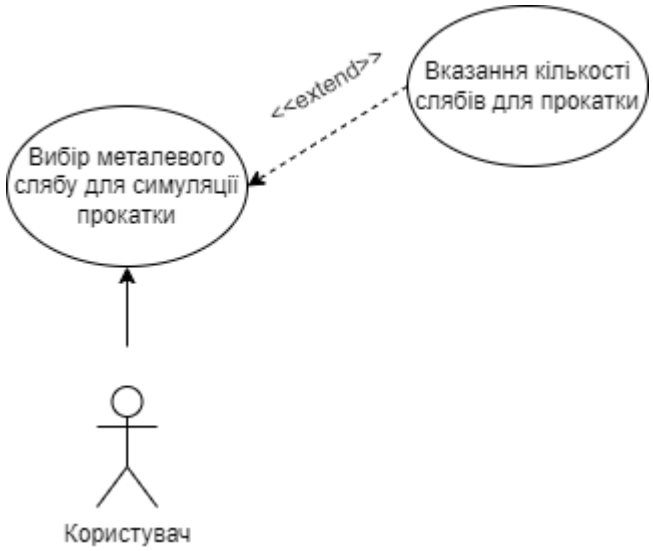


Рисунок 2.1.2 – Діаграма прецедентів програми симуляції прокатки металевих слябів

2.2 Внутрішнє проектування

2.2.1 Аналіз зовнішніх специфікацій системи

2.2.1.1 Моделювання словника системи

Розглянувши різні варіанти взаємодії користувача з програмою можемо виділити базові сутності для програми перегляду деталей прокатного стану.

Виділені сутності:

З’єднання з сервером, Команди сервера, Налаштування сервера, Додавання деталей, Перегляд історії деталі, Перегляд навантажень деталі, Заміна деталі, Відправка деталі у смітник, Вибраний рядок у таблиці, Перегляд деталей у стані, Вікно з’єднання з сервером, Перегляд прокатаних смуг, Вікно вибору дій.

Виділені обов’язки:

З’єднання з сервером – основний клас для з’єднання клієнту з сервером та надання функцій для роботи з ним.

Команди сервера – клас-помічник для класу «З’єднання з сервером», зчитує та зберігає у собі команди для серверу, які записані у файлі.

Налаштування сервера – клас-помічник для класу «З’єднання з сервером», зчитує та зберігає у собі налаштування підключення серверу.

Додавання деталей – основний клас для додавання будь-якої деталі на склад з вказанням кількості та максимальним пробігом, у випадку з валком.

Перегляд історії деталі – основний клас для перегляду історії замін по вибраній деталі.

Перегляд навантажень деталі – основний клас для перегляду навантажень та швидкості роботи вибраної деталі.

Заміна деталі – основний клас для заміни вибраної деталі в прокатному стані на таку саму зі складу.

Відправка деталі у смітник – основний клас для списання вже зламаної деталі та відправки її у смітник.

Вибраний рядок у таблиці – клас-помічник для різних класів, але в основному для «Перегляд деталей у стані», записує рядки, які вибирає користувач при використанні таблиць з інформацією.

Перегляд деталей у стані – основний клас для перегляду повного обладнання прокатного стану, історії, навантажень та швидкостей, заміни деталей та відправки на списання, тобто у смітник.

Вікно з’єднання з сервером – основний клас для взаємодії з користувачем для отримання від нього, або надання йому інформації про сервер та його налаштування.

Перегляд прокатаних смуг – основний клас для перегляду вже прокатаних смуг прокатним станом.

Вікно вибору дій – основний клас головного меню для надання можливості вибору дій над прокатним станом користувачем.

Атрибути та методи для виконання обов’язків для програми перегляду деталей прокатного стану наведені в табл. 2.2.1.

Таблиця 2.2.1 – Перегляд прокатного стану

Сутність	Атрибути	Методи
З’єднання з сервером	serverCommandsPath – константа, що зберігає шлях до файлу з командами; sqlConnection – поле, що зберігає підключення до серверу; serverSettings – поле, що зберігає в собі налаштування серверу; serverCommands – поле, що зберігає в собі запити до серверу;	DatabaseConnection – конструктор створення класу, записує дані в поля; ~DatabaseConnection – деструктор класу, закриває з’єднання з сервером під час знищення; IsConnectionOk – перевіряє статус з’єднання; GetSelectDetail – універсальна функція для вибірки деталей за розташуванням; GetSelectRoller – вибірка валків; GetSelectShaft – вибірка шпинделів; GetSelectGearCage – вибірка шестеренних клітей; GetSelectClutch – вибірка муфт;

Продовження таблиці 2.2.1

	connectionString – поле в якій зберігається строка підключення до серверу; ConnectionTime – поле, що зберігає час в який було з'єднано клієнт з сервером; sqlDataAdapter – поле, що зберігає дані з серверу у зрозумілому форматі для програми; dataTable – поле, в яке конвертуються отримані дані в табличний формат; sqlCommand – поле, яке зберігає результат запиту.	GetSelectReductionGear – вибірка редукторів; GetSelectElectricMotor – вибірка електродвигунів; GetSelectStand – вибірка повного робочого стенду; GetDetailLog – вибірка навантажень та швидкостей по вибраній деталі; GetDetailHistory – вибірка історії по вибраній деталі; SwapFunc – універсальна функція заміни старої деталі на нову зі складу; RollerSwap – заміна старого валку на новий; ShaftSwap – заміна старого шпинделя на новий; GearCageSwap – заміна старої шестеренної кліті на нову; ClutchSwap – заміна старої муфти на нову; ReductionGearSwap – заміна старого редуктора на новий; ElectricMotorSwap – заміна старого електродвигуна на новий; InsertFunc – універсальна функція додавання деталей на склад; InsertRoller – додавання валку до складу;

Продовження таблиці 2.2.1

		InsertShaft – додавання шпинделя до складу; InsertGearCage – додавання шестеренної кліті до складу; InsertClutch – додавання муфти до складу; InsertReductionGear – додавання редуктора до складу; InsertElectricMotor – додавання електродвигуна до складу; SelectStrips – вибірка прокатаних металевих смуг; RelocationFunc – універсальна функція для зміни місцерозташування деталі; RollerRelocation – зміна місцерозташування валку; ShaftRelocation – зміна місцерозташування шпинделя; GearCageRelocation – зміна місцерозташування шестеренної кліті; ClutchRelocation – зміна місцерозташування муфти; ReductionGearRelocation – зміна місцерозташування редуктора; ElectricMotorRelocation – зміна місцерозташування електродвигуна.

Продовження таблиці 2.2.1

Команди сервера	SelectRoller – поле, яке зберігає процедуру виведення валків у строковому форматі для подальшого використання; SelectShaft – поле, яке зберігає процедуру виведення шпинделів у строковому форматі для подальшого використання; SelectGearCage – поле, яке зберігає процедуру виведення шестеренних клітей у строковому форматі для подальшого використання; SelectClutch – поле, яке зберігає процедуру виведення муфт у строковому форматі для подальшого використання; SelectReductionGear – поле, яке зберігає процедуру виведення	

Продовження таблиці 2.2.1

	редукторів у строковому форматі для подальшого використання; SelectElectricMotor – поле, яке зберігає процедуру виведення електродвигунів у строковому форматі для подальшого використання; SelectStand – поле, яке зберігає процедуру виведення повного робочого стану в строковому форматі для подальшого використання; SelectDetailLog – поле, що зберігає процедуру виведення навантажень та швидкостей по будь-якій деталі у строковому форматі; SelectDetailHistory – поле, що зберігає процедуру виведення	

Продовження таблиці 2.2.1

	історії по будь-якій деталі у строковому форматі; RollerSwap – поле процедури заміни валків у строковому форматі; ShaftSwap – поле процедури заміни шпинделів у строковому форматі; GearCageSwap – поле процедури заміни шестеренних клітей у строковому форматі; ClutchSwap – поле процедури заміни муфт у строковому форматі; ReductionGearSwap – поле процедури заміни редукторів у строковому форматі; ElectricMotorSwap – поле процедури заміни електродвигунів у строковому форматі; InsertRoller – поле процедури додавання	

Продовження таблиці 2.2.1

	валків на склад у строковому форматі; InsertShaft – поле процедури додавання шпинделів на склад у строковому форматі; InsertGearCage – поле процедури додавання шестеренних клітей на склад у строковому форматі; InsertClutch – поле процедури додавання муфт на склад у строковому форматі; InsertReductionGear – поле процедури додавання редукторів на склад у строковому форматі; InsertElectricMotor – поле процедури додавання електродвигунів на склад у строковому форматі; SelectStrips – поле процедури вибірки	

Продовження таблиці 2.2.1

	прокатаних металевих смуг у строковому форматі; RollerRelocation – поле процедури зміни місцерозташування валків у строковому форматі; ShaftRelocation – поле процедури зміни місцерозташування шпинделів у строковому форматі; GearCageRelocation – поле процедури зміни місцерозташування шестеренних клітей у строковому форматі; ClutchRelocation – поле процедури зміни місцерозташування муфт у строковому форматі; ReductionGearRelocation – поле процедури зміни місцерозташування	

Продовження таблиці 2.2.1

	редукторів у строковому форматі; ElectricMotorRelocation – поле процедури зміни місцерозташування електродвигунів у строковому форматі.	
Налаштування сервера	ServerName – поле назви серверу; DatabaseName – поле назви бази даних.	ServerSettings – конструктор створення класу та для надання значень полів за замовчуванням.
Додавання деталей	databaseConnection – поле класу з’єднання з сервером;	DetailAddWindow – конструктор для створення класу із заповненням полів; ComboBoxInit – заповнення comboBox даними для подальшого їх вибору; buttonBack_Click – закриття поточного вікна; buttonAdd_Click – додавання вказаної деталі на склад; comboBoxDetailType_SelectionChanged – змінює інтерфейсну частину, в залежності від вибору; textBoxMileage_PreviewTextInput – контроль введення тільки цифр.

Продовження таблиці 2.2.1

Перегляд історії деталей		DetailHistoryWindow – конструктор класу та вікна із заповненням таблиці історією деталі, яка була попередньо вибрана.
Перегляд навантажень деталі		DetailLogWindow – конструктор класу та вікна із заповненням таблиці навантаженнями та швидкостями деталі, яка була попередньо вибрана.
Заміна деталі	detailsWindow – поле, яке зберігає посилання на попереднє вікно для подальшого його використання; databaseConnection – поле, яке зберігає клас для з'єднання з сервером; selectedIdsGrid1 – поле класу із вибраними ідентифікаторами таблиці 1; selectedIdsGrid2 – поле класу із вибраними ідентифікаторами таблиці 2.	DetailSwapWindow – конструктор класу та вікна із заповненням полів змінними; buttonClose_Click – закриття поточного вікна; TimePickerInit – ініціалізація та заповнення comboBox та timePicker даними для вибору, а також, часом відкриття поточного вікна для подальшого використання в якості даних при заміні деталі; buttonConfirm_Click – підтвердження факту заміни деталі.

Продовження таблиці 2.2.1

Відправка деталі у смітник	databaseConnection – поле, яке зберігає клас для з'єднання з сервером; selectedIds – поле класу із вибраними ідентифікаторами таблиці 2; detailsWindow – поле, яке зберігає посилання на попереднє вікно для подальшого його використання.	DetailToGarbageWindow – конструктор класу та вікна із заповненням полів вхідними даними; buttonConfirm_Click – відправка вказаної деталі у смітник; buttonBack_Click – закриття поточного вікна.
Вибраний рядок у таблиці	DetailId – поле зберігає унікальний ідентифікатор деталі; DetailTypeId – поле зберігає ідентифікатор типу деталі; DetailLocationId – поле зберігає ідентифікатор місцерозташування деталі; DetailStandTypeId – поле зберігає	ToString – перевантаження стандартної функція для отримання інформації про клас; GetTitle – надає коротку інформацію про вибрану деталь.

Продовження таблиці 2.2.1

	ідентифікатор типу кліті; DetaiName – поле зберігає назву деталі.	
Перегляд деталей у стані	databaseConnection – поле зберігає клас для з'єднання з сервером; selectedIdsJournal – поле зберігає рядок, який був обраний останній у будь-якій таблиці; selectedIdsGrid1 – поле зберігає рядок, який був обраний останній у першій таблиці; selectedIdsGrid2 – поле зберігає рядок, який був обраний останній у другій таблиці.	DetailsWindow – конструктор класу та вікна із заповненням полів вхідними даними; ClockWorking – виведення годинника на форму; ComboBoxStandInit – ініціалізація comboBox за типом кліті; ComboBoxTypeInit – ініціалізація comboBox за типом перегляду; buttonBack_Click – закрити поточне вікно та повернутися на попереднє; GetIds – отримання обраних користувачем ідентифікаторів за вказаною таблицею; buttonLog_Click – відкриття форми для перегляду навантажень та швидкостей обраної деталі; buttonHistory_Click – відкриття форми для перегляду історії обраної деталі; buttonDetailSwap_Click – відкриття форми для заміни старої деталі на нову;

Продовження таблиці 2.2.1

		buttonToGarbage_Click – відкриття форми для відправки зламаної деталі у смітник; RollerDataGrid – заповнення першої таблиці інформацією про встановлені валки в прокатному стані та другої про запасні деталі такого типу на складі; ShaftDataGrid – заповнення першої таблиці інформацією про встановлені шпинделя в прокатному стані та другої про запасні деталі такого типу на складі; GearCageDataGrid – заповнення першої таблиці інформацією про встановлені шестеренні кліті в прокатному стані та другої про запасні деталі такого типу на складі; ClutchDataGrid – заповнення першої таблиці інформацією про встановлені муфти в прокатному стані та другої про запасні деталі такого типу на складі; ReductionGearDataGrid – заповнення першої таблиці інформацією про встановлені редуктори в прокатному стані та другої про запасні деталі такого типу на складі;

Продовження таблиці 2.2.1

		ElectricMotorDataGrid – заповнення першої таблиці інформацією про встановлені електродвигуни в прокатному стані та другої про запасні деталі такого типу на складі; StandDataGrid – заповнення першої таблиці інформацією про повний набір деталей за вибраною групою клітей; Grid1VisualImprove – візуальне покращення першої таблиці; Grid2VisualImprove – візуальне покращення другої таблиці; DataGridRefresh – інформаційне оновлення таблиць; VariablesRefresh – оновлення строкових індексів; FullStandProposition – виведення пропозиції заміни деталі при повному відображені групи кліті; comboBoxStand_SelectionChanged – оновлення інформації на формі при зміні обраних індексів у comboBox; comboBoxType_SelectionChanged – оновлення інформації на формі при зміні обраних індексів у comboBox; dataGridRollers1_SelectionChanged – оновлення останньо-обраного рядку,

Продовження таблиці 2.2.1

		останньо-обраного рядку за першою таблицею та оновлення пропозицій заміни деталей при зміні індексу в таблиці; dataGridRollers2_SelectionChanged – оновлення останньо-обраного рядку та останньо-обраного рядку за другою таблицею при зміні індексу в таблиці; dataGridRollers1_PreviewMouseDown – оновлення останньо-обраного рядку при натисненні кнопки миші; dataGridRollers2_PreviewMouseDown– оновлення останньо-обраного рядку при натисненні кнопки миші.
Вікно з'єднання з сервером	serverSettingsPath – константа зберігає шлях до файлу з налаштування серверу; serverSettings – поле клас, якого зберігає дані з налаштуванням серверу.	SqlConnectionWindow – конструктор класу та вікна, що заповняє поля при створені; buttonConnect_Click – підключення до серверу.
Перегляд прокатаних смуг	databaseConnection – поле зберігає клас для з'єднання з сервером.	StripsWindow – конструктор класу та вікна із заповненням таблиці вже прокатаними металевими смугами.

Продовження таблиці 2.2.1

Вікно вибору дії	databaseConnection – поле зберігає клас для з’єднання з сервером.	MainWindow – конструктор класу та вікна із заповненням полів; ClockWorking – виведення годинника на форму; buttonDetails_Click – вікно перегляду встановлених деталей; buttonAddDetails_Click – вікно додавання деталей на склад; buttonStrips_Click – вікно перегляду вже прокатаних металевих смуг.

Розглянувши різні варіанти взаємодії користувача з програмою можемо виділити базові сутності для програми симуляції прокатки металевих слябів.
Виділені сутності:

З’єднання з сервером, Команди сервера, Налаштування сервера, Прокатка металевого слябу, Вікно вибору металевого слябу.

Виділені обов’язки:

З’єднання з сервером – основний клас для з’єднання клієнту з сервером та надання функцій для роботи з ним.

Команди сервера – клас-помічник для класу «З’єднання з сервером», зчитує та зберігає у собі команди для серверу, які записані у файлі.

Налаштування сервера – клас-помічник для класу «З’єднання з сервером», зчитує та зберігає у собі налаштування підключення серверу.

Прокатка металевого слябу – основний клас, який реалізує прокатку вже вибраного металевого слябу та за вказаною його кількістю.

Вікно вибору металевого слябу – основний клас, який надає можливість вибору металевого слябу та його кількості для подальшої прокатки.

Атрибути та методи для виконання обов’язків для програми симуляції прокатки наведені в табл. 2.2.2.

Таблиця 2.2.2 – Перегляд симуляції прокатки

Сутність	Атрибути	Методи
З’єднання з сервером	dbSettingsPath – константа, що зберігає шлях до файлу з налаштуванням серверу; dbCommandsPath – константа, що зберігає шлях до файлу з командами; dbSettings – поле, що зберігає клас з налаштуваннями сервера; dbCommands – поле, що зберігає клас з командами до серверу; sqlConnection – поле, що зберігає з’єднання з сервером.	DataConnection – конструктор класу із заповненням полів; IsConnectionOk – перевірка статусу з’єднання з сервером; GetSelectSteel – вибірка металевих слябів, які можуть бути прокатаними; RollingSteel – прокатка металевого слябу та перетворення його на смугу; IncrementRolledStrips – збільшення лічильника кількості прокатаних слябів; GetRollingTime – визначення часу, як довго буде відбуватися прокатка одного слябу; GetLastStrip – отримання ідентифікатора останньої смуги, яка була прокатана; GetStandIds – отримання ідентифікаційних номерів за вказаним місцерозташуванням; GetLoads – отримання навантажень за вказаним номером металевого слябу;

Продовження таблиці 2.2.2

		InsertLogOneDetail – додавання запису навантажень та швидкостей в базу даних для вказаної деталі; InsertLog – додавання записів навантажень та швидкостей на цілий ряд деталей; DetailLogging – додавання записів навантажень та швидкостей на цілий прокатний стенд; IncreaseMileageTwoRoller – збільшення прокатного шляху для пари валків; IncreaseMileageRollers – збільшення прокатного шляху для всіх валків у прокатному стенді.
Команди сервера	SelectSteel – поле, яке зберігає процедуру виведення металевих слябів, які можна прокатати у строковому форматі; RollingSteel – поле, яке зберігає процедуру прокатки металевих	

Продовження таблиці 2.2.2

	слябів у строковому форматі; IncrementRolledStrips – поле, яке зберігає процедуру збільшення кількості прокатаних смуг у строковому форматі; GetRollingTime – поле, яке зберігає процедуру отримання часу прокатки одного металевого слябу у строковому форматі; SelectStand – поле, яке зберігає процедуру отримання ідентифікаторів вибраних деталей за розташуванням у строковому форматі; SelectLoads – поле, яке зберігає процедуру отримання навантажень по вибраному металевому слябу у строковому форматі;	

Продовження таблиці 2.2.2

	InsertDetailLog – поле, яке зберігає процедуру введення запису навантажень на швидкостей у базу даних у строковому форматі; GetLastStrip – поле, яке зберігає процедуру отримання металевого слябу прокатаного найостаннішим у строковому форматі; IncreaseMileageRoller – поле, яке зберігає процедуру збільшення пробігу валку у строковому форматі.	
Налаштування сервера	ServerName – поле, яке зберігає назву серверу; DatabaseName – поле, яке зберігає назву бази даних.	
Прокатка металевого слябу	dataConnection – поле, яке зберігає клас для з’єднання з сервером; selectedId – поле, яке зберігає ідентифікатор	RollingSteelWindow – конструктор класу та вікна із заповнення полів; RollingBlock – прокатка вказаних металевих слябів;

Продовження таблиці 2.2.2

	обраного металевого слябу; rollingCount – поле, яке зберігає кількість металевих слябів, які потрібно прокатати.	buttonConfirm_Click – підтвердження та запуск функції прокатки; buttonBack_Click – закриття поточного вікна.
Вікно вибору металевого слябу	dataConnection – поле, яке зберігає клас для з'єднання з сервером; selectedId	MainWindow – конструктор класу та вікна із заповнення полів; DatabaseInit – створення з'єднання з базою даних; buttonRolling_Click – відкриття вікна для прокатки металевих слябів; dataGridSteel_SelectionChanged – зміна ідентифікатора металевого слябу вибраного користувачем.

2.2.1.2 Моделювання розподілу обов'язків у системі

Класи, які працюють спільно для досягнення різного типу задач різної складності для програми перегляду деталей у прокатному стані:

- З'єднання з сервером, Команди сервера, Налаштування сервера, Вікно з'єднання з сервером – відповідають за з'єднання з сервером, його налаштування та запити;
- Додавання деталей, З'єднання з сервером, Вікно вибору дій – надають можливість за допомогою графічного інтерфейсу додавати отримані деталі на склад;
- Перегляд історії деталі, З'єднання з сервером, Вибраний рядок у таблиці,

Перегляд деталей у стані – виведення історії деталі у графічну таблицю;

- Перегляд навантажень деталі, З’єднання з сервером, Вибраний рядок у таблиці, Перегляд деталей у стані – виведення інформації щодо навантажень та швидкостей деталі у графічну таблицю;
- Заміна деталі, З’єднання з сервером, Вибраний рядок у таблиці, Перегляд деталей у стані – заміна обраної старої деталі на обрану нову;
- Відправка деталі у смітник, З’єднання з сервером, Вибраний рядок у таблиці, Перегляд деталей у стані – відправка обраної зламаної деталі у смітник;
- Перегляд деталей у стані, З’єднання з сервером, Вікно вибору дій – показ встановлених деталей у прокатний стан за вказаною групою клітей;
- Перегляд прокатаних смуг, З’єднання з сервером, Вікно вибору дій – показ вже прокатаних металевих смуг.

Моделюємо типи зв’язків у табл. 2.2.4

Таблиця 2.2.4 – Моделювання залежності

Клас, що зв’язується	Клас із котрим зв’язуються	Тип зв’язку
З'єднання з сервером	Команди сервера	Асоціація
	Налаштування сервера	Асоціація
Перегляд історії деталі	З’єднання з сервером	Асоціація
	Вибраний рядок у таблиці	Асоціація
Перегляд навантажень деталі	З’єднання з сервером	Асоціація
	Вибраний рядок у таблиці	Асоціація
Заміна деталі	З’єднання з сервером	Асоціація
	Вибраний рядок у таблиці	Асоціація

Продовження таблиці 2.2.4

Відправка деталі у смітник	З’єднання з сервером	Асоціація
	Вибраний рядок у таблиці	Асоціація
Перегляд деталей у стані	З'єднання з сервером	Асоціація
	Перегляд історії деталі	Асоціація
	Перегляд навантажень деталі	Асоціація
	Заміна деталі	Асоціація
	Відправка деталі у смітник	Асоціація
Перегляд прокатаних смуг	З'єднання з сервером	Асоціація
Вікно вибору дій	З'єднання з сервером	Асоціація
	Перегляд деталей у стані	Асоціація
	Перегляд прокатаних смуг	Асоціація

Класи, які працюють спільно для досягнення різного типу задач різної складності для програми симуляції прокатки металевих слябів:

- З’єднання з сервером, Команди сервера, Налаштування сервера – надають можливість відправляти та отримувати запити різного призначення;
- Прокатка металевого слябу, З’єднання з сервером – прокатка вибраного металевого слябу за вказаною кількістю.
- Вікно вибору металевого слябу, З’єднання з сервером – можливість перегляду та вибору металевого слябу для прокатки із вказанням кількості.

Моделюємо типи зв’язків у табл. 2.2.5

Таблиця 2.2.5 – Моделювання залежності

Клас, що зв’язується	Клас із котрим зв’язуються	Тип зв’язку
З’єднання з сервером	Команди сервера	Асоціація
	Налаштування сервера	Асоціація
Вікно вибору металевого слябу	З’єднання з сервером	Асоціація
	Прокатка металевого слябу	Асоціація
Прокатка металевого слябу	З’єднання з сервером	Асоціація

2.2.1.3 Визначення призначень об’єктів за допомогою CRC карток

Класи можуть бути представлені у вигляді CRC карток. Розглянемо ці картки для програми перегляду прокатного стану у табл. 2.2.6 – 2.2.18.

Таблиця 2.2.6 – CRC карта класу «З'єднання з сервером»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає зв’язок з сервером	Команди сервера Налаштування сервера

Таблиця 2.2.7 – CRC карта класу «Команди сервера»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Зберігає запити для подальшого використання	Немає

Таблиця 2.2.8 – CRC карта класу «Налаштування сервера»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Зберігає назву сервера та назву бази даних	Немає

Таблиця 2.2.9 – CRC карта класу «Додавання деталей»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість додавати нові деталі на склад	З’єднання з сервером

Таблиця 2.2.10 – CRC карта класу «Перегляд історії деталей»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість переглянути історію обраної деталі	З’єднання з сервером Вибраний рядок у таблиці

Таблиця 2.2.11 – CRC карта класу «Перегляд навантажень деталі»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість переглянуть історію навантажень та швидкостей обраної деталі	З’єднання з сервером Вибраний рядок у таблиці

Таблиця 2.2.12 – CRC карта класу «Заміна деталі»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість замінити стару деталь на нову такого ж типу	З’єднання з сервером Вибраний рядок у таблиці Перегляд деталей у стані

Таблиця 2.2.13 – CRC карта класу «Відправка деталі у смітник»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість списати деталь та відправити її у смітник	З’єднання з сервером Вибраний рядок у таблиці

Таблиця 2.2.14 – CRC карта класу «Вибраний рядок у таблиці»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає програмі розуміння, який рядок у таблиці був обраний користувачем	Немає

Таблиця 2.2.15 – CRC карта класу «Перегляд деталей у стані»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість переглянути будь-яку встановлену деталь у прокатному стані	З’єднання з сервером Вибраний рядок у таблиці Перегляд історії деталей Перегляд навантажень деталі Заміна деталі Відправка деталі у смітник

Таблиця 2.2.16 – CRC карта класу «Вікно з’єднання з сервером»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість користувачу підключитись до вказаного ним серверу	З’єднання з сервером Вікно вибору дій

Таблиця 2.2.17 – CRC карта класу «Перегляд прокатаних смуг»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість переглянути смуги, які вже було прокатані	З’єднання з сервером

Таблиця 2.2.18 – CRC карта класу «Вікно вибору дій»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість користувачу вибрати дію: переглянути смуги, додати деталі на склад або переглянути деталі на прокатному стані	З’єднання з сервером Перегляд прокатаних смуг Перегляд деталей у стані Додавання деталей

Розглянемо CRC картки для програми симуляції прокатки металевих слябів у табл. 2.2.19 – 2.2.23.

Таблиця 2.2.19 – CRC карта класу «З'єднання з сервером»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає зв’язок з сервером	Команди сервера Налаштування сервера

Таблиця 2.2.20 – CRC карта класу «Команди сервера»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Зберігає запити для подальшого використання	Немає

Таблиця 2.2.21 – CRC карта класу «Налаштування сервера»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Зберігає назву сервера та назву бази даних	Немає

Таблиця 2.2.22 – CRC карта класу «Прокатка металевого слябу»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Виконує симуляцію прокатки вибраного металевого слябу у вказаній кількості	З’єднання з сервером

Таблиця 2.2.23 – CRC карта класу «Вікно вибору металевого слябу»

Базовий клас	Похідні класи (нащадки)
Немає	Немає
Обов’язки	Зв’язки
Надає можливість вибрати сляб, по якому потрібно виконати симуляцію прокатки за вказаною кількістю	З’єднання з сервером Прокатка металевого слябу

2.2.1.4 Побудова об’єктної моделі (діаграма класів)

Представимо діаграму класів, як заключний етап моделювання програми перегляду деталей у прокатному стані (рис. 2.2.1.4).

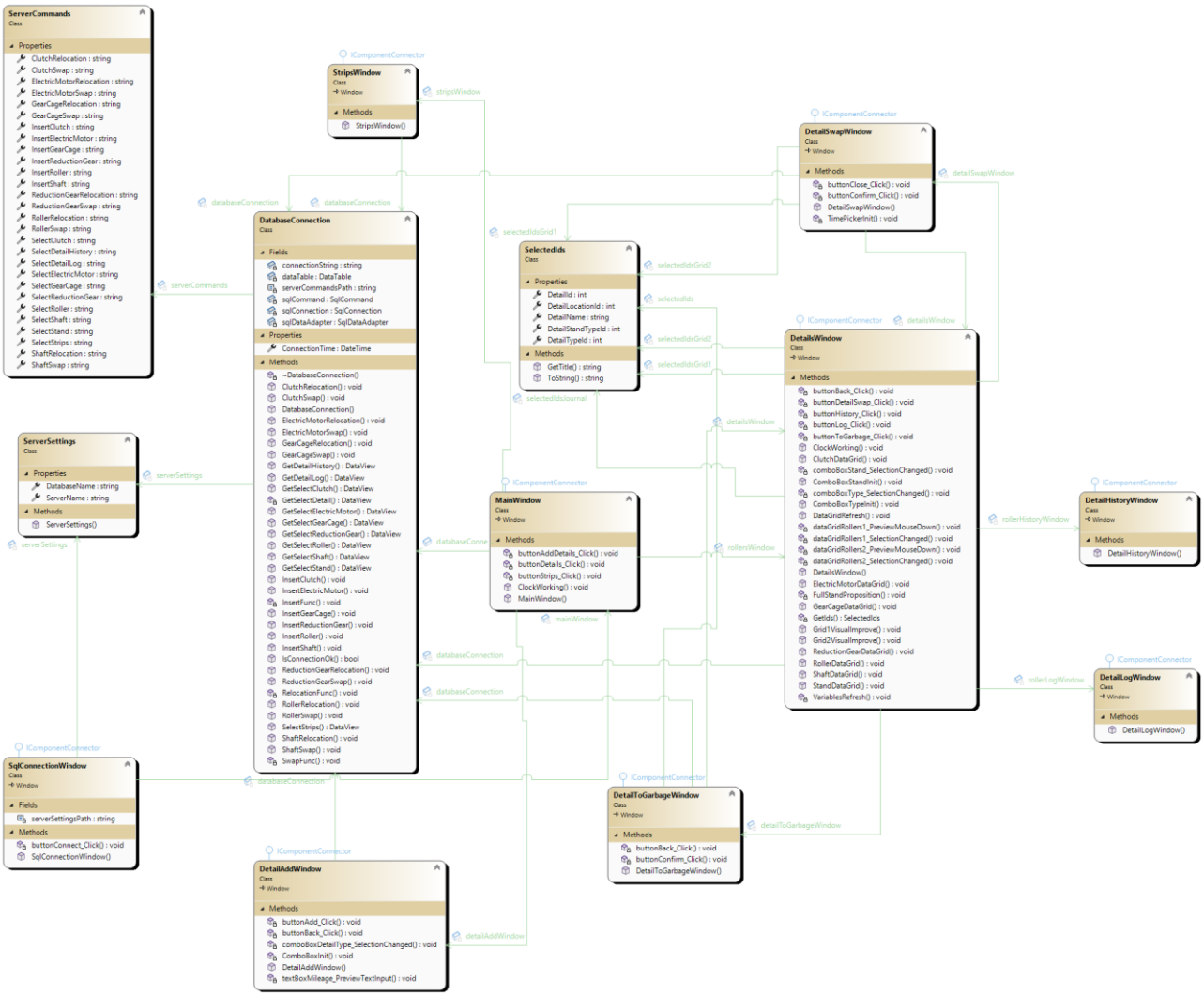


Рисунок 2.2.1.4 – Діаграма класів. Перегляд деталей на стані.

Представимо діаграму класів, як заключний етап моделювання програми симуляції прокатки металевих слябів (рис. 2.2.1.5).

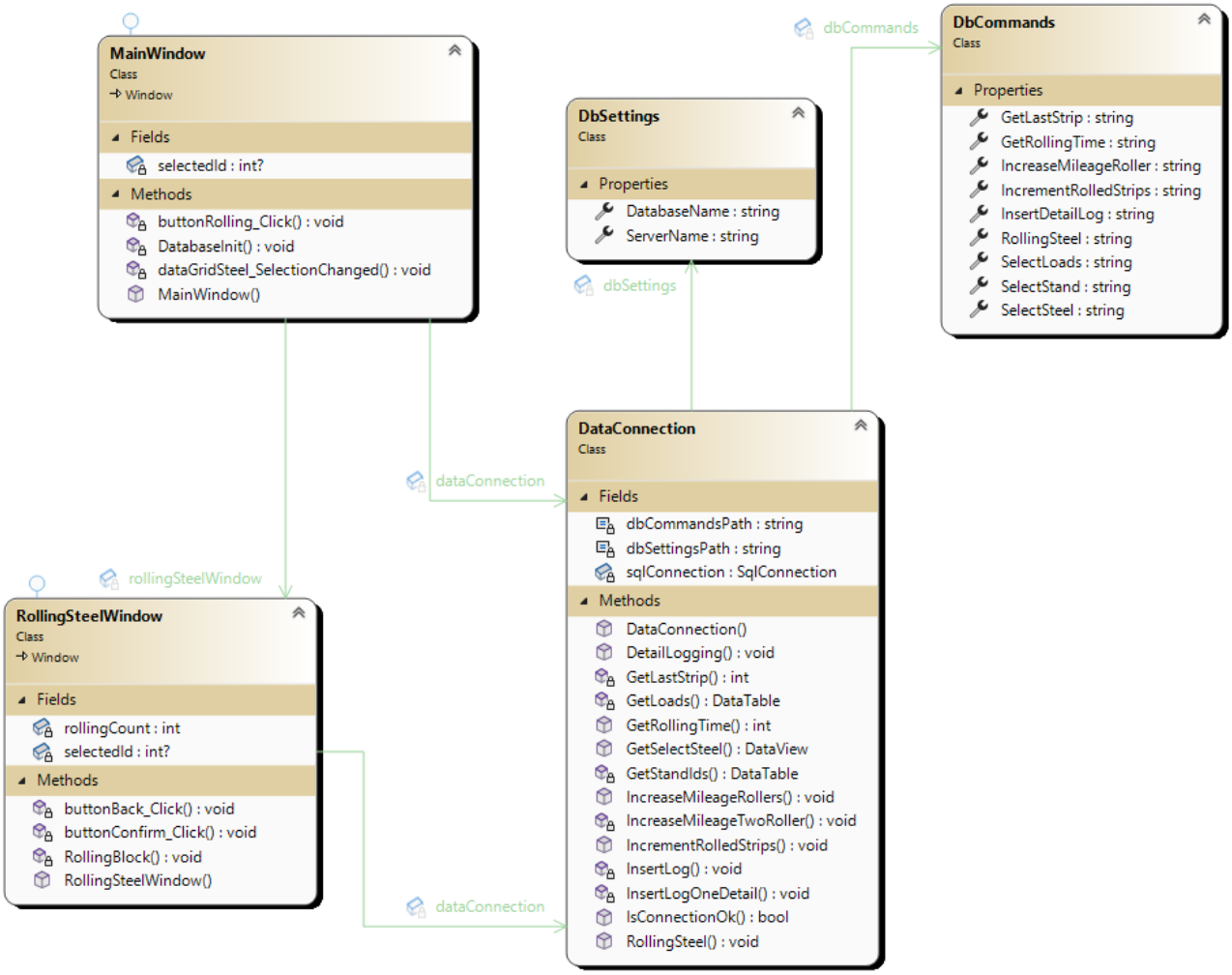


Рисунок 2.2.1.5 – Діаграма класів. Симуляція прокатки металевих слябів.

2.2.2 Проектування інтерфейсу користувача

Розглядаючи діаграму прецедентів рис. 2.1.1 можна зробити висновок, що окремі форми застосунку повинні виконувати наступні функції для програми перегляду прокатного стану:

- Вікно вибору дій;
- Перегляд прокатаних смуг;
- Додавання деталей на склад;
- Перегляд стану встановлених деталей;
- Перегляд історії вказаної деталі;
- Перегляд навантажень та швидкостей вказаної деталі;

- Заміна вказаної деталі;
- Відправка вказаної деталі на металобрухт.

Таким чином програмний продукт для зручності користування повинен складатися з наступник вікон:

- Вікно головного меню (рис.2.2.2);
- Вікно додавання деталей (рис.2.2.3);
- Вікно перегляду прокатаних смуг (рис.2.2.4);
- Вікно перегляду встановлених деталей та складу (рис.2.2.5);
- Вікно історії деталі (рис.2.2.6);
- Вікно навантажень та швидкостей деталі (рис.2.2.7);
- Вікно заміни деталі (рис.2.2.8).

Ескізи форм для програми перегляду прокатного стану.

Вікно головного меню повинно надавати такі можливості як: перегляд деталей у прокатному стані, можливість додавання нових деталей на склад та перегляд прокатаних металевих слябів у смуги. Головне меню розділяє вкладений функціонал у програмі. Ескіз форми зображено на (рис. 2.2.2).

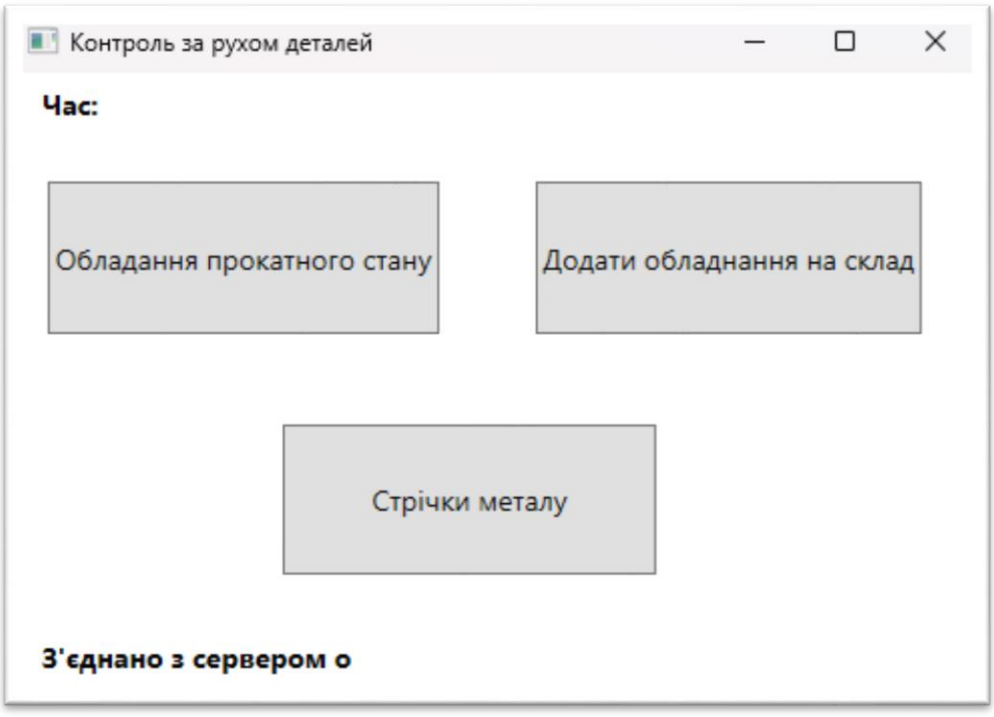


Рисунок 2.2.2 – Вікно головного меню

Вікно додавання деталі повинно надавати такі можливості як: можливість повернутись на попереднє вікно, вказання типу деталі та тип кліті до якої ця деталь відноситься, максимальний пробіг прокатки для валку наданий виробником цієї деталі та кількість деталей, яка буде додана на зберігання. Ескіз форми зображено на (рис 2.2.3).

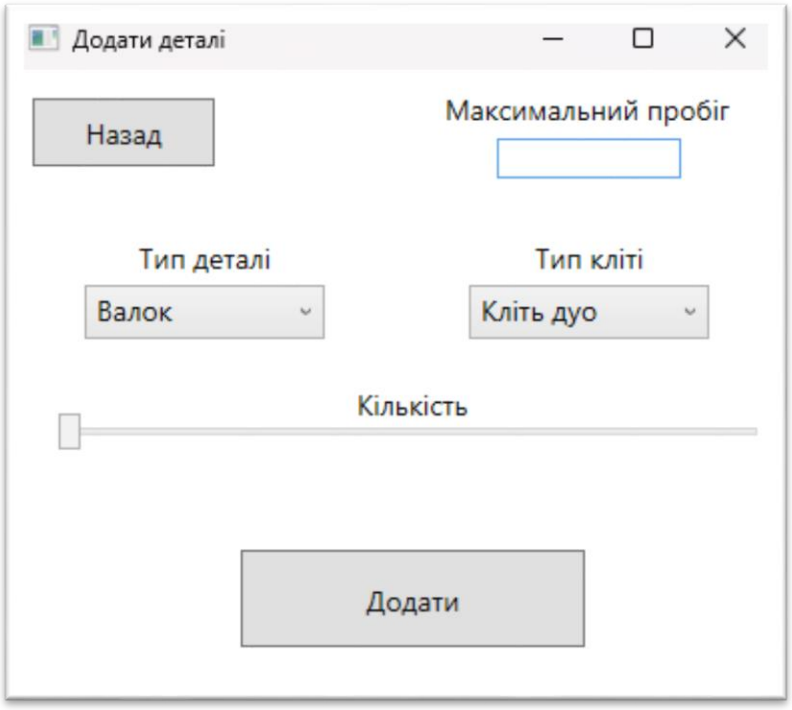


Рисунок 2.2.3 – Вікно додавання деталі

Вікно перегляду прокатаних смуг повинно надавати можливість перегляду вхідних параметрів слябу, вихідних параметрів вже смуги, початок і кінець прокатки слябу за його ідентифікаційним номером. Все це складає один рядок в табличному записі. Ескіз форми зображено на (рис. 2.2.4).

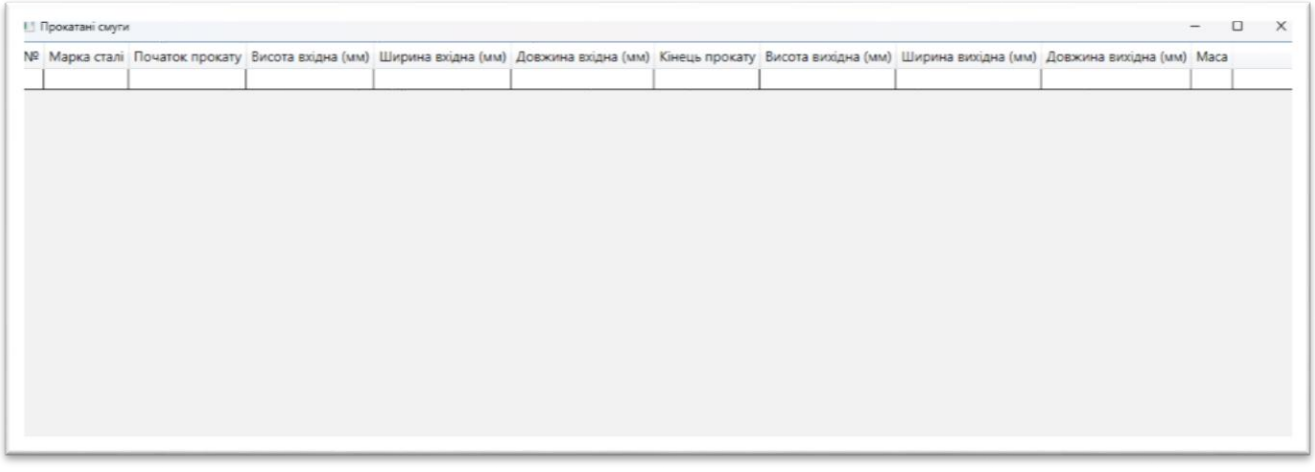


Рисунок 2.2.4 – Вікно перегляду прокатаних смуг

Вікно перегляду встановлених деталей є одним з найважливіших складових програми. Тому, воно повинно надавати найрізноманітніший функціонал одночасно. Такий як: обрання групи клітей для огляду та тип огляду, а саме, або огляд кліті та її деталей, або огляд тільки окремих деталей, виведення у ліву таблицю складових деталей прокатного стану, а у праву – деталей зі складу для майбутньої заміни, можливістю маніпулювання, а саме, вибору рядків таблиць для перегляду історії деталі, навантажень та швидкостей деталі, заміна обраної деталі на нову та відправка деталі обраної в правій таблиці на металобрухт. Також, інтерфейс повинен мати зворотній зв’язок з користувачем для покращення відчуття від користування. Ескіз форми зображено на (рис. 2.2.5).

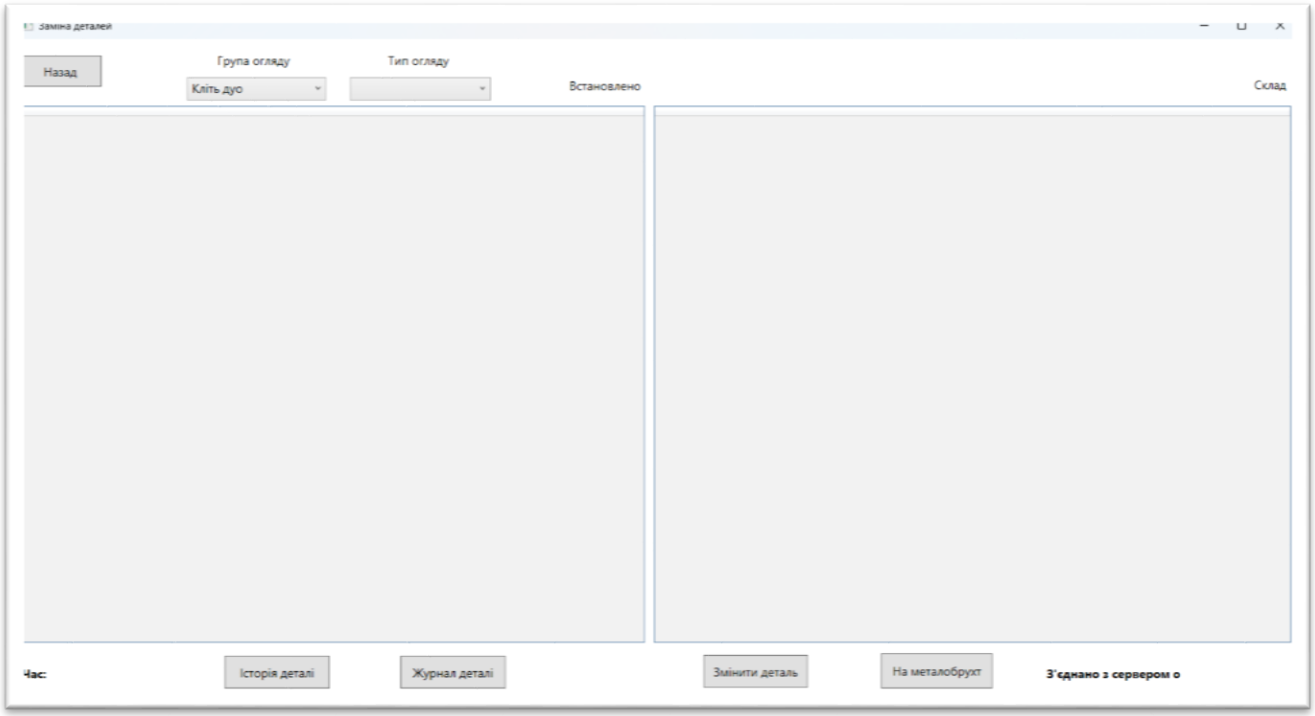


Рисунок 2.2.5 – Вікно перегляду встановлених деталей та складу

Вікно перегляду історії деталі повинно надавати можливість перегляду номеру запису, часу заміни деталі, місцерозташування звідки деталь була зняти, місцерозташування куди деталь була переміщена та вказання причина такої заміни. Кожна подія описується окремим записом у таблиці. Ескіз форми зображено на (рис 2.2.6).

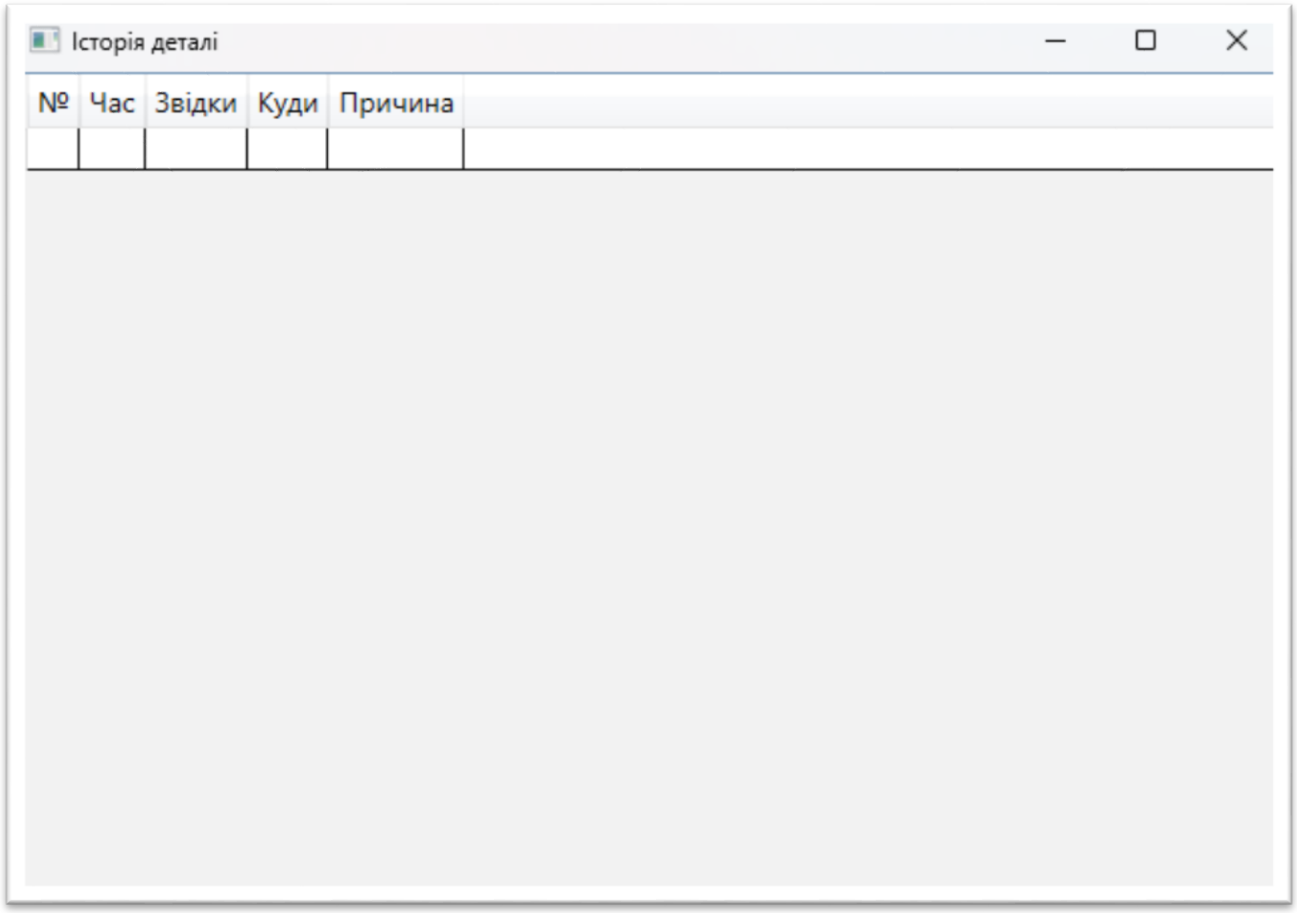


Рисунок 2.2.6 – Вікно перегляду історії

Вікно перегляду навантажень та швидкостей деталі повинно надавати можливість перегляду номеру запису, ідентифікатор смуги за якої виконувалась прокатка, марку сталі смуги, навантаження та швидкість у вказаний проміжок часу. Кожна подія описується окремим записом у таблиці. Ескіз форми зображено на (рис. 2.2.7).

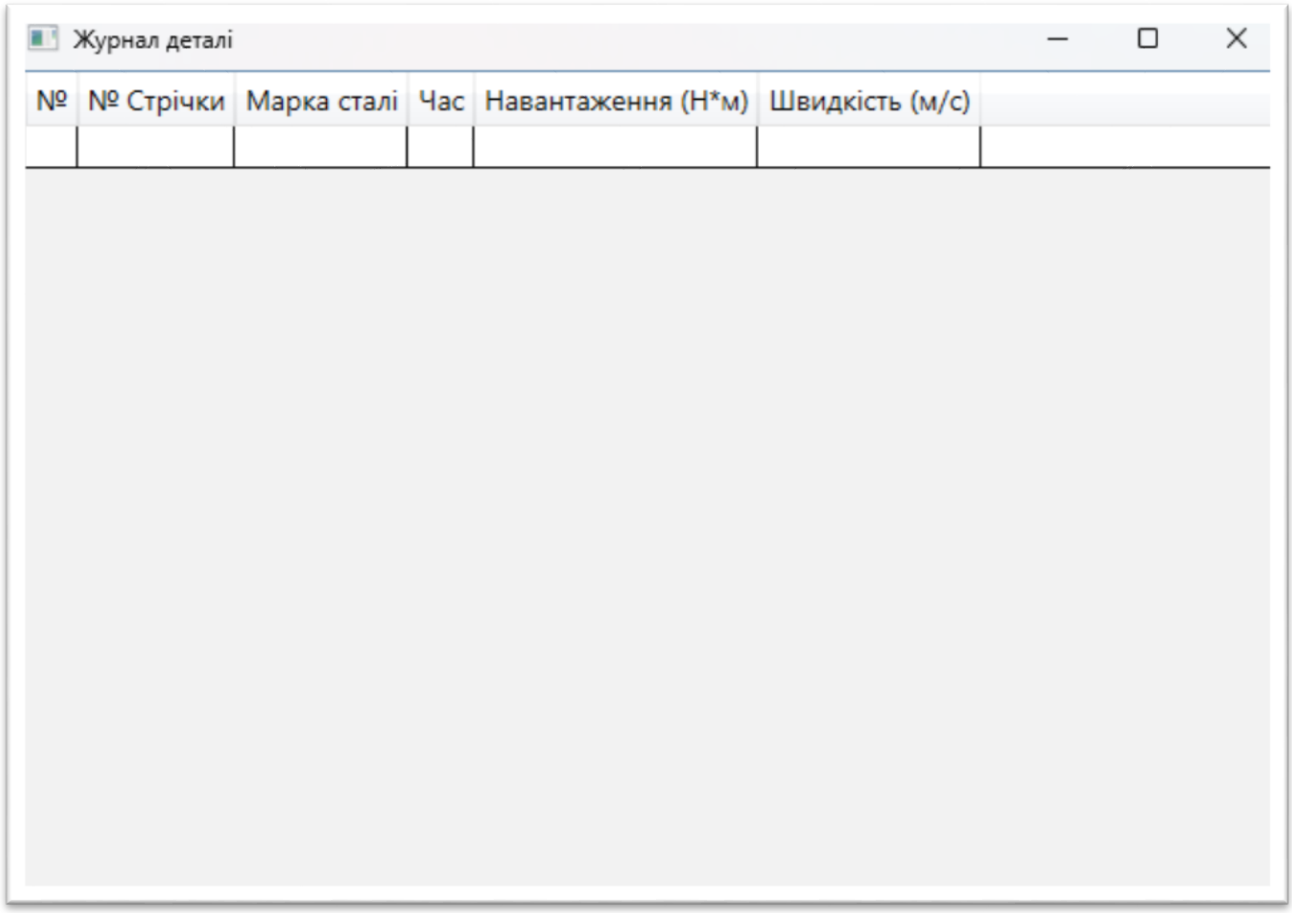


Рисунок 2.2.7 – Вікно перегляду навантажень та швидкостей деталі (журнал)

Вікно заміни деталей є однією з найважливіших частин сталої працездатності прокатного стану. Тому, воно повинно надавати можливість, за попередньо вибраними деталями, надати користувачу розуміння яка деталь на яку буде змінена, надати можливість вказати причину заміни та можливість вказати дату та час події включно до секунд. Ескіз форми зображено на (рис 2.2.8).

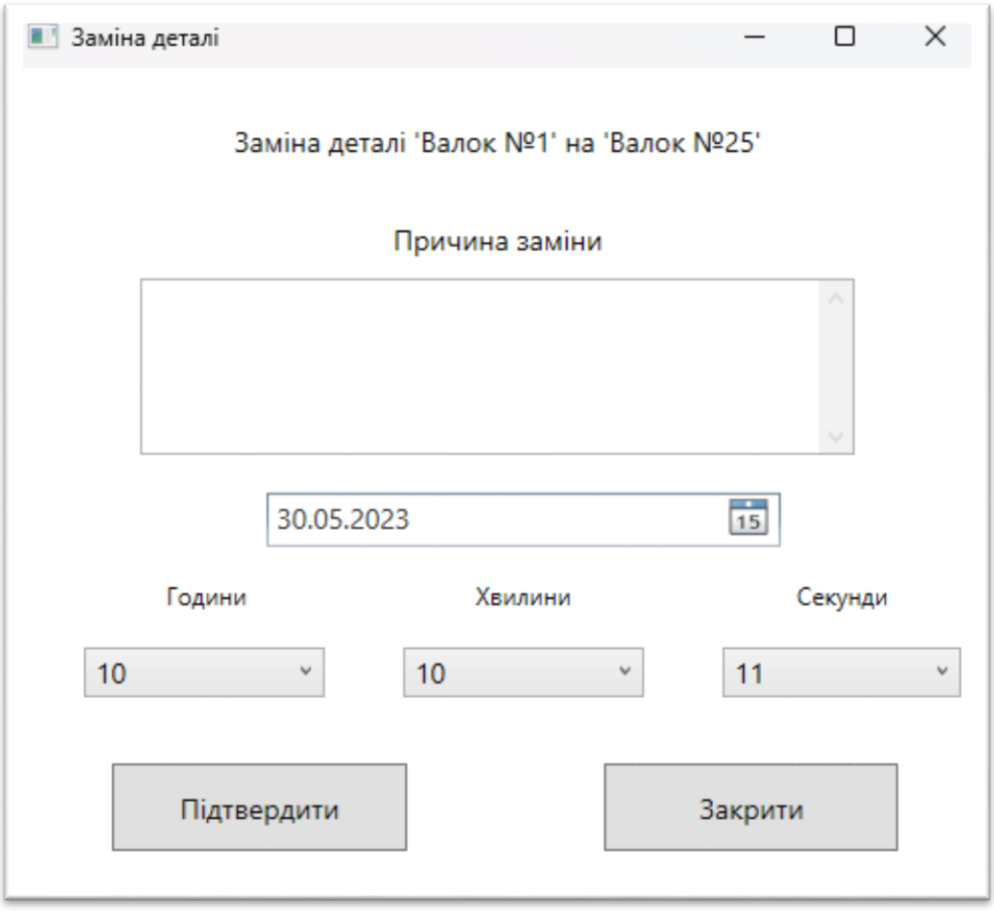


Рисунок 2.2.8 – Вікно заміни деталі

Розглянемо створену раніше діаграму прецедентів рис. 2.1.2 для програми симуляції прокатки металевих слябів та виділимо наступні функції:

- Вибір слябу для прокатки;
- Вибір кількості слябів.

Таким чином програма, яка буде виконувати симуляцію прокатки металевих слябів у смуги повинна мати наступне вікно:
Вікно вибору слябу.

Вікно вибори слябу також відповідає і за кількісну складову. Тобто, надає можливість зробити дві або більше прокаток обраних слябів та перетворити їх на смуги. Ескіз форми зображено на (рис. 2.2.9).

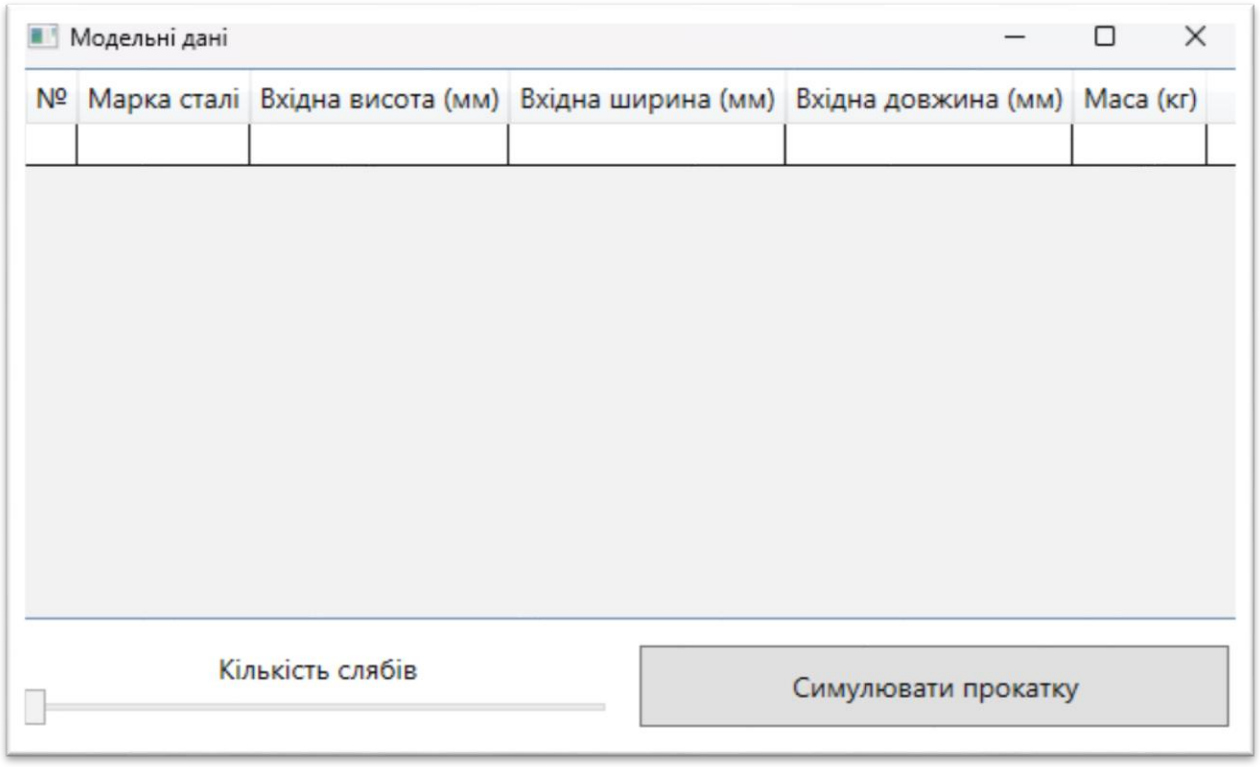


Рисунок 2.2.9 – Вікно симуляції прокатки слябу з вказанням кількості

2.2.3 Вибір мови програмування та бази даних

Для створення даного програмного забезпечення була обрана мова програмування C#. Ця мова є об’єктно-орієнтованою та використовує безпечну систему типізації. Синтаксис мови дуже схожий на такі мови як: C, C++ та Java, і це не є чимось незвичайним, бо ця мова була створена, як покращення недоліків пращурів, із вирішенням їх проблем та додаванням нових технічних можливостей. C# чудово підходить для розробки додатків різного розміру. Надає можливість співпрацювати з будь-якою базою даних, писати багатоплатформений код та співпрацювати з дуже зручним та функціональним графічним фреймворком. Вибір цієї мови для розробки програмного забезпечення є чудовим рішенням адже більшість комп’ютерів користувачів працює на операційній системі Windows, тому створювати програму на мові для операційної системи вона була створена дуже зручно та надійно.

В якості графічної підсистеми – WPF (Windows Presentation Foundation). Графічна підсистема, яка є більш сучасним та удосконаленим нащадком WinForms. Складається з декларативної мови розмітки XAML, що основана на

XML, і це значно спрощує створення інтерфейсу для користувача. Розробник завдяки мови розмітки може створювати, редагувати та видаляти елементи інтерфейсу не використовуючи візуальний редактор форми, який вбудований у Visual Studio. WPF надає можливість створювати інтерфейс, який легко масштабується та може бути дуже гарно оформлений розробником, що покращить відчуття від використання користувачем.

Системою управління базою даних було обрано Microsoft SQL Server 2022. Для запитів використовує мову Transact SQL. Вона чудово підходить для роботи з мовою програмування C#, бо має одного й того ж самого розробника. Тому, сумісність та функціонал у цьому випадку на найвищому рівні.

2.2.4 Схема бази даних, таблиці та процедури

2.2.4.1 Проектування таблиць

Під час проектування схеми бази даних за предметною областю були виділені наступні таблиці:

- RollingStandType – таблиця-довідник, що надає розуміння до якої групи клітей відноситься деталь;
- DetailType – таблиця-довідник, що надає розуміння якого типу обрана деталь;
- DetailLocation – таблиця-довідник, що надає розуміння яке місцерозташування має обрана деталь;
- DetailHistory – таблиця, яка зберігає дані змін місцерозташування по кожній деталі;
- Roller – таблиця, яка зберігає дані про валки та їх характеристики;
- Shaft – таблиця, яка зберігає дані про шпинделя та їх характеристики;
- GearCage – таблиця, яка зберігає дані про шестеренні кліті та їх характеристики;
- Clutch – таблиця, яка зберігає дані про муфти та їх характеристики;
- ReductionGear – таблиця, яка зберігає дані про редуктора та їх характеристики;

- ElectricMotor – таблиця, яка зберігає дані про електродвигуни та їх характеристики;
- Steel – таблиця, яка зберігає дані про металеві сляби;
- Strip – таблиця, яка зберігає дані про прокатані металеві сляби у смуги ;
- DetailLog – таблиця, яка зберігає дані про навантаження та швидкості роботи кожної деталі;
- Loads – таблиця, яка зберігає дані про навантаження та швидкості за кожним металевим слябом.

2.2.4.2 Проектування атрибутів

В ході аналізу прокатного стану та особливостей складського обліку були виділені такі атрибути:

- RollingStandType: id (PK), typeDescription;
- DetailType: id (PK), typeDescription;
- DetailLocation: id (PK), locationDescription;
- DetailHistory: id (PK), detailTypeId (FK), detailId, replaceTime, replaceFrom (FK), replaceTo (FK), replaceReason;
- Roller: id (PK), detailTypeId (FK), standTypeId (FK), detailLocationId (FK), installationDate, ironStripsDone, mileageCurrent, mileageMax;
- Shaft: id (PK), detailTypeId (FK), standTypeId (FK), detailLocationId (FK), installationDate, ironStripsDone, mileageCurrent, mileageMax;
- GearCage: id (PK), detailTypeId (FK), standTypeId (FK), detailLocationId (FK), installationDate, ironStripsDone, mileageCurrent, mileageMax;
- Clutch: id (PK), detailTypeId (FK), standTypeId (FK), detailLocationId (FK), installationDate, ironStripsDone, mileageCurrent, mileageMax;
- ReductionGear: id (PK), detailTypeId (FK), standTypeId (FK), detailLocationId (FK), installationDate, ironStripsDone, mileageCurrent, mileageMax;
- ElectricMotor: id (PK), detailTypeId (FK), standTypeId (FK), detailLocationId (FK), installationDate, ironStripsDone, mileageCurrent, mileageMax;
- Steel: id (PK), RTIME, steelGrade, inHeight, inWidth, inLength, countStrips,

outHeight, outWidth, outLength, mass;

- Strip: id (PK), steelId (FK), steelGrade, inHeight, inWidth, inLength, outHeight, outWidth, outLength, mass, productionStartTime, productionEndTime;
- DetailLog: id (PK), detailTypeId (FK), detailId, stripId (FK), logTime, valueLoad, valueRpm;
- Loads: id (PK), steelId (FK), load1, load2, load3, load4, load5, load6, load7, load8, load9, load10, load11, load12, speed1, speed2, speed3, speed4, speed5, speed6, speed7, speed8, speed9, speed10, speed11, speed12.

2.2.4.3 Проектування збережених процедур

Для сталого функціонування програми за предметною областю, були виділені наступні збережені процедури:

- SelectRoller – збережена процедура для надання вибірки валків із таблиці за вказаними параметрами;
- SelectShaft – збережена процедура для надання вибірки шпинделів із таблиці за вказаними параметрами;
- SelectGearCage – збережена процедура для надання вибірки шестеренних клітей із таблиці за вказаними параметрами;
- SelectClutch – збережена процедура для надання вибірки муфт із таблиці за вказаними параметрами;
- SelectReductionGear – збережена процедура для надання вибірки редукторів із таблиці за вказаними параметрами;
- SelectElectricMotor – збережена процедура для надання вибірки електродвигунів із таблиці за вказаними параметрами;
- SelectStand – збережена процедура для надання вибірки повного прокатного механізму за вказаними параметрами;
- SelectDetailLog – збережена процедура для надання вибірки навантажень та швидкостей за вказаною деталлю;
- SelectDetailHistory – збережена процедура для надання вибірки історії місцерозташувань за вказаною деталлю;

- AddDetailHistory – збережена процедура для додавання історії заміни на деталь;
- RollerSwap – збережена процедура для заміни старого валку на новий;
- ShaftSwap – збережена процедура для заміни старого шпинделю на новий;
- GearCageSwap – збережена процедура для заміни старої шестеренної кліті на нову;
- ClutchSwap – збережена процедура для заміни старої муфти на нову;
- ReductionGearSwap – збережена процедура для заміни старого редуктора на новий;
- ElectricMotorSwap – збережена процедура для заміни старого електродвигуна на новий;
- InsertRoller – збережена процедура для додавання нового валку в таблицю;
- InsertShaft – збережена процедура для додавання нового шпинделю в таблицю;
- InsertGearCage – збережена процедура для додавання нової шестеренної кліті в таблицю;
- InsertClutch – збережена процедура для додавання нової муфти в таблицю;
- InsertReductionGear – збережена процедура для додавання нового редуктора в таблицю;
- InsertElectricMotor – збережена процедура для додавання нового електродвигуна в таблицю;
- SelectStrips – збережена процедура для вибірки прокатаних металевих смуг;
- SelectSteel – збережена процедура для вибірки металевих слябів;
- RollingSteel – збережена процедура для прокатки металевого слябу в смугу;
- IncrementRolledStrips – збережена процедура для збільшення кількості прокатаних поліс на встановлених деталях;
- GetRollingTime – збережена процедура для підрахунку часу, який займає

прокатка слябу;

- InsertDetailLog – збережена процедура для додавання навантажень та швидкостей по кожній деталі у таблицю;
- SelectLoads – збережена процедура для отримання навантажень та швидкостей по кожному металевому слябу;
- GetLastStrip – збережена процедура для отримання ідентифікатора останньої прокатаної смуги;
- IncreaseMileageRoller – збережена процедура для збільшення прокатного пробігу на встановлених валках;
- RollerRelocation – збережена процедура для зміни місцерозташування валку;
- ShaftRelocation – збережена процедура для зміни місцерозташування шпинделю;
- GearCageRelocation – збережена процедура для зміни місцерозташування шестеренної кліті;
- ClutchRelocation – збережена процедура для зміни місцерозташування муфти;
- ReductionGearRelocation – збережена процедура для зміни місцерозташування редуктора;
- ElectricMotorRelocation – збережена процедура для зміни місцерозташування електродвигуна;

2.2.4.4 Діаграма зв’язків таблиць

Для того, щоб більш детальніше зрозуміти структуру бази даних розглянемо зв’язки таблиць (рис. 2.2.4.4).

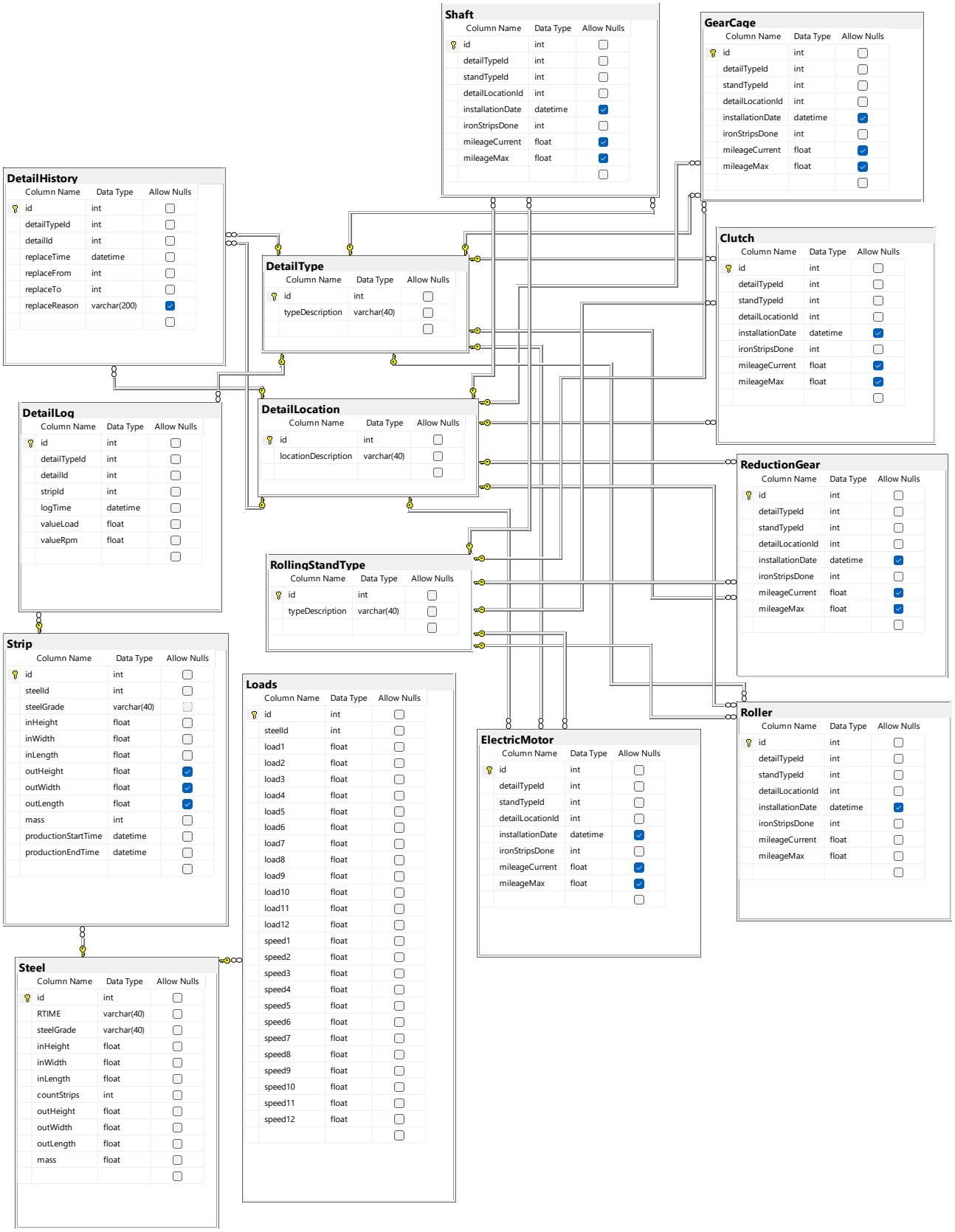


Рисунок 2.2.4.4 – Діаграма зв’язків таблиць

3 ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

Створене програмне забезпечення є ERP системою. Тестування такої програми є доволі складним процесом адже вхідні дані можуть бути тими самим, але вихідні дані кожен раз відрізнятися. Тому, в цьому випадку буде використовуватися метод тестування чорної скриньки. Такий спосіб наддасть розуміння в правильності реалізацій функцій, вірній співпраці з базою даних, коректності роботи інтерфейсу та продуктивності програми.

Розроблені та виконані тести для програм перегляду прокатного стану та симуляції прокатки слябів наведені нижче:

Таблиця 3.1 – Тест №1

Номер тесту	Зміст тесту	Вхідні дані	Очікуваний результат
1	Перевірка можливості програми підключитись до бази даних	Назва серверу: KYRYLO\SQLEXPRESS Назва бази даних: RollingMechanism	Успішне підключення до бази даних та відкриття головного меню програми

Отриманий результат зображено на рис. 3.1:

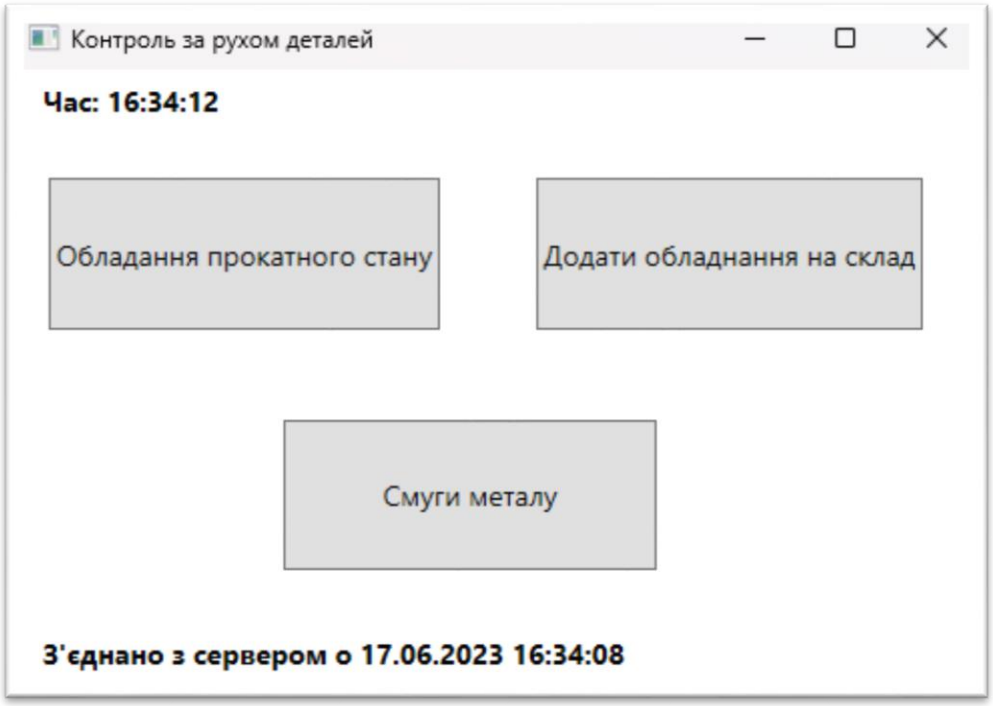


Рисунок 3.1 – Отриманий результат для тесту 1

Висновок до тесту: очікуваний результат повність збігається з отриманим. Програма успішно виконала підключення до бази даних та відкрила головну форму.

Таблиця 3.2 – Тест №2

Номер тесту	Зміст тесту	Вхідні дані	Очікуваний результат
2	Перегляд групи деталей за чорновою групою клітей	Групи огляду: Кліть чорнова Тип огляду: Повний огляд	Виведення деталей, які утворюють чорнову групу клітей

Отриманий результат зображено на рис. 3.2:

№	Деталь	Розташування	Дата установки	Прокатано	Ресурс (%)	
5	Валок	Валок чорновий верхній(3)	16-06-2023 10:13:38	0	100	
6	Валок	Валок чорновий нижній(3)	16-06-2023 10:13:38	0	100	
5	Шпиндель	Шпиндель чорновий верхній(3)	16-06-2023 10:13:38	0		
6	Шпиндель	Шпиндель чорновий нижній(3)	16-06-2023 10:13:38	0		
3	Шестеренна кліть	Шестеренна кліть чорнова(3)	16-06-2023 10:13:38	0		
3	Муфта	Муфта чорнова(3)	16-06-2023 10:13:38	0		
3	Редуктор	Редуктор чорновий(3)	16-06-2023 10:13:38	0		
3	Електродвигун	Електродвигун чорновий(3)	16-06-2023 10:13:38	0		
7	Валок	Валок чорновий верхній(4)	16-06-2023 10:13:38	0	100	
8	Валок	Валок чорновий нижній(4)	16-06-2023 10:13:38	0	100	
7	Шпиндель	Шпиндель чорновий верхній(4)	16-06-2023 10:13:38	0		
8	Шпиндель	Шпиндель чорновий нижній(4)	16-06-2023 10:13:38	0		
4	Шестеренна кліть	Шестеренна кліть чорнова(4)	16-06-2023 10:13:38	0		
4	Муфта	Муфта чорнова(4)	16-06-2023 10:13:38	0		
4	Редуктор	Редуктор чорновий(4)	16-06-2023 10:13:38	0		
4	Електродвигун	Електродвигун чорновий(4)	16-06-2023 10:13:38	0		
9	Валок	Валок чорновий верхній(5)	16-06-2023 10:13:38	0	100	
10	Валок	Валок чорновий нижній(5)	16-06-2023 10:13:38	0	100	
9	Шпиндель	Шпиндель чорновий верхній(5)	16-06-2023 10:13:38	0		
10	Шпиндель	Шпиндель чорновий нижній(5)	16-06-2023 10:13:38	0		
5	Шестеренна кліть	Шестеренна кліть чорнова(5)	16-06-2023 10:13:38	0		
5	Муфта	Муфта чорнова(5)	16-06-2023 10:13:38	0		
5	Редуктор	Редуктор чорновий(5)	16-06-2023 10:13:38	0		
5	Електродвигун	Електродвигун чорновий(5)	16-06-2023 10:13:38	0		
11	Валок	Валок чорновий верхній(6)	16-06-2023 10:13:38	0	100	
12	Валок	Валок чорновий нижній(6)	16-06-2023 10:13:38	0	100	
11	Шпиндель	Шпиндель чорновий верхній(6)	16-06-2023 10:13:38	0		

Рисунок 3.2 – Отриманий результат для тесту 2

Висновок до тесту: очікуваний результат повністю збігається з отриманим. Програма успішно вивела деталі, які складають чорнову групу клітей.

Таблиця 3.3 – Тест №3

Номер тесту	Зміст тесту	Вхідні дані	Очікуваний результат
3	Заміна деталі, а саме, валку на новий зі складу	Деталь у першій таблиці: Валок №1 Деталь у другій таблиці: Валок №25	Успішна заміна та запис історії для кожної деталі

Отриманий результат зображено на рис. 3.3:

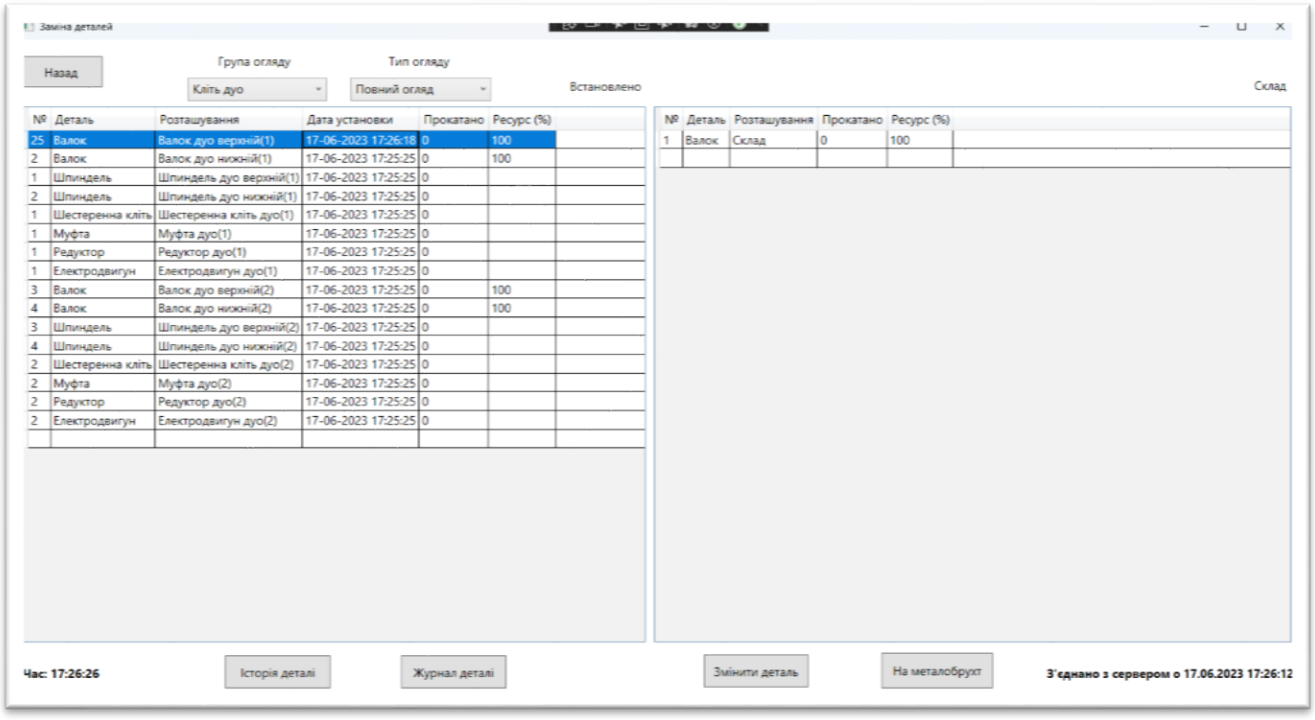


Рисунок 3.3 – Отриманий результат для тесту 3

Висновок до тесту: очікуваний результат повністю збігається з отриманим. Валок №1 був успішно замінений на Валок №25 у прокатному стані.

Таблиця 3.4 – Тест №4

Номер тесту	Зміст тесту	Вхідні дані	Очікуваний результат
4	Перегляд історії деталі замін деталі, а саме, валку.	Деталь у другій таблиці: Валок №1	Виведення історії замін деталі та їх причин

Отриманий результат зображено на рис 3.4:

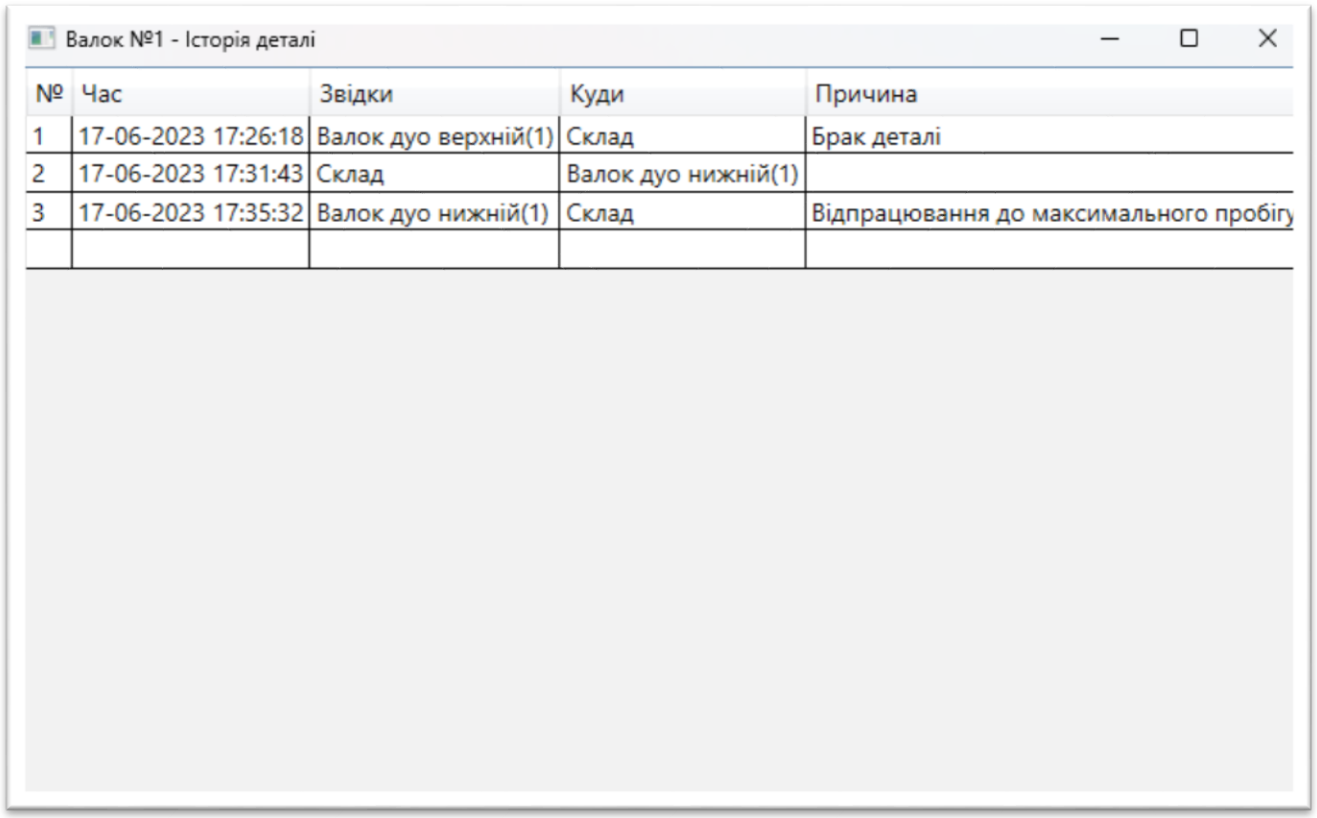


Рисунок 3.4 – Отриманий результат для тесту 4

Висновок до тесту: очікуваний результат повністю збігається з отриманим. Програма успішно вивела історію та причини замін деталі Валок №1.

Таблиця 3.5 – Тест №5

Номер тесту	Зміст тесту	Вхідні дані	Очікуваний результат
5	Додавання деталей на склад з вказанням їх кількості	Тип деталі: Муфта Тип кліті: Кліть Чорнова Кількість: 4	Успішне додавання муфт у кількості 4 одиниці на склад

Отриманий результат зображено на рис. 3.5 – 3.6:

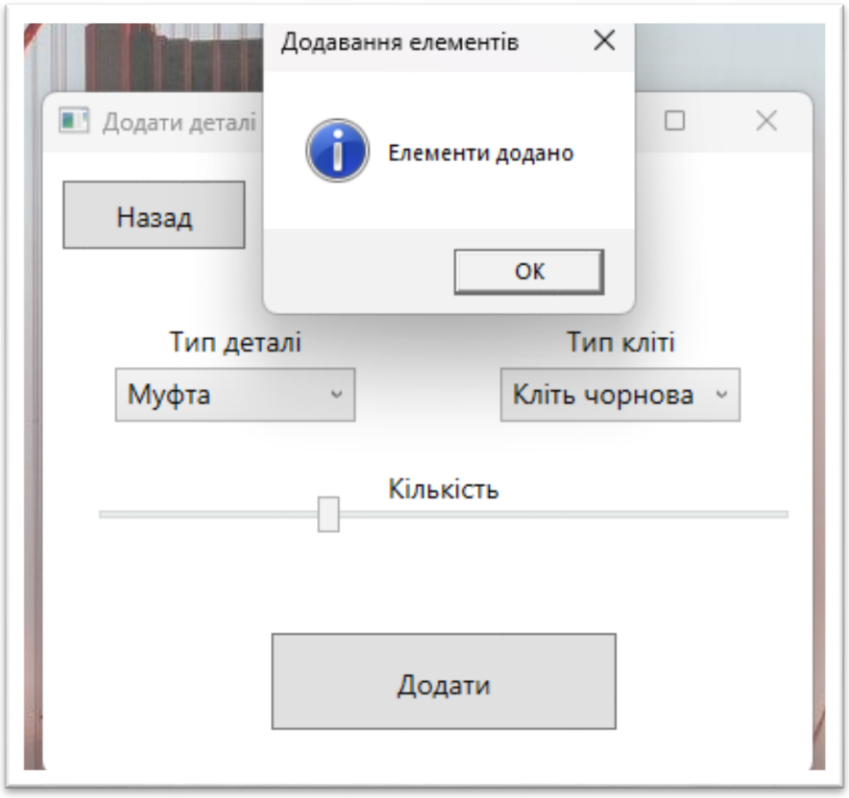


Рисунок 3.5 – Отриманий результат для тесту 5

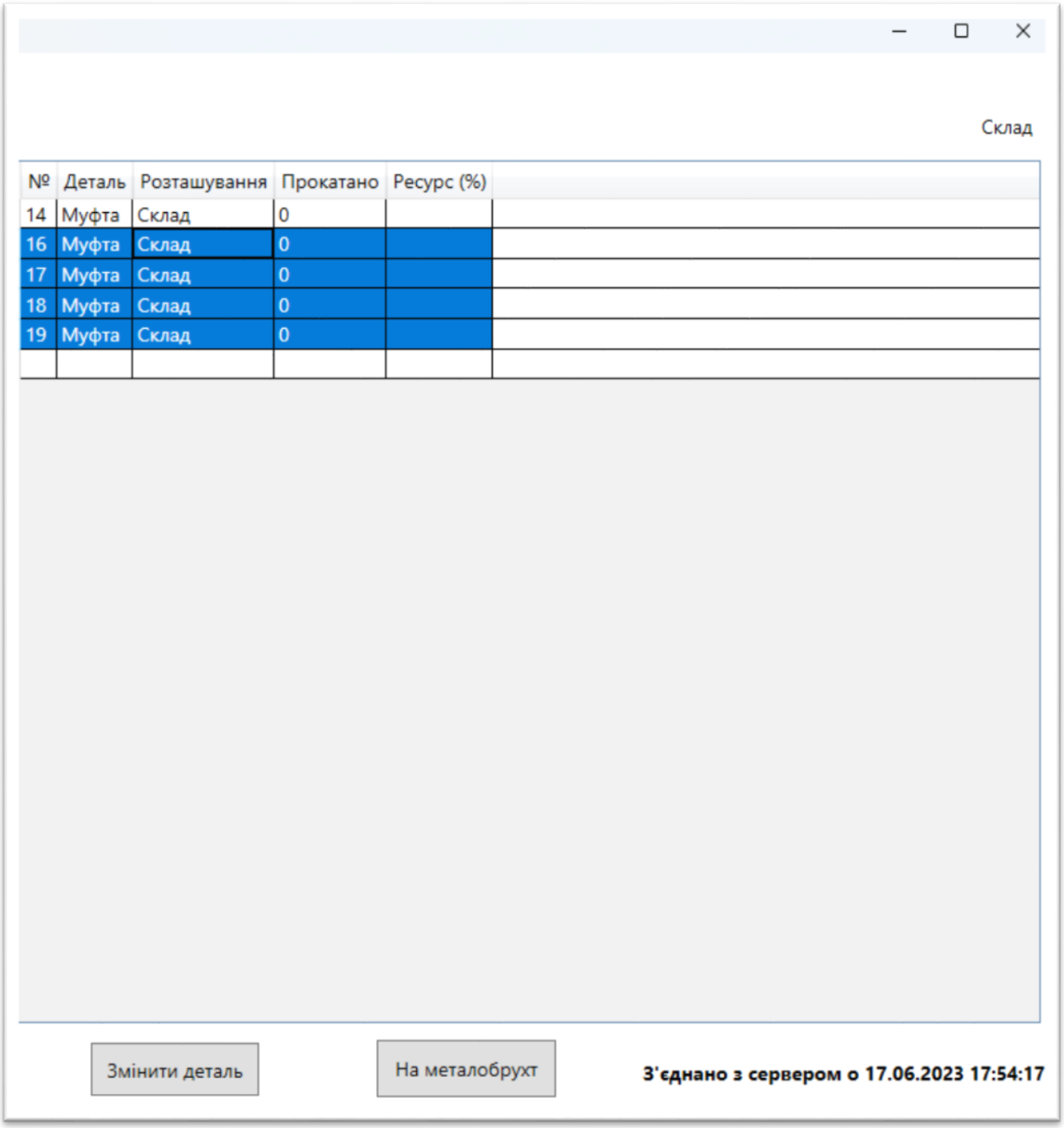


Рисунок 3.6 – Отриманий результат для тесту 5

Висновок до тесту: очікуваний результат повність збігається з отриманим. Деталь «Муфта» типу «Кліть чорнова» була успішна додана на склад у кількості одиниць 4.

Таблиця 3.6 – Тест №6

Номер тесту	Зміст тесту	Вхідні дані	Очікуваний результат
6	Відправка деталі на металобрухт	Група огляду: Кліть чистова Тип огляду: Муфта Деталь у другій таблиці: Муфта №15	Успішна відправка обраної деталі на металобрухт

Отриманий результат зображено на рис. 3.7:

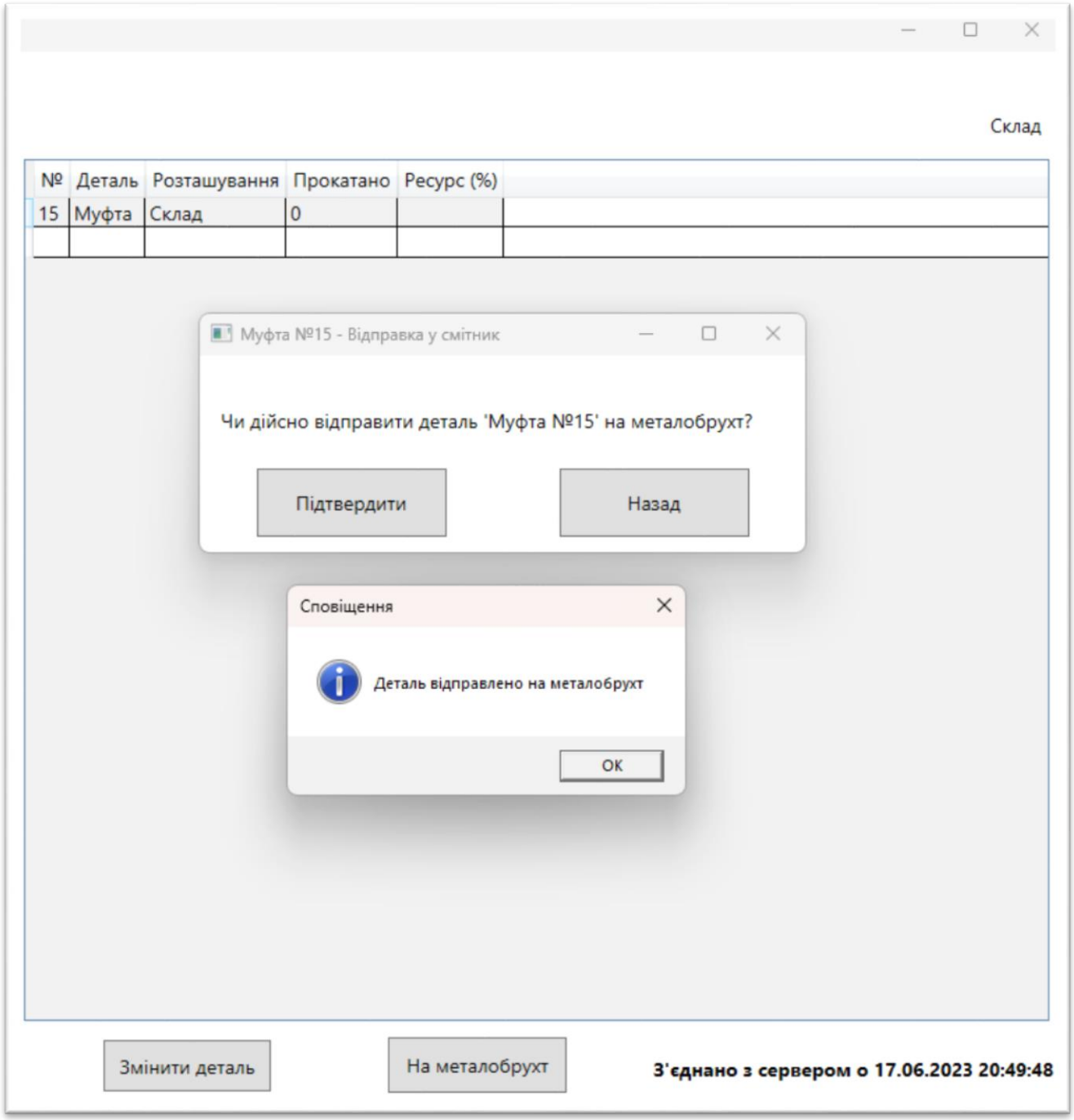


Рисунок 3.7 – Отриманий результат для тесту 6

Висновок до тесту: очікуваний результат повністю збігається з отриманим. Обрана деталь була успішно перенесена зі складу на металобрухт.

Таблиця 3.7 – Тест №7

Номер тесту	Зміст тесту	Вхідні дані	Очікуваний результат
7	Прокатка металевого слябу у смугу	Номер слябу у таблиці: 3 Кількість слябів: 1	Успішна прокатка металевого слябу у смугу

Отриманий результат на рис. 3.8:

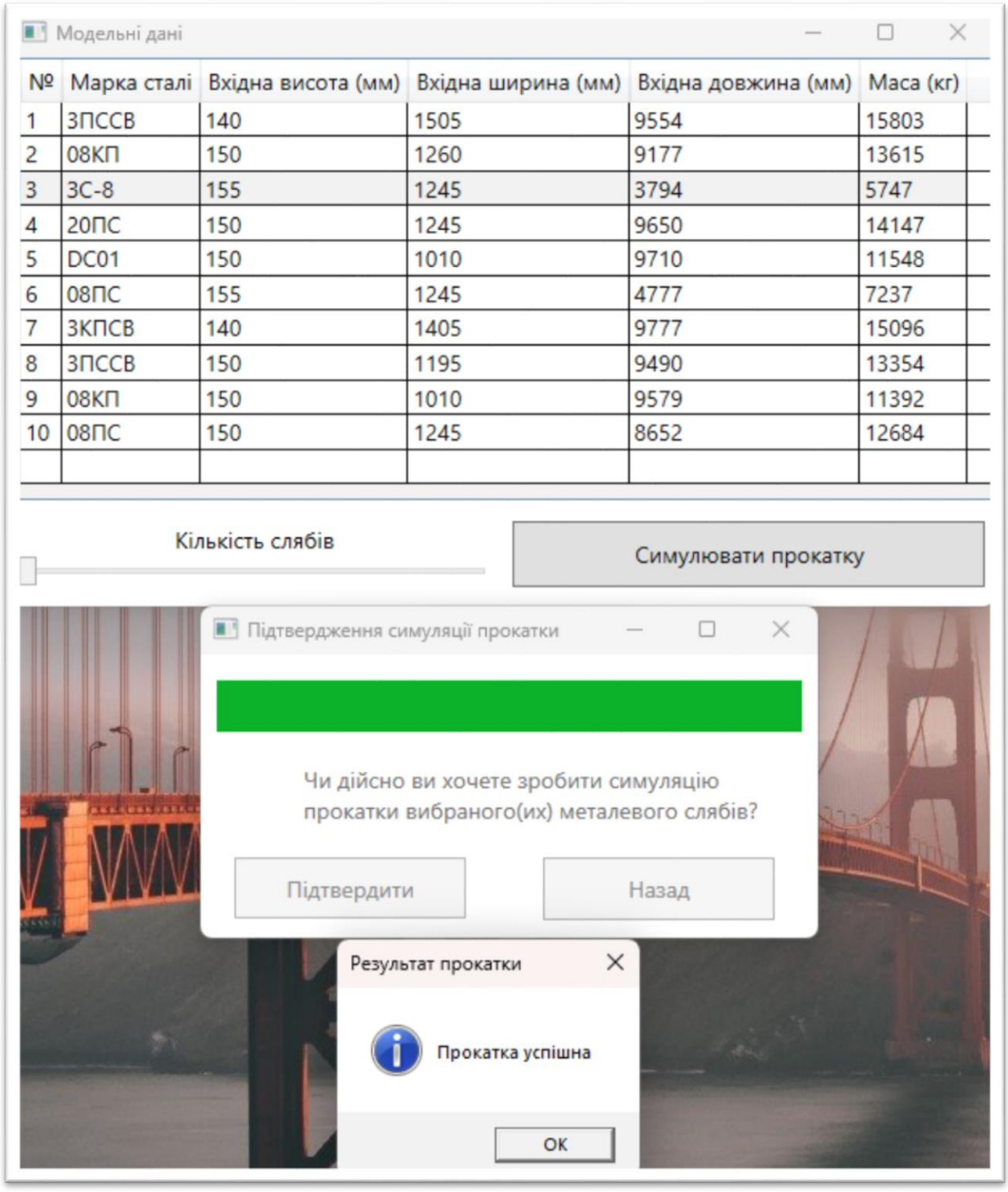


Рисунок 3.8 – Отриманий результат для тесту 7

Висновок до тесту: очікуваний результат повністю збігається з отриманим. Сляб з порядковим номером «3» з маркою сталі «ЗС-8» був успішно прокатаний.

Таблиця 3.8 – Тест №8

Номер тесту	Зміст тесту	Вхідні дані	Очікуваний результат
8	Перегляд даних про смугу, яка була прокатана в попередньому тесті	Вибір в головному меню: Смуги металу	Успішно прокатаний сляб у смугу з відображенням старих, нових характеристик та часу початку-кінця прокатки

Отриманий результат на рис. 3.9:

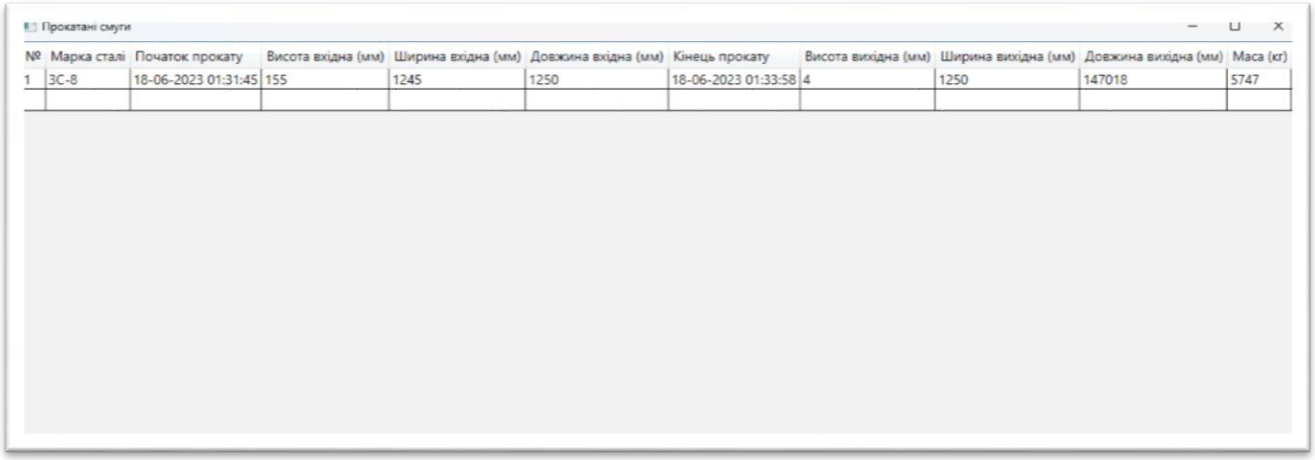


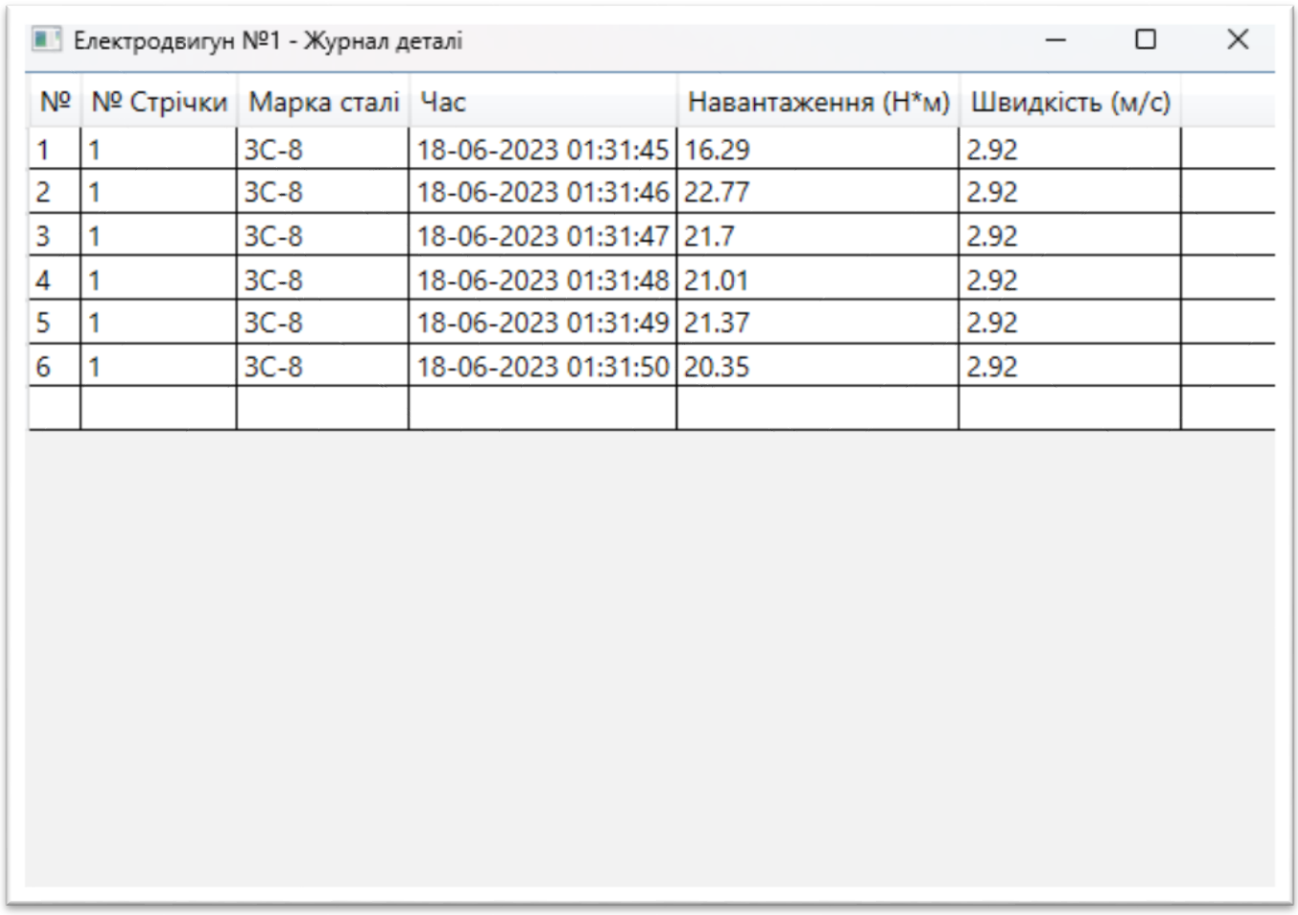
Рисунок 3.9 – Отриманий результат для тесту 8

Висновок до тесту: очікуваний результат повністю збігається з отриманим. Сляб було успішно прокатано, записано часовий проміжок роботи та нові параметри вже смуги.

Таблиця 3.9 – Тест №9

Номер тесту	Зміст тесту	Вхідні дані	Очікуваний результат
9	Перегляд навантажень та швидкостей деталі під час прокатки слябу з попереднього тесту	Група огляду: Кліть дуо Тип огляду: Повний огляд Деталь у першій таблиці: Електродвигун №1	Можливий перегляд даних про навантаження та швидкості деталі під час прокатки металевого слябу

Отриманий результат на рис. 3.10:



№	№ Стрічки	Марка сталі	Час	Навантаження (Н*м)	Швидкість (м/с)	
1	1	3С-8	18-06-2023 01:31:45	16.29	2.92	
2	1	3С-8	18-06-2023 01:31:46	22.77	2.92	
3	1	3С-8	18-06-2023 01:31:47	21.7	2.92	
4	1	3С-8	18-06-2023 01:31:48	21.01	2.92	
5	1	3С-8	18-06-2023 01:31:49	21.37	2.92	
6	1	3С-8	18-06-2023 01:31:50	20.35	2.92	

Рисунок 3.10 – Отриманий результат для тесту 9

Висновок до тесту: очікуваний результат повністю збігається з отриманим. Дані часу роботи, навантаження та швидкості записані вірно та збігається з початком прокатки з попереднього тесту.

ВИСНОВКИ

Результатом роботи є створений програмний комплекс на базі фреймворку .NET. Розроблено дві програми, а саме, програма для перегляду обладнання прокатного стану та ведення складського обліку і програма для симуляції прокатки металевих слябів у металеві смуги.

Ці два додатки були розроблені на мові C#, з використанням графічної підсистеми Windows Presentation Foundation (WPF) та системи управління базами даних (СУБД) Microsoft SQL Server.

Всі використанні технології виходять з «одного стеку», тобто одного розробника, а саме, Microsoft. Це є величезним плюсом при розробці подібних, доволі складних програм адже зручність та легкість створення програмного забезпечення є неймовірною. Кожна технологія має зворотну сумісність і програмісту не потрібно вирішувати проблеми, які можуть виникнути через різні архітектури.

Мова C# та графічна підсистема WPF чудово підійшли до цієї програми. Адже ця мова є сучасною, об’єктно-орієнтованою, зі строгою типізацію та з величезним набором можливостей та технологій. Головне, вона підтримується Microsoft, які в свою чергу кожен рік-два її оновлюють, виправляють попередні помилки та додають нові функції. WPF є також чудовою графічною підсистемою, яка надає можливість створювати зручний та сучасний інтерфейс, який в свою чергу є дуже надійним при роботі, та легким в розумінні й написанні через використані у собі XAML файлів для опису інтерфейсу, що є декларативною мовою розмітки подібною до XML.

Microsoft SQL Server є популярним прикладом системи управління базами даних. Де в якості мови запитів використовується Transact-SQL, яка була спільно розроблена Microsoft та Sybase. Використана СУБД виконала всі поставлені під час проектування задачі. А через свою сумісність з .NET фреймворком, її використання є легшим чим здається.

Створений програмний продукт є ERP системою з архітектурним шаблоном MVC (англ. Model-view-controller, укр. Модель–представлення–контролер). Тобто, програма виконує задачі бізнес логіки та підтримує подальші зміни або розширення. Цей шаблон сприяє впорядкованості структури програми та робить її більш зрозумілою.

Обидві програми реалізовані як «тонкий клієнт», а саме, є незалежними від СУБД. Тобто, вся бізнес логіка побудована в системі управління базами даних. І це само по собі зменшує обсяг самої програми, покращує її надійність та можливість вдосконалення. Такий підхід дозволяє легко пристосовуватися до змін у базі даних, або взагалі до зміни СУБД.

Отже, розробка подібного додатка є дуже кропітким і складним процесом. Розробнику дуже важливо спочатку приділяти уваги проектування системи, а вже після цього розпочинати розробку. Для того, щоб мінімізувати можливі складності сумісності була й обрана платформа .NET. Але, не обійшлося і без проблем, так, наприклад, візуальний компонент у графічній підсистемі WPF під назвою «dataGrid», який є однією з найважливіших частин обох програм, має скудний функціонал, деякі функції не працюють, а іноді, взагалі не виконують задачі на яку вони спроможні. Не дивлячись на складності при розробці програма створена такою, як була спроектована на початку. В цьому велика доля заслуги чудової платформи .NET та кропіткої роботи програміста-розробника та керівника проекту.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Rolling (metalworking) [Virtual Resource] / AManWithNoPlan / BrownHairedGirl / Abductive /. – Edit: 2023 – 23 March – Access Mode: URL: https://en.wikipedia.org/wiki/Rolling_(metalworking)
2. Early History of Rolling [Virtual Resource] / The National Iron & Steel Heritage Museum / – Edit: 2012 – 16 June – Access Mode: URL: https://steelmuseum.org/206_Mill_Exhibit_2018/early_history.cfm
3. Enterprise resource planning (ERP) [Virtual Resource] / AManWithNoPlan / William Avery / Marek Prasoł /. – Edit: 2023 – 25 May – Access Mode: URL: https://en.wikipedia.org/wiki/Enterprise_resource_planning
4. SAP SE [Virtual Resource] / Oleh Konkevych / Ehlla / Vittalio / Yuriz /. – Edit: 2022 – 2 December – Access Mode: URL: https://uk.wikipedia.org/wiki/SAP_SE
5. Oracle Corporation [Virtual Resource] / Lxlalexlxl / E. Motenko / Taromsky / Green Zero /. – Edit: 2022 – 12 November – Access Mode: URL: https://uk.wikipedia.org/wiki/Oracle_Corporation
6. C Sharp [Virtual Resource] / Alter73 / Olion17 / VictorAnyakin / Lxlalexlxl /. – Edit: 2023 – 8 January – Access Mode: URL: https://uk.wikipedia.org/wiki/C_Sharp
7. Windows Presentation Foundation [Virtual Resource] / Vovkulaka rtm / Serge3ling / VanyaS /. – Edit: 2022 – 16 May – Access Mode: URL: https://uk.wikipedia.org/wiki/Windows_Presentation_Foundation
8. Microsoft SQL Server [Virtual Resource] / Liubov Bovan / Alessot / Vlasenko D /. – Edit: 2023 – 24 February – Access Mode: URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server
9. Transact-SQL [Virtual Resource] / Deineka / Vlasenko D / Vovkulaka rtm /. – Edit: 2022 – 27 April – Access Mode: URL: https://uk.wikipedia.org/wiki/Transact-SQL
10. Working with JSON [Virtual Resource] / MDN contributors /. – Edit: 2023 – 10

May – Access Mode: URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

11. Stored Procedures (Database Engine) / WilliamDAssafMSFT / rwestMSFT / rothja /. – Edit: 2023 – 4 March – Access Mode: URL: https://learn.microsoft.com/en-us/sql/relational-databases/stored-procedures/stored-procedures-database-engine?view=sql-server-ver16
12. АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ В ОБЛАСТІ ДИНАМІКИ ТА ДІАГНОСТИКИ ОБЛАДНАННЯ ПРОКАТНИХ СТАНОВ [Virtual Resource] / Павло Крот /. – Edit: 2006 – 22 August – Access Mode: URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/22434/32-Krot.pdf?sequence=
13. Model–view–controller [Virtual Resource] / Mindmatrix / LizardJr8 / William Avery /. – Edit: 2023 – 6 June – Access Mode: URL: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

ДОДАТОК А

Технічне завдання

ЗАТВЕРДЖЕНО
1116130.01315-01-ЛЗ

СИСТЕМА КОНТРОЛЮ ЗА РУХОМ ДЕТАЛЕЙ НА МЕТКОМБІНАТІ ТА ОЦІНКА ЇХ ЗАЛИШКОВОГО РЕСУРСА

Технічне завдання

1116130.01315-01

Листів 11

2023

ЗМІСТ

1 ВВЕДЕННЯ

Програмне забезпечення для перегляду прокатного стану та складського обліку. Надає можливість покращення роботи підприємства зменшуючи фінансові витрати та підвищуючи ефективність виробництва. Бере на себе фактор людської помилки та покращує розроблену продукцію.

Причина виникнення такого продукту є складність ERP систем та закритість технологічного процесу виробництва.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки слугує бакалаврський проект та наказ ректора від 19.06.2023 №1209ст ректора Українського державного університету науки і технологій «Про призначення наукових керівників та затвердження тем бакалаврських робіт» за спеціальністю «121 Інженерія програмного забезпечення» факультету «Комп’ютерні технології і системи» кафедри «Комп’ютерні інформаційні технології».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення:

Розроблене програмне забезпечення спрощує керування прокатним станом на підприємстві. Полегшення розуміння принципу дії механізмів та їх надійності.

Експлуатаційне призначення:

Спрощення роботи персоналу на підприємстві через використання розробленого програмного забезпечення. Зменшення помилок людського фактору та покращення фінансового стану виробництва через більш вдале використання ресурсів.

4 ВИМОГИ ДО ПРОГРАМИ

4.1 Вимоги до функціональних характеристик

Зрозумілий інтерфейс користування: підпис функціональних кнопок, зворотній зв’язок від інтерфейсу, виведення таблиць в зручному оформлені та надійність роботи системи.

4.2 Вимоги до надійності

- Наявність архівної копії тексту програми на зовнішньому носії;
- Забезпечення стійкого функціонування програми;
- Контроль вхідних і вихідних даних;
- Захист від копіювання.

4.3 Умови експлуатації

Температура повітря від +10 до 30 градусів за Цельсієм. Відносна вологість до 40-50%.

Обслуговування потрібне якщо з програмою виникли труднощі. Для цього потрібен розробник, який ознайомлений з архітектурою програми та особливістю підприємства.

4.4 Вимоги до складу і параметрів технічних засобів

- 64-розядний процесор частотою від 1800 МГц;
- 2 ГБ ОЗУ;
- 350 МБ вільного місця на накопичувачі;
- Пристрій вводу: миша та клавіатура;
- Пристрій виводу: монітор.

4.5 Вимоги до інформаційної і програмної сумісності

Програма працює на операційній системі Windows версій 7, 8, 8.1, 10, 11. Написана на мові C# при підтримці .NET фреймворку версії 6.0 та СУБД Microsoft SQL Server 2022 року.

4.6 Вимоги до маркувань і упаковки

Яровий К.В.
СИСТЕМА КОНТРОЛЮ ЗА РУХОМ ДЕТАЛЕЙ НА МЕТКОМБІНАТІ ТА ОЦІНКА ЇХ ЗАЛИШКОВОГО РЕСУРСА
Україна. м. Дніпро 49100
2023

Вимоги до упаковки – повинна захищати від будь-яких негативних впливів та забруднення.

4.7 Вимоги до транспортування і зберігання

Транспортування товару необхідно організовувати таким чином, щоб у дорозі він не забруднився та не зіпсувався. Тому, для цього необхідно мати спеціалізований транспорт та використовувати упаковку.

Зберігати товар при температурі від +15 до +25 градусів по Цельсію, при низькій вологості до 20%.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації мають входити:

- специфікація;
- текст програми.

Програмна документація повинна відповідати вимогам ДСТУ.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиця А.1 – Стадії та етапи розробки

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Вступ	14.09.22 – 28.10.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	29.10.22 – 10.03.23	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	11.03.23 – 31.04.23	
4	Постановка задачі, технічне завдання	01.05.23 – 07.05.23	30%
5	Техніко-економічні показники	08.05.23 – 14.05.23	
6	Розробка інструментальних засобів дослідження	15.05.23 – 21.05.23	
7	Виконання досліджень	22.05.23 – 28.05.23	60%
8	Оформлення тез доповідей	29.05.23 – 02.06.23	
9	Оформлення статті у фаховий журнал	03.06.23 – 07.06.23	
10	Оформлення пояснювальної записки	08.06.23 – 11.06.23	
11	Розробка демонстраційних матеріалів	12.06.23 – 18.06.23	100%
12	Подання кваліфікаційної роботи до кафедри	22.06.23	
13	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	28.06.23	

7 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль виконання програмного забезпечення виконує керівник розробки Разносілін В.В. Прийом роботи здійснюється уповноваженою комісією.

8 БІБЛІОГРАФІЧНИЙ СПИСОК

Івченко Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна. – Д.: Вид-во Дніпропетровського національного університету залізничного транспорту імені академіка В. Лазаряна, 2009. - 38 с.

ДОДАТОК Б

Специфікація

ЗАТВЕРДЖЕНО
1116130.01315-01-ЛЗ

СИСТЕМА КОНТРОЛЮ ЗА РУХОМ ДЕТАЛЕЙ НА МЕТКОМБІНАТІ ТА ОЦІНКА ЇХ ЗАЛИШКОВОГО РЕСУРСА

Технічне завдання

1116130.01315-01

Листів 2

2023

Специфікації

Таблиця Б.1 – Специфікації

Позначення	Найменування	Примітка
1116130.01315-01-ЛЗ	Документація	
1116130.01315-01	Лист затвердження	
1116130.01315-01-ЛЗ	Технічне завдання	
1116130.01315-01	Лист затвердження	
1116130.01315-01 12 01-ЛЗ	Специфікація	
1116130.01315-01 12 01	Лист затвердження	
	Текст програми	

ДОДАТОК В

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
19.06.2023

СИСТЕМА КОНТРОЛЮ ЗА РУХОМ ДЕТАЛЕЙ НА МЕТКОМБІНАТІ ТА ОЦІНКА ЇХ ЗАЛИШКОВОГО РЕСУРСА

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01315-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
07.12.22
Керівник розробки
_____ Валентин РАЗНОСІЛІН
07.12.22
Виконавець
_____ Кирило ЯРОВИЙ
07.12.22
Нормконтролер
_____ Світлана ВОЛКОВА
07.12.22

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
19.06.2023

СИСТЕМА КОНТРОЛЮ ЗА РУХОМ ДЕТАЛЕЙ НА МЕТКОМБІНАТІ ТА ОЦІНКА ЇХ ЗАЛИШКОВОГО РЕСУРСА

Специфікація
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01315-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
07.12.22
Керівник розробки
_____ Валентин РАЗНОСІЛІН
07.12.22
Виконавець
_____ Кирило ЯРОВИЙ
07.12.22
Нормконтролер
_____ Світлана ВОЛКОВА
07.12.22

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
19.06.2023

СИСТЕМА КОНТРОЛЮ ЗА РУХОМ ДЕТАЛЕЙ НА МЕТКОМБІНАТІ ТА ОЦІНКА ЇХ ЗАЛИШКОВОГО РЕСУРСА

Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01315-01 12 01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
07.12.22
Керівник розробки
_____ Валентин РАЗНОСІЛІН
07.12.22
Виконавець
_____ Кирило ЯРОВИЙ
07.12.22
Нормконтролер
_____ Світлана ВОЛКОВА
07.12.22

2023

ДОДАТОК Г

Текст програми

ЗАТВЕРДЖЕНО
1116130.01315-01 12 01-ЛЗ

СИСТЕМА КОНТРОЛЮ ЗА РУХОМ ДЕТАЛЕЙ НА МЕТКОМБІНАТІ ТА ОЦІНКА ЇХ ЗАЛИШКОВОГО РЕСУРСА

Текст програми

1116130.01315-01 12 01

Листів 32

2023

Програма 1 – перегляд прокатного стану та облік деталей.

DatabaseConnection.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Text.Json;

namespace ClientProgram.DatabaseConnectionFolder
{
    public class DatabaseConnection
    {
        private const string serverCommandsPath = "../../../JsonFiles/ServerCommands.json";
        private SqlConnection sqlConnection;
        private ServerSettings serverSettings;
        private ServerCommands serverCommands;
        private string connectionString;
        public DateTime ConnectionTime { get; private set; }

        private SqlDataAdapter sqlDataAdapter;
        private DataTable dataTable;
        private SqlCommand sqlCommand;

        public DatabaseConnection(ServerSettings serverSettings)
        {
            this.serverSettings = serverSettings;
            serverCommands = (ServerCommands)JsonSerializer.Deserialize(File.ReadAllText(serverCommandsPath), typeof(ServerCommands));

            connectionString = $"Server={this.serverSettings.ServerName};Database={this.serverSettings.DatabaseName};integrated security=SSPI";
            sqlConnection = new SqlConnection(connectionString);
            ConnectionTime = DateTime.Now;
        }

        ~DatabaseConnection()
        {
            sqlConnection.Close();
        }

        public bool IsConnectionOk()
        {
            try
            {
                sqlConnection.Open();
            }
            catch (SqlException)
            {
                return false;
            }

            return true;
        }

        private DataView GetSelectDetail(string command, int locStart, int locEnd, int standTypeId)
        {
```

```
            sqlCommand = new
SqlCommand(command,
sqlConnection);

sqlCommand.Parameters.Add(new
SqlParameter("@locStart", locStart));

sqlCommand.Parameters.Add(new
SqlParameter("@locEnd", locEnd));

sqlCommand.Parameters.Add(new
SqlParameter("@standTypeId",
standTypeId));

            sqlDataAdapter = new
SqlDataAdapter(sqlCommand);
            dataTable = new DataTable();
            sqlDataAdapter.Fill(dataTable);

            return dataTable.DefaultView;
        }

        public DataView
GetSelectRoller(int locStart, int
locEnd, int standTypeId)
        {
            return
GetSelectDetail(serverCommands.Sele
ctRoller, locStart, locEnd,
standTypeId);
        }

        public DataView
GetSelectShaft(int locStart, int locEnd,
int standTypeId)
        {
            return
GetSelectDetail(serverCommands.Sele
ctShaft, locStart, locEnd, standTypeId);
        }

        public DataView
GetSelectGearCage(int locStart, int
locEnd, int standTypeId)
        {
            return
GetSelectDetail(serverCommands.Sele
ctGearCage, locStart, locEnd,
standTypeId);
        }

        public DataView
GetSelectClutch(int locStart, int
locEnd, int standTypeId)
        {
            return
GetSelectDetail(serverCommands.Sele
ctClutch, locStart, locEnd,
standTypeId);
        }

        public DataView
GetSelectReductionGear(int locStart,
int locEnd, int standTypeId)
        {
            return
GetSelectDetail(serverCommands.Sele
ctReductionGear, locStart, locEnd,
standTypeId);
        }

        public DataView
GetSelectElectricMotor(int locStart, int
locEnd, int standTypeId)
        {
            return
GetSelectDetail(serverCommands.Sele
ctElectricMotor, locStart, locEnd,
standTypeId);
        }
```

```
        public DataView
GetSelectStand(int locStart, int locEnd)
        {
            sqlCommand = new
SqlCommand(serverCommands.Select
Stand, sqlConnection);

sqlCommand.Parameters.Add(new
SqlParameter("@locStart", locStart));

sqlCommand.Parameters.Add(new
SqlParameter("@locEnd", locEnd));

            sqlDataAdapter = new
SqlDataAdapter(sqlCommand);
            dataTable = new DataTable();
            sqlDataAdapter.Fill(dataTable);

            return dataTable.DefaultView;
        }

        public DataView GetDetailLog(int
detailTypeId, int detailId)
        {
            sqlCommand = new
SqlCommand(serverCommands.Select
DetailLog, sqlConnection);

sqlCommand.Parameters.Add(new
SqlParameter("@detailTypeId",
detailTypeId));

sqlCommand.Parameters.Add(new
SqlParameter("@detailId", detailId));

            sqlDataAdapter = new
SqlDataAdapter(sqlCommand);
            dataTable = new DataTable();
            sqlDataAdapter.Fill(dataTable);

            return dataTable.DefaultView;
        }

        public DataView
GetDetailHistory(int detailTypeId, int
detailId)
        {
            sqlCommand = new
SqlCommand(serverCommands.Select
DetailHistory, sqlConnection);

sqlCommand.Parameters.Add(new
SqlParameter("@detailTypeId",
detailTypeId));

sqlCommand.Parameters.Add(new
SqlParameter("@detailId", detailId));

            sqlDataAdapter = new
SqlDataAdapter(sqlCommand);
            dataTable = new DataTable();
            sqlDataAdapter.Fill(dataTable);

            return dataTable.DefaultView;
        }

        private void SwapFunc(string
command, int detailTypeId, int idFrom,
int idTo, DateTime replaceTime, int
replaceFrom, int replaceTo, string
replaceReason)
        {
            sqlCommand = new
SqlCommand(command,
sqlConnection);


sqlCommand.Parameters.Add(new
SqlParameter("@detailTypeId",
detailTypeId));
```

```
sqlCommand.Parameters.Add(new
SqlParameter("@idFrom", idFrom));

sqlCommand.Parameters.Add(new
SqlParameter("@idTo", idTo));

sqlCommand.Parameters.Add(new
SqlParameter("@replaceTime",
replaceTime));

sqlCommand.Parameters.Add(new
SqlParameter("@replaceFrom",
replaceFrom));

sqlCommand.Parameters.Add(new
SqlParameter("@replaceTo",
replaceTo));

sqlCommand.Parameters.Add(new
SqlParameter("@replaceReason",
replaceReason));


sqlCommand.ExecuteNonQuery();
        }

        public void RollerSwap(int
detailTypeId, int idFrom, int idTo,
DateTime replaceTime, int
replaceFrom, int replaceTo, string
replaceReason)
        {

SwapFunc(serverCommands.RollerSw
ap, detailTypeId, idFrom, idTo,
replaceTime, replaceFrom, replaceTo,
replaceReason);
        }

        public void ShaftSwap(int
detailTypeId, int idFrom, int idTo,
DateTime replaceTime, int
replaceFrom, int replaceTo, string
replaceReason)
        {

SwapFunc(serverCommands.ShaftSwa
p, detailTypeId, idFrom, idTo,
replaceTime, replaceFrom, replaceTo,
replaceReason);
        }

        public void GearCageSwap(int
detailTypeId, int idFrom, int idTo,
DateTime replaceTime, int
replaceFrom, int replaceTo, string
replaceReason)
        {

SwapFunc(serverCommands.GearCage
Swap, detailTypeId, idFrom, idTo,
replaceTime, replaceFrom, replaceTo,
replaceReason);
        }

        public void ClutchSwap(int
detailTypeId, int idFrom, int idTo,
DateTime replaceTime, int
replaceFrom, int replaceTo, string
replaceReason)
        {

SwapFunc(serverCommands.ClutchSw
ap, detailTypeId, idFrom, idTo,
replaceTime, replaceFrom, replaceTo,
replaceReason);
        }

        public void
ReductionGearSwap(int detailTypeId,
```

```
int idFrom, int idTo, DateTime
replaceTime, int replaceFrom, int
replaceTo, string replaceReason)
        {

SwapFunc(serverCommands.Reduction
GearSwap, detailTypeId, idFrom, idTo,
replaceTime, replaceFrom, replaceTo,
replaceReason);
        }

        public void
ElectricMotorSwap(int detailTypeId,
int idFrom, int idTo, DateTime
replaceTime, int replaceFrom, int
replaceTo, string replaceReason)
        {

SwapFunc(serverCommands.ElectricM
otorSwap, detailTypeId, idFrom, idTo,
replaceTime, replaceFrom, replaceTo,
replaceReason);
        }

        private void InsertFunc(string
command, int standTypeId)
        {
            sqlCommand = new
SqlCommand(command,
sqlConnection);

sqlCommand.Parameters.Add(new
SqlParameter("@standTypeId",
standTypeId));

sqlCommand.ExecuteNonQuery();
        }

        public void InsertRoller(int
standTypeId, float mileageMax)
        {
            sqlCommand = new
SqlCommand(serverCommands.Insert
Roller, sqlConnection);

sqlCommand.Parameters.Add(new
SqlParameter("@standTypeId",
standTypeId));

sqlCommand.Parameters.Add(new
SqlParameter("@mileageMax",
mileageMax));

sqlCommand.ExecuteNonQuery();
        }

        public void InsertShaft(int
standTypeId)
        {

InsertFunc(serverCommands.InsertSha
ft, standTypeId);
        }

        public void InsertGearCage(int
standTypeId)
        {

InsertFunc(serverCommands.InsertGea
rCage, standTypeId);
        }

        public void InsertClutch(int
standTypeId)
        {

InsertFunc(serverCommands.InsertClut
ch, standTypeId);
        }

        public void
InsertReductionGear(int standTypeId)
```

```
        {

InsertFunc(serverCommands.InsertRed
uctionGear, standTypeId);
        }

        public void
InsertElectricMotor(int standTypeId)
        {

InsertFunc(serverCommands.InsertElec
tricMotor, standTypeId);
        }

        public DataView SelectStrips()
        {
            sqlCommand = new
SqlCommand(serverCommands.Select
Strips, sqlConnection);

            sqlDataAdapter = new
SqlDataAdapter(sqlCommand);
            dataTable = new DataTable();
            sqlDataAdapter.Fill(dataTable);

            return dataTable.DefaultView;
        }

        private void
RelocationFunc(string command, int
detailId, int locFrom, int locTo)
        {
            sqlCommand = new
SqlCommand(command,
sqlConnection);

sqlCommand.Parameters.Add(new
SqlParameter("@detailId", detailId));

sqlCommand.Parameters.Add(new
SqlParameter("@locFrom", locFrom));

sqlCommand.Parameters.Add(new
SqlParameter("@locTo", locTo));

sqlCommand.ExecuteNonQuery();
        }

        public void RollerRelocation(int
detailId, int locFrom, int locTo)
        {

RelocationFunc(serverCommands.Roll
erRelocation, detailId, locFrom,
locTo);
        }

        public void ShaftRelocation(int
detailId, int locFrom, int locTo)
        {

RelocationFunc(serverCommands.Shaf
tRelocation, detailId, locFrom, locTo);
        }

        public void
GearCageRelocation(int detailId, int
locFrom, int locTo)
        {

RelocationFunc(serverCommands.Gear
CageRelocation, detailId, locFrom,
locTo);
        }

        public void ClutchRelocation(int
detailId, int locFrom, int locTo)
        {

RelocationFunc(serverCommands.Clut
chRelocation, detailId, locFrom,
locTo);
```

```
        }

        public void
ReductionGearRelocation(int detailId,
int locFrom, int locTo)
        {

RelocationFunc(serverCommands.Red
uctionGearRelocation, detailId,
locFrom, locTo);
        }

        public void
ElectricMotorRelocation(int detailId,
int locFrom, int locTo)
        {

RelocationFunc(serverCommands.Elec
tricMotorRelocation, detailId, locFrom,
locTo);
        }
    }
}
```

ServerCommands.cs

```
namespace
ClientProgram.DatabaseConnectionFol
der
{
    public class ServerCommands
    {
        public string SelectRoller { get;
set; } = "";
        public string SelectShaft { get;
set; } = "";
        public string SelectGearCage {
get; set; } = "";
        public string SelectClutch { get;
set; } = "";
        public string SelectReductionGear
{ get; set; } = "";
        public string SelectElectricMotor
{ get; set; } = "";
        public string SelectStand { get;
set; } = "";
        public string SelectDetailLog {
get; set; } = "";
        public string SelectDetailHistory {
get; set; } = "";
        public string RollerSwap { get;
set; } = "";
        public string ShaftSwap { get; set;
} = "";
        public string GearCageSwap {
get; set; } = "";
        public string ClutchSwap { get;
set; } = "";
        public string ReductionGearSwap
{ get; set; } = "";
        public string ElectricMotorSwap {
get; set; } = "";
        public string InsertRoller { get;
set; } = "";
        public string InsertShaft { get; set;
} = "";
        public string InsertGearCage {
get; set; } = "";
        public string InsertClutch { get;
set; } = "";
        public string InsertReductionGear
{ get; set; } = "";
        public string InsertElectricMotor {
get; set; } = "";
        public string SelectStrips { get;
set; } = "";
        public string RollerRelocation {
get; set; } = "";
        public string ShaftRelocation {
get; set; } = "";
        public string GearCageRelocation
{ get; set; } = "";
```

```
        public string ClutchRelocation {
get; set; } = "";
        public string
ReductionGearRelocation { get; set; }
= "";
        public string
ElectricMotorRelocation { get; set; } =
"";
    }
}
```

ServerSettings.cs

```
namespace
ClientProgram.DatabaseConnectionFol
der
{
    public class ServerSettings
    {
        public string ServerName { get;
set; }
        public string DatabaseName { get;
set; }

        public ServerSettings()
        {
            ServerName = "";
            DatabaseName = "";
        }
    }
}
```

DetailAddWindow.xaml.cs

```
using
ClientProgram.DatabaseConnectionFol
der;
using
System.Text.RegularExpressions;
using System.Windows;

namespace
ClientProgram.DetailAddWindowFold
er
{
    public partial class
DetailAddWindow : Window
    {
        private DatabaseConnection
databaseConnection;

        public
DetailAddWindow(DatabaseConnectio
n databaseConnection)
        {
            InitializeComponent();
            this.databaseConnection =
databaseConnection;
            ComboBoxInit();
        }

        private void ComboBoxInit()
        {

comboBoxDetailType.Items.Add("Вал
ок");

comboBoxDetailType.Items.Add("Шп
индель");

comboBoxDetailType.Items.Add("Ше
стеренна кліть");

comboBoxDetailType.Items.Add("Му
фта");

comboBoxDetailType.Items.Add("Ред
уктор");

comboBoxDetailType.Items.Add("Еле
ктродвигун");
```

```
comboBoxDetailType.SelectedIndex = 0;


comboBoxStandType.Items.Add("Кліть дуо");

comboBoxStandType.Items.Add("Кліть чорнова");

comboBoxStandType.Items.Add("Кліть чистова");

comboBoxStandType.SelectedIndex = 0;
    }

    private void buttonBack_Click(object sender, RoutedEventArgs e)
    {
        MainWindow mainWindow = new MainWindow(databaseConnection);

mainWindow.WindowStartupLocation = WindowStartupLocation.CenterScreen;
        mainWindow.Show();
        Close();
    }

    private void buttonAdd_Click(object sender, RoutedEventArgs e)
    {
        int detailType = comboBoxDetailType.SelectedIndex + 1;
        int standType = comboBoxStandType.SelectedIndex + 1;
        int detailCount = (int)sliderCount.Value;

        if (detailType == 1)
        {
            if (textBoxMileage.Text.Length <= 0) return;

            for (int i = 0; i < detailCount; i++)

databaseConnection.InsertRoller(standType, float.Parse(textBoxMileage.Text));
        }
        else if (detailType == 2)
        {
            for (int i = 0; i < detailCount; i++)

databaseConnection.InsertShaft(standType);
        }
        else if (detailType == 3)
        {
            for (int i = 0; i < detailCount; i++)

databaseConnection.InsertGearCage(standType);
        }
        else if (detailType == 4)
        {
            for (int i = 0; i < detailCount; i++)
```

```
databaseConnection.InsertClutch(standType);
            }
            else if (detailType == 5)
            {
                for (int i = 0; i < detailCount; i++)
                    databaseConnection.InsertReductionGear(standType);
            }
            else if (detailType == 6)
            {
                for (int i = 0; i < detailCount; i++)
                    databaseConnection.InsertElectricMotor(standType);
            }

            MessageBox.Show("Елементи додано", "Додавання елементів", MessageBoxButton.OK, MessageBoxImage.Information);
        }

        private void comboBoxDetailType_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)
        {
            if (comboBoxDetailType.SelectedIndex == 0)
            {
                textBoxMileage.Visibility = Visibility.Visible;
                labelMileageMax.Visibility = Visibility.Visible;
            }
            else
            {
                textBoxMileage.Visibility = Visibility.Hidden;
                labelMileageMax.Visibility = Visibility.Hidden;
            }
        }

        private void textBoxMileage_PreviewTextInput(object sender, System.Windows.Input.TextCompositionEventArgs e)
        {
            Regex regex = new Regex("[^0-9]+");
            e.Handled = regex.IsMatch(e.Text);
        }
    }
}
```

DetailHistoryWindow.xaml.cs

```
using ClientProgram.DatabaseConnectionFolder;
using ClientProgram.DetailsWindowFolder.SelectedIdsFolder;
using System.Windows;

namespace ClientProgram.DetailsWindowFolder.DetailHistoryWindowFolder
{
    public partial class DetailHistoryWindow : Window
    {
```

```
        public
DetailHistoryWindow(DatabaseConnec
tion databaseConnection, SelectedIds
selectedIds)
        {
            InitializeComponent();
            Title =
$"{selectedIds.GetTitle()} - Історія
деталі";
            dataGridHistory.ItemsSource =
databaseConnection.GetDetailHistory(s
electedIds.DetailTypeId,
selectedIds.DetailId);
        }
    }
}
```

DetailLogWindow.xaml.cs

```
using
ClientProgram.DatabaseConnectionFol
der;
using
ClientProgram.DetailsWindowFolder.S
electedIdsFolder;
using System.Windows;

namespace
ClientProgram.DetailsWindowFolder.
DetailLogWindowFolder
{
    public partial class
DetailLogWindow : Window
    {
        public
DetailLogWindow(DatabaseConnectio
n databaseConnection, SelectedIds
selectedIds)
        {
            InitializeComponent();
            Title =
$"{selectedIds.GetTitle()} - Журнал
деталі";
            dataGridLog.ItemsSource =
databaseConnection.GetDetailLog(sele
ctedIds.DetailTypeId,
selectedIds.DetailId);
        }
    }
}
```

DetailSwapWindow.xaml.cs

```
using
ClientProgram.DatabaseConnectionFol
der;
using
ClientProgram.DetailsWindowFolder.S
electedIdsFolder;
using System;
using System.Windows;

namespace
ClientProgram.DetailsWindowFolder.
DetailSwapFolder
{
    public partial class
DetailSwapWindow : Window
    {
        private DetailsWindow
detailsWindow;
        private DatabaseConnection
databaseConnection;
        private SelectedIds
selectedIdsGrid1;
        private SelectedIds
selectedIdsGrid2;

        public
DetailSwapWindow(DatabaseConnecti
on databaseConnection, SelectedIds
selectedIdsGrid1, SelectedIds
```

```
selectedIdsGrid2, DetailsWindow
detailsWindow)
        {
            InitializeComponent();
            this.databaseConnection =
databaseConnection;
            this.selectedIdsGrid1 =
selectedIdsGrid1;
            this.selectedIdsGrid2 =
selectedIdsGrid2;
            this.detailsWindow =
detailsWindow;

            labelInformation.Content =
$"Заміна деталі
\'{selectedIdsGrid1.GetTitle()}\' на
\'{selectedIdsGrid2.GetTitle()}\'";

            TimePickerInit();
        }

        private void
buttonClose_Click(object sender,
RoutedEventArgs e)
        {
            Close();
        }

        private void TimePickerInit()
        {
            for (int i = 0; i < 24; i++)

comboBoxHours.Items.Add(i);

            for (int i = 0; i < 60; i++)
            {

comboBoxMinutes.Items.Add(i);

comboBoxSeconds.Items.Add(i);
            }

            comboBoxHours.SelectedIndex
=
int.Parse(DateTime.Now.ToString("H
H"));

comboBoxMinutes.SelectedIndex =
int.Parse(DateTime.Now.ToString("m
m"));

comboBoxSeconds.SelectedIndex =
int.Parse(DateTime.Now.ToString("ss"
));

            datePicker1.Text =
DateTime.Now.ToString("dd.MM.yyy
y");
        }

        private void
buttonConfirm_Click(object sender,
RoutedEventArgs e)
        {
            int detailTypeId =
selectedIdsGrid1.DetailTypeId;
            int idFrom =
selectedIdsGrid1.DetailId;
            int idTo =
selectedIdsGrid2.DetailId;
            DateTime replaceTime =
DateTime.Parse($"{datePicker1.Text}
{comboBoxHours.Text}:{comboBoxM
inutes.Text}:{comboBoxSeconds.Text
}"); ;
            int replaceFrom =
selectedIdsGrid1.DetailLocationId;
            int replaceTo =
selectedIdsGrid2.DetailLocationId;
            string replaceReason =
textBoxReason.Text;
```

```
            if
(selectedIdsGrid1.DetailTypeId == 1)

databaseConnection.RollerSwap(detail
TypeId, idFrom, idTo, replaceTime,
replaceFrom, replaceTo,
replaceReason);
            else if
(selectedIdsGrid1.DetailTypeId == 2)

databaseConnection.ShaftSwap(detailT
ypeId, idFrom, idTo, replaceTime,
replaceFrom, replaceTo,
replaceReason);
            else if
(selectedIdsGrid1.DetailTypeId == 3)

databaseConnection.GearCageSwap(de
tailTypeId, idFrom, idTo, replaceTime,
replaceFrom, replaceTo,
replaceReason);
            else if
(selectedIdsGrid1.DetailTypeId == 4)

databaseConnection.ClutchSwap(detail
TypeId, idFrom, idTo, replaceTime,
replaceFrom, replaceTo,
replaceReason);
            else if
(selectedIdsGrid1.DetailTypeId == 5)

databaseConnection.ReductionGearSw
ap(detailTypeId, idFrom, idTo,
replaceTime, replaceFrom, replaceTo,
replaceReason);
            else if
(selectedIdsGrid1.DetailTypeId == 6)

databaseConnection.ElectricMotorSwa
p(detailTypeId, idFrom, idTo,
replaceTime, replaceFrom, replaceTo,
replaceReason);


detailsWindow.DataGridRefresh();
            Close();
        }
    }
}
```

DetailToGarbageWindow.xaml.cs

```
using
ClientProgram.DatabaseConnectionFol
der;
using
ClientProgram.DetailsWindowFolder.S
electedIdsFolder;
using System.Windows;

namespace
ClientProgram.DetailsWindowFolder.
DetailToGarbageWindowFolder
{
    public partial class
DetailToGarbageWindow : Window
    {
        private DatabaseConnection
databaseConnection;
        private SelectedIds selectedIds;
        private DetailsWindow
detailsWindow;

        public
DetailToGarbageWindow(DatabaseCo
nnection databaseConnection,
SelectedIds selectedIds,
DetailsWindow detailsWindow)
        {
            InitializeComponent();
            this.databaseConnection =
databaseConnection;
```

```
        this.selectedIds = selectedIds;
        this.detailsWindow =
detailsWindow;
        Title =
$"{selectedIds.GetTitle()} - Відправка
у смітник";
        labelInformation.Content =
$"Чи дійсно відправити деталь
\'{selectedIds.GetTitle()}\' на
металобрухт?";
    }

    private void
buttonConfirm_Click(object sender,
RoutedEventArgs e)
    {
        int detailTypeId =
selectedIds.DetailTypeId;
        int detailId =
selectedIds.DetailId;

        if (detailTypeId == 1)
databaseConnection.RollerRelocation(
detailId, 0, -1);
        else if (detailTypeId == 2)
databaseConnection.ShaftRelocation(d
etailId, 0, -1);
        else if (detailTypeId == 3)
databaseConnection.GearCageRelocati
on(detailId, 0, -1);
        else if (detailTypeId == 4)
databaseConnection.ClutchRelocation(
detailId, 0, -1);
        else if (detailTypeId == 5)
databaseConnection.ReductionGearRel
ocation(detailId, 0, -1);
        else if (detailTypeId == 6)
databaseConnection.ElectricMotorRelo
cation(detailId, 0, -1);

        MessageBox.Show("Деталь
відправлено на металобрухт",
"Сповіщення",
MessageBoxButton.OK,
MessageBoxImage.Information);

detailsWindow.DataGridRefresh();
        Close();
    }

    private void
buttonBack_Click(object sender,
RoutedEventArgs e)
    {
        Close();
    }
  }
}
```

SelectedIds.cs

```
namespace
ClientProgram.DetailsWindowFolder.S
electedIdsFolder
{
  public class SelectedIds
  {
    public int DetailId { get; set; }
    public int DetailTypeId { get; set;
}
    public int DetailLocationId { get;
set; }
    public int DetailStandTypeId {
get; set; }
    public string DetailName { get;
set; }

    public override string ToString()
    {
      string ret = $"id: {DetailId}\n";
      ret += $"detailTypeId:
{DetailTypeId}\n";
```

```
            ret += $"detailLocationId:
{DetailLocationId}\n";
            ret += $"standTypeId:
{DetailStandTypeId}\n";
            ret += $"{DetailName}";

            return ret;
        }

        public string GetTitle()
        {
            return $"{DetailName}
№{DetailId}";
        }
    }
}
```

DetailWindow.xaml.cs

```
using
ClientProgram.DatabaseConnectionFol
der;
using
ClientProgram.DetailsWindowFolder.
DetailHistoryWindowFolder;
using
ClientProgram.DetailsWindowFolder.
DetailLogWindowFolder;
using
ClientProgram.DetailsWindowFolder.
DetailSwapFolder;
using
ClientProgram.DetailsWindowFolder.
DetailToGarbageWindowFolder;
using
ClientProgram.DetailsWindowFolder.S
electedIdsFolder;
using System;
using System.Data;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;

namespace
ClientProgram.DetailsWindowFolder
{
    public partial class DetailsWindow :
Window
    {
        private DatabaseConnection
databaseConnection;
        private SelectedIds?
selectedIdsJournal;
        private SelectedIds?
selectedIdsGrid1;
        private SelectedIds?
selectedIdsGrid2;

        public
DetailsWindow(DatabaseConnection
databaseConnection)
        {
            InitializeComponent();
            this.databaseConnection =
databaseConnection;
            selectedIdsJournal = null;
            selectedIdsGrid1 = null;
            selectedIdsGrid2 = null;

            ComboBoxStandInit();
            ComboBoxTypeInit();
            Task.Run(new
Action(ClockWorking));
            labelConnectionTime.Content =
"З'єднано з сервером о " +
databaseConnection.ConnectionTime.T
oString("dd.MM.yyyy HH:mm:ss");
        }

        public void ClockWorking()
        {
            string time;
            while (true)
```

```
        {
            time = "Час: " +
DateTime.Now.ToString("HH:mm:ss")
;
            Dispatcher.BeginInvoke(()
=> labelClock.Content = time);
            Thread.Sleep(1000);
        }
    }

    public void ComboBoxStandInit()
    {

comboBoxStand.Items.Add("Кліть
дуо");

comboBoxStand.Items.Add("Кліть
чорнова");

comboBoxStand.Items.Add("Кліть
чистова");
        comboBoxStand.SelectedIndex
= 0;
    }

    public void ComboBoxTypeInit()
    {

comboBoxType.Items.Add("Повний
огляд");

comboBoxType.Items.Add("Валок");

comboBoxType.Items.Add("Шпиндел
ь");

comboBoxType.Items.Add("Шестерен
на кліть");

comboBoxType.Items.Add("Муфта");

comboBoxType.Items.Add("Редуктор
");

comboBoxType.Items.Add("Електрод
вигун");
        comboBoxType.SelectedIndex
= -1;
    }

    private void
buttonBack_Click(object sender,
RoutedEventArgs e)
    {
        MainWindow mainWindow =
new
MainWindow(databaseConnection);

mainWindow.WindowStartupLocation
=
WindowStartupLocation.CenterScreen;
        mainWindow.Show();
        Close();
    }

    private SelectedIds?
GetIds(DataGrid dataGrid)
    {
        if (dataGrid.SelectedIndex == -
1 || (dataGrid.SelectedItem as
DataRowView) == null)
            return null;

        return new SelectedIds()
        {
            DetailId =
(int)(dataGrid.SelectedItem as
DataRowView).Row.ItemArray[0],
            DetailTypeId =
(int)(dataGrid.SelectedItem as
DataRowView).Row.ItemArray[1],
```

```
            DetailLocationId =
(int)(dataGrid.SelectedItem as
DataRowView).Row.ItemArray[2],
            DetailStandTypeId =
(int)(dataGrid.SelectedItem as
DataRowView).Row.ItemArray[3],
            DetailName =
(string)(dataGrid.SelectedItem as
DataRowView).Row.ItemArray[4]
        };
    }

    private void
buttonLog_Click(object sender,
RoutedEventArgs e)
    {
        if (selectedIdsJournal == null)
        {

MessageBox.Show("Потрібно
вибрати деталь", "Сповіщення",
MessageBoxButton.OK,
MessageBoxImage.Information);
            return;
        }

        DetailLogWindow
rollerLogWindow = new
DetailLogWindow(databaseConnection
, selectedIdsJournal);

rollerLogWindow.WindowStartupLoca
tion =
WindowStartupLocation.CenterScreen;
        rollerLogWindow.Show();
    }

    private void
buttonHistory_Click(object sender,
RoutedEventArgs e)
    {
        if (selectedIdsJournal == null)
        {

MessageBox.Show("Потрібно
вибрати деталь", "Сповіщення",
MessageBoxButton.OK,
MessageBoxImage.Information);
            return;
        }

        DetailHistoryWindow
rollerHistoryWindow = new
DetailHistoryWindow(databaseConnec
tion, selectedIdsJournal);

rollerHistoryWindow.WindowStartupL
ocation =
WindowStartupLocation.CenterScreen;
        rollerHistoryWindow.Show();
    }

    private void
buttonDetailSwap_Click(object sender,
RoutedEventArgs e)
    {
        if (selectedIdsGrid1 == null ||
selectedIdsGrid2 == null)
        {

MessageBox.Show("Потрібно
вибрати деталь в правій і лівій
таблиці", "Сповіщення",
MessageBoxButton.OK,
MessageBoxImage.Information);
            return;
        }
        else if
(selectedIdsGrid1.DetailTypeId !=
selectedIdsGrid2.DetailTypeId) return;
```

```
            DetailSwapWindow
detailSwapWindow = new
DetailSwapWindow(databaseConnecti
on, selectedIdsGrid1, selectedIdsGrid2,
this);

detailSwapWindow.WindowStartupLo
cation =
WindowStartupLocation.CenterScreen;
            detailSwapWindow.Show();
        }

        private void
buttonToGarbage_Click(object sender,
RoutedEventArgs e)
        {
            if (selectedIdsGrid2 == null)
            {

MessageBox.Show("Потрібно
вибрати деталь у лівій таблиці",
"Сповіщення",
MessageBoxButton.OK,
MessageBoxImage.Information);
                return;
            }

            DetailToGarbageWindow
detailToGarbageWindow = new
DetailToGarbageWindow(databaseCon
nection, selectedIdsGrid2, this);

detailToGarbageWindow.WindowStart
upLocation =
WindowStartupLocation.CenterScreen;

detailToGarbageWindow.Show();
        }

        public void RollerDataGrid(int
standTypeId)
        {
            dataGridRollers1.ItemsSource
=
databaseConnection.GetSelectRoller(1,
100, standTypeId);
            dataGridRollers2.ItemsSource
=
databaseConnection.GetSelectRoller(0,
0, standTypeId);
        }

        public void ShaftDataGrid(int
standTypeId)
        {
            dataGridRollers1.ItemsSource
=
databaseConnection.GetSelectShaft(1,
100, standTypeId);
            dataGridRollers2.ItemsSource
=
databaseConnection.GetSelectShaft(0,
0, standTypeId);
        }

        public void GearCageDataGrid(int
standTypeId)
        {
            dataGridRollers1.ItemsSource
=
databaseConnection.GetSelectGearCag
e(1, 100, standTypeId);
            dataGridRollers2.ItemsSource
=
databaseConnection.GetSelectGearCag
e(0, 0, standTypeId);
        }

        public void ClutchDataGrid(int
standTypeId)
        {
```

```
            dataGridRollers1.ItemsSource
=
databaseConnection.GetSelectClutch(1,
100, standTypeId);
            dataGridRollers2.ItemsSource
=
databaseConnection.GetSelectClutch(0,
0, standTypeId);
        }

        public void
ReductionGearDataGrid(int
standTypeId)
        {
            dataGridRollers1.ItemsSource
=
databaseConnection.GetSelectReductio
nGear(1, 100, standTypeId);
            dataGridRollers2.ItemsSource
=
databaseConnection.GetSelectReductio
nGear(0, 0, standTypeId);
        }

        public void
ElectricMotorDataGrid(int
standTypeId)
        {
            dataGridRollers1.ItemsSource
=
databaseConnection.GetSelectElectric
Motor(1, 100, standTypeId);
            dataGridRollers2.ItemsSource
=
databaseConnection.GetSelectElectric
Motor(0, 0, standTypeId);
        }

        public void StandDataGrid(int
locStart, int locEnd)
        {
            dataGridRollers1.ItemsSource
=
databaseConnection.GetSelectStand(lo
cStart, locEnd);
            dataGridRollers2.ItemsSource
= null;
        }

        public void Grid1VisualImprove()
        {
            if
(dataGridRollers1.Columns.Count <=
0) return;


dataGridRollers1.Columns[1].Visibility
= Visibility.Hidden;

dataGridRollers1.Columns[2].Visibility
= Visibility.Hidden;

dataGridRollers1.Columns[3].Visibility
= Visibility.Hidden;


dataGridRollers1.Columns[0].Header =
"№";

dataGridRollers1.Columns[4].Header =
"Деталь";

dataGridRollers1.Columns[5].Header =
"Розташування";

dataGridRollers1.Columns[6].Header =
"Дата установки";

dataGridRollers1.Columns[7].Header =
"Прокатано";
```

```
dataGridRollers1.Columns[8].Header =
"Ресурс (%)";
        }

        public void Grid2VisualImprove()
        {
            if
(dataGridRollers2.Columns.Count <=
0) return;


dataGridRollers2.Columns[1].Visibility
= Visibility.Hidden;

dataGridRollers2.Columns[2].Visibility
= Visibility.Hidden;

dataGridRollers2.Columns[3].Visibility
= Visibility.Hidden;

dataGridRollers2.Columns[6].Visibility
= Visibility.Hidden;


dataGridRollers2.Columns[0].Header =
"№";

dataGridRollers2.Columns[4].Header =
"Деталь";

dataGridRollers2.Columns[5].Header =
"Розташування";

dataGridRollers2.Columns[7].Header =
"Прокатано";

dataGridRollers2.Columns[8].Header =
"Ресурс (%)";
        }

        public void DataGridRefresh()
        {
            if
(comboBoxStand.SelectedIndex == 0)
            {
                if
(comboBoxType.SelectedIndex == 0)
StandDataGrid(1, 16);
                else if
(comboBoxType.SelectedIndex == 1)
RollerDataGrid(1);
                else if
(comboBoxType.SelectedIndex == 2)
ShaftDataGrid(1);
                else if
(comboBoxType.SelectedIndex == 3)
GearCageDataGrid(1);
                else if
(comboBoxType.SelectedIndex == 4)
ClutchDataGrid(1);
                else if
(comboBoxType.SelectedIndex == 5)
ReductionGearDataGrid(1);
                else if
(comboBoxType.SelectedIndex == 6)
ElectricMotorDataGrid(1);
            }
            else if
(comboBoxStand.SelectedIndex == 1)
            {
                if
(comboBoxType.SelectedIndex == 0)
StandDataGrid(17, 48);
                else if
(comboBoxType.SelectedIndex == 1)
RollerDataGrid(2);
                else if
(comboBoxType.SelectedIndex == 2)
ShaftDataGrid(2);
```

```
            else if
(comboBoxType.SelectedIndex == 3)
GearCageDataGrid(2);
            else if
(comboBoxType.SelectedIndex == 4)
ClutchDataGrid(2);
            else if
(comboBoxType.SelectedIndex == 5)
ReductionGearDataGrid(2);
            else if
(comboBoxType.SelectedIndex == 6)
ElectricMotorDataGrid(2);
        }
        else if
(comboBoxStand.SelectedIndex == 2)
        {
            if
(comboBoxType.SelectedIndex == 0)
StandDataGrid(49, 96);
            else if
(comboBoxType.SelectedIndex == 1)
RollerDataGrid(3);
            else if
(comboBoxType.SelectedIndex == 2)
ShaftDataGrid(3);
            else if
(comboBoxType.SelectedIndex == 3)
GearCageDataGrid(3);
            else if
(comboBoxType.SelectedIndex == 4)
ClutchDataGrid(3);
            else if
(comboBoxType.SelectedIndex == 5)
ReductionGearDataGrid(3);
            else if
(comboBoxType.SelectedIndex == 6)
ElectricMotorDataGrid(3);
        }

        Grid1VisualImprove();
        Grid2VisualImprove();
    }

    private void VariablesRefresh()
    {
        dataGridRollers1.SelectedIndex
= -1;
        dataGridRollers2.SelectedIndex
= -1;
        selectedIdsJournal = null;
        selectedIdsGrid1 = null;
        selectedIdsGrid2 = null;
    }

    private void
FullStandProposition()
    {
        if
(comboBoxType.SelectedIndex > 0 ||
selectedIdsGrid1 == null) return;

        int typeId =
selectedIdsGrid1.DetailTypeId;
        int standTypeId =
selectedIdsGrid1.DetailStandTypeId;

        if (typeId == 1)
dataGridRollers2.ItemsSource =
databaseConnection.GetSelectRoller(0,
0, standTypeId);
        else if (typeId == 2)
dataGridRollers2.ItemsSource =
databaseConnection.GetSelectShaft(0,
0, standTypeId);
        else if (typeId == 3)
dataGridRollers2.ItemsSource =
databaseConnection.GetSelectGearCag
e(0, 0, standTypeId);
        else if (typeId == 4)
dataGridRollers2.ItemsSource =
databaseConnection.GetSelectClutch(0,
0, standTypeId);
```

```
            else if (typeId == 5)
dataGridRollers2.ItemsSource =
databaseConnection.GetSelectReductio
nGear(0, 0, standTypeId);
            else if (typeId == 6)
dataGridRollers2.ItemsSource =
databaseConnection.GetSelectElectric
Motor(0, 0, standTypeId);
            else
dataGridRollers2.ItemsSource = null;

            Grid2VisualImprove();
        }

        private void
comboBoxStand_SelectionChanged(ob
ject sender,
SelectionChangedEventArgs e)
        {
            DataGridRefresh();
            VariablesRefresh();
        }

        private void
comboBoxType_SelectionChanged(obj
ect sender,
SelectionChangedEventArgs e)
        {
            DataGridRefresh();
            VariablesRefresh();
        }

        private void
dataGridRollers1_SelectionChanged(o
bject sender,
SelectionChangedEventArgs e)
        {
            selectedIdsGrid1 =
GetIds(dataGridRollers1);
            FullStandProposition();
            selectedIdsGrid1 =
GetIds(dataGridRollers1);
            selectedIdsJournal =
GetIds(dataGridRollers1);
        }

        private void
dataGridRollers2_SelectionChanged(o
bject sender,
SelectionChangedEventArgs e)
        {
            selectedIdsJournal =
GetIds(dataGridRollers2);
            selectedIdsGrid2 =
GetIds(dataGridRollers2);
        }

        private void
dataGridRollers1_PreviewMouseDown
(object sender,
System.Windows.Input.MouseButtonE
ventArgs e)
        {
            selectedIdsJournal =
GetIds(dataGridRollers1);
        }

        private void
dataGridRollers2_PreviewMouseDown
(object sender,
System.Windows.Input.MouseButtonE
ventArgs e)
        {
            selectedIdsJournal =
GetIds(dataGridRollers2);
        }
    }
}
```

SqlConnectionWindow.xaml.cs

```
using
ClientProgram.DatabaseConnectionFol
der;
using System.IO;
using System.Text.Json;
using System.Windows;

namespace
ClientProgram.SqlConnectionFolder
{
   public partial class
SqlConnectionWindow : Window
   {
      private const string
serverSettingsPath =
"../../../JsonFiles/ServerSettings.json";
      private ServerSettings?
serverSettings;

      public SqlConnectionWindow()
      {
         InitializeComponent();
         WindowStartupLocation =
WindowStartupLocation.CenterScreen;

         serverSettings =
(ServerSettings)JsonSerializer.Deserial
ize(File.ReadAllText(serverSettingsPat
h), typeof(ServerSettings));

         textBoxServer.Text =
serverSettings.ServerName;
         textBoxDatabase.Text =
serverSettings.DatabaseName;
      }

      private void
buttonConnect_Click(object sender,
RoutedEventArgs e)
      {
         serverSettings = new
ServerSettings()
         {
            ServerName =
textBoxServer.Text,
            DatabaseName =
textBoxDatabase.Text,
         };

File.WriteAllText(serverSettingsPath,
JsonSerializer.Serialize(serverSettings)
);

         DatabaseConnection
databaseConnection = new
DatabaseConnection(serverSettings);
         if
(databaseConnection.IsConnectionOk()
)
         {
            MainWindow mainWindow
= new
MainWindow(databaseConnection);

mainWindow.WindowStartupLocation
=
WindowStartupLocation.CenterScreen;
            mainWindow.Show();
         }
         else
         {

MessageBox.Show("Помилка
з'єднання", "Connection Error",
MessageBoxButton.OK,
MessageBoxImage.Error);
         }
         Close();
      }
   }
}
```

StripsWindow.xaml.cs

```
using
ClientProgram.DatabaseConnectionFol
der;
using System.Windows;

namespace
ClientProgram.StripsWindowFolder
{
   public partial class StripsWindow :
Window
   {
      private DatabaseConnection
databaseConnection;

      public
StripsWindow(DatabaseConnection
databaseConnection)
      {
         InitializeComponent();
         this.databaseConnection =
databaseConnection;
         dataGridStrips.ItemsSource =
databaseConnection.SelectStrips();
      }
   }
}
```

MainWindow.xaml.cs

```
using
ClientProgram.DatabaseConnectionFol
der;
using
ClientProgram.DetailAddWindowFold
er;
using
ClientProgram.DetailsWindowFolder;
using System;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using
ClientProgram.StripsWindowFolder;

namespace ClientProgram
{
   public partial class MainWindow :
Window
   {
      public DatabaseConnection
databaseConnection;

      public
MainWindow(DatabaseConnection
databaseConnection)
      {
         InitializeComponent();
         this.databaseConnection =
databaseConnection;
         labelConnectionTime.Content =
"З'єднано з сервером о " +
databaseConnection.ConnectionTime.T
oString("dd.MM.yyyy HH:mm:ss");

         // Clock init
         Task.Run(new
Action(ClockWorking));
      }

      public void ClockWorking()
      {
         string time;
         while (true)
         {
            time = "Час: " +
DateTime.Now.ToString("HH:mm:ss")
;
            Dispatcher.BeginInvoke(()
=> labelClock.Content = time);
            Thread.Sleep(1000);
         }
```

```
        }

        private void
buttonDetails_Click(object sender,
RoutedEventArgs e)
        {
            DetailsWindow rollersWindow
= new
DetailsWindow(databaseConnection);

rollersWindow.WindowStartupLocatio
n =
WindowStartupLocation.CenterScreen;
            rollersWindow.Show();
            Close();
        }

        private void
buttonAddDetails_Click(object sender,
RoutedEventArgs e)
        {
            DetailAddWindow
detailAddWindow = new
DetailAddWindow(databaseConnectio
n);

detailAddWindow.WindowStartupLoc
ation =
WindowStartupLocation.CenterScreen;
            detailAddWindow.Show();
            Close();
        }

        private void
buttonStrips_Click(object sender,
RoutedEventArgs e)
        {
            StripsWindow stripsWindow =
new
StripsWindow(databaseConnection);

stripsWindow.WindowStartupLocation
=
WindowStartupLocation.CenterScreen;
            stripsWindow.Show();
        }
    }
}
```

Програма 2 – Симуляція прокатки металевих слябів.

```
DataConnection.cs
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Text.Json;

namespace
DataSender.DataConnectionFolder
{
    public class DataConnection
    {
        private const string dbSettingsPath
= "../../../Json/DbSettings.json";
        private const string
dbCommandsPath =
"../../../Json/DbCommands.json";

        private DbSettings? dbSettings;
        private DbCommands?
dbCommands;

        private SqlConnection
sqlConnection;

        public DataConnection()
        {
```

```
            dbSettings =
(DbSettings?)JsonSerializer.Deserialize
(File.ReadAllText(dbSettingsPath),
typeof(DbSettings));
            dbCommands =
(DbCommands?)JsonSerializer.Deseria
lize(File.ReadAllText(dbCommandsPat
h), typeof(DbCommands));

            string connectionString =
$"Server={dbSettings?.ServerName};
Database={dbSettings?.DatabaseName
};integrated security=SSPI";
            sqlConnection = new
SqlConnection(connectionString);
        }

        public bool IsConnectionOk()
        {
            try
            {
                sqlConnection.Open();
            }
            catch (SqlException)
            {
                return false;
            }

            return true;
        }

        public DataView GetSelectSteel()
        {
            SqlCommand sqlCommand =
new
SqlCommand(dbCommands?.SelectSte
el, sqlConnection);
            SqlDataAdapter
sqlDataAdapter = new
SqlDataAdapter(sqlCommand);
            DataTable dataTable = new
DataTable();
            sqlDataAdapter.Fill(dataTable);

            return dataTable.DefaultView;
        }

        public void RollingSteel(int? id,
DateTime productionStartTime,
DateTime productionEndTime)
        {
            SqlCommand sqlCommand =
new
SqlCommand(dbCommands?.RollingSt
eel, sqlConnection);

sqlCommand.Parameters.Add(new
SqlParameter("@id", id));

sqlCommand.Parameters.Add(new
SqlParameter("@productionStartTime"
, productionStartTime));

sqlCommand.Parameters.Add(new
SqlParameter("@productionEndTime",
productionEndTime));


sqlCommand.ExecuteNonQuery();
        }

        public void
IncrementRolledStrips()
        {
            SqlCommand sqlCommand =
new
SqlCommand(dbCommands?.Incremen
tRolledStrips, sqlConnection);

sqlCommand.ExecuteNonQuery();
        }
```

```
        public int GetRollingTime(int? steelId)
        {
            SqlCommand sqlCommand = new SqlCommand(dbCommands?.GetRollingTime, sqlConnection);

            sqlCommand.Parameters.Add(new SqlParameter("@steelId", steelId));
            SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

            int seconds = 0;
            while (sqlDataReader.Read())
                seconds = (int)sqlDataReader["Seconds"];
            sqlDataReader.Close();

            return seconds;
        }

        private int GetLastStrip()
        {
            SqlCommand sqlCommand = new SqlCommand(dbCommands?.GetLastStrip, sqlConnection);
            SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

            int ret = 0;
            while (sqlDataReader.Read())
                ret = (int)sqlDataReader["id"];
            sqlDataReader.Close();

            return ret;
        }

        private DataTable GetStandIds(int locStart, int locEnd)
        {
            SqlCommand sqlCommand = new SqlCommand(dbCommands?.SelectStand, sqlConnection);

            sqlCommand.Parameters.Add(new SqlParameter("@locStart", locStart));

            sqlCommand.Parameters.Add(new SqlParameter("@locEnd", locEnd));
            SqlDataAdapter idAdapter = new SqlDataAdapter(sqlCommand);
            DataTable idTable = new DataTable();
            idAdapter.Fill(idTable);

            return idTable;
        }

        private DataTable GetLoads(int? selectedId)
        {
            SqlCommand sqlCommand2 = new SqlCommand(dbCommands?.SelectLoads, sqlConnection);

            sqlCommand2.Parameters.Add(new SqlParameter("@steelId", selectedId));
            SqlDataAdapter loadsAdapter = new SqlDataAdapter(sqlCommand2);
            DataTable loadsTable = new DataTable();
            loadsAdapter.Fill(loadsTable);

            return loadsTable;
        }
```

```
        private void
InsertLogOneDetail(int detailTypeId,
int detailId, int? stripId, DateTime
logTime, double valueLoad, double
valueRpm)
        {
            SqlCommand sqlInsert = new
SqlCommand(dbCommands?.InsertDet
ailLog, sqlConnection);
            sqlInsert.Parameters.Add(new
SqlParameter("@detailTypeId",
detailTypeId));
            sqlInsert.Parameters.Add(new
SqlParameter("@detailId", detailId));
            sqlInsert.Parameters.Add(new
SqlParameter("@stripId", stripId));
            sqlInsert.Parameters.Add(new
SqlParameter("@logTime", logTime));
            sqlInsert.Parameters.Add(new
SqlParameter("@valueLoad",
valueLoad));
            sqlInsert.Parameters.Add(new
SqlParameter("@valueRpm",
valueRpm));
            sqlInsert.ExecuteNonQuery();
        }

        private void InsertLog(int stripId,
DataTable stand, object loadObject,
object speedObject, DateTime
logTime)
        {
            double load =
(double)loadObject;
            double speed =
(double)speedObject;

            // roller 1
            int detailId =
(int)stand.Rows[0]["id"];
            int detailTypeId =
(int)stand.Rows[0]["detailTypeId"];

InsertLogOneDetail(detailTypeId,
detailId, stripId, logTime, load * 11.96,
speed);

            // roller 2
            detailId =
(int)stand.Rows[1]["id"];
            detailTypeId =
(int)stand.Rows[1]["detailTypeId"];

InsertLogOneDetail(detailTypeId,
detailId, stripId, logTime, load * 11.96,
speed);

            // shaft1
            detailId =
(int)stand.Rows[2]["id"];
            detailTypeId =
(int)stand.Rows[2]["detailTypeId"];

InsertLogOneDetail(detailTypeId,
detailId, stripId, logTime, load * 5.075,
speed);

            // shaft2
            detailId =
(int)stand.Rows[3]["id"];
            detailTypeId =
(int)stand.Rows[3]["detailTypeId"];

InsertLogOneDetail(detailTypeId,
detailId, stripId, logTime, load * 5.075,
speed);

            // gearCage
            detailId =
(int)stand.Rows[4]["id"];
```

```
        detailTypeId =
(int)stand.Rows[4]["detailTypeId"];

InsertLogOneDetail(detailTypeId,
detailId, stripId, logTime, load * 2.5,
speed);

        // clutch
        detailId =
(int)stand.Rows[5]["id"];
        detailTypeId =
(int)stand.Rows[5]["detailTypeId"];

InsertLogOneDetail(detailTypeId,
detailId, stripId, logTime, load * 1.375,
speed);

        // reductionGear
        detailId =
(int)stand.Rows[6]["id"];
        detailTypeId =
(int)stand.Rows[6]["detailTypeId"];

InsertLogOneDetail(detailTypeId,
detailId, stripId, logTime, load, speed *
4.56);

        // electricMotor
        detailId =
(int)stand.Rows[7]["id"];
        detailTypeId =
(int)stand.Rows[7]["detailTypeId"];

InsertLogOneDetail(detailTypeId,
detailId, stripId, logTime, load, speed *
4.56);
    }

    public void DetailLogging(int?
selectedId, DateTime
productionStartTime)
    {
        DataTable loads =
GetLoads(selectedId);
        int stripId = GetLastStrip();

        for (int i = 0; i <
loads.Rows.Count; i++)
        {
            DataRow row =
loads.Rows[i];
            DateTime dateTime =
productionStartTime.AddSeconds(i);

            if ((double)row["load1"] >
0.0D)
                InsertLog(stripId,
GetStandIds(1, 8), row["load1"],
row["speed1"], dateTime);
            if ((double)row["load2"] >
0.0D)
                InsertLog(stripId,
GetStandIds(9, 16), row["load2"],
row["speed2"], dateTime);
            if ((double)row["load3"] >
0.0D)
                InsertLog(stripId,
GetStandIds(17, 24), row["load3"],
row["speed3"], dateTime);
            if ((double)row["load4"] >
0.0D)
                InsertLog(stripId,
GetStandIds(25, 32), row["load4"],
row["speed4"], dateTime);
            if ((double)row["load5"] >
0.0D)
                InsertLog(stripId,
GetStandIds(33, 40), row["load5"],
row["speed5"], dateTime);
            if ((double)row["load6"] >
0.0D)
```

```
                InsertLog(stripId,
GetStandIds(41, 48), row["load6"],
row["speed6"], dateTime);
            if ((double)row["load7"] >
0.0D)
                InsertLog(stripId,
GetStandIds(49, 56), row["load7"],
row["speed7"], dateTime);
            if ((double)row["load8"] >
0.0D)
                InsertLog(stripId,
GetStandIds(57, 64), row["load8"],
row["speed8"], dateTime);
            if ((double)row["load9"] >
0.0D)
                InsertLog(stripId,
GetStandIds(65, 72), row["load9"],
row["speed9"], dateTime);
            if ((double)row["load10"] >
0.0D)
                InsertLog(stripId,
GetStandIds(73, 80), row["load10"],
row["speed10"], dateTime);
            if ((double)row["load11"] >
0.0D)
                InsertLog(stripId,
GetStandIds(81, 88), row["load11"],
row["speed11"], dateTime);
            if ((double)row["load12"] >
0.0D)
                InsertLog(stripId,
GetStandIds(89, 96), row["load12"],
row["speed12"], dateTime);
          }
      }

      private void
IncreaseMileageTwoRoller(DataTable
stand, object mileageObject)
      {
          double mileage =
(double)mileageObject;

          SqlCommand sqlCommand =
new
SqlCommand(dbCommands?.Increase
MileageRoller, sqlConnection);
          int rollerUpId =
(int)stand.Rows[0]["id"];

sqlCommand.Parameters.Add(new
SqlParameter("@rollerId",
rollerUpId));

sqlCommand.Parameters.Add(new
SqlParameter("@mileage", mileage));

sqlCommand.ExecuteNonQuery();

          int rollerDownId =
(int)stand.Rows[1]["id"];
          sqlCommand = new
SqlCommand(dbCommands?.Increase
MileageRoller, sqlConnection);

sqlCommand.Parameters.Add(new
SqlParameter("@rollerId",
rollerDownId));

sqlCommand.Parameters.Add(new
SqlParameter("@mileage", mileage));

sqlCommand.ExecuteNonQuery();
      }

      public void
IncreaseMileageRollers(int?
selectedId)
      {
          DataTable loads =
GetLoads(selectedId);
```

```
        for (int i = 0; i <
loads.Rows.Count; i++)
        {
            DataRow row =
loads.Rows[i];

            if ((double)row["load1"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(1, 2), row["speed1"]);
            if ((double)row["load2"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(9, 10), row["speed2"]);
            if ((double)row["load3"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(17, 18), row["speed3"]);
            if ((double)row["load4"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(25, 26), row["speed4"]);
            if ((double)row["load5"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(33, 34), row["speed5"]);
            if ((double)row["load6"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(41, 42), row["speed6"]);
            if ((double)row["load7"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(49, 50), row["speed7"]);
            if ((double)row["load8"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(57, 58), row["speed8"]);
            if ((double)row["load9"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(65, 66), row["speed9"]);
            if ((double)row["load10"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(73, 74), row["speed10"]);
            if ((double)row["load11"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(81, 82), row["speed11"]);
            if ((double)row["load12"] >
0.0D)

IncreaseMileageTwoRoller(GetStandId
s(89, 90), row["speed12"]);
        }
      }
   }
}
```

DbCommands.cs

```
namespace
DataSender.DataConnectionFolder
{
   class DbCommands
   {
      public string SelectSteel { get; set;
} = "";
```

```
        public string RollingSteel { get;
set; } = "";
        public string
IncrementRolledStrips { get; set; } =
"";
        public string GetRollingTime {
get; set; } = "";
        public string SelectStand { get;
set; } = "";
        public string SelectLoads { get;
set; } = "";
        public string InsertDetailLog {
get; set; } = "";
        public string GetLastStrip { get;
set; } = "";
        public string
IncreaseMileageRoller { get; set; } =
"";
    }
}
```

DbSettings.cs

```
namespace
DataSender.DataConnectionFolder
{
    class DbSettings
    {
        public string ServerName { get;
set; } = "";
        public string DatabaseName { get;
set; } = "";
    }
}
```

RollingSteelWindow.xaml.cs

```
using
DataSender.DataConnectionFolder;
using System;
using System.ComponentModel;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;

namespace
DataSender.RollingSteelWindowFolde
r
{
    public partial class
RollingSteelWindow : Window
    {
        private DataConnection
dataConnection;
        private int? selectedId;
        private int rollingCount;

        public
RollingSteelWindow(DataConnection
dataConnection, int? selectedId, int
rollingCount)
        {
            InitializeComponent();
            this.dataConnection =
dataConnection;
            this.selectedId = selectedId;
            this.rollingCount =
rollingCount;

            progressBar1.Minimum = 0;
            progressBar1.Maximum =
rollingCount;
        }

        private void RollingBlock()
        {
            DateTime productionStartTime
= DateTime.Now;
            DateTime productionEndTime
= DateTime.Now;

            for (int i = 0; i < rollingCount;
i++)
            {
```

```
            productionStartTime =
productionEndTime;
            productionEndTime =
productionStartTime.AddSeconds(data
Connection.GetRollingTime(selectedId
));


dataConnection.RollingSteel(selectedId
, productionStartTime,
productionEndTime);

dataConnection.IncrementRolledStrips(
);

dataConnection.DetailLogging(selected
Id, productionStartTime);

dataConnection.IncreaseMileageRoller
s(selectedId);

            Dispatcher.BeginInvoke(()
=> progressBar1.Value++);
        }

        Dispatcher.Invoke(() =>

MessageBox.Show("Прокатка
успішна", "Результат прокатки",
MessageBoxButton.OK,
MessageBoxImage.Information));

        Dispatcher.BeginInvoke(() =>
Close());
    }

    private void
buttonConfirm_Click(object sender,
RoutedEventArgs e)
    {
        Task.Run(() =>
RollingBlock());
        IsEnabled = false;
    }

    private void
buttonBack_Click(object sender,
RoutedEventArgs e)
    {
        Close();
    }
  }
}

MainWindow.xaml.cs
using
DataSender.DataConnectionFolder;
using
DataSender.RollingSteelWindowFolde
r;
using System;
using System.Data;
using System.Windows;

namespace DataSender
{
  public partial class MainWindow :
Window
  {
    private DataConnection
dataConnection;
    private int? selectedId;

    public MainWindow()
    {
        InitializeComponent();
        WindowStartupLocation =
WindowStartupLocation.CenterScreen;
        dataConnection = new
DataConnection();
        DatabaseInit();
```

```
            dataGridSteel.ItemsSource =
dataConnection.GetSelectSteel();
            selectedId = null;
        }

        private void DatabaseInit()
        {
            if
(!dataConnection.IsConnectionOk())
            {

MessageBox.Show("Помилка
з'єднання з сервером",
"SqlConnection",
MessageBoxButton.OK,
MessageBoxImage.Error);
                Close();
            }
        }

        private void
buttonRolling_Click(object sender,
RoutedEventArgs e)
        {
            if (selectedId == null)
            {

MessageBox.Show("Потрібно
вибрати пункт у табдиці",
"Помилка", MessageBoxButton.OK,
MessageBoxImage.Error);
                return;
            }
            else
            {
                RollingSteelWindow
rollingSteelWindow = new
RollingSteelWindow(dataConnection,
selectedId, (int)sliderCount.Value);

rollingSteelWindow.WindowStartupLo
cation =
WindowStartupLocation.CenterScreen;
                rollingSteelWindow.Show();
            }
        }

        private void
dataGridSteel_SelectionChanged(objec
t sender,
System.Windows.Controls.SelectionCh
angedEventArgs e)
        {
            if (dataGridSteel.SelectedIndex
== -1 || (dataGridSteel.SelectedItem as
DataRowView) == null)
                return;
            selectedId =
(int?)(dataGridSteel.SelectedItem as
DataRowView).Row.ItemArray[0];
        }
    }
}
```

ДОДАТОК Ґ

Текст скриптів

ЗАТВЕРДЖЕНО
1116130.01315-01 12 01-ЛЗ

СИСТЕМА КОНТРОЛЮ ЗА РУХОМ ДЕТАЛЕЙ НА МЕТКОМБІНАТІ ТА ОЦІНКА ЇХ ЗАЛИШКОВОГО РЕСУРСА

Текст скриптів

1116130.01315-01 12 01

Листів 18

2023

```
CreateDB.sql
/*********************************
*********************************
**********************
	Database re-create
*********************************
*********************************
**********************/

USE master;

IF DB_ID('RollingMechanism') IS
NOT NULL
BEGIN
	DROP DATABASE
RollingMechanism
END

IF DB_ID('RollingMechanism') IS
NULL
BEGIN
	CREATE DATABASE
RollingMechanism
END

IF DB_ID('RollingMechanism') IS
NOT NULL
BEGIN
	USE RollingMechanism
END

/*********************************
*********************************
**********************
	enum Tables
*********************************
*********************************
**********************/

-- enum Table for rollers type
CREATE TABLE RollingStandType(
	id INT IDENTITY(1, 1)
PRIMARY KEY,
	typeDescription VARCHAR(40)
NOT NULL,
	-- 1 кліть дуо
	-- 2 кліть чорнова
	-- 3 кліть чистова
);

CREATE TABLE DetailType(
	id INT IDENTITY(1, 1)
PRIMARY KEY,
	typeDescription VARCHAR(40)
NOT NULL,
	-- 1 roller
	-- 2 shaft
	-- 3 gear cage
	-- 4 clutch
	-- 5 reduction gear
	-- 6 electric motor
);

-- enum Table for location where -1
broken, 0 in storage and etc
CREATE TABLE DetailLocation(
	id INT IDENTITY(-1, 1)
PRIMARY KEY,
	locationDescription
VARCHAR(40) NOT NULL,
	-- ...
);

/*********************************
*********************************
**********************
	Detail log and history
*********************************
*********************************
**********************/

CREATE TABLE DetailHistory(
```

```
	id INT IDENTITY(1, 1)
PRIMARY KEY,
	detailTypeId INT FOREIGN
KEY REFERENCES DetailType(id)
NOT NULL,
	detailId INT NOT NULL,
	replaceTime DATETIME NOT
NULL,
	replaceFrom INT FOREIGN
KEY REFERENCES
DetailLocation(id) NOT NULL,
	replaceTo INT FOREIGN KEY
REFERENCES DetailLocation(id)
NOT NULL,
	replaceReason VARCHAR(200)
NULL,
);

/**********************************
**********************************
**********************
	Mechanism
**********************************
**********************************
**********************/

CREATE TABLE Roller(
	id INT IDENTITY(1, 1)
PRIMARY KEY,
	detailTypeId INT FOREIGN
KEY REFERENCES DetailType(id)
NOT NULL,
	standTypeId INT FOREIGN
KEY REFERENCES
RollingStandType(id) NOT NULL,
	detailLocationId INT FOREIGN
KEY REFERENCES
DetailLocation(id) NOT NULL,
	installationDate DATETIME
NULL,
	ironStripsDone INT NOT
NULL,
	mileageCurrent FLOAT NOT
NULL,
	mileageMax FLOAT NOT
NULL,
);

CREATE TABLE Shaft(
	id INT IDENTITY(1, 1)
PRIMARY KEY,
	detailTypeId INT FOREIGN
KEY REFERENCES DetailType(id)
NOT NULL,
	standTypeId INT FOREIGN
KEY REFERENCES
RollingStandType(id) NOT NULL,
	detailLocationId INT FOREIGN
KEY REFERENCES
DetailLocation(id) NOT NULL,
	installationDate DATETIME
NULL,
	ironStripsDone INT NOT
NULL,
	mileageCurrent FLOAT NULL,
	mileageMax FLOAT NULL,
);

CREATE TABLE GearCage(
	id INT IDENTITY(1, 1)
PRIMARY KEY,
	detailTypeId INT FOREIGN
KEY REFERENCES DetailType(id)
NOT NULL,
	standTypeId INT FOREIGN
KEY REFERENCES
RollingStandType(id) NOT NULL,
	detailLocationId INT FOREIGN
KEY REFERENCES
DetailLocation(id) NOT NULL,
```

```
	installationDate DATETIME
NULL,
	ironStripsDone INT NOT
NULL,
	mileageCurrent FLOAT NULL,
	mileageMax FLOAT NULL,
);

CREATE TABLE Clutch(
	id INT IDENTITY(1, 1)
PRIMARY KEY,
	detailTypeId INT FOREIGN
KEY REFERENCES DetailType(id)
NOT NULL,
	standTypeId INT FOREIGN
KEY REFERENCES
RollingStandType(id) NOT NULL,
	detailLocationId INT FOREIGN
KEY REFERENCES
DetailLocation(id) NOT NULL,
	installationDate DATETIME
NULL,
	ironStripsDone INT NOT
NULL,
	mileageCurrent FLOAT NULL,
	mileageMax FLOAT NULL,
);

CREATE TABLE ReductionGear(
	id INT IDENTITY(1, 1)
PRIMARY KEY,
	detailTypeId INT FOREIGN
KEY REFERENCES DetailType(id)
NOT NULL,
	standTypeId INT FOREIGN
KEY REFERENCES
RollingStandType(id) NOT NULL,
	detailLocationId INT FOREIGN
KEY REFERENCES
DetailLocation(id) NOT NULL,
	installationDate DATETIME
NULL,
	ironStripsDone INT NOT
NULL,
	mileageCurrent FLOAT NULL,
	mileageMax FLOAT NULL,
);

CREATE TABLE ElectricMotor(
	id INT IDENTITY(1, 1)
PRIMARY KEY,
	detailTypeId INT FOREIGN
KEY REFERENCES DetailType(id)
NOT NULL,
	standTypeId INT FOREIGN
KEY REFERENCES
RollingStandType(id) NOT NULL,
	detailLocationId INT FOREIGN
KEY REFERENCES
DetailLocation(id) NOT NULL,
	installationDate DATETIME
NULL,
	ironStripsDone INT NOT
NULL,
	mileageCurrent FLOAT NULL,
	mileageMax FLOAT NULL,
);

/*******************************
*******************************
**********************
	Metal blocks and strips
*******************************
*******************************
**********************/
CREATE TABLE Steel(
	id INT PRIMARY KEY,
	RTIME VARCHAR(40) NOT
NULL,
	steelGrade VARCHAR(40) NOT
NULL,
```

```
        inHeight FLOAT NOT NULL,
        inWidth FLOAT NOT NULL,
        inLength FLOAT NOT NULL,
        countStrips INT NOT NULL,
        outHeight FLOAT NOT NULL,
        outWidth FLOAT NOT NULL,
        outLength FLOAT NOT NULL,
        mass FLOAT NOT NULL
);

CREATE TABLE Strip(
        id INT IDENTITY(1, 1)
PRIMARY KEY,
        steelId INT FOREIGN KEY
REFERENCES Steel(id) NOT NULL,
        steelGrade VARCHAR(40) NOT
NULL,
        inHeight FLOAT NOT NULL,
        inWidth FLOAT NOT NULL,
        inLength FLOAT NOT NULL,
        outHeight FLOAT NULL,
        outWidth FLOAT NULL,
        outLength FLOAT NULL,
        mass INT NOT NULL,
        productionStartTime
DATETIME NOT NULL,
        productionEndTime DATETIME
NOT NULL,
);

CREATE TABLE DetailLog(
        id INT IDENTITY(1, 1)
PRIMARY KEY,
        detailTypeId INT FOREIGN
KEY REFERENCES DetailType(id)
NOT NULL,
        detailId INT NOT NULL,
        stripId INT FOREIGN KEY
REFERENCES Strip(id) NOT NULL,
        logTime DATETIME NOT
NULL,
        valueLoad FLOAT NOT NULL,
        valueRpm FLOAT NOT NULL,
);

CREATE TABLE Loads(
        id INT IDENTITY(1, 1)
PRIMARY KEY,
        steelId INT FOREIGN KEY
REFERENCES Steel(id) NOT NULL,
        load1 FLOAT NOT NULL,
        load2 FLOAT NOT NULL,
        load3 FLOAT NOT NULL,
        load4 FLOAT NOT NULL,
        load5 FLOAT NOT NULL,
        load6 FLOAT NOT NULL,
        load7 FLOAT NOT NULL,
        load8 FLOAT NOT NULL,
        load9 FLOAT NOT NULL,
        load10 FLOAT NOT NULL,
        load11 FLOAT NOT NULL,
        load12 FLOAT NOT NULL,
        speed1 FLOAT NOT NULL,
        speed2 FLOAT NOT NULL,
        speed3 FLOAT NOT NULL,
        speed4 FLOAT NOT NULL,
        speed5 FLOAT NOT NULL,
        speed6 FLOAT NOT NULL,
        speed7 FLOAT NOT NULL,
        speed8 FLOAT NOT NULL,
        speed9 FLOAT NOT NULL,
        speed10 FLOAT NOT NULL,
        speed11 FLOAT NOT NULL,
        speed12 FLOAT NOT NULL
);

StoredProceduresDB.sql
USE RollingMechanism;

/*******************************
*******************************
**********************
```

```
        Roller
**********************************
**********************************
**********************/
GO
CREATE PROCEDURE SelectRoller(
        @locStart INT,
        @locEnd INT,
        @standTypeId INT
) AS
SELECT Roller.id,
Roller.detailTypeId,
Roller.detailLocationId,
Roller.standTypeId,
        DetailType.typeDescription,
        DetailLocation.locationDescripti
on,
        FORMAT(Roller.installationDat
e, 'dd-MM-yyyy HH:mm:ss'),
        Roller.ironStripsDone,
        ROUND((100.0 -
(Roller.mileageCurrent /
Roller.mileageMax * 100)), 2) AS
'Condition'
FROM Roller
        INNER JOIN RollingStandType
ON Roller.standTypeId =
RollingStandType.id
        INNER JOIN DetailLocation ON
Roller.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
Roller.detailTypeId = DetailType.id
        WHERE standTypeId =
@standTypeId AND detailTypeId = 1
AND detailLocationId >= @locStart
AND detailLocationId <= @locEnd
        ORDER BY detailLocationId
ASC;
```

```
/**********************************
**********************************
**********************
        Shaft
**********************************
**********************************
**********************/
GO
CREATE PROCEDURE SelectShaft(
        @locStart INT,
        @locEnd INT,
        @standTypeId INT
) AS
SELECT Shaft.id, Shaft.detailTypeId,
Shaft.detailLocationId,
Shaft.standTypeId,
        DetailType.typeDescription,
        DetailLocation.locationDescripti
on,
        FORMAT(Shaft.installationDate
, 'dd-MM-yyyy HH:mm:ss'),
        Shaft.ironStripsDone,
        Shaft.mileageCurrent AS
'Condition'
FROM Shaft
        INNER JOIN RollingStandType
ON Shaft.standTypeId =
RollingStandType.id
        INNER JOIN DetailLocation ON
Shaft.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
Shaft.detailTypeId = DetailType.id
        WHERE standTypeId =
@standTypeId AND detailTypeId = 2
AND detailLocationId >= @locStart
AND detailLocationId <= @locEnd
        ORDER BY detailLocationId
ASC;
```

```
/*********************************
*********************************
**********************
        GearCage
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
SelectGearCage(
        @locStart INT,
        @locEnd INT,
        @standTypeId INT
) AS
SELECT GearCage.id,
GearCage.detailTypeId,
GearCage.detailLocationId,
GearCage.standTypeId,
        DetailType.typeDescription,
        DetailLocation.locationDescripti
on,
        FORMAT(GearCage.installation
Date, 'dd-MM-yyyy HH:mm:ss'),
        GearCage.ironStripsDone,
        GearCage.mileageCurrent AS
'Condition'
FROM GearCage
        INNER JOIN RollingStandType
ON GearCage.standTypeId =
RollingStandType.id
        INNER JOIN DetailLocation ON
GearCage.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
GearCage.detailTypeId = DetailType.id
        WHERE standTypeId =
@standTypeId AND detailTypeId = 3
AND detailLocationId >= @locStart
AND detailLocationId <= @locEnd
        ORDER BY detailLocationId
ASC;
```

```
/*********************************
*********************************
**********************
        Clutch
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE SelectClutch(
        @locStart INT,
        @locEnd INT,
        @standTypeId INT
) AS
SELECT Clutch.id,
Clutch.detailTypeId,
Clutch.detailLocationId,
Clutch.standTypeId,
        DetailType.typeDescription,
        DetailLocation.locationDescripti
on,
        FORMAT(Clutch.installationDat
e, 'dd-MM-yyyy HH:mm:ss'),
        Clutch.ironStripsDone,
        Clutch.mileageCurrent AS
'Condition'
FROM Clutch
        INNER JOIN RollingStandType
ON Clutch.standTypeId =
RollingStandType.id
        INNER JOIN DetailLocation ON
Clutch.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
Clutch.detailTypeId = DetailType.id
        WHERE standTypeId =
@standTypeId AND detailTypeId = 4
AND detailLocationId >= @locStart
AND detailLocationId <= @locEnd
        ORDER BY detailLocationId
ASC;
```

```
/*********************************
*********************************
**********************
        ReductionGear
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
SelectReductionGear(
        @locStart INT,
        @locEnd INT,
        @standTypeId INT
) AS
SELECT ReductionGear.id,
ReductionGear.detailTypeId,
ReductionGear.detailLocationId,
ReductionGear.standTypeId,
        DetailType.typeDescription,
        DetailLocation.locationDescripti
on,
        FORMAT(ReductionGear.install
ationDate, 'dd-MM-yyyy HH:mm:ss'),
        ReductionGear.ironStripsDone,
        ReductionGear.mileageCurrent
AS 'Condition'
FROM ReductionGear
        INNER JOIN RollingStandType
ON ReductionGear.standTypeId =
RollingStandType.id
        INNER JOIN DetailLocation ON
ReductionGear.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
ReductionGear.detailTypeId =
DetailType.id
        WHERE standTypeId =
@standTypeId AND detailTypeId = 5
AND detailLocationId >= @locStart
AND detailLocationId <= @locEnd
        ORDER BY detailLocationId
ASC;

/*********************************
*********************************
**********************
        ElectricMotor
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
SelectElectricMotor(
        @locStart INT,
        @locEnd INT,
        @standTypeId INT
) AS
SELECT ElectricMotor.id,
ElectricMotor.detailTypeId,
ElectricMotor.detailLocationId,
ElectricMotor.standTypeId,
        DetailType.typeDescription,
        DetailLocation.locationDescripti
on,
        FORMAT(ElectricMotor.installa
tionDate, 'dd-MM-yyyy HH:mm:ss'),
        ElectricMotor.ironStripsDone,
        ElectricMotor.mileageCurrent
AS 'Condition'
FROM ElectricMotor
        INNER JOIN RollingStandType
ON ElectricMotor.standTypeId =
RollingStandType.id
        INNER JOIN DetailLocation ON
ElectricMotor.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
ElectricMotor.detailTypeId =
DetailType.id
        WHERE standTypeId =
@standTypeId AND detailTypeId = 6
```

```
AND detailLocationId >= @locStart
AND detailLocationId <= @locEnd
	ORDER BY detailLocationId
ASC;

/*********************************
*********************************
**********************
	Select full stand
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE SelectStand(
	@locStart INT,
	@locEnd INT
) AS
(SELECT Roller.id,
Roller.detailTypeId,
Roller.detailLocationId,
Roller.standTypeId,
	DetailType.typeDescription,
DetailLocation.locationDescription,
	FORMAT(Roller.installationDat
e, 'dd-MM-yyyy HH:mm:ss'),
Roller.ironStripsDone,
	ROUND((100.0 -
(Roller.mileageCurrent /
Roller.mileageMax * 100)), 2) AS
'Condition'
	FROM Roller
	INNER JOIN RollingStandType
ON Roller.standTypeId =
RollingStandType.id
	INNER JOIN DetailLocation ON
Roller.detailLocationId =
DetailLocation.id
	INNER JOIN DetailType ON
Roller.detailTypeId = DetailType.id
	WHERE Roller.detailLocationId
>= @locStart AND
Roller.detailLocationId <= @locEnd
UNION
SELECT Shaft.id, Shaft.detailTypeId,
Shaft.detailLocationId,
Shaft.standTypeId,
	DetailType.typeDescription,
DetailLocation.locationDescription,
	FORMAT(Shaft.installationDate
, 'dd-MM-yyyy HH:mm:ss'),
Shaft.ironStripsDone,
	Shaft.mileageCurrent AS
'Condition'
	FROM Shaft
	INNER JOIN RollingStandType
ON Shaft.standTypeId =
RollingStandType.id
	INNER JOIN DetailLocation ON
Shaft.detailLocationId =
DetailLocation.id
	INNER JOIN DetailType ON
Shaft.detailTypeId = DetailType.id
	WHERE Shaft.detailLocationId
>= @locStart AND
Shaft.detailLocationId <= @locEnd
UNION
SELECT GearCage.id,
GearCage.detailTypeId,
GearCage.detailLocationId,
GearCage.standTypeId,
	DetailType.typeDescription,
DetailLocation.locationDescription,
	FORMAT(GearCage.installation
Date, 'dd-MM-yyyy HH:mm:ss'),
GearCage.ironStripsDone,
	GearCage.mileageCurrent AS
'Condition'
	FROM GearCage
```

```
        INNER JOIN RollingStandType
ON GearCage.standTypeId =
RollingStandType.id
        INNER JOIN DetailLocation ON
GearCage.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
GearCage.detailTypeId = DetailType.id
        WHERE
GearCage.detailLocationId >=
@locStart AND
GearCage.detailLocationId <=
@locEnd
UNION
SELECT Clutch.id,
Clutch.detailTypeId,
Clutch.detailLocationId,
Clutch.standTypeId,
        DetailType.typeDescription,
DetailLocation.locationDescription,
        FORMAT(Clutch.installationDat
e, 'dd-MM-yyyy HH:mm:ss'),
Clutch.ironStripsDone,
        Clutch.mileageCurrent AS
'Condition'
        FROM Clutch
        INNER JOIN RollingStandType
ON Clutch.standTypeId =
RollingStandType.id
        INNER JOIN DetailLocation ON
Clutch.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
Clutch.detailTypeId = DetailType.id
        WHERE Clutch.detailLocationId
>= @locStart AND
Clutch.detailLocationId <= @locEnd
UNION
SELECT ReductionGear.id,
ReductionGear.detailTypeId,
ReductionGear.detailLocationId,
ReductionGear.standTypeId,
        DetailType.typeDescription,
DetailLocation.locationDescription,
        FORMAT(ReductionGear.install
ationDate, 'dd-MM-yyyy HH:mm:ss'),
ReductionGear.ironStripsDone,
        ReductionGear.mileageCurrent
AS 'Condition'
        FROM ReductionGear
        INNER JOIN RollingStandType
ON ReductionGear.standTypeId =
RollingStandType.id
        INNER JOIN DetailLocation ON
ReductionGear.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
ReductionGear.detailTypeId =
DetailType.id
        WHERE
ReductionGear.detailLocationId >=
@locStart AND
ReductionGear.detailLocationId <=
@locEnd
UNION
SELECT ElectricMotor.id,
ElectricMotor.detailTypeId,
ElectricMotor.detailLocationId,
ElectricMotor.standTypeId,
        DetailType.typeDescription,
DetailLocation.locationDescription,
        FORMAT(ElectricMotor.installa
tionDate, 'dd-MM-yyyy HH:mm:ss'),
ElectricMotor.ironStripsDone,
        ElectricMotor.mileageCurrent
AS 'Condition'
        FROM ElectricMotor
        INNER JOIN RollingStandType
ON ElectricMotor.standTypeId =
RollingStandType.id
```

```
        INNER JOIN DetailLocation ON
ElectricMotor.detailLocationId =
DetailLocation.id
        INNER JOIN DetailType ON
ElectricMotor.detailTypeId =
DetailType.id
        WHERE
ElectricMotor.detailLocationId >=
@locStart AND
ElectricMotor.detailLocationId <=
@locEnd) ORDER BY
detailLocationId ASC

/*********************************
*********************************
**********************
        Select detail log
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
SelectDetailLog(
        @detailTypeId INT,
        @detailId INT
) AS
SELECT
        ROW_NUMBER()
OVER(ORDER BY DetailLog.id ASC)
AS '№',
        Strip.id AS '№ Стрічки',
        Strip.steelGrade AS 'Марка
сталі',
        FORMAT(DetailLog.logTime,
'dd-MM-yyyy HH:mm:ss') AS 'Час',
        ROUND(valueLoad, 2) AS
'Навантаження (Н*м)',
        ROUND(DetailLog.valueRpm,
2) AS 'Швидкість (м/с)'
FROM DetailLog
        INNER JOIN Strip ON Strip.id =
DetailLog.stripId
        WHERE detailTypeId =
@detailTypeId AND detailId =
@detailId;

/*********************************
*********************************
**********************
        Select detail history
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
SelectDetailHistory(
        @detailTypeId INT,
        @detailId INT
) AS
SELECT
        ROW_NUMBER()
OVER(ORDER BY DetailHistory.id
ASC) AS '№',
        FORMAT(replaceTime, 'dd-
MM-yyyy HH:mm:ss') AS 'Час',
        d1.locationDescription AS
'Звідки',
        d2.locationDescription AS
'Куди',
        DetailHistory.replaceReason AS
'Причина'
FROM DetailHistory
        INNER JOIN DetailLocation d1
ON d1.id = replaceFrom
        INNER JOIN DetailLocation d2
ON d2.id = replaceTo
        WHERE detailTypeId =
@detailTypeId AND detailId =
@detailId;
```

```
/*********************************
*********************************
**********************
        Add detail history
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
AddDetailHistory(
        @detailTypeId INT,
        @detailId INT,
        @replaceTime DATETIME,
        @replaceFrom INT,
        @replaceTo INT,
        @replaceReason
VARCHAR(200)
) AS
INSERT INTO DetailHistory
VALUES(@detailTypeId, @detailId,
@replaceTime, @replaceFrom,
@replaceTo, @replaceReason);

/*********************************
*********************************
**********************
        Roller swap
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE RollerSwap(
        @detailTypeId INT,
        @idFrom INT,
        @idTo INT,
        @replaceTime DATETIME,
        @replaceFrom INT,
        @replaceTo INT,
        @replaceReason
VARCHAR(200)
) AS
UPDATE Roller SET detailLocationId
= @replaceTo, installationDate =
NULL WHERE id = @idFrom;
UPDATE Roller SET detailLocationId
= @replaceFrom, installationDate =
@replaceTime WHERE id = @idTo;
EXEC AddDetailHistory
@detailTypeId, @idFrom,
@replaceTime, @replaceFrom,
@replaceTo, @replaceReason;
EXEC AddDetailHistory
@detailTypeId, @idTo,
@replaceTime, @replaceTo,
@replaceFrom, NULL;

/*********************************
*********************************
**********************
        Shaft swap
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE ShaftSwap(
        @detailTypeId INT,
        @idFrom INT,
        @idTo INT,
        @replaceTime DATETIME,
        @replaceFrom INT,
        @replaceTo INT,
        @replaceReason
VARCHAR(200)
) AS
UPDATE Shaft SET detailLocationId
= @replaceTo, installationDate =
NULL WHERE id = @idFrom;
UPDATE Shaft SET detailLocationId
= @replaceFrom, installationDate =
@replaceTime WHERE id = @idTo;
EXEC AddDetailHistory
@detailTypeId, @idFrom,
```

```
@replaceTime, @replaceFrom,
@replaceTo, @replaceReason;
EXEC AddDetailHistory
@detailTypeId, @idTo,
@replaceTime, @replaceTo,
@replaceFrom, NULL;

/*********************************
*********************************
**********************
        GearCage swap
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
GearCageSwap(
        @detailTypeId INT,
        @idFrom INT,
        @idTo INT,
        @replaceTime DATETIME,
        @replaceFrom INT,
        @replaceTo INT,
        @replaceReason
VARCHAR(200)
) AS
UPDATE GearCage SET
detailLocationId = @replaceTo,
installationDate = NULL WHERE id =
@idFrom;
UPDATE GearCage SET
detailLocationId = @replaceFrom,
installationDate = @replaceTime
WHERE id = @idTo;
EXEC AddDetailHistory
@detailTypeId, @idFrom,
@replaceTime, @replaceFrom,
@replaceTo, @replaceReason;
EXEC AddDetailHistory
@detailTypeId, @idTo,
@replaceTime, @replaceTo,
@replaceFrom, NULL;

/*********************************
*********************************
**********************
        Clutch swap
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE ClutchSwap(
        @detailTypeId INT,
        @idFrom INT,
        @idTo INT,
        @replaceTime DATETIME,
        @replaceFrom INT,
        @replaceTo INT,
        @replaceReason
VARCHAR(200)
) AS
UPDATE Clutch SET detailLocationId
= @replaceTo, installationDate =
NULL WHERE id = @idFrom;
UPDATE Clutch SET detailLocationId
= @replaceFrom, installationDate =
@replaceTime WHERE id = @idTo;
EXEC AddDetailHistory
@detailTypeId, @idFrom,
@replaceTime, @replaceFrom,
@replaceTo, @replaceReason;
EXEC AddDetailHistory
@detailTypeId, @idTo,
@replaceTime, @replaceTo,
@replaceFrom, NULL;

/*********************************
*********************************
**********************
        ReductionGear swap
```

```
**********
**********
**********/
GO
CREATE PROCEDURE
ReductionGearSwap(
	@detailTypeId INT,
	@idFrom INT,
	@idTo INT,
	@replaceTime DATETIME,
	@replaceFrom INT,
	@replaceTo INT,
	@replaceReason
VARCHAR(200)
) AS
UPDATE ReductionGear SET
detailLocationId = @replaceTo,
installationDate = NULL WHERE id =
@idFrom;
UPDATE ReductionGear SET
detailLocationId = @replaceFrom,
installationDate = @replaceTime
WHERE id = @idTo;
EXEC AddDetailHistory
@detailTypeId, @idFrom,
@replaceTime, @replaceFrom,
@replaceTo, @replaceReason;
EXEC AddDetailHistory
@detailTypeId, @idTo,
@replaceTime, @replaceTo,
@replaceFrom, NULL;

/**********
**********
**********
	ElectricMotor swap
**********
**********
**********/
GO
CREATE PROCEDURE
ElectricMotorSwap(
	@detailTypeId INT,
	@idFrom INT,
	@idTo INT,
	@replaceTime DATETIME,
	@replaceFrom INT,
	@replaceTo INT,
	@replaceReason
VARCHAR(200)
) AS
UPDATE ElectricMotor SET
detailLocationId = @replaceTo,
installationDate = NULL WHERE id =
@idFrom;
UPDATE ElectricMotor SET
detailLocationId = @replaceFrom,
installationDate = @replaceTime
WHERE id = @idTo;
EXEC AddDetailHistory
@detailTypeId, @idFrom,
@replaceTime, @replaceFrom,
@replaceTo, @replaceReason;
EXEC AddDetailHistory
@detailTypeId, @idTo,
@replaceTime, @replaceTo,
@replaceFrom, NULL;

/**********
**********
**********
	Insert Roller
**********
**********
**********/
GO
CREATE PROCEDURE InsertRoller(
	@standTypeId INT,
	@mileageMax FLOAT
) AS
```

```
INSERT INTO Roller VALUES (1,
@standTypeId, 0, NULL, 0, 0,
@mileageMax);

/*********************************
*********************************
**********************
        Insert Shaft
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE InsertShaft(
        @standTypeId INT
) AS
INSERT INTO Shaft VALUES (2,
@standTypeId, 0, NULL, 0, NULL,
NULL);

/*********************************
*********************************
**********************
        Insert GearCage
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
InsertGearCage(
        @standTypeId INT
) AS
INSERT INTO GearCage VALUES (3,
@standTypeId, 0, NULL, 0, NULL,
NULL);

/*********************************
*********************************
**********************
        Insert Clutch
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE InsertClutch(
        @standTypeId INT
) AS
INSERT INTO Clutch VALUES (4,
@standTypeId, 0, NULL, 0, NULL,
NULL);

/*********************************
*********************************
**********************
        Insert ReductionGear
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
InsertReductionGear(
        @standTypeId INT
) AS
INSERT INTO ReductionGear
VALUES (5, @standTypeId, 0, NULL,
0, NULL, NULL);

/*********************************
*********************************
**********************
        Insert ElectricMotor
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
InsertElectricMotor(
        @standTypeId INT
) AS
```

```
INSERT INTO ElectricMotor
VALUES (6, @standTypeId, 0, NULL,
0, NULL, NULL);

/*********************************
*********************************
**********************
        Select strips
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE SelectStrips
AS
SELECT
        Strip.id AS '№',
        Steel.steelGrade AS 'Марка
сталі',
        FORMAT(Strip.productionStart
Time, 'dd-MM-yyyy HH:mm:ss') AS
'Початок прокату',
        Strip.inHeight AS 'Висота
вхідна (мм)',
        Strip.inWidth AS 'Ширина
вхідна (мм)',
        Strip.outWidth AS 'Довжина
вхідна (мм)',
        FORMAT(Strip.productionEndT
ime, 'dd-MM-yyyy HH:mm:ss') AS
'Кінець прокату',
        Strip.outHeight AS 'Висота
вихідна (мм)',
        Strip.outWidth AS 'Ширина
вихідна (мм)',
        Strip.outLength AS 'Довжина
вихідна (мм)',
        Strip.mass AS 'Маса (кг)'
FROM Strip
        INNER JOIN Steel ON Steel.id
= Strip.steelId

/*********************************
*********************************
**********************
        Select steel
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE SelectSteel
AS SELECT
        Steel.id AS '№',
        Steel.steelGrade AS 'Марка
сталі',
        Steel.inHeight AS 'Вхідна
висота (мм)',
        Steel.inWidth AS 'Вхідна
ширина (мм)',
        Steel.inLength AS 'Вхідна
довжина (мм)',
        Steel.mass AS 'Маса (кг)'
FROM Steel;

/*********************************
*********************************
**********************
        Rolling steel
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE RollingSteel(
        @id INT,
        @productionStartTime
DATETIME,
        @productionEndTime
DATETIME
) AS
INSERT INTO Strip
        SELECT
        id, steelGrade, inHeight,
inWidth,  inLength,
```

```
        outHeight, outWidth, outLength,
mass,
        @productionStartTime,
@productionEndTime
        FROM Steel WHERE @id =
Steel.id;

/**********************************
**********************************
**********************
        Increment rolled strips
**********************************
**********************************
**********************/
GO
CREATE PROCEDURE
IncrementRolledStrips
AS
UPDATE Roller SET ironStripsDone =
ironStripsDone + 1 WHERE
detailLocationId > 0;
UPDATE Shaft SET ironStripsDone =
ironStripsDone + 1 WHERE
detailLocationId > 0;
UPDATE GearCage SET
ironStripsDone = ironStripsDone + 1
WHERE detailLocationId > 0;
UPDATE Clutch SET ironStripsDone
= ironStripsDone + 1 WHERE
detailLocationId > 0;
UPDATE ReductionGear SET
ironStripsDone = ironStripsDone + 1
WHERE detailLocationId > 0;
UPDATE ElectricMotor SET
ironStripsDone = ironStripsDone + 1
WHERE detailLocationId > 0;

/**********************************
**********************************
**********************
        Get rolling time
**********************************
**********************************
**********************/
GO
CREATE PROCEDURE
GetRollingTime(
        @steelId INT
) AS
SELECT COUNT(*) AS Seconds
FROM Loads WHERE @steelId =
Loads.steelId;

/**********************************
**********************************
**********************
        Insert detail log
**********************************
**********************************
**********************/
GO
CREATE PROCEDURE
InsertDetailLog(
        @detailTypeId INT,
        @detailId INT,
        @stripId INT,
        @logTime DATETIME,
        @valueLoad FLOAT,
        @valueRpm FLOAT
) AS
INSERT INTO DetailLog
VALUES(@detailTypeId, @detailId,
@stripId, @logTime, @valueLoad,
@valueRpm);

/**********************************
**********************************
**********************
        Select loads
**********************************
**********************************
**********************/
```

```
GO
CREATE PROCEDURE SelectLoads(
	@steelId INT
) AS
SELECT
	id,
	load1, load2, load3, load4, load5,
load6,
	load7, load8, load9, load10,
load11, load12,
	speed1, speed2, speed3, speed4,
speed5, speed6,
	speed7, speed8, speed9, speed10,
speed11, speed12
FROM Loads WHERE steelId =
@steelId;

/**********************************
**********************************
**********************
	Get last strip id
**********************************
**********************************
**********************/
GO
CREATE PROCEDURE GetLastStrip
AS
SELECT * FROM Strip WHERE id =
(SELECT MAX(id) FROM Strip);

/**********************************
**********************************
**********************
	Increase mileage roller
**********************************
**********************************
**********************/
GO
CREATE PROCEDURE
IncreaseMileageRoller(
	@rollerId INT,
	@mileage FLOAT
) AS
UPDATE Roller SET
Roller.mileageCurrent =
Roller.mileageCurrent + @mileage
WHERE Roller.id = @rollerId;

/**********************************
**********************************
**********************
	Roller relocation
**********************************
**********************************
**********************/
GO
CREATE PROCEDURE
RollerRelocation(
	@detailId INT,
	@locFrom INT,
	@locTo INT
) AS
UPDATE Roller SET
Roller.detailLocationId = @locTo
WHERE Roller.id = @detailId;

/**********************************
**********************************
**********************
	Shaft relocation
**********************************
**********************************
**********************/
GO
CREATE PROCEDURE
ShaftRelocation(
	@detailId INT,
	@locFrom INT,
	@locTo INT
) AS
```

```
UPDATE Shaft SET
Shaft.detailLocationId = @locTo
WHERE Shaft.id = @detailId;

/*********************************
*********************************
**********************
        GearCage relocation
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
GearCageRelocation(
        @detailId INT,
        @locFrom INT,
        @locTo INT
) AS
UPDATE GearCage SET
GearCage.detailLocationId = @locTo
WHERE GearCage.id = @detailId;

/*********************************
*********************************
**********************
        Clutch relocation
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
ClutchRelocation(
        @detailId INT,
        @locFrom INT,
        @locTo INT
) AS
UPDATE Clutch SET
Clutch.detailLocationId = @locTo
WHERE Clutch.id = @detailId;

/*********************************
*********************************
**********************
        ReductionGear relocation
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
ReductionGearRelocation(
        @detailId INT,
        @locFrom INT,
        @locTo INT
) AS
UPDATE ReductionGear SET
ReductionGear.detailLocationId =
@locTo WHERE ReductionGear.id =
@detailId;

/*********************************
*********************************
**********************
        ElectricMotor relocation
*********************************
*********************************
**********************/
GO
CREATE PROCEDURE
ElectricMotorRelocation(
        @detailId INT,
        @locFrom INT,
        @locTo INT
) AS
UPDATE ElectricMotor SET
ElectricMotor.detailLocationId =
@locTo WHERE ElectricMotor.id =
@detailId;
```